Vikramaditya Story - Revisited with a Dose of Graph Theory

Mukkai S. Krishnamoorthy and Narsingh Deo

Contents

Dedication

To our teachers and to our students

copyright

Acknowledgements

The first author thanks Professor Narsingh Deo (second author) for teaching him Graph Theory and for being a source of inspiration to him.

The first author is also thankful for the numerous constructive comments he received from many. He especially acknowledges the help of Dr. Janaki Krishnamoorthy, Dr. Raju Krishnamoorthy, Prof. Gurpur Prabhu, Prof. Sam Kim, Dr. Srinivasan Rajan, Dr. Allen Lavoie, Prof. H.S. Mani, Dr. Luis Tari, Dr. T. V. Raman, Mr. T. Thomas and Prof. S. Narayanan. Additionally, he wants to thank Dr. David Goldschmidt for going through the entire manuscript and making numerous changes.

Chapter 1

Introduction

The goal of this text is to teach mathematical and computer science concepts through a series of stories designed for students in grades 7-10.

The storyline is similar to the Vikramaditya stories, also called the Vetala tales, in Indian folklore. These stories are believed to have taken place in the 11th century BCE.[1] Collectively, there are 21 stories, one taking place each night. Each story begins with a series of questions, and the protagonist has to successfully answer those questions to set himself free.

Our stories start on October 31st, Halloween. They take place in a hamlet called Royt, a small college town somewhere in Upstate New York. Royt was a prosperous town a hundred years ago, but these days the town has many dilapidated buildings and a rather imposing cemetery. Ajur and his parents live in this hamlet. Ajur has a pet dog named Jura, who accompanies Ajur wherever he goes. In the cemetery lives Rishnak, a ghost, who was a tyrant but now has good intentions.

1.1 Characters

Ajur - A young boy who is interested in mathematics but easily gets bored.
Jura - Ajur's pet dog.
Kinaja - An angel who helps Ajur overcome challenges posed by Rishnak.
Rishnak - A ghost with a mathematical bent of mind from whom Ajur wants to escape by solving various mathematical challenges.

[1] https://en.wikipedia.org/wiki/List_of_Vetala_Tales

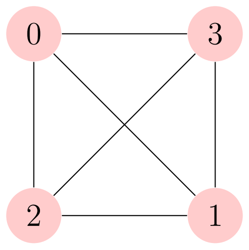


Figure 1.1: A graph with four vertices and six edges

1.2 Notation

Graphs , also known as networks , occur naturally in many different applications. They are abstractions of a relation between any two objects, i.e., a binary relation. Objects can, for example, be people, cities, countries, or webpages. These objects are represented by **vertices**, usually drawn as dots or circles on a page. Relations may exist between any two different objects. These relations are represented as lines connecting the two vertices and are called **edges** .

We will illustrate with three examples. In our first example [Figure 1.1], there are four vertices representing objects, labeled with the numbers 0, 1, 2, and 3 respectively. There are six relations (or edges); these relations are {0,1}, {0,2}, {0,3}, {1,2}, {1,3}, and {2,3}. These relations are symmetric, meaning if there is a relation between vertex 0 and vertex 1, then there is also a relation between vertex 1 and vertex 0. Such graphs are called **undirected** graphs.

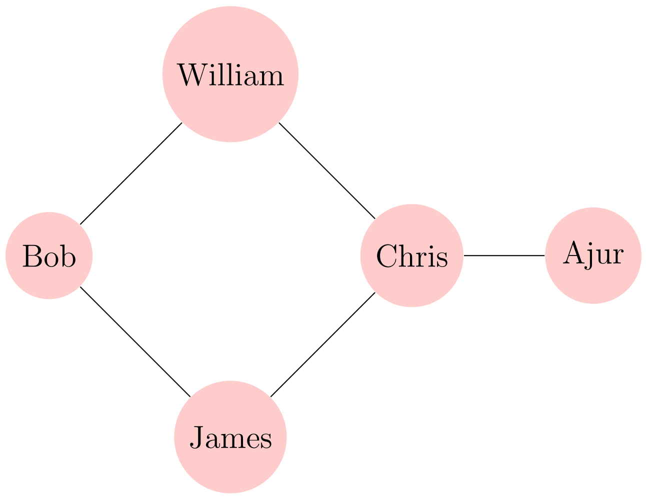


Figure 1.2: A friendship graph with five vertices and five edges

In our next example [Figure 1.2], we have five vertices representing five people, namely Bob, William, James, Chris, and Ajur. The edges in the graph represent friendship. Chris is friends with William, James, and Ajur. Bob is friends with William and James. These friendship relations are depicted as a graph with five vertices and five edges.

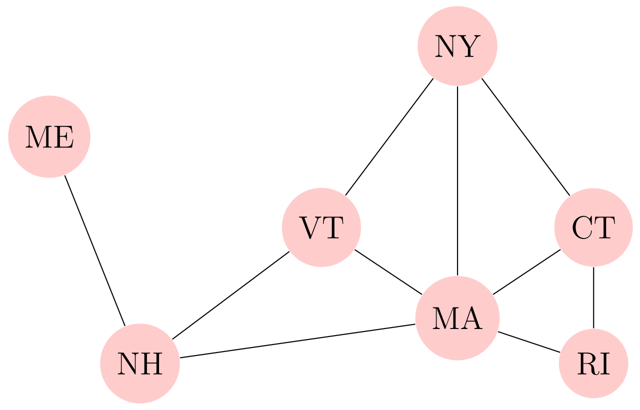


Figure 1.3: A graph representing seven Northeastern states in the United States, with seven vertices and 10 edges

In our third example [Figure 1.3], we have seven Northeastern states in the United States, namely New York (NY), Connecticut (CT), Vermont (VT), Maine (ME), Massachusetts (MA), Rhode Island (RI), and New Hampshire (NH). The relationship represented in this graph is the sharing of a border with another state. Therefore, NY borders CT, VT, and MA. CT additionally borders RI and MA. VT additionally borders MA and NH. ME only borders NH. MA additionally borders RI and NH. These state-border relations are depicted as a graph with seven vertices and 10 edges.

One other term to introduce here. If there is an edge between two vertices (e.g., between NY and VT in Figure 1.3), we say that these two vertices are **adjacent**. More specifically, NY is adjacent to VT. And VT is adjacent to NY.

Chapter 2

A Trip to the Cemetery

It was Halloween, a cold and dreary fall afternoon. Ajur was interested in visiting the famous cemetery in Royt as any teenager with an adventurous spirit would be wont to do. His pet dog Jura, a Labrador, was eager to follow. As Ajur examined the old headstones, he lost track of time, and it grew dark. He felt tired, so he and Jura sat under a tree and fell asleep.

Deep in sleep, they were awakened by a loud noise. The noise was from Rishnak, a ghost who lived in the cemetery. Rishnak had died many years ago; he used to teach mathematics and graph theory to talented high school students all throughout the area. When Rishnak spotted Ajur in the cemetery, he thought to himself, "Here is an unusual teenager." Perhaps he could test whether this youngster was proficient in mathematics. And if he was indeed proficient, Rishnak could reward him with his magical powers. When Rishnak appeared in front of Ajur, Jura jumped up and started barking. Ajur sat up with a start.

Rishnak was afraid he would intimidate Ajur by being brusque. Jura's barking became louder and louder, so Rishnak started talking in a soft voice. He gently asked what grade Ajur was in. Ajur replied that he was in the 8th grade. Rishnak asked what subjects Ajur liked most in school, to which Ajur proudly said, "mathematics." Rishnak smiled. He told the boy that he had been cursed to live as a ghost, and his curse could only be lifted if he could reward a youngster who could answer all of his questions.

Ajur was intrigued by Rishnak's plight. Standing up now, Ajur was excited for the possibility of rewards and the opportunity to help lift the curse on Rishnak, a poor ghost. Ajur also thought he would be able to tell his friends about his adventure.

Chapter 3

Degrees

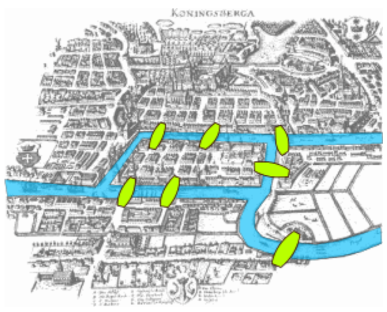
Rishnak asked Ajur whether he knew about graphs and trees in mathematics. Ajur jumped up and down and proclaimed that he knew all about graph theory. He continued that graphs are often used to represent relations of a set of objects. Smiling broadly, he stated that the famous mathematician, Euler, was the father of graph theory. He was eager to show off his knowledge and described the famous Königsberg bridge problem. Königsberg (now Kaliningrad in modern day Russia) was on the banks of the Pregel River. There were seven bridges crossing this river, connecting the two sides of the river with two islands as shown in [Figure 3.1].[1] Rishnak emphasized that modeling this real word problem as a mathematical problem is an important component of understanding and producing a solution. Using a stick, Ajur drew a graph of this in the dirt.

He continued, stating that the problem was to start from the vertex labeled 0 and to walk across all bridges once and only once, finally returning to start vertex 0. Ajur could not contain his enthusiasm and asked Rishnak how to do it. Rishnak gently reminded Ajur that he was the one asking the questions and that Ajur had to respond with the correct responses. Ajur nodded his head.

Rishnak told Ajur, "I am going to ask you a series of questions, then I'll observe whether I have any chance of having my curse lifted."

Rishnak then asked Ajur, "In a class of 33 students, what is the maximum and minimum number of friends a student can have? And since friend is a mutual relation, this means that if A is a friend of B, then B is a friend

[1]Bogdan Giuşcă - Public domain (PD), based on the image



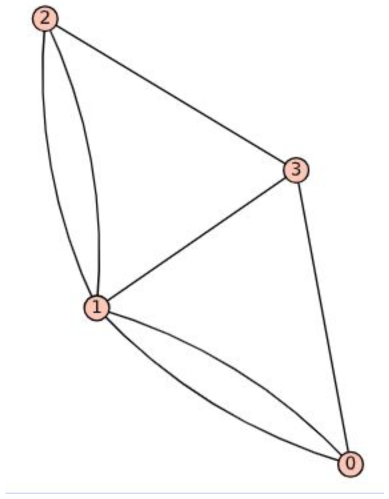


Figure 3.1: Original Königsberg Bridge and a representation by a graph

of A. In other words, a student cannot be a friend to herself!"

Ajur replied nonchalantly that the maximum number of friends is 32 and the minimum number is zero.

Rishnak then asked Ajur, "Will there be a student in the class with 32 friends and a student with zero friends?"

Ajur smiled to himself, seeing right away that Rishnak was a clever ghost. Ajur replied, "How is that possible? If a student has 32 friends, she is friends with everyone else, so everyone else would have at least one friend. So there cannot be a student with zero friends."

Ajur paused and thought for a few seconds, then said, "Similarly, if there is a student with zero friends, there are only 32 students remaining, so another student can have at most 31 friends."

Rishnak was pleased to have found someone who seemed to have the ability to help him. His questioning continued. "Can all 33 students have a distinct number of friends? In other words, can each student have a distinct number of friends that differs from every other student?"

Ajur responded immediately by saying this was not possible because with 33 distinct students, the distinct number of friends each student in the class could have would be $\{0, 1, 2, \cdots, 32\}$. He smiled and said, "And there are 33 numbers in this set, but alas, it contains both a students with 32 friends and a student with zero friends, which is not possible."

Rishnak asked Ajur, "Can there be just two students with the same number of friends, with the other students all having a distinct numbers of friends? And in this case, all of the students have at least one friend."

Ajur frowned. This was not as easy a problem. There was no longer a student with zero friends. Ajur said, "Give me a moment to reason this out." He knew that the maximum number could only be 32 and the minimum had to be one, otherwise there would not be 32 distinct numbers. But there are 33 students.

"Aha," said Ajur, "by the pigeonhole principle , which states that if there are more pigeons than boxes, then one box must contain at least two pigeons, there has to be two students with the same number of friends."

Rishnak said nothing, then asked, "Are you sure that's it?"

Ajur quickly said, "No, wait, there's more." He reasoned about what the numbers could be for each student. He matched 32 with one, i.e., the student with one friend has to be a friend of a student with 32 friends. Then he matched 31 with two, meaning the student with two friends has to be a friend of a student with 32 friends and a student with 31 friends. Continuing this argument, he matched 30 with three, 29 with four, $\cdots$, 18 with 15, and finally 17 with 16. He said, "We have an even number of students accounted for, but an odd number of total students. So that means there are two students with the same number of friends."

Rishnak moved on to the next question. "If I asked five students in a class of six students how many friends each of them have and they all gave distinct numbers greater than zero, how many friends does the student who was not asked have?"

Ajur thought about this. "Okay, if all five students had distinct numbers greater than zero, they had to be $\{1, 2, 3, 4, 5\}$. Let's call the students A(pu), B(art), C(arla), D(uma), and E(rnie), with five, four, three, two, and one friend, respectively." As he spoke, Ajur drew a new diagram in the dirt. "Let F(ermat) be the sixth student. A is friends with B, C, D, E, and F. B is friends with C, D, and F. B is already friends with A. And E has only one friend, namely A. Now C is friends with F. C is already friends with A and B. And D and E have their friends quota counted already. So it is easy to see from the graph that F has exactly three friends, namely A, B, and C."

Ajur smiled and waved his hand majestically at the graph he drew [Figure 3.2].

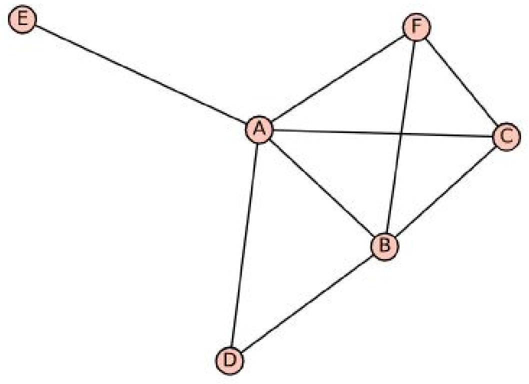


Figure 3.2: A graph with six students, with five students having 5, 4, 3, 2, and 1 friend

Rishnak was impressed, but wanted to test Ajur even further. He asked whether there can be an odd number of students with a grand total of an odd number of friends. Ajur said that this was impossible as the sum of all numbers of friends across all students has to be even. He said, "Look at it this way. If A is a friend of B, then this friendship is counted twice since B is also a friend of A. Hence the sum of all friends is even. We know that students with an even number of friends will contribute an even number of friendships (twice as many). And this in turn implies that the number of students with an odd number of friends also has to be even since doubling an odd number always gives you an even number."

Rishnak smiled, then said, "Good. Next, can you draw a graph of a class with five students having one, one, two, two, and two friends?"

Ajur knelt and whipped up the graph in the dirt [Figure 3.3].

Rishnak asked, "Can you draw a different graph with the same number of friend relationships, meaning the same degree sequence ?"[2]

Ajur took no time to respond and drew another graph [Figure 3.4]. Jura had long ago fallen asleep but now stirred. He barked, trying to tell Ajur that he wanted him to play. Rishnak was happy to hear the answers Ajur gave. He saw a ray of hope that his curse could maybe finally be lifted.

[2]Hakimi has given a method of constructing a graph with a given degree sequence

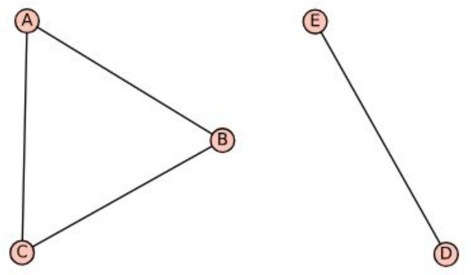


Figure 3.3: A graph with five students having 1, 1, 2, 2, and 2 friends

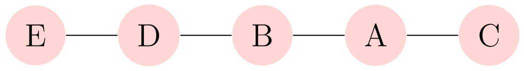


Figure 3.4: Another graph with five students having 1, 1, 2, 2, and 2 friends

Question for the first day

Rishnak stood tall in front of Ajur. "Okay, here is the question for the first night. It has two parts. Answer correctly and we'll continue again tomorrow. First, can you provide a degree sequence [3] that is the degree sequence of exactly one graph? And ignore the labels on the vertices. They do not matter."

Ajur started to answer, but Rishnak raised his hand, silencing him. "Wait, here is part two of the question. Is there also a degree sequence with exactly one graph that realizes it that has at least one edge?"

Ajur frowned. "Let me think."

Before you turn the page, try to come up with answers of your own!

[3] A degree sequence is a collection of the degrees of all the vertices.

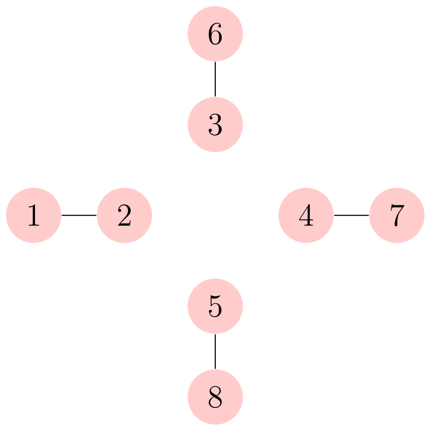


Figure 3.5: A unique graph with degree sequence $\{1, 1, 1, 1, 1, 1, 1, 1\}$

Answer for the first day

Ajur said, "For the first question, that's easy. A degree sequence of all zeros can have only one graphical realization."

Rishnak nodded. "And to the second question? A degree sequence with exactly one graph that realizes it that has at least one edge?"

Ajur thought for a moment, then responded, "Yes, for a sequence of even length, a degree sequence of all ones would always be realized by a unique graph." He used his stick to draw an example graph with a degree sequence of eight ones [Figure 3.5].

Rishnak told Ajur to come back the next night. Ajur stood agape as Rishnak disappeared into a cold mist.

Chapter 4

Trees and Rooted Trees

Rishnak found Ajur and his dog Jura walking along a row of graves. Ajur was reading the inscriptions on the tombstones. Each headstone had the names of a family (parents, wives, husbands) and the dates of birth and of death. Ajur did not know so many relatives could be buried in such a small area. He then thought of how different cultures honor their departed ones.[1] As Ajur's mind wandered through these family trees, he remembered the definitions of a tree and a rooted tree. He was talking to Jura, saying that a rooted tree in graph theory looks like a normal tree with one distinguished vertex. A rooted tree in real life has its root at the bottom, whereas a rooted tree in graph theory is drawn with a root at its top. Both convey the same information. [Figure 4.1][Figure 4.2]

"Here are two drawings of the same information." Ajur sketched in the dirt as Jura watched, his tail wagging fiercely. Ajur explained that each vertex in a rooted tree has just one parent vertex, except for the root vertex, which has no parent. A rooted tree can also be thought of as a graph. There are some restrictions, but they will become clearer as the story proceeds.

Rishnak caught up with Ajur and Jura as he had been following them quietly. Rishnak asked Ajur, "How many edges does a rooted tree with seven vertices have?"

Ajur reasoned that since each vertex other than the root has exactly one parent vertex and there are no other edges, the number of edges will be six.

[1]He remembered seeing the movie Coco in 2017 about how people in Mexico remember their departed ones. He had also heard how Hindus in India go to the city of Varanasi (Banaras) to perform rituals to thank and honor their deceased forefathers.

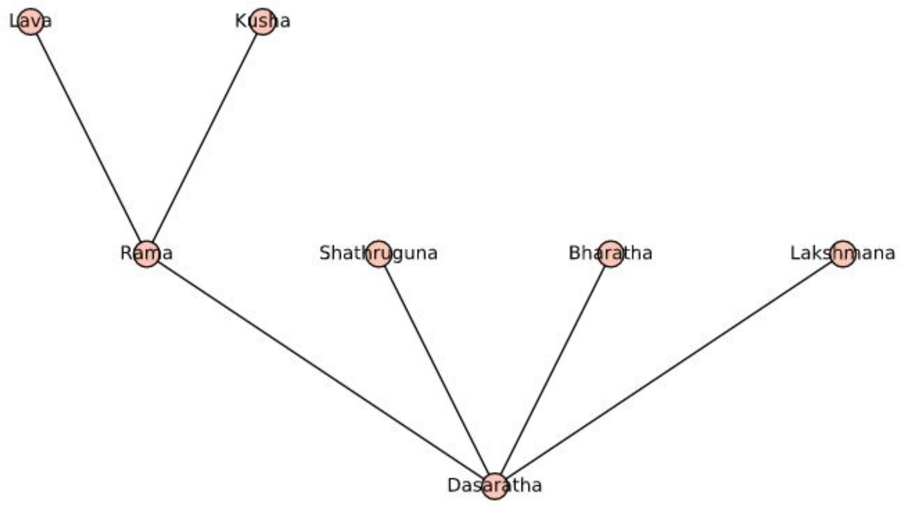


Figure 4.1: A tree drawn with its root at the bottom

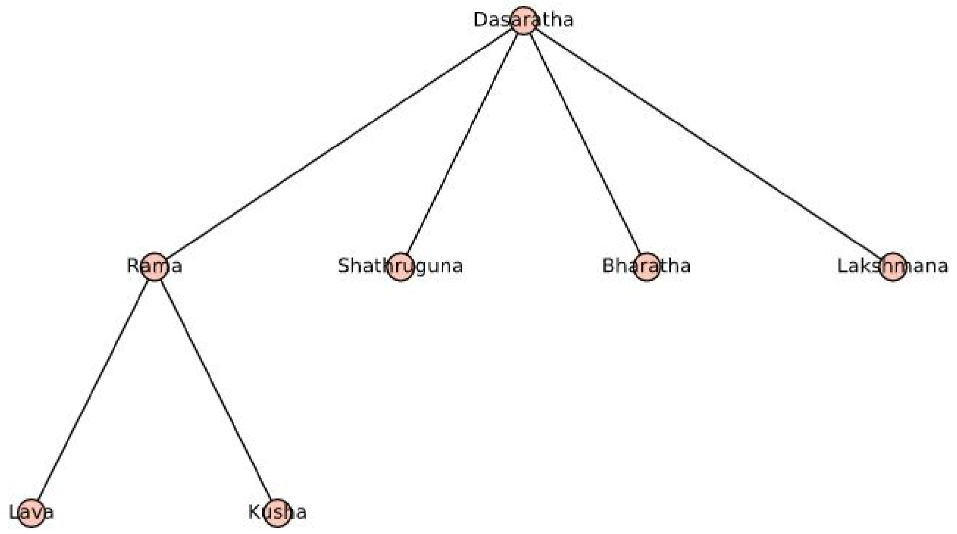


Figure 4.2: The same tree drawn with its root at the top

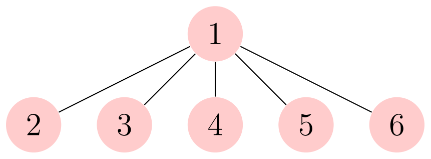


Figure 4.3: A tree with six vertices and five leaf vertices; the vertex labeled 1 is the root vertex

Rishnak then asked how many edges a rooted tree with 1000 vertices has.

Without blinking, Ajur shot back with his answer. “Exactly 999.”

Impressed, but not unduly impressed, Rishnak asked how many edges there are in a rooted tree with n vertices.

Nonchalantly, Ajur replied, “It is $n - 1$ by the same argument. I mean each of the $n - 1$ vertices has just one edge connected to its parent. For each vertex other than the root vertex, there is a parent vertex and zero or more child vertices. The descendants of a vertex are all of its children, grandchildren, great grandchildren, and so on. Analogously, for each vertex, the ancestors of that vertex are its parent, grandparent, great grandparent, and so on.”

Ajur looked up at the trees around him. “A vertex with no child vertices is known as a leaf vertex .”

Rishnak asked Ajur, “What is the largest number of leaf vertices a tree with six vertices can have?”

Ajur immediately responded that the number is five. He drew a rooted tree in the dirt [Figure 4.3].

Rishnak asked, “What is the smallest number of leaf vertices a tree with six vertices can have?”

Ajur knew this answer too. He drew another rooted tree [Figure 4.4].

Rishnak said, “We get a lot of lightning and thunderstorms here, especially during the summer months. Lightning affects tall objects in an open area, especially objects that conduct electricity.[2] Lightning conductors are usually at the top of buildings and have less resistance than the building. Therefore lightning passes through the conductor.”

[2]Benjamin Franklin had demonstrated the electrical nature of lightning.

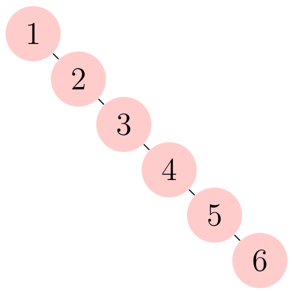


Figure 4.4: A tree with six vertices and one leaf vertex labeled 6; the vertex labeled 1 is the root vertex

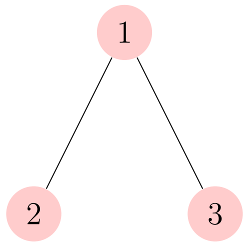


Figure 4.5: A resistance tree with three vertices and two leaf vertices as ground

In a dazzling light display, Rishnak drew a rooted tree with three vertices [Figure 4.5]. Rishnak said that each edge has a resistance of 1 ohm, and vertices labeled 2 and 3 are grounded. “So Ajur, what is the effective resistance of this resistance tree?”

Ajur remembered his physics and realized the two resistances were in parallel. He said, “The effective resistance is $\frac{1}{2}$ ohms.”

Rishnak asked, “And how do you know that?”

Ajur said, “Well, intuitively, there are two paths the current can take to reach ground and so the resistance splits evenly between the two paths.”

Rishnak waved his hands through the air and drew another graph that dazzled in front of Ajur’s eyes [Figure 4.6]. “What is the effective resistance for the following rooted tree?”

Ajur said that he had previously computed the effective resistances for the two “subtrees” rooted at the vertices labeled 2 and 3 to be $\frac{1}{2}$. From

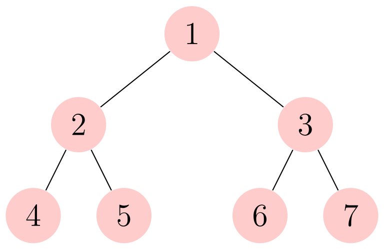


Figure 4.6: A resistance tree with seven vertices and four leaf vertices as ground

the root, there are two parallel paths with equal resistance of $1 + \frac{1}{2}$ (there is a series connection from root vertex 1 to the vertices labeled 2 and 3). Therefore, the effective resistance is half that, or $\frac{3}{4}$ ohms.

Ajur continued, "There is a pattern here. For the resistance tree with 15 vertices, so adding one more level or increasing the height of this tree, the effective resistance will be $\frac{7}{8}$."

Rishnak then asked what happens if the tree is of infinite height, a really tall lightning conductor!

Ajur was perplexed. But he reasoned that this could be formulated as a recurrence relation. "Okay," he said, "let R be the effective resistance of this infinite tree . The root has two children, each of which will have a resistance of R ohms. The root vertex is connected to the child vertex with a resistance of 1 ohm. Here's the equation." He sketched the following equation in the dirt:

$$R = \frac{R+1}{2}$$

Simplifying this, he concluded that the effective resistance was 1 ohm. "A tree of infinite height, with its very low resistance, is like a really tall and effective lightning conductor."[3]

[3]Ajur was right, but because of the large number of joints, this solution may not be a practical one!

Rishnak stated that there was another way of getting the same result. Smiling, he said, “From an earlier example, you can generalize the resistance to be $\frac{2^h-1}{2^h}$, where h is the height of the rooted tree. The height here is the longest of path lengths from the root vertex to all of its leaf vertices. What can that equation be simplified to?”

Ajur thought for a moment, then said, “It can be further simplified to $1 - \frac{1}{2^h}$.”

Rishnak smiled again. “And as h goes to infinity, the $\frac{1}{2^h}$ term goes to zero and hence the resistance is simply 1.”

Ajur stood in awe. He learned the important lesson that there are often multiple ways of finding a solution, and each approach may provide a new insight.

Ajur asked Rishnak, “Can you construct a tree with infinite height (so with an edge resistance of 1 ohm) so that the effective resistance is $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, or more generally any fraction less than or equal to 1?”

Rishnak frowned. “I’m the one asking the questions.”[4]

Rishnak cleared his throat and continued. “A tree is like a rooted tree but with no root vertex. A tree can be drawn in any manner. The degree of a vertex is the number of edges incident on that vertex. The leaf or pendant vertex has a degree of 1. So in this tree [Figure 4.4], both vertices labeled 1 and 6 are leaf or pendant vertices, while all other vertices have degree 2.”

Rishnak told Ajur that one important property of a tree is that there are no cycles in it. Ajur could easily understand this from a genealogy perspective—a person cannot be an ancestor as well as a descendant of himself or herself. Ajur thought further and said, “There is only one path between any two vertices in a tree.”

Of course Ajur assumed that the edges were undirected, which Rishnak knew. Rishnak asked, “How did you infer that there is a unique path between two vertices in a tree?”

Ajur promptly replied that if there are two paths between any two vertices, there will be a cycle, which cannot exist in a tree. Since there is a unique path between two vertices, one can compute the distance between two vertices as the number of edges in that path.

Ajur provided clarifications with some examples. As one example, in this first tree that I drew [Figure 4.3], the distance between the vertex

[4]Of course Rishnak knew the answer and suggests that you answer this by thinking of every vertex with more than two child vertices.

labeled 1 and any other vertex is 1. And the distance between the vertex labeled 2 and the vertex labeled 6 is 2."

Rishnak nodded as Ajur continued. "In this other tree [Figure 4.4], the distance between the vertex labeled 1 and the vertex labeled 6 is 5, while the distance between the vertex labeled 2 and the vertex labeled 5 is 3, and the distance between the vertex labeled—"

"Okay, I got it," boomed Rishnak.

Question for the second day

Rishnak stood tall. "Ajur, here is the question for the second night. Can you construct an infinite tree with an effective resistance of $\frac{3}{5}$?"

Before you turn the page, try to come up with an answer of your own!

Answer for the second day

Ajur repeated the question to himself, wondering how to solve it. “An infinite tree with an effective resistance of $\frac{3}{5}$?”

Rishnak said, “Well?”

Ajur said, “A way to think about this is to consider an infinite tree in which each vertex has six children and each edge is of resistance 3 ohms. We can add a series of three 1 ohm edges to get to 3 ohms.”

Rishnak said, “Are you sure that will work?”

Ajur said, “Yes, with this tree, we will get this recurrence equation.” He quickly sketched out the recurrence in the dirt:

$$R = \frac{R+3}{6}$$

Here, each child vertex will have a resistance of R ohms (by symmetry, as they look like the original tree). Simplifying this, we get $\frac{5R}{6} = \frac{3}{6}$, which if we solve for R gets us to a resistance of $R = \frac{3}{5}$ ohms.”

Rishnak was very pleased. Jura barked and they called it a night.

Chapter 5

Subgraphs

Rishnak wandered the cemetery, looking for Ajur. As he searched, he saw a headstone with the name Schossow. Rishnak recognized the name from the "Instant Insanity" puzzle[1], and just as Rishnak was thinking it would be an interesting topic to discuss with Ajur, Jura the dog, eager to explore, nudged Ajur awake from a nap.

Before long, Ajur and Jura were strolling along the path when Rishnak startled Ajur (as ghosts tend to do).

Rishnak asked Ajur what he knew about subgraphs. Ajur said that he was familiar with subsets. "And since a graph has both a vertex set and an edge set, I think I can deduce what a subgraph is. Given some graph $G = (V, E)$ with vertex set V and edge set E, then take any subset X of V and consider all edges in E for which both end vertices are in X."

Ajur picked up a stick and drew a graph in the dirt [Figure 5.1].

He said, "From this first graph, I could define a separate vertex subset $V' = \{1, 2, 3, 5\}$ to form another graph, say $G' = (V', E')$, which is a subgraph of the first."

He drew a second graph [Figure 5.2].

Rishnak laughed and said that the subgraph Ajur drew was called an *induced subgraph* . He said, "It is called that because all of the edges are included in the vertex subset. You do have the flexibility of choosing only a subset of these edges, but there is one condition: for each edge in the chosen

[1]This is also known as Katzenjammer, (Great) Tantalizer, Face-4, Cube-4, Bognar Balls, Taktikolor, Frantic, Diabolical, Damblocks, and Symington's Puzzle. A patent was awarded to Schossow in 1990.

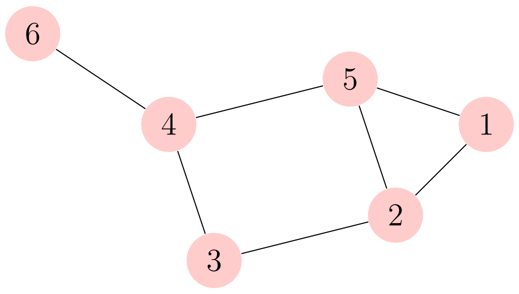


Figure 5.1: Example graph with six vertices and seven edges

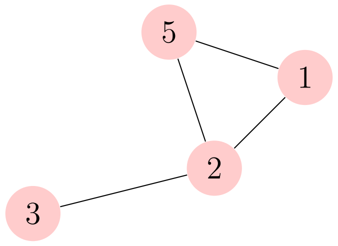


Figure 5.2: Induced subgraph of the graph shown in Figure 5.1, with vertices in set $V' = \{1, 2, 3, 5\}$ and all edges between these vertices present from the original graph

subset of edges, the end vertices must be in the chosen vertex subset. Let me show you."

Rishnak drew a graph in the air in a dazzling light display [Figure 5.3]. "Here is a subgraph of your original graph. And note that a subgraph with no vertices and no edges is also an induced subgraph (and subgraph) of any graph."

Ajur nodded his head and yawned. Rishnak scowled and said, "Time to learn something new, Ajur, then see if you can still answer my questions."

Ajur straightened as Rishnak continued. "A walk from a vertex i to a vertex j is an alternating sequence of vertices and edges. Every edge in that walk is incident between vertices preceding and succeeding that edge. For example, in that graph that you drew"—he pointed down to the dirt [Figure 5.1]—"a walk from vertex 6 to vertex 1 could be $6-(6,4)-4-(4,3)-3-(3,2)-2-(2,1)-1$. The edges are represented as a vertex pair. And if

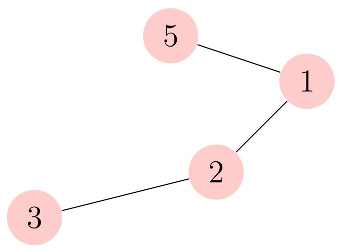


Figure 5.3: A subgraph of Figure 5.1

those edges have names as labels, we could use that label instead. Another walk from vertex 6 to vertex 1 could be $6 - (6,4) - 4 - (4,5) - 5 - (5,1) - 1$."

Ajur listened intently, then asked, "Can a walk use the same edge more than once?"

Rishnak said, "No. The only other condition that a walk has (besides an edge being incident on a preceding and a succeeding vertex) is that all edges must be distinct. The vertices in a walk can be repeated, though. Have a look at this graph." In a flash of light, Rishnak drew another graph [Figure 5.4]. "A walk from vertex 1 to vertex 8 could be $1 - (1,2) - 2 - (2,4) - 4 - (4,6) - 6 - (6,8) - 8 - (8,2) - 2 - (2,3) - 3 - (3,4) - 4 - (4,5) - 5 - (5,6) - 6 - (6,7) - 7 - (7,8) - 8$."

Ajur tried to pay attention but was naturally getting bored. He interjected, "In your walk, you have visited all the edges in the graph, much like the Königsberg Bridge Problem."[2]

Rishnak smiled and nodded. "And if the starting and ending vertices in a walk are the same, it is known as a closed walk. If all of the edges in a closed walk are distinct, then it is known as a cycle."

Ajur remembered the idea of a cycle from yesterday's discussion of trees.

Rishnak asked Ajur to list two cycles from the graph that sparkled in front of him [Figure 5.4].

Ajur had no trouble at all in listing two cycles as $1 - (1,2) - 2 - (2,8) - 8 - (8,1) - 1$ and $2 - (2,4) - 4 - (4,6) - 6 - (6,8) - 8 - (8,2) - 2$. Rishnak taught Ajur that often the edges are omitted when describing a walk, so the cycles could be written simply as $(1,2,8,1)$ and $(2,4,6,8,2)$. And even simpler, we can state that a walk is a cycle and omit the repetitive last vertex, so we have cycles $(1,2,8)$ and $(2,4,6,8)$.

[2]As mentioned earlier in Chapter 3 [Figure 3.1].

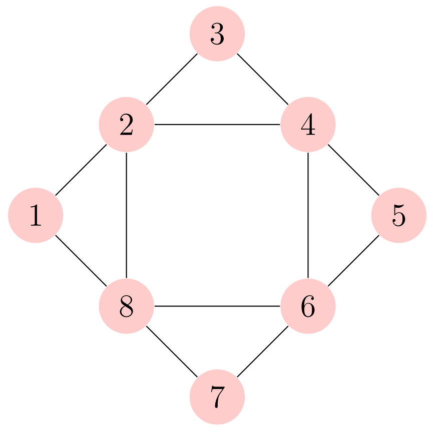


Figure 5.4: Example graph with eight vertices and 12 edges

Rishnak beamed as he went on. “Of course a cycle and a walk are examples of subgraphs with some added conditions. These added conditions make the study of subgraphs very interesting. If in a cycle, all the vertices of the original graph are present, then that cycle is known as a Hamiltonian Cycle . For example, in this graph”—he again referred to the graph that shone in front of him [Figure 5.4]—“the cycle $(1, 2, 3, 4, 5, 6, 7, 8, 1)$ is a Hamiltonian Cycle of this other graph”—he whisked his hands through the air to produce another graph [Figure 5.5]—“And if all the edges in a walk are distinct then it is known as a path.”

Ajur tried to keep up. “Okay, so in that first graph”—he pointed to the graph [Figure 5.4]—“an example path is $1-(1,2)-2-(2,3)-3$ or simply $(1, 2, 3)$. Let me draw this path.” He drew the path in the dirt [Figure 5.6].

Rishnak nodded, then continued, “If there is a path between every pair of vertices in a subgraph, then the subgraph is said to be connected. If a subgraph contains no cycles and is connected, the subgraph is a tree. And if such a tree contains all of the vertices then it is known as a spanning tree since it spans all vertices. Watch closely, here’s a spanning tree.”

Rishnak transformed the original graph [Figure 5.4] into a new one with fewer edges [Figure 5.7].

Rishnak continued, “A subgraph in which the degree of every vertex is 1 is said to be a matching. An example of a subgraph that is a matching for this original graph”—he again showed the original graph [Figure 5.4]—“is

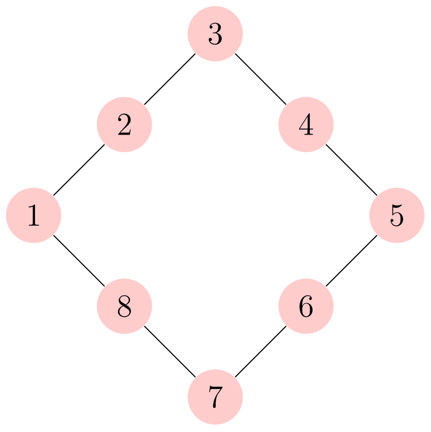


Figure 5.5: A subgraph of Figure 5.4, which forms a Hamiltonian Cycle

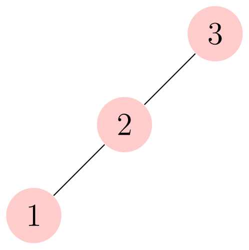


Figure 5.6: A subgraph of Figure 5.4, which forms a tree

this other graph that contains pairs of vertices." Rishnak moved his hands and the graph reduced to one with only three edges [Figure 5.8].

"If the subgraph contains all vertices and the degree of every vertex is 1, then it is called a perfect matching . Here's an example of a subgraph that is a perfect matching." A new graph appeared, this time with all of the original vertices but only four edges [Figure 5.9].

Rishnak asked Ajur how many perfect matchings there were in the original graph [Figure 5.4].

Ajur thought for a bit, his brain catching up with everything Rishnak was showing him. He saw that vertices 1, 3, 5, and 7 have degree 2. Therefore, one of those incident on 1, 3, 5, and 7 would have to be selected. "There are exactly two perfect matchings."

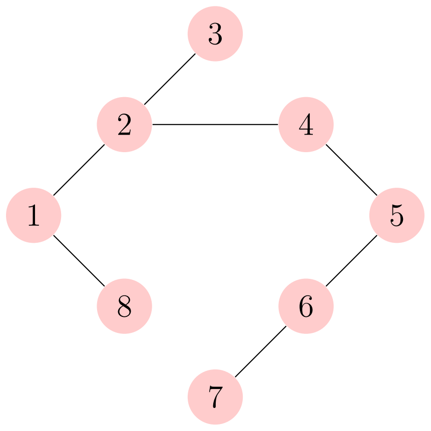


Figure 5.7: A subgraph of Figure 5.4, which forms a spanning tree

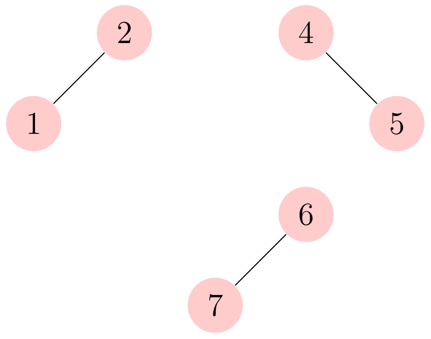


Figure 5.8: A subgraph of Figure 5.4, which forms a matching

Rishnak had more to teach Ajur. He said, “A graph is connected if there is a path between every pair of vertices in that graph. Otherwise the graph is disconnected . Watch closely as here are two graphs, the first being a connected graph [Figure 5.10], the second a graph that is not connected [Figure 5.11].”

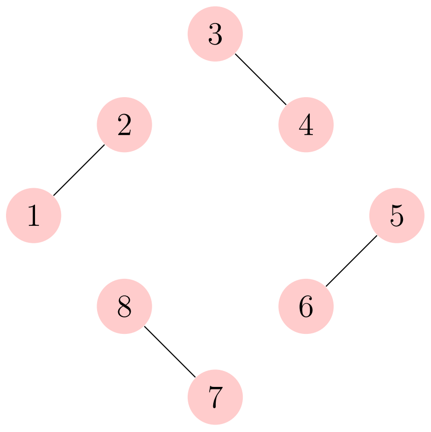


Figure 5.9: A subgraph of Figure 5.4, which forms a perfect matching

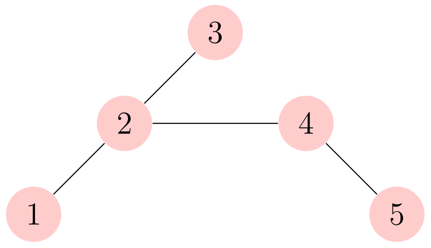


Figure 5.10: A connected graph

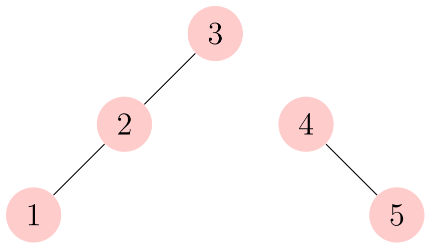


Figure 5.11: A graph that is not connected

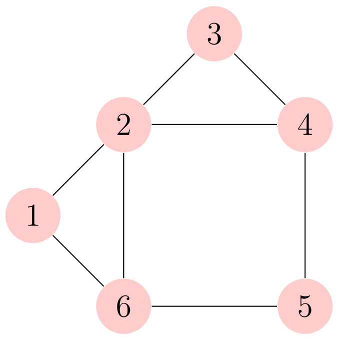


Figure 5.12: Can you list the cycles in this graph?

Question for the third day

Rishnak said, "At last we come to the question for the third day. Can you list the cycles in this graph? And state the length of each cycle?" He splayed his hands and a new graph appeared in front of him [Figure 5.12]

Before you turn the page, try to come up with an answer of your own!

Answer for the third day

Ajur scratched his head and studied the graph that Rishnak showed him. At length, he said, “I think I see six cycles in the graph.” He proceeded to list them.

- Two cycles of length 3: $(1, 2, 6), (2, 3, 4)$
- One cycle of length 4: $(2, 4, 5, 6)$
- Two cycles of length 5: $(2, 3, 4, 5, 6), (1, 2, 4, 5, 6)$
- One cycle of length 6: $(1, 2, 3, 4, 5, 6)$

Rishnak was again pleased as this was the correct answer. He smiled and noticed that Ajur was getting restless, and so was Jura, so they called it a night.

Chapter 6

Eulerian Paths and Cycles

Ajur was still tired after listening to so many definitions and examples the day before. He complained to Jura that their recent conversation with Rishnak was too similar to a boring math class. He said, "I prefer to solve problems that are fun."

Rishnak was near and overheard Ajur. Sighing, Rishnak did agree that the previous day's exchanges were dry. His ghost friends were right—he was making the beautiful subject of graph theory dull and monotonous. So for the next session, Rishnak decided to reach Ajur a more interesting problem, the existence of an Eulerian walk.

Recall that a connected graph is a graph in which there is a path between every possible pair of vertices. Given this, an *Eulerian walk* in a connected graph is a walk that includes every edge exactly once. This is also known as an *Eulerian path* and this path can visit vertices more than once if need be. If the starting vertex and the ending vertex are the same then the walk is called a *closed Eulerian walk* or an *Eulerian cycle*. The problem is named in honor of Leonhard Euler, the first person to describe this (in the 1700s!).

The question of whether or not a given connected graph has an Eulerian walk (or a closed Eulerian walk) is among the oldest problems in graph theory. "Ajur will love this problem," thought Rishnak as he searched for Ajur in the cemetery.

It did not take long for Rishnak to catch up with Ajur and Jura as they walked along a desolate path in a far corner of the cemetery.

Rishnak flashed his hands to produce a new graph [Figure 6.1], then asked Ajur, "In this graph, is there a walk starting from vertex 2 and ending at vertex 4 that travels through all of the edges *exactly once*?"

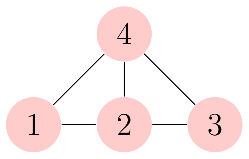


Figure 6.1: Example graph with four vertices and five edges

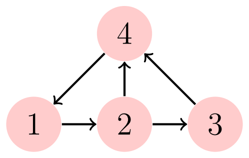


Figure 6.2: Eulerian walk $(2, 3, 4, 1, 2, 4)$ from vertex 2 to vertex 4 of the graph shown in Figure 6.1

Ajur studied the graph, frowning.

Rishnak continued, “If you find one, know that it is called an Eulerian walk. This is just like asking if you can trace all the edges once and only once without lifting your pen—or your stick.”

Ajur noticed that there was a cycle. “I see the cycle $(2, 3, 4, 1)$ and after this cycle, we’re back at vertex 2 with just one edge left, the edge $(2, 4)$. Can we visit a vertex more than once?”

Rishnak nodded.

Jumping up, Ajur grabbed a stick and drew a graph in the dirt [Figure 6.2], using arrows to show how he traversed the edges with vertex 2 as the starting point. “There’s an Eulerian walk by combining the cycle and that last remaining edge. The path is $2 - (2, 3) - 3 - (3, 4) - 4 - (4, 1) - 1 - (1, 2) - 2 - (2, 4) - 4$ or just $(2, 3, 4, 1, 2, 4)$ with edges omitted.”

Rishnak nodded again. “This is an Eulerian walk. If we were able to start and end on the same vertex, it would be a closed Eulerian walk.”

Ajur smiled. “I see. So there is no closed Eulerian walk in this graph because that would necessarily imply that every vertex had even degree!”

Rishnak also smiled. “That’s correct, Ajur.”

In a flash of light, Rishnak showed Ajur another graph [Figure 6.3] and said, “Here’s a more challenging problem for you. In this graph, is there a

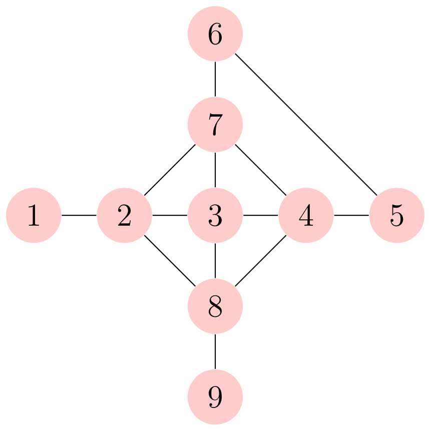


Figure 6.3: Example graph with nine vertices and 13 edges

walk from vertex 1 that ends at vertex 9 and traverses every edge exactly once? In other words, is there an Eulerian walk from vertex 1 to vertex 9?"

Ajur stood perplexed. He tried to reason out an approach he could take, saying out loud, "Not all of the vertices have an even degree, so that means there is no closed Eulerian walk." He studied the graph further. "But seven of the nine vertices *do* have an even degree. Vertices 1 and 9 both have an odd degree of 1, so they would definitely be the start and end vertices."

Rishnak was pleased.

Ajur continued, "Let me look for cycles. I see cycles $(2, 3, 7)$, $(7, 6, 5, 4)$, and $(3, 4, 8)$, none of which have any edge in common. Aha, I think I got it. I can combine the cycles."

Ajur excitedly drew a graph in the dirt [Figure 6.4], again using arrows to show the path to follow. "The Eulerian walk is then $1 - (1, 2) - 2 - (2, 3) - 3 - (3, 4) - 4 - (4, 8) - 8 - (8, 3) - 3 - (3, 7) - 7 - (7, 6) - 6 - (6, 5) - 5 - (5, 4) - 4 - (4, 7) - 7 - (7, 2) - 2 - (2, 8) - 8 - (8, 9) - 9$ or we can just say $(1, 2, 3, 4, 8, 3, 7, 6, 5, 4, 7, 2, 8, 9)$."

Ajur added again that there was no closed Eulerian walk in the graph because not all of the vertices had an even degree. But an Eulerian walk can start from a vertex with an odd degree and then end at a vertex with an odd degree.

Rishnak said, "Exactly right. There are exactly two such vertices with odd degrees, and all other vertices must have even degrees. All other vertices

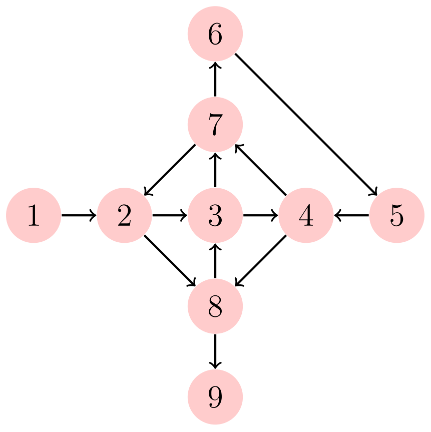


Figure 6.4: Eulerian walk $(1, 2, 3, 4, 8, 3, 7, 6, 5, 4, 7, 2, 8, 9)$ from vertex 1 to vertex 9 of the graph shown in Figure 6.3

become intermediate vertices in the Eulerian walk, meaning that for every edge coming into the vertex there is a corresponding edge leaving that vertex."

Rishnak asked Ajur how he would modify the original graph [Figure 6.3] to have a closed Eulerian walk.

Ajur reasoned this out quickly. He said, "There are exactly two vertices with odd degrees, namely vertices 1 and 9. If we were to add an edge between these two vertices, then every vertex would have an even degree and a closed Eulerian walk would be possible."

He drew the graph in the dirt [Figure 6.5]. "The closed Eulerian walk would be $(1, 2, 3, 4, 8, 3, 7, 6, 5, 4, 7, 2, 8, 9, 1)$."

Rishnak smiled. "Let's go back to this first graph." With a flash of his hands, the graph formed in front of him [Figure 6.1]. "How would you modify this graph to have a closed Eulerian walk?"

Ajur now had a problem. The two vertices with odd degrees—vertices 2 and 4—already had an edge between them. He said, "I don't think we can do it."

Rishnak said, "Have you ever heard of a *multigraph*?"

Ajur shook his head.

Rishnak continued, "In a multigraph, there can be more than one edge between any pair of vertices."

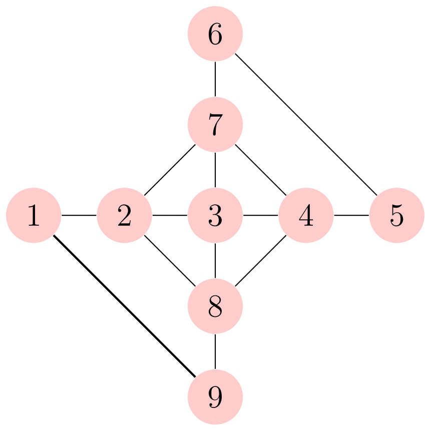


Figure 6.5: The graph from Figure 6.3 with additional edge $(1, 9)$ enabling closed Eulerian walk $(1, 2, 3, 4, 8, 3, 7, 6, 5, 4, 7, 2, 8, 9, 1)$

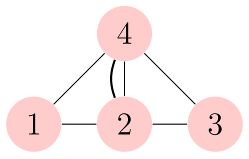


Figure 6.6: The graph from Figure 6.1 with additional edge $(2, 4)$ added to form a multigraph with closed Eulerian walk $(2, 3, 4, 1, 2, 4, 2)$

Ajur pondered this for a moment, then said, "Oh, I can add another edge between vertex 2 and vertex 4 to get a closed Eulerian walk"—he hurriedly drew a new graph [Figure 6.6]—"it's $(2, 3, 4, 1, 2, 4, 2)$."

Rishnak asked Ajur whether he knew about directed graphs.

Ajur nodded and said, "In a directed graph, an edge (x, y) only goes from vertex x to vertex y. I mean it doesn't also go back from vertex y to vertex x." He drew an example graph to show this [Figure 6.7].

Rishnak said, "Right. And instead of talking about the degree of a vertex, we then have an *in-degree* and an *out-degree* of a vertex. The number of edges coming into a vertex is the in-degree of that vertex, while the number of edges going out of a vertex is the out-degree."

Rishnak drew a new graph in a dazzling display of light [Figure 6.7], then said, "In this directed graph, vertex 1 has an in-degree of 2 and an

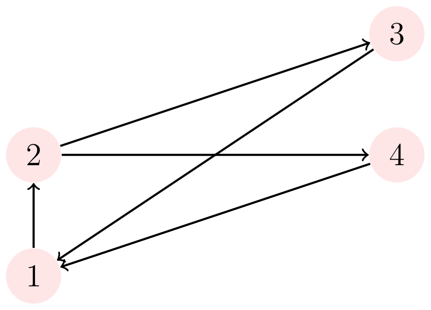


Figure 6.7: Example directed graph with four vertices and five directed edges

out-degree of 1. Vertex 3 has an in-degree and an out-degree 1, same with vertex 4. And a directed graph is said to be *strongly connected* if there is a directed path from every vertex to every other vertex. This example graph is indeed strongly connected."

Ajur marveled at the graph in front of him. He said, "And an Eulerian walk can only exist in a strongly connected directed graph, right?"

Rishnak said, "Precisely."

Anticipating Rishnak's next question, Ajur said that there was no closed Eulerian walk in this directed graph because there was no way to traverse each edge exactly once and also start and end on the same vertex.

Rishnak said, "Is there an Eulerian walk from vertex 2 to vertex 1?"

Ajur thought about this, remembering he could try tracing a path without having to raise his pen (or stick) at any point. He reasoned in a manner similar to what he did before. Ajur said, "I see the cycle $(2, 3, 1, 2)$, so if I start at vertex 2, I can write the Eulerian walk as $(2, 3, 1, 2, 4, 1)$."

Rishnak smiled, then said, "How could we have a closed Eulerian walk?"

Ajur said, "We can't in this graph. To have a closed Eulerian walk, we need a collection of cycles that do not share any edges."

Rishnak said, "Correct. That is called an *edge-disjoint cycle*, a cycle in which no edge is common. In such a case, the in-degree of each vertex must be the same as its out-degree."

Ajur nodded and said, "I see! This is similar to the condition for an undirected graph in which a closed Eulerian walk can only exist if the degree of each vertex is even."

Rishnak nodded. “Keep going.”

Ajur said, “For an Eulerian walk in a directed graph, it has to start from a vertex with an out-degree that is one greater than its in-degree. And it must end at a vertex with an in-degree that is one greater than its out-degree.”

Rishnak said, “Exactly. And for all other vertices, the in-degree must equal the out-degree.”

It was getting late, but Rishnak wanted to share more about how one could use Eulerian walks—and he wanted to keep Ajur interested. “Consider the following problem. Let’s say you want to construct a string of zeros and ones such that all four of the possible two-bit strings occur as a substring. Note that there are exactly four two-bit strings consisting of zeros and ones, namely 00, 01, 10, and 11. As an example, the string 00110 contains all four of these two-bit substrings.”

Ajur nodded, following Rishnak so far anyways.

Rishnak continued, “Suppose instead, we wanted to build a *circular* string that contained all possible two-bit sub-strings. Such a string is known as a De Bruijn sequence. And this problem is actually closely related to constructing a closed Eulerian walk. Have a look at this directed graph”—he flashed a new graph [Figure 6.8] in front of Ajur—“If we construct an Eulerian walk on this directed graph, we would get such a string.”

Ajur frowned. “How would that work?”

Rishnak said, “For this directed graph, there are two vertices with labels 0 and 1. Each directed edge also has a label, which is added to the generated string when the edge is traversed. And the endpoint of that directed edge is the vertex with the same label.”

Ajur understood. “I see. But why is there a directed edge with label 0 from vertex 0 to itself that—”

Rishnak chimed in, “That is called a *self loop*.”

Ajur continued, “A self loop, okay. If we start at vertex 0, then the idea behind this self loop is to append the edge label 0 to the generated string. And we end up still at vertex 0. Similarly, there is an edge with label 1 from vertex 1 to itself. And there is an edge with label 1 from vertex 0 to endpoint vertex 1, and vice versa.”

Rishnak nodded and said, “Yes. Notice that each vertex has an in-degree of 2 and an out-degree of 2. Therefore, we know that this directed graph has a closed Eulerian walk from vertex 0 back to vertex 0—and that walk is 0110. Remember that each character comes from an edge label.”

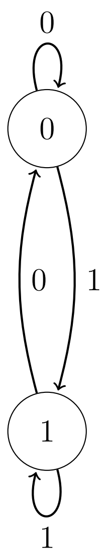


Figure 6.8: An Eulerian walk on this directed graph will yield a De Bruijn sequence, which contains all sub-strings of length 2 over binary alphabet $\{0, 1\}$

Ajur smiled and said, "And from this, we get all four substrings of length 2, namely 01, 11, 10, and 00, with that last one generated by taking the last character and the first character since it is a closed walk."

Rishnak then asked Ajur to construct a De Bruijn sequence that contained all substrings of length 3.

Ajur thought about this and rephrased the question. "Do you mean to obtain a string of zeros and ones such that all eight three-bit strings occur as a substring?"

Rishnak nodded and said, "Right, what would the eight three-bit strings be?"

Ajur knew the answer to be 000, 001, 010, 011, 100, 101, 110, and 111 by counting in binary (base 2). Ajur continued, "Okay, I want to construct a graph from which an Eulerian walk would naturally yield such a sequence, right?"

Rishnak nodded.

Ajur worked out the graph in the dirt using his stick, looking for a pattern to follow. He started with four vertices labeled 00, 01, 10, and 11.

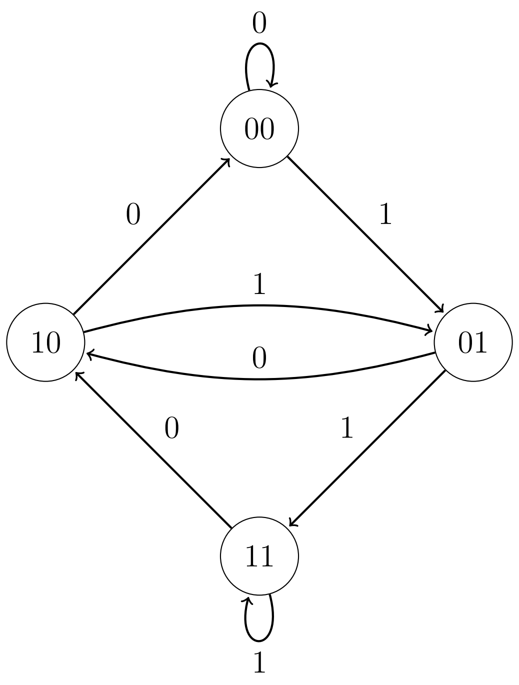


Figure 6.9: A closed Eulerian walk on this directed graph will yield a De Bruijn sequence that contains all substrings of length 3 over binary alphabet $\{0, 1\}$

He drew a directed edge from vertex 00 to itself with label 0 because if you get a 0, you can append 0 to the vertex label 00, then drop the first character. There is also an edge with label 1 from vertex 00 to vertex 01.

"Aha!" exclaimed Ajur. "You said that every vertex must have an out-degree of 2 and an in-degree of 2." Ajur drew the rest of the directed graph [Figure 6.9] with swift strokes of his stick. "Starting from vertex 00, an Eulerian walk could generate $0 - 1 - 0 - 1 - 1 - 1 - 0 - 0$ from the edge labels. This walk is also a closed Eulerian walk since we start and end with the same vertex."

Rishnak tilted his head and said, "And?"

Ajur continued, "And from the walk, we end up with all of the strings of length 3 over alphabet $\{0, 1\}$ as substrings of the Eulerian walk."

Rishnak said, "Good."

Rishnak stretched his arms out wide. He asked, “Are you sharp enough for something else brand new?”

Ajur smiled and said, “Yes.”

Rishnak said that another application of the Eulerian walk is in the creation of mazes or labyrinths (as known in Greece) or bhul bulaiyah (as known in India). He said, “It is quite easy to enter a maze, but it is difficult to get out of one. In so many stories of old, there are reports of people dying in mazes because they were unable to find their way out. One was the angel Kinaja, who was then a human trapped in a maze. She had figured out how to get out but was too exhausted to walk—and so she died.”

Ajur felt a wave of sadness.

Rishnak quickly said, “Do not despair, for Kinaja told me that those who studied graph theory and understood Eulerian walks could find their way out of any maze.”

Ajur said that he, too, knew about mazes, having seen a video of a psychology lab experiment with mice running through an elaborate maze. He had also been in a maze as a young boy when his parents took him to a corn field maze.

Eager to show off his knowledge of mazes and poetry, Ajur quoted Robert Frost’s famous poem *The Road not Taken*, which could also be related to traversing a maze—Ajur recited:

> Two roads diverged in a wood, and I—
> I took the one less traveled by,
> And that has made all the difference.

Ajur took a bow.

But Rishnak was getting impatient, as ghosts tend to do, and he wanted to stay focused on the maze and on graphs. Rishnak waved his hands and a glimmering picture of a maze appeared [Figure 6.10].

Rishnak said, “You can construct a graph from this maze by placing a vertex at each place where there is an open space to move or where you have a choice to make, meaning multiple paths you could follow.”

Ajur gazed at the maze in front of him.

Rishnak continued, “Join vertices together with an edge if there is a corridor connecting them and there are no vertices already between them. In a maze without any loops, this will result in a tree since no cycles will be present.”

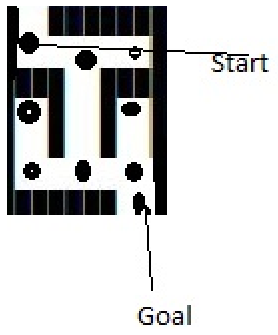


Figure 6.10: A simple maze with a start and a goal, with intermediate steps (vertices) marked for clarity

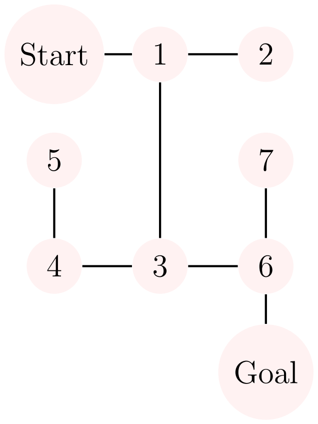


Figure 6.11: A graph (tree actually) corresponding to the maze shown in Figure 6.10

Ajur understood. He drew a graph [Figure 6.11] in the dirt that matched the maze Rishnak showed him. "But how does a Eulerian walk make any sense here?"

Rishnak smiled, happy to see Ajur's desire to learn. Rishnak said, "You can make this graph Eulerian by traversing each edge twice. Each time you traverse an edge, you leave a bread crumb to mark it. Once you reach a dead end, you turn back. And if an edge has two bread crumbs, that means

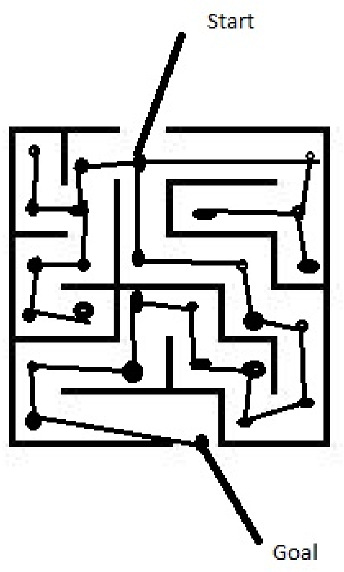


Figure 6.12: A maze with a start and a goal, with the corresponding graph representation drawn inside the maze

you do not traverse that edge any more."

Rishnak's smile broadened as he continued, "This was precisely the strategy Kinaja described to me for how to get out of a maze. In the beginning, all of the paths are free of bread crumbs. Whenever one takes a path (if there is one available to take that has no bread crumbs), you mark the edge with a bread crumb. And if there are no paths available without any bread crumbs, you take the path in which there is only one bread crumb present, placing bread crumbs as you go. Eventually you will reach the end goal as there is an Eulerian walk present in the underlying graph of the maze."

Ajur said, "Let me see if I understand." He drew another maze in the dirt, then drew the graph *inside the maze* [Figure 6.12]. "Like this?"

Rishnak beamed. "Precisely."

Question for the fourth day

Rishnak said, “The time has come for me to ask you the question for the fourth day.”

Ajur straightened, ready for his question.

Rishnak said, “It has two parts. Going back to this earlier graph”—he flashed the De Bruijn sequence graph for substrings of length 3 from earlier [Figure 6.9]—“first can you write a string (different than the one we already came up with) that contains all substrings of length 3 over binary alphabet $\{0, 1\}$?”

Ajur nodded. “And?”

Rishnak said, “Second, can you construct a directed graph that will yield a De Bruijn sequence that contains all substrings of length 4 over the same alphabet?”

Before you turn the page, try to come up with an answer of your own!

$$\begin{tikzcd}
{} & 001 \arrow[rrr, "1"] \arrow[dr, "0"] & {} & {} & 011 \arrow[dr, "1"] \arrow[dd, "0"] & {} \\
000 \arrow[loop left, "0"] \arrow[ur, "1"] & {} & 010 \arrow[r, "1"] \arrow[dl, "0"] & 101 \arrow[l, bend left, "0"] \arrow[ur, "1"] & {} & 111 \arrow[loop right, "1"] \arrow[dl, "0"] \\
{} & 100 \arrow[uu, "1"] \arrow[ul, "0"] & {} & {} & 110 \arrow[lll, "0"] \arrow[ul, "1"] & {}
\end{tikzcd}$$

Figure 6.13: A graph to generate a De Bruijn sequence of length 4

Answer for the fourth day

For length 3, Ajur used the directed graph in front of him [Figure 6.9] and wrote a closed Eulerian walk as $0-0-0-1-0-1-1-1$. "This one contains all substrings of length 3. Going left to right, they are 000, 001, 010, 101, 011, 111, 110, and 100."

Rishnak smiled and said, "Keep going."

For length 4, Ajur cleared a wide patch in the dirt and drew a new graph [Figure 6.13]. Once he was done, he checked to be sure that the graph was Eulerian by verifying that the in-degree and out-degree of each vertex was 2.

"Here is the graph and from this graph"—he traced with his stick—"the sequence is $0-0-0-0-1-0-1-0-0-1-1-0-1-1-1-1$ and it contains all the substrings of length 4, which are 0000, 0001, 0010, 0101, 1010, 0100, 1001, 0011, 0110, 1101, 1011, 0111, 1111, 1110, 1100, and 1000."

Rishnak was happy with Ajur's answers. He said, "Well done."

Ajur smiled to himself. He was impressed by the power of the Eulerian walk and wanted a walk to be named after him someday, too! The sun was setting and it was time for Ajur and Jura to go home.

Chapter 7

Hamiltonian Paths and Cycles

Ajur walked with Jura, thinking what he could do to inspire a problem to be named after him. Rishnak found Ajur to be an interesting individual to discuss puzzles based on graph theory with. After the discussion of the Eulerian walk, Rishnak decided to introduce a closely related graph-theoretic construction, the concepts of Hamiltonian paths and Hamiltonian cycles.

Rishnak appeared in front of Ajur and immediately began defining the notion of a Hamiltonian path. "In what's called a Hamiltonian path, each of the vertices of the given graph are visited exactly once. The length of such a path is the number of edges in that path. So, Ajur, what is the length of a Hamiltonian path in a graph with n vertices?"

Ajur said, "That's easy. A Hamiltonian path in a graph of n vertices will have a path length of $n - 1$."

Rishank said, "Right. And a Hamiltonian cycle is a Hamiltonian path that forms a cycle. Therefore, the length of a Hamiltonian cycle in a graph with n vertices is n. Let's start with this graph." Rishnak presented a graph [Figure 7.1] in the air in front of Ajur. "Tell me, Ajur, is there a Hamiltonian cycle in this graph,[1] and if so, what edges form the cycle?"

Ajur thought about Rishnak's question for a few seconds, then picked up a stick and drew the graph and the Hamiltonian cycle in the dirt [Figure 7.2].

Rishnak told Ajur that his solution was correct but not unique. "There are more solutions than just that one.[2] And do you know why a Hamiltonian

[1]This graph is a cube graph of length 3.

[2]Can you find a few more Hamiltonian cycles that are different from the one described by Ajur?

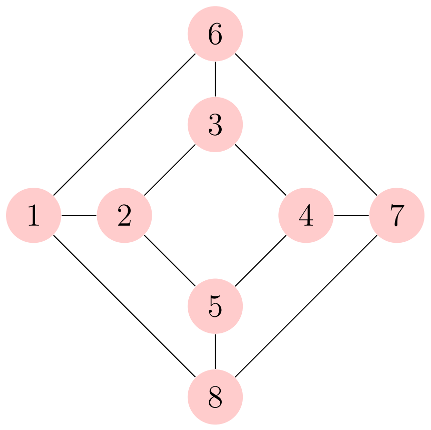


Figure 7.1: Cube graph (of length 3) with eight vertices and 12 edges

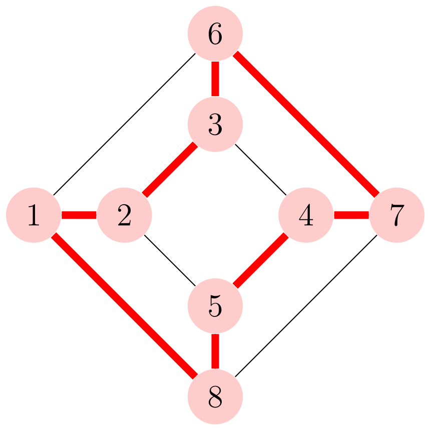


Figure 7.2: Cube graph of Figure 7.1 with a Hamiltonian cycle marked in thick edges

cycle is called a Hamiltonian cycle?"

Ajur frowned. "No, but I suppose you'll tell me?"

Rishnak laughed and said, "The mathematician Sir William Rowan Hamilton wanted to find a cycle to visit all of the vertices of a dodecahedron, which is a three-dimensional structure with 20 vertices and 30 edges. It's also one of the five platonic solids."

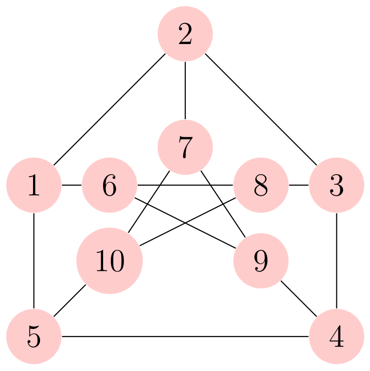


Figure 7.3: The Petersen graph with 10 vertices and 15 edges

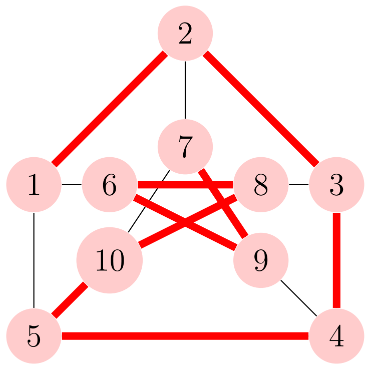


Figure 7.4: The Petersen graph of Figure 7.3 with a Hamiltonian path marked in thick edges

Rishnak flashed his hands and a new graph [Figure 7.3] appeared in dazzling lights in front of Ajur. Rishnak said, “Is there a Hamiltonian cycle in this graph, which by the way is a well-known graph called the Petersen graph, named after mathematician Julius Petersen.”

Ajur studied this graph for a long time, but he was not able to find a Hamiltonian cycle in the graph. He sighed and said, “I don’t see a cycle, though I do see a Hamiltonian path.”

Ajur drew the graph and a Hamiltonian path in the dirt [Figure 7.4].

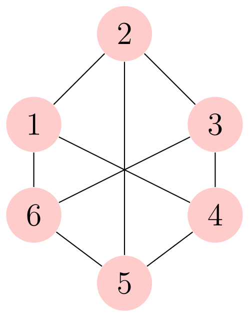


Figure 7.5: A bipartite graph with six vertices and nine edges, comprising vertex partition sets $A = (1, 3, 5)$ and $B = (2, 4, 6)$

He said, "And that's not the only Hamiltonian path in this graph, is it?[3] There are a few others."

Rishnak smiled and assured Ajur that the Petersen graph indeed does not have a Hamiltonian cycle. He said, "Remember the Eulerian cycle, which traverses every edge exactly once? We can easily test whether a graph has an Eulerian cycle by just testing to see whether the degree of every vertex is even. Unfortunately, there is no easy way to test whether a given graph has a Hamiltonian cycle."

Rishnak continued, "Speaking of cycles, there is a special class of graphs called *bipartite graphs* in which every cycle is of even length. Further, in a bipartite graph, the vertex set is partitioned into two sets A and B such that every edge has one end vertex in A and its other end vertex in B."

Rishnak flashed his hands to form a new graph in the air before Ajur [Figure 7.5].

Ajur immediately saw that all of the cycles had even lengths, in this case 4 and 6. He said, "And every edge in the cycle must go from one partition set to the other. These are the only possible edges. That's why the length of each cycle must be even, right?"

Rishnak nodded and asked Ajur what the two vertex partitions were for this new graph. "All of the edges must go from one partition set to the other."

After a little thought, Ajur said, "One partition contains vertices 1, 3, and 5, while the other partition contains vertices 2, 4, and 6." He drew a

[3]Can you find the other Hamiltonian paths in the Petersen graph?

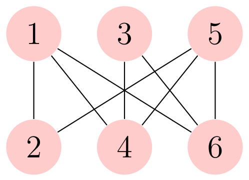


Figure 7.6: The bipartite graph from Figure 7.5 reorganized to emphasize vertex partition sets $A = (1, 3, 5)$ and $B = (2, 4, 6)$

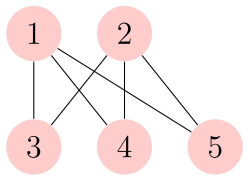


Figure 7.7: A bipartite graph with five vertices and six edges

new version of the graph to illustrate what he meant [Figure 7.6].

Ajur said, “This graph also has a Hamiltonian cycle”—he frowned, deep in thought—“Wait, every tree is a bipartite graph, too, since a tree contains no cycles. And since a tree does not have any cycles, it has no Hamiltonian cycles either.”

Rishnak smiled. “That is true, Ajur. But let’s get back to bipartite graphs. Can you draw a graph that is not a tree that also does not have a Hamiltonian cycle?”

Ajur wanted to show that understood these new concepts, but he struggled to see how a bipartite graph fit in here. After a few failed attempts, he said, “Aha, I see”—he quickly drew a new graph [Figure 7.7]—“this graph does not have a Hamiltonian cycle. If there was a Hamiltonian cycle, the vertices in the cycle would have to alternate between the two vertex partitions, but one vertex partition has only two vertices while the other vertex partition has three vertices.”

Rishnak smiled broadly, appreciating Ajur’s logical thinking. He said, “Okay, here’s another puzzle for you, Ajur. And I heard this one on the radio program Car Talk, which is broadcast on public radio.[4] There are nine

[4]This next problem is attributed to Bruce Robinson, a professor of Civil and Envi-

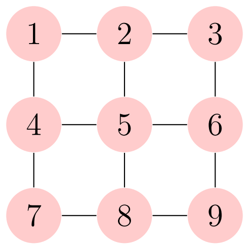


Figure 7.8: A 3×3 grid representing nine apartments with each apartment modeled as a vertex and each edge denoting an adjacency relation between apartments

jealous people who live in apartments of a building that we can represent as a 3×3 grid. And we can draw this a graph with each vertex representing one apartment."

Rishnak flashed his hands to form a new graph [Figure 7.8]. He continued, "The apartments are numbered 1 through 9, starting in the upper left-hand corner. Each person is jealous of his adjacent neighbor, so we use edges to represent jealous neighbors wanting to move above or below, or to the right or the left. The question here is very simple. What is the fewest number of total moves that can accomplish this and make everyone happy?"

Ajur thought about Rishnak's question. It must have something to do with bipartite graphs. He said, "Since there are nine vertices, the two partitions would not be of equal size. So it's not possible to have everyone move."

Rishnak smiled and said, "Good, but state your argument more clearly to make sure it is correct. What else can you add here?"

Ajur frowned, thinking for a minute how else he could show this to be true. At length, he said, "The given graph has a Hamiltonian path but not a Hamiltonian cycle. So instead of a 3×3 grid of apartments, if we had a 4×3 grid or a 3×4 grid, we would have a Hamiltonian cycle (and a bipartite graph) so it would be easy for all of them to move to an adjacent apartment."

ronmental Engineering at the University of Tennessee.

3	10	21	16	5
20	15	4	11	22
9	2	23	6	17
14	19	8	25	12
1	24	13	18	7

Table 7.1: A valid knight's tour for a 5×5 chessboard starting with move 1 at the bottom left-hand corner

Rishnak again smiled and said, "Good, Ajur. Let's look at another problem, this one back to our friend Euler. He proposed the *Knight's tour* problem on a chessboard. You probably already know a chessboard is an 8×8 square grid. Place a knight in any square. From that square, the knight has to repeatedly move[5] and successfully visit all of the squares exactly once."

Ajur raised his eyebrows in astonishment.

Rishnak continued, "You may think of this problem as finding a Hamiltonian path in a 64-vertex graph with two vertices adjacent if there is a valid knight's move from one vertex to the other. There is no easy way to find a knight's tour[6] other than what's called an *exhaustive* search, meaning we systematically explore every possible set of moves until we find a valid knight's tour or explore and exhaust all possibilities."

Ajur marveled at how one problem could be translated into another problem, then solved in a new way.

Rishnak stretched his arms out wide and a jumbling of bright letters appeared in front of Ajur [Table 7.2]. Rishnak said, "Can you find the message that is encoded in this grid of letters?"

Rishnak continued, "There are poems from different cultures, including India and China, that are actually similar to the Knight's tour problem. You can find the message by following a knight's tour."

Ajur was intrigued and resolved to read and understand these poems. Rishnak added that one first needs to verify that a knight's tour is possible

[5]To be a valid move, a knight may only move either two squares vertically up or down, then one square horizontally left or right, or one square vertically up or down, then two squares horizontally left or right. More colloquially, a knight moves in a 2×1 L shape.

[6]The solution presented in Table 7.1 is a move-by-move knight's tour for a 5×5 chessboard starting from the bottom left-hand corner.

i	t	t	t	b	l
r	h	d	e	u	s
h	a	y	e	d	o
a	o	e	p	r	n
s	n	f	o	l	l
f	v	d	i	e	u

Table 7.2: Can you find the message encoded in this grid of letters?

for a 3×3 or 4×4 chessboard. “Be careful, though,” warned Rishnak, “for checking whether a knight’s tour is possible may take a long time.”

Ajur asked, “How do you mean?”

Rishnak said, “There are many methods for checking whether a given graph has a Hamiltonian cycle. Unfortunately these conditions are not exhaustive, meaning they are not enough. Let me teach you a simple test to know whether a graph has a Hamiltonian cycle or not.”

Ajur felt his heart race in excitement.

Rishnak continued, “A well-known theorem states that if the degree of every vertex in a graph with n vertices is at least $\frac{n}{2}$, then the graph must have a Hamiltonian cycle. The proof consists of three parts:

1. We must show that the graph is connected.
2. The length of the longest path in any graph with n vertices is less than or equal to $n - 1$.
3. We identify $\mathcal{P}$ as a longest path, from which we can find a Hamiltonian cycle.”

Ajur thought about what Rishnak had said. Quite a lot to take in and understand! Therefore, Ajur followed a systematic approach. He said, “Okay first, the graph has to be connected. Otherwise, there would be at least two connected components, one of which would have size at most $\frac{n}{2}$, which violates the given degree condition.”

Rishnak nodded.

Ajur continued, “Next, in this connected graph, we need to find the longest path. Let’s say we find it and it’s $v_1, v_2, \ldots, v_k$. Since this is the longest path, all neighbors of start vertex v_1 and end vertex v_k must be in

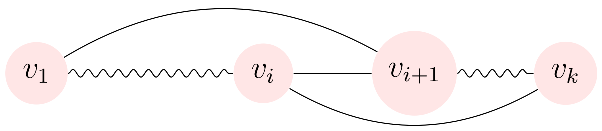


Figure 7.9: Graphical description of the argument used by Ajur to show that if the degree of every vertex in a graph with n vertices is at least $\frac{n}{2}$, then the graph must have a Hamiltonian cycle; here, solid lines represent edges, whereas squiggly lines represent paths

this path. Otherwise, we could extend that path, which would mean the path we started with was not the longest path—a contradiction."

Rishnak nodded again.

Ajur thought about he to proceed from there, then said, "Let's see, at least $\frac{n}{2}$ vertices in the path $v_2, v_3, \ldots, v_k$ are adjacent to v_1. Let those adjacent vertices of v_1 form set $S_1 = \{v_i, v_j, \ldots, v_m\}$. The size of this set is at least $\frac{n}{2}$. For each of these vertices, let's then consider their preceding vertices in the path. Let that set be S_2. The size of that set is also at least $\frac{n}{2}$. Following this through, the vertices adjacent to v_k also have to be a set of size at least $\frac{n}{2}$—and they are among $v_1, v_2, \ldots, v_{k-1}$."

Ajur paused. "But if none of the vertices adjacent to v_k are in set S_2, then the vertices adjacent to v_k would have size less than $\frac{n}{2}$"—Ajur frowned, wondering how he could show this—"and we know this because $k - 1 - \frac{n}{2} < \frac{n}{2}$. Aha, I see it"—Ajur drew a picture in the dirt [Figure 7.9]—"We would have the situation shown here."

Rishnak smiled and said, "Go on, what then?"

Ajur said, "Then this creates cycle $v_k, v_i, \ldots, v_1, v_{i+1} \ldots, v_k$. The claim is that this is a Hamiltonian cycle. If not, there is at least one vertex in this cycle that will be adjacent to some other vertex, but this will result in a path that is longer than the path we originally started with—a contradiction!"

Ajur jumped up and down with joy. This was a tough problem to crack and he was able to follow along. He loved the subtle and clever arguments used in the proof.

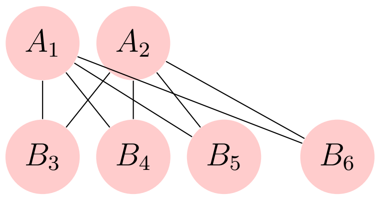


Figure 7.10: Does complete bipartite graph $K_{2,4}$ have a Hamiltonian path?

Question for the fifth day

Rishnak waved his hands to form a new graph [Figure 7.10]. He said, "Here is a complete bipartite graph, which we can abbreviate as $K_{2,4}$ since it has two vertices in set A and four vertices in set B. More specifically, vertices A_1 and A_2 are adjacent to vertices B_3, B_4, B_5, and B_6, and there are no other edges."

Rishnak cleared his throat, then said, "Is there a Hamiltonian path in $K_{2,4}$?"

Before you turn the page, try to come up with an answer of your own!

Answer for the fifth day

Ajur studied the graph in front of him. At length, he said, "In a Hamiltonian path, vertices have to alternate between the vertices in the two different partition sets. If we start with a vertex in the larger set B, then the Hamiltonian path must alternate as in $B - A - B - A - B - A - B$, which would require there to be three vertices in partition set A. Since there are only two vertices in A, there cannot be a Hamiltonian path in $K_{2,4}$."

Rishnak smiled. "Correct."

Ajur wanted more questions to answer and puzzles to solve, but Jura, who was patiently waiting nearby, was beginning to get restless, so Ajur decided to call it a day.

Chapter 8

Graph and Subgraph Isomorphism

Rishnak eagerly searched for Ajur because he wanted to share more new concepts with him. It did not take long for Rishnak to find Ajur and Jura walking along the bank of a (presumably haunted) pond in the cemetery. Rishnak immediately started the session with a small variant on what they had already been discussing.

Rishnak said, “A graph whose vertices are labeled is called a *labeled graph*, while one without labels for vertices is called an *unlabeled graph*. Two labeled graphs are equivalent if they are identical, for example these two graphs”—he flashed his hands to form a pair of labeled graphs in the air in front of Ajur [Figure 8.1] [Figure 8.2]—“are equivalent.”

Ajur studied the two graphs for a moment, then nodded.

Rishnak waved his hands again and a third graph appeared [Figure 8.3]. “This third graph is not equivalent to the other two graphs because there is no longer an edge between vertices 1 and 2 or between vertices 3 and 4. Or you could say that the vertex labels are different.”

Ajur said “I see the differences, but when are two graphs the same?”

Rishnak smiled and said, “Good question. Two graphs are called *isomorphic*—meaning structurally the same—if they are equivalent under a vertex relabeling . For example, if in the third graph [Figure 8.3], vertex 2 is relabeled as vertex 3 and similarly vertex 3 is relabeled as vertex 2, then the third graph becomes equivalent or isomorphic to the other two graphs.”

Ajur studied the third graph again and said, “Aha, I see.”

Rishnak continued, “We can also say that if a graph G is equivalent to a

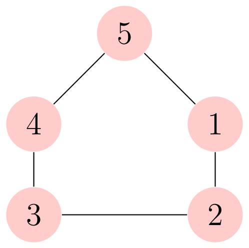


Figure 8.1: A labeled graph with five vertices and five edges

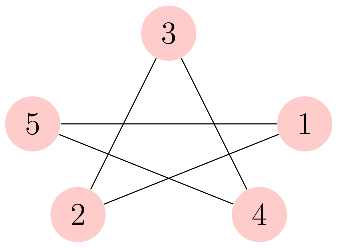


Figure 8.2: A second labeled graph with five vertices and five edges that is equivalent to the graph shown in Figure 8.1

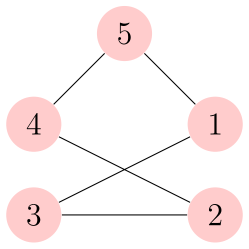


Figure 8.3: A third labeled graph with five vertices and five edges that is not equivalent to the graphs shown in Figure 8.1 and Figure 8.2 because the vertex labels are differ from that of either of the two other graphs

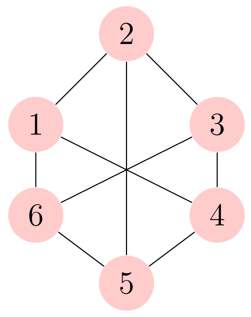


Figure 8.4: A bipartite graph with six vertices and nine edges

graph H, and H is equivalent to another graph M, then G is equivalent to graph M. In other words, the relation of graphs being equivalent or isomorphic is *transitive*. And to test whether two graphs are isomorphic is actually a very hard problem, though it is not as hard as finding a Hamiltonian cycle in a graph."

Ajur thought about this. If two graphs are isomorphic, they should have the same number of vertices, the same number of edges, the same degree sequences, the same length for the longest cycle, the same length for the shortest cycle, and so on. He repeated these to Rishnak, then asked, "Can we use some of these properties to show to graphs are isomorphic?"

Rishnak furled his brow and said, "Unfortunately not. These properties of graphs are called *graph invariants*, but by no mean are these invariants exhaustive. We do not know of a single easily computable invariant that can be used to test whether or not two given graphs are isomorphic."

Ajur frowned at this, frustrated that there were unsolved problems like this.

Rishnak said, "It is much easier to detect that two graphs are not isomorphic. Can you draw two graphs with the same number of vertices and the same number of edges that are not isomorphic?"

Ajur thought a bit then grabbed a stick and drew two graphs in the dirt [Figure 8.4] [Figure 8.5]. He said, "Both these graphs have six vertices and nine edges, but the first graph is bipartite—we know this because all cycles are of even length—while the other graph is not bipartite because it has a cycle of length 3. Therefore, these two graphs are not isomorphic."

Rishnak smiled.

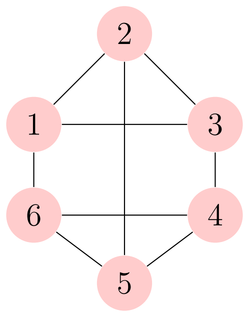


Figure 8.5: A non-bipartite graph with six vertices and nine edges

Ajur thought for awhile, then said that partial solutions to the graph isomorphism problem would be very useful if you wanted to test whether a given graph is in a collection of graphs, a problem that comes up in chemistry.

Rishnak was pleased that Ajur was thinking about some of the practical applications of graph theory. Rishnak said, "There is an easier method to test for isomorphism between two rooted trees. Let's start with how we can encode a labeled tree. We'll use an approach called a Prüfer code. The construction is iterative, and the Prüfer code of a labeled tree with n vertices is a code of length $n-2$. Here's the algorithm, but first, remember that a leaf of a tree is a vertex that is connected to only one other vertex.

1. Let Prüfer code P_c be an empty string.
2. Find leaf vertex v with the smallest label, then identify vertex w that connects vertex v to the rest of tree.
3. Remove v from the tree and append w to Prüfer code P_c.
4. Repeat from step 2 until we are left with only two vertices."

Ajur raised his eyebrows and said, "Wow, that's a lot to take in."

Rishnak laughed and said, "Try it on this rooted tree." He waved his hands and a new graph [Figure 8.6] appeared in front of Ajur.

Ajur took a deep breath and said, "Okay, the smallest leaf is vertex 4. Its adjacent vertex is labeled 2, so we append 2 to Prüfer code P_c and remove vertex 4. The next smallest leaf vertex is labeled 5 and is adjacent

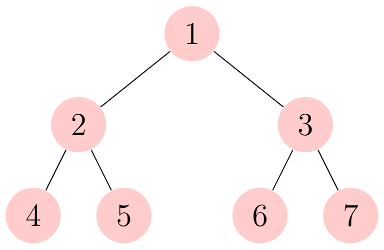


Figure 8.6: A rooted tree with seven vertices, of which four vertices are leaf vertices

to vertex 2. Therefore, we append 2 to P_c and remove vertex 5. Now vertex 2 is the smallest leaf vertex and its adjacent vertex is vertex 1, so we append 1 to P_c and remove vertex 2. At this point, P_c is 221, but we're not done yet."

Rishnak smiled and said, "Right, go on."

Ajur said, "Wait, vertex 1 is now the smallest leaf vertex, but it was originally the root of the tree. Is that okay?"

Rishnak said, "Yes, that's part of the algorithm."

Ajur shrugged and continued, "Okay, vertex 1 is next and its adjacent vertex is vertex 3, so vertex 1 is removed and we append 3 to P_c. The next smallest leaf is then vertex 6 and its adjacent vertex is vertex 3, so we again append 3 to P_c. We stop when there are only two vertices left, so we are done. The Prüfer code for the tree is $P_c = 22133$."

Rishnak nodded and said, "Good, since the length of P_c is 5, we can immediately conclude that there are seven vertices in the tree. Further, we can also conclude that vertices labeled 4, 5, 6, and 7 are the leaf vertices since they do not appear in the given Prüfer code."

Ajur asked, "What about unlabeled trees? Can we find a Prüfer code for an unlabeled tree?"

Rishnak replied with a resounding yes and explained the steps to follow for a rooted tree. He said, "For any tree, one needs to first choose an appropriate root vertex, which we can call the *center* of the tree. Here's the basic algorithm to follow.

1. Assign each leaf vertex a label of 01.

2. Let x be a non-leaf vertex. For all of the children of x, sort these

labels in order. Assign vertex x with a label that is a 0, followed by the concatenation of the sorted labels of its children, followed by a 1.

3. Repeat step 2 until you label the root vertex."

Rishnak knew this was a lot to follow. He said, "Have another look at this tree [Figure 8.6]. If this was an unlabeled graph, then leaf vertices 4, 5, 6, and 7 would all be labeled 01. Then vertex 2 would be labeled 001011. Same with vertex 3. Finally, root vertex 1 would have the label 00010110010111, which would serve as the Prüfer code for this tree."

Ajur tried his best to follow. After some time, he said, "I think I see. Using these Prüfer codes, we could compare them to see if they match, which would show that the two rooted trees are the same—I mean, isomorphic."

Rishnak smiled and said, "Precisely. And closely related to the problem of graph isomorphism is a problem that has plenty of uses in real life, not just in the ghoulish realm."

Ajur chuckled at the bad joke.

Rishnak continued, "Instead of asking whether two graphs are structurally the same and therefore isomorphic, given two graphs G and H, the *subgraph isomorphism* problem is to determine whether there is a subgraph of G that is isomorphic to H."

Ajur said, "A subgraph of G. I understand. This approach could be used to test whether graph G with n vertices has a Hamiltonian cycle by choosing graph H to be a cycle of length n and asking whether there is a subgraph of G isomorphic to H."

Rishnak nodded and said, "Yes, and that is the reason why the subgraph isomorphism problem is hard. On the other hand, if both G and H are labeled graphs, there are heuristics to test whether H occurs in G as a subgraph. This type of labeled subgraph isomorphism problem has many applications in medical imaging and chemical structure identification."

Ajur marveled at how connected and useful graphs and subgraphs were.

Question for the sixth day

Rishnak said, "Okay, Ajur, the time has come. Here is the question for the sixth day. Can you draw a tree with a Prüfer code of 232?"

Before you turn the page, try to come up with an answer of your own!

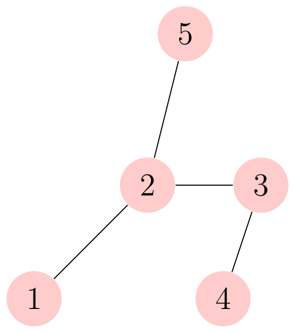


Figure 8.7: Tree with 5 vertices with a Prüfer code of 232

Answer for the sixth day

Ajur nodded and said, “Yes, first of all, I know that this will be a tree with five vertices because the Prüfer code has a length of 3.”

Without hesitating, Ajur drew a tree in the dirt [Figure 8.7]. He said, “This is the tree.”

Rishnak was happy with Ajur’s answer, and Ajur was eager to go home, with Jura at his side, to think about all of the applications graphs and subgraphs might have in the world.

Chapter 9

Planar Graphs

Ajur was so enamored by Hamiltonian cycles, he was eager to meet Rishnak and learn more about them, but Rishnak wanted to introduce a different concept: drawing graphs.

Ajur said, “I already know how to draw graphs.”

Rishnak said, “Wait, Ajur, there’s much more to it. Listen. A planar drawing of a graph is one in which no edges cross each other (except at their endpoint vertices). By definition, a *planar graph* is a graph for which there exists at least one planar drawing. And if a graph does not have any possible planar drawings, we call it a *non-planar graph*.”

In a dazzling flash of light, Rishnak waved his hands and two graphs appeared [Figure 9.1] [Figure 9.2]. He said, “For example, here are two drawings of complete graph K_4, the 4 in this case meaning we have four

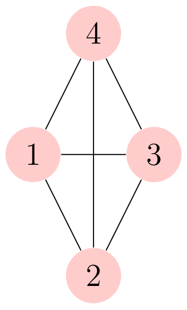


Figure 9.1: A drawing of complete graph K_4 in which two edges cross

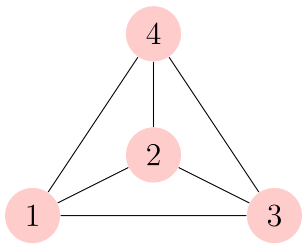


Figure 9.2: A planar drawing of K_4 in which no edges cross

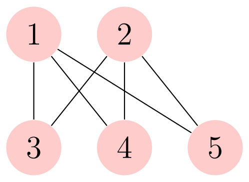


Figure 9.3: Bipartite graph $K_{2,3}$ with five vertices and six edges in which four edges cross

vertices. Notice that the second graph [Figure 9.2] has no edges that cross, so we call K_4 a planar graph. In other words, it displays nicely on a two-dimensional plane."

Rishnak asked, "So, Ajur, can you draw this graph"—he waved his hands and a new graph appeared [Figure 9.3]—"as a planar graph, that is, with no edges that cross?"

Ajur studied the graph and found the task to be a bit challenging. Then, he thought of moving vertex 2 down below vertex 4, thinking that would help separate the crossed edges. He picked up a stick and drew a graph in the dirt [Figure 9.4], smiling as he realized he had solved the problem.

Impressed, Rishnak waved his hands and another new graph appeared [Figure 9.5]. He asked Ajur whether this new graph had a planar representation or not.

Ajur worked at this problem for some time, struggling to find a planar drawing.

At length, he said, "I can only come up with this graph"—he pointed to the graph he had drawn [Figure 9.6]—"but there's one edge crossing I can't get rid of."

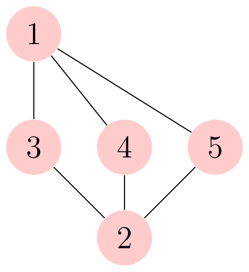


Figure 9.4: A planar drawing of bipartite graph $K_{2,3}$ with five vertices and six edges

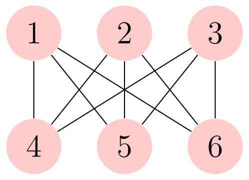


Figure 9.5: Bipartite graph $K_{3,3}$ with six vertices and nine edges in which seven of the edges cross

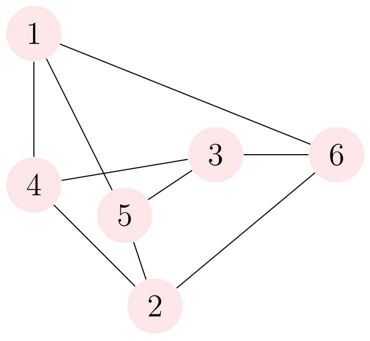


Figure 9.6: A drawing of bipartite graph $K_{3,3}$ in which just two edges edges cross

Rishnak told Ajur that bipartite graph $K_{3,3}$ does not have a planar representation. He said, "There is a well-known related puzzle credited to Sam Loyd, an American recreational mathematician and puzzler. Let's say there are three houses, which we can represent as vertices 1, 2, and 3 of $K_{3,3}$, drawn on paper. Below them are three vertices 4, 5, and 6 representing gas, water, and electricity suppliers. The aim of the puzzle is to draw lines to get each utility into every house, but—"

Ajur chimed in, "But without crossing any of the lines." Ajur laughed at the puzzle, thinking that he should read up on Loyd's puzzle books the next time he visited his local library.

Rishnak cleared his throat, not entirely happy that Ajur cut him off. He said, "If two graphs G and H are isomorphic and if G is planar, can you say that H is also planar?"

Ajur said, "Yes, if H is isomorphic to G, then each vertex of H corresponds to some vertex of G. By just relabeling the vertices of G in its planar drawing, we can obtain a planar drawing of H."

Rishnak nodded and asked, "What can we say about trees?"

Ajur thought for a moment, then said, "That's easy. All trees have a planar representation[1] because there are no cycles."

Rishnak said, "Right, trees are indeed easy to embed in a plane as there are no cycles. There are some interesting space-filling planar representations of trees, for example this one"—he waved his hands and an intricate planar drawing appeared in front of Ajur [Figure 9.7].

Ajur marveled at the drawing.

Rishnak continued, "One way to get a planar drawing of a graph is to first embed the longest cycle in a plane, then try to place the rest of the vertices so that no edges cross one another. The longest cycle will divide the plane into two regions, an inner region (inside the cycle) and an outer region (outside the cycle)."

Ajur said, "So the edges have to go either inside or outside the longest cycle."

Rishnak said, "Precisely. By trying out various possibilities, backtracking as necessary, you can eventually get a planar representation of a graph—if one exists. Of course, this process is easier said than done."

Ajur asked, "Is there any easier way?"

[1]Ajur liked monkeys and monkeys liked trees. It follows by transitivity that Ajur liked trees.

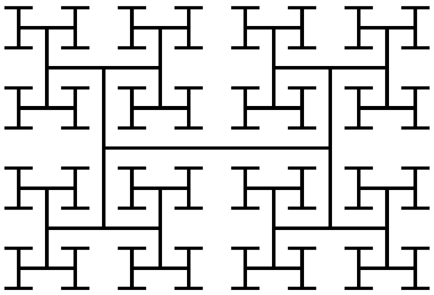


Figure 9.7: A planar drawing of a tree, which is self-similar

Rishnak said, "There is in fact an efficient method of placing the edges, without any backtracking, but the details are rather complicated."

After a few minutes of silence in which Ajur thought about how this might work, Rishnak said, "Remember Euler?"

Ajur nodded.

Rishnak waved his hands and a new graph appeared [Figure 9.8]. He said, "Euler discovered a remarkable relationship between planar graphs and three-dimensional solids. This planar graph represents a tetrahedron—a pyramid."

Ajur asked, "How is that a pyramid? It's two-dimensional."

Rishnak chuckled and said, "If you imagine stretching vertex $a3$ out toward you, you end up with a three-dimensional solid with four sides, hence the name tetrahedron."

Ajur studied the graph and said, "Oh, I see now."

Rishnak continued, "Think of each region in a planar representation as the *face* of such a three-dimensional solid. Then Euler's formula relating the number of edges e, the number of vertices n, and the number of faces f of a planar graph is simply this." Rishnak waved his hands and Euler's equation appeared as follows:

$$f - e + n = 2 \tag{9.1}$$

Rishnak said, "Let me explain this a bit. Each region corresponds to a cycle in the graph that surrounds that region. It also defines the external region, which is sometimes also called the infinite region. We know there are $n - 1$ edges in a tree, so if we can find that tree in the given graph, then the rest of the $e - (n - 1)$ edges will form part of a cycle. This gets

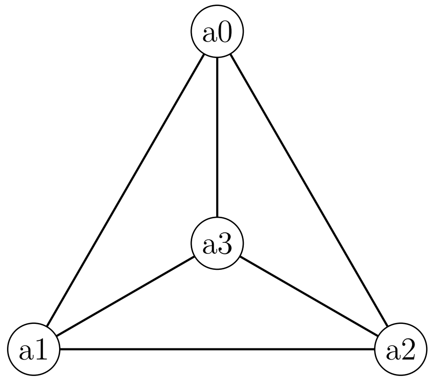


Figure 9.8: Planar graph representing a tetrahedron

tricky and we will have to explore this at some point in the future.[2] Each of these cycles correspond to an internal region—in other words, an internal face—and one external face. From this, Euler's equation (9.1) follows."

As Rishnak spoke, he drew the following equations in the air:

$$\begin{aligned} \text{internal faces} &= e-(n-1) \\ \text{external faces} &= 1 \\ \text{total faces} &= \text{internal faces } + \text{ external faces} \\ &= e-n+2 \qquad (9.2) \end{aligned}$$

Ajur studied the equations and understood, at least for a tetrahedron. He said, "Okay, I've constructed Platonic solids with Zometool[3]. Does this equation work for all of these?"

Rishnak smiled and said, "Ah yes, the Platonic solids, named after Plato, the ancient Greek philosopher. And yes, Euler's equation applies to the five Platonic solids. In other words, the graph theory behind Euler's formula can be used to relate vertices, edges, and faces of the Platonic solids."

Rishnak waved his hands and next to the graph corresponding to the tetrahedron [Figure 9.8], four more graphs appeared [Figure 9.10] [Figure 9.9] [Figure 9.11] [Figure 9.12]. He said, "Ajur, can you write down

[2]Rishnak and Ajur end up discussing this in Chapter 11 when they talk about spanning trees in a connected graph and how to choose one.

[3]Zometool is a commercial construction kit for making general polyhedra.

the number of vertices n, the number of edges e, and the number of faces f for each of these?"

Ajur looked at each graph in turn, then systematically wrote the values in the dirt to create his table [Table 9.1].

Rishnak smiled and said, "Good. All Platonic solids have a planar representation, as you can see in these graphs. The reason this works is that every solid fits within a sphere in such a way that no point or vertex passes through what would be the North pole. From this, one may stereographically project the graph onto the plane."

Solid	n	e	f
Tetrahedron	4	6	4
Cube	8	12	6
Octohedron	6	12	8
Dodecahedron	20	30	12
Icosahedron	12	30	20

Table 9.1: Parameters of Euler's equation for the Platonic solids

Rishnak said, "Notice that each edge appears in exactly two faces. If each face is a triangle—therefore a cycle of length 3—the graph is called a *maximal planar graph*. From this, we can deduce $e = \frac{3f}{2}$, or $f = \frac{2e}{3}$. Substituting this for f in Euler's equation (9.1)"—Rishnak waved his hands and equations floated in the air in front of Ajur—"we can solve for e."

$$\begin{aligned}
\frac{2e}{3} &= e - n + 2 \\
-\frac{e}{3} &= -n + 2 \\
\frac{e}{3} &= n - 2 \\
e &= 3n - 6 \qquad (9.3)
\end{aligned}$$

Rishnak paused after showing this equation (9.3), giving Ajur time to think about it, then said, "By using this equation, we can show that complete graph K_5"—he flashed his hands and a new graph appeared [Figure 9.13]—"is non-planar. This graph has five vertices and 10 edges,

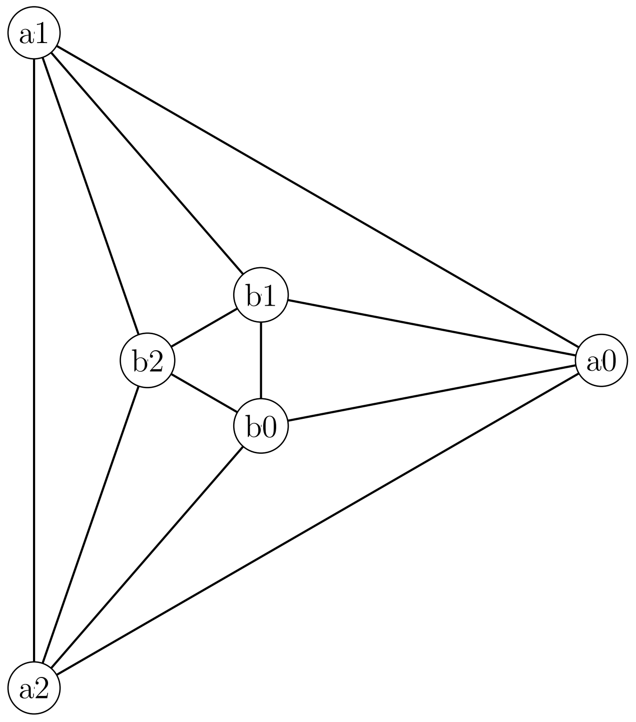


Figure 9.9: Planar graph representing a octahedron

so $n = 5$ and $e = 10$. But from the equation, if it were planar, then the number of edges would have to be at most 9."

Ajur raised his eyebrows and said, "Aha, I see how that works. Nice!"

Rishnak asked Ajur, "Can you use Euler's equation to show that complete bipartite graph $K_{3,3}$ [Figure 9.5] with its six vertices is non-planar?"

Ajur was alert and knew that a bipartite graph only has even cycle lengths. He said, "To have the maximum number of edges, each face has to have a cycle length of 4. Therefore, $2e = 4f$ or $f = \frac{e}{2}$."

Ajur scribbled down equations in the dirt and said, "Substituting for f in Euler's equation $f = e - n + 2$ (9.1), we get $-\frac{e}{2} = n - 2$, which we can write as $e = 2n - 4$."

Rishnak smiled as Ajur went on, "Substituting $n = 6$, we get $e = 8$. That's the maximum number of edges possible, but $K_{3,3}$ has nine edges, which is more than eight. Therefore, we have proven that $K_{3,3}$ is non-planar!"

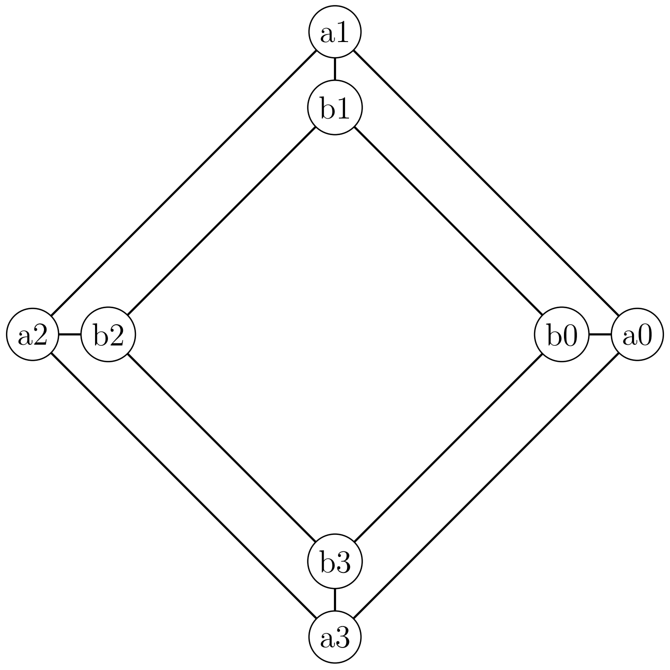


Figure 9.10: Planar graph representing a cube

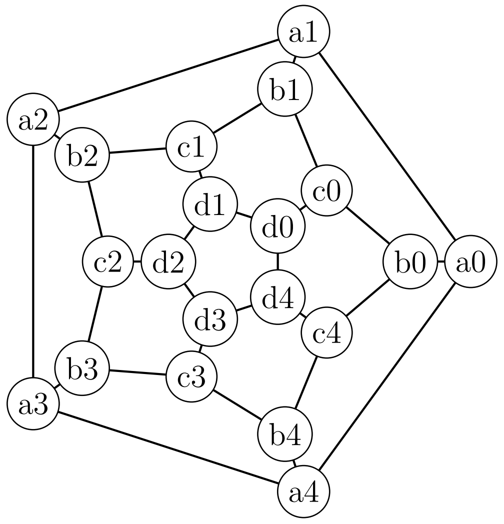


Figure 9.11: Planar graph representing a dodecahedron

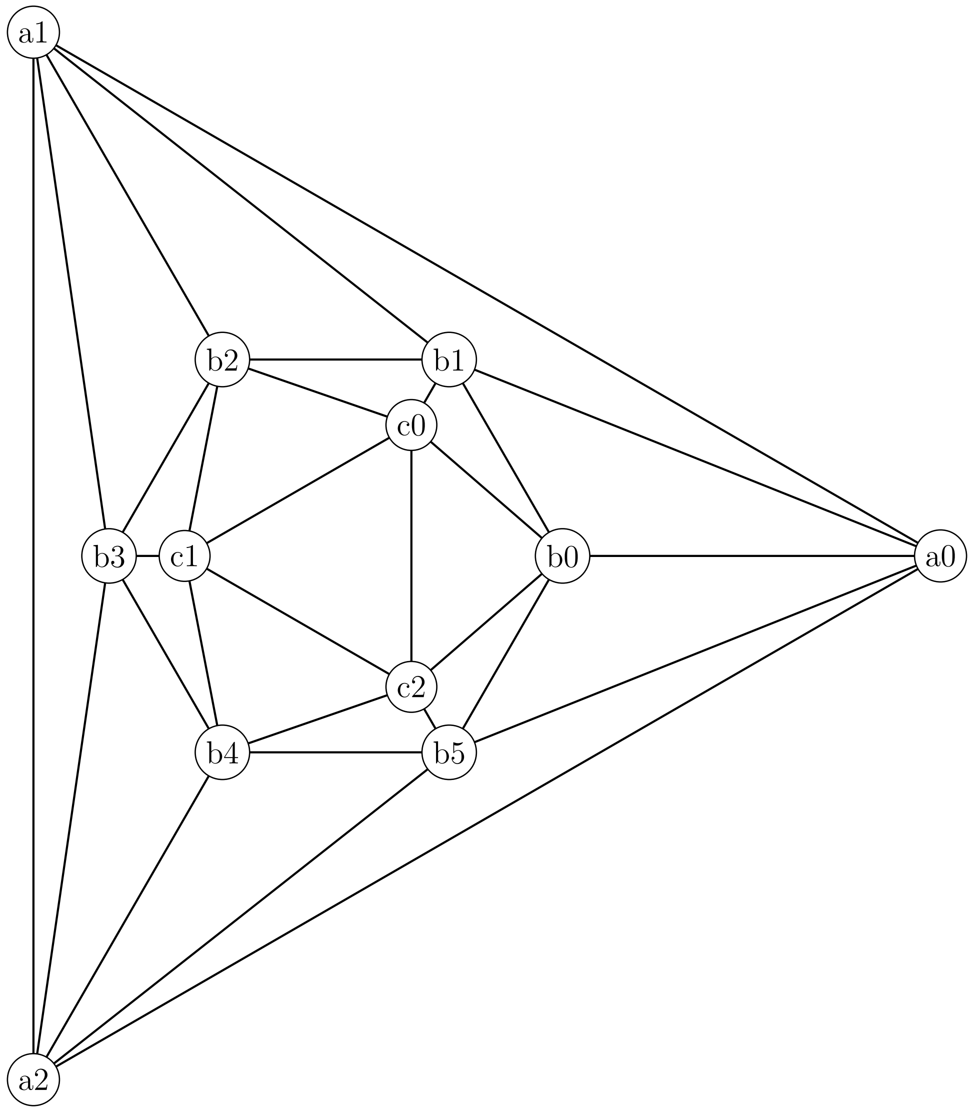


Figure 9.12: Planar graph representing a icosahedron

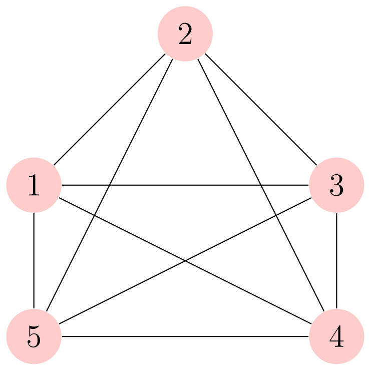


Figure 9.13: Complete graph K_5 with five vertices and 10 edges

Rishnak's smile broadened. He said, "Good, Ajur. And here's one more interesting fact. It turns out that any planar graph has not only a planar representation that we can draw, but also a representation in which each edge can be drawn as a straight line. Something to think about after we depart for the night."

Ajur thought for a moment, then said, "And every subgraph of a planar graph must also be a planar graph, right?"

Rishnak said, "Yes, Ajur. Also, the operation of dividing an edge by introducing a new vertex of degree 2 will not change whether it is planar or non-planar. We could also do this twice to essentially add a new edge and not change whether the graph is planar or non-planar."

Ajur frowned and said, "I don't understand."

Rishnak said, "Have a look at this graph"—he waved his hands and a new graph appeared [Figure 9.14]—"from the original tetrahedron graph [Figure 9.2], we can divide an edge by adding a vertex or, in this case, we can add two vertices with a new edge between them. In the end, we still have a planar graph."

Ajur studied the graph in front of him until he understood.

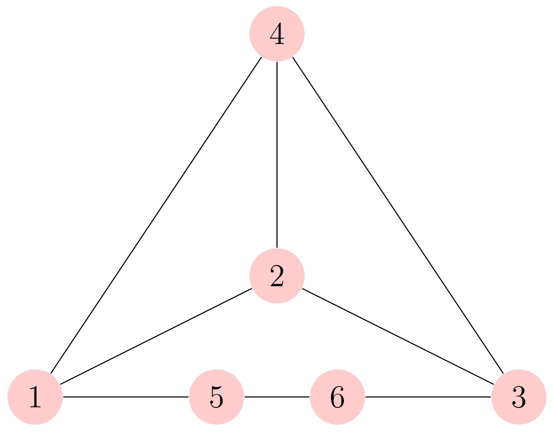


Figure 9.14: A division of original edge $(1, 3)$ by adding two new vertices (5 and 6) to the planar graph shown in Figure 9.2

Question for the seventh day

Rishnak stretched his arms out and said, “We have covered a lot today. You should be ready to answer the question for the seventh day. Can you show that in a planar graph, there exists at least one vertex of degree 5?”

Before you turn the page, try to come up with an answer of your own!

Answer for the seventh day

Ajur thought about this. He remembered Euler's equation (9.1) and said, "Each edge occurs in exactly two faces. And the smallest face could be a cycle of length 3. Therefore, substituting $f = \frac{2e}{3}$, we get $\frac{2e}{3} = e - n + 2$, which results in $\frac{e}{3} = n - 2$ or just $e = 3n - 6$, so then[4]—"

Ajur frowned, wondering where to go with this line of thinking. At length, he said, "This tells us that $3n - 6$ is the maximum number of edges a planar graph can have, so if all vertices have degree 6 or more, then $e \geq 3n$, contradicting that the graph is planar."

Rishnak was happy with Ajur's answer. Jura appeared to be getting restless, so Rishnak decided that they had had enough for that session. They parted for the night.

[4]Ajur essentially derived Euler's equation (9.3) once again

Chapter 10

Graph Coloring

Rishnak was impressed with Ajur's logical reasoning ability and went searching for him. It did not take long for Rishnak to find him—ghosts can move rather quickly. He found Ajur and Jura sitting on a stone bench. Without hesitation, Rishnak asked Ajur if he liked coloring.

Ajur enthusiastically responded that he did indeed like to color and enjoyed its calming effect.

Rishnak was pleased with this response and introduced the next topic. He said, "Let's talk today about graph coloring. A proper *vertex coloring* of a graph is a coloring of vertices such that no adjacent vertices—I mean vertices connected by an edge—have the same color."

Rishnak flashed his hands and a graph appeared in front of Ajur, but this graph gleamed with different colors [Figure 10.1]. He said, "This is an example of a proper vertex coloring of a graph using four colors."

Ajur marveled at the graph in front of him.

Rishnak continued, "An interesting problem is to determine the smallest number of colors to properly vertex color a graph. Can you do better than four colors for this graph, Ajur?"

Ajur jumped up from the stone bench and grabbed a stick. He said, "Yes, I already see one with three colors." He drew a new graph in the dirt, using fallen leaves to show the three different colors [Figure 10.2].

Rishnak asked, "How do you know three is the minimum number of colors that you need for that graph?"

Ajur thought for a moment, then said, "Three colors are needed because vertices 1, 2, and 4 are mutually adjacent, so those vertices need three distinct colors."

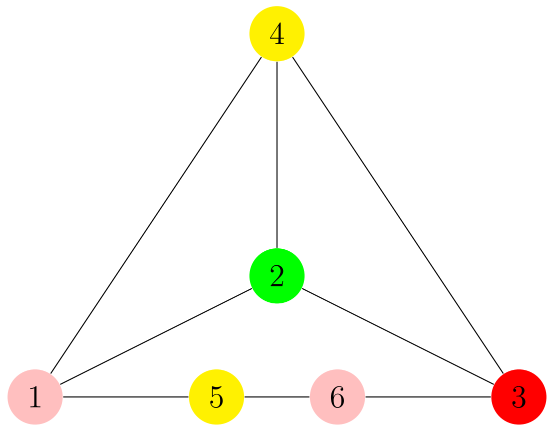


Figure 10.1: A proper coloring of the vertices of a graph using four colors

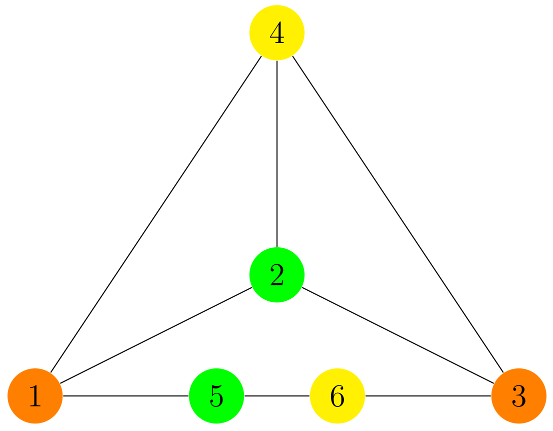


Figure 10.2: A proper coloring of the vertices of the graph shown in Figure 10.1 using only three colors

Rishnak smiled and asked Ajur to properly color complete bipartite graph $K_{3,3}$.

Ajur jumped at the opportunity and quickly drew graph $K_{3,3}$ in the dirt, then used orange and yellow leaves to color the graph [Figure 10.3].

Rishnak said, “Good. How did you come to that answer so quickly?”

Ajur said, “In a bipartite graph, vertices are partitioned into two sets A and B. Since the edges always go from a vertex in A to a vertex in B, all

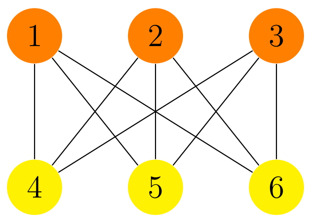


Figure 10.3: Two coloring of complete bipartite graph $K_{3,3}$ with six vertices and nine edges

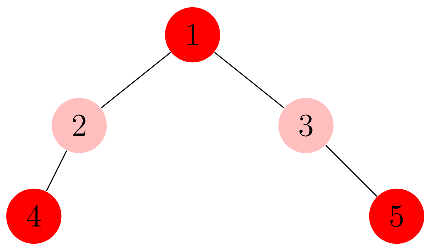


Figure 10.4: Two coloring of a tree

of the vertices in A can be colored with one color, say orange, while all of the vertices in B can be colored with another color, say yellow."

Rishnak said, "Right, and since you are using leaves, let's talk about trees. Since all trees are also bipartite graphs, trees can always be colored with just two colors." Rishnak waved his hands and a brightly colored red and pink graph appeared in front of Ajur [Figure 10.4].

Ajur grew more curious and asked, "Can you also color edges? And if so, is there a concept of adjacent edges?"

Rishnak always believed that asking the right questions is important because it is a signal that one's understanding is deepening. He responded, "Yes, two edges are adjacent if they are incident to the same vertex. Consider this graph"—he waved his hands and a new graph appeared [Figure 10.5]—"Edge e_1 is adjacent to edge e_2 since both are incident at vertex 1. Edge e_1 is also adjacent to edges e_3, e_4, and e_5 because all of these edges are incident at vertex 2."

Ajur exclaimed, "Then the maximum degree of this graph, which is four, tells us that we need four colors for the edge covering."

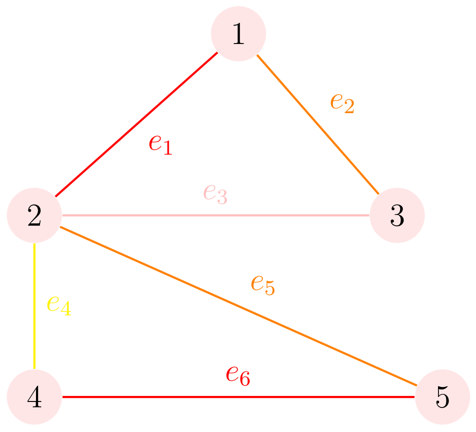


Figure 10.5: Four edge coloring of a graph

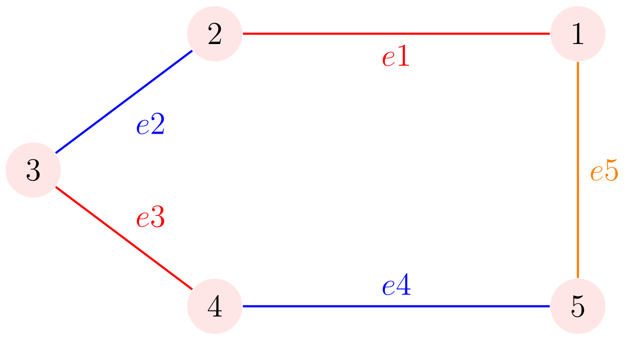


Figure 10.6: Three edge coloring of a cycle of length 5

Rishnak smiled and said, “Your observation regarding the maximum degree of a graph is good, but the maximum degree of graph, call it Δ, does not imply that the graph has a Δ-coloring. Think of this graph”–Rishnak waved his hands and a new graph appeared [Figure 10.6]–“and notice that it is a cycle of length 5 with a maximum degree of $\Delta = 2$, but it needs three colors to color its edges.”

Ajur frowned and said, “Oh right, I see. I was hoping there was a simple mathematical explanation here.”

Rishnak said, “There is. A graph with maximum degree Δ can be edge colored with either Δ or $\Delta + 1$ colors.”

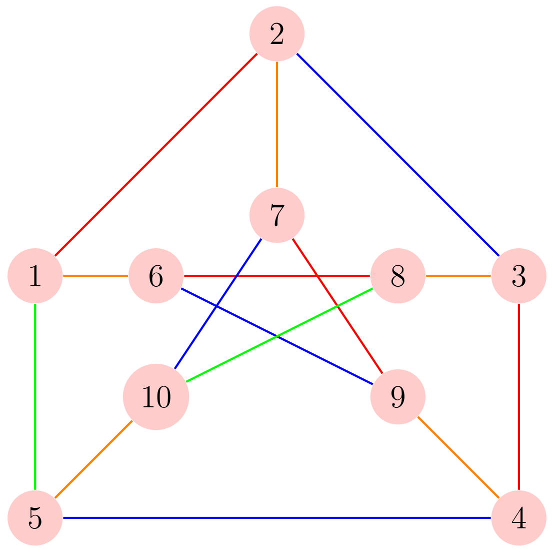


Figure 10.7: Four edge coloring of the Petersen graph, a regular graph of degree 3, and therefore a Snark

Ajur's frown turned into a smile.

Rishnak also smiled, then said, "A regular graph with degree 3—meaning a graph with all of its vertices having a degree of 3, also known as a *trivalent graph*—that needs four edge colors for a proper edge coloring is known as a *Snark*."

Ajur laughed. He had heard of Snarks from a poem titled *The Hunting of the Snarks* by his dad's favorite author, Lewis Carroll.

Rishnak smiled and said, "Strange name, yes, but graph theorists often have a whimsical sense of humor, and since four-edge-colorable trivalent graphs are awfully elusive, they named these graphs Snarks."

Ajur thought for a moment, curious to find one for himself. He asked, "Is the Petersen graph a Snark?"

Surprised that Ajur remembered, Rishnak replied that the Petersen graph indeed needs four colors to properly color the edges of that graph. He waved his hands and the graph appeared in front of Ajur [Figure 10.7].

Rishnak said, "We can also convert an edge coloring problem to a vertex coloring problem. From a given graph G for which you want to color the edges, we can construct a new graph H in which edges of G become vertices

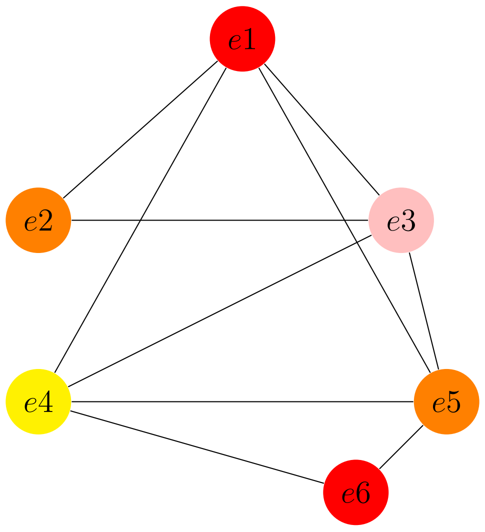


Figure 10.8: Four vertex coloring of a graph that corresponds to the four edge coloring of the graph shown in Figure 10.5

of H, and two vertices of H are adjacent if the corresponding edges in G are incident on the same vertex of G."

Rishnak flashed his hands and said, "Here, let me show you. The edge coloring of this graph"—he waved his hands and a previous graph appeared [Figure 10.5]—"can be converted into a vertex coloring in this graph"—Rishnak waved his hands again and a new graph appeared [Figure 10.8].

Ajur studied the two graphs. He said, "I see, but the edges in the new graph H don't exactly correspond to the vertices of G. Instead, there's an edge in H for each adjacent pair of edges in G."

Rishnak nodded, happy to see Ajur understood.

Rishnak said, "Here's a new problem for you, Ajur, somewhat related to the edge coloring problem. In a group of six people, say Alexis, Bailey, Charles, Danny, Elaine, and Francis, each pair of individuals could be either a friend or an enemy. For example, Charles might be friends with Alexis, Bailey, and Charles, but enemies with the other three. Can you prove that in a group of six people, there will be at least three people who are mutual friends with one another or mutual enemies with one another?"

Ajur thought about this, then said, "I could try to model this problem as a complete graph with six vertices, coloring the 15 edges either red or

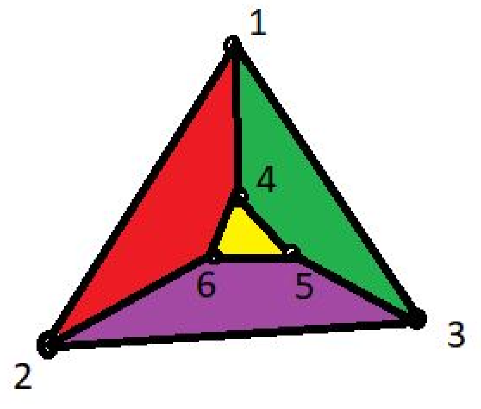


Figure 10.9: A graph with four regions or faces, with each region or face colored a different color

blue. We could use red to denote enemies and blue to denote friends. No matter how we color the edges, there will always be a red triangle of three vertices or a blue triangle of three vertices, right?"

Rishnak nodded. He was impressed with Ajur's ability to translate the given problem into a graph-theoretic problem even if he could not solve it completely.[1]

Ajur asked, "What other graph coloring problems are there?"

Rishnak said, "A quite useful one is the map coloring problem. The idea is to color the regions (or faces of a planar graph since maps are usually drawn on a plane) such that no two adjacent regions (or faces) are colored the same. Here is an example of a map coloring of a graph with six vertices and nine edges."

Rishnak waved his hands an a new graph appeared, its four faces colored in [Figure 10.9]. He said, "The four faces are $(1, 2, 4, 6)$, $(1, 3, 5, 4)$, $(2, 3, 5, 6)$ and $(4, 5, 6)$. Each of these regions share a border or edge with other regions. Therefore, all of these four regions are mutually adjacent. Often the outside region is also colored. And in this graph, the outside region could be colored yellow just like the center region. We know this because the outside region does not share a border with region $(4, 5, 6)$."

[1]You can try to work out the full solution.

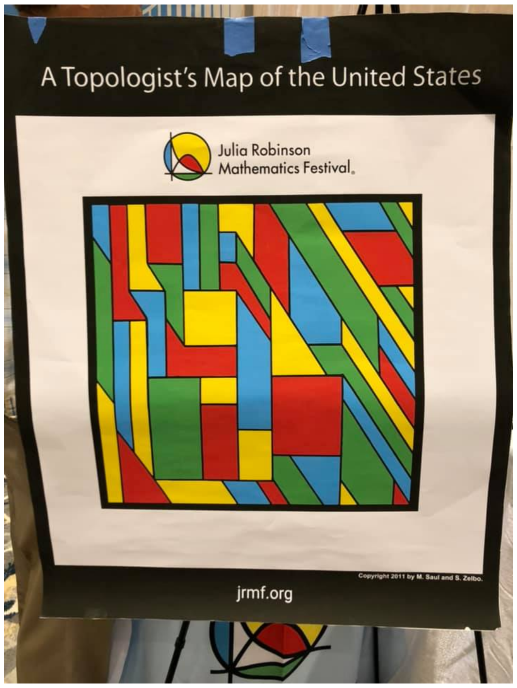

A Topologist's Map of the United States
Julia Robinson Mathematics Festival
Copyright 2011 by M. Saul and S. Zelbo.
jrmf.org

Figure 10.10: USA Map with face (i.e., state) adjacencies preserved, colored using only four colors

Ajur said, "So we still need only four colors."

Rishnak nodded. He said, "One of the most famous theorems about map coloring is that every planar graph can be map colored with four colors. That is, we need only four colors to color every region such that no two adjacent regions have the same color. Here, have a look at this example map of the USA, colored using only four colors."

Rishnak waved his hands and a new image appeared in front of Ajur [Figure 10.10]. Ajur frowned and said, "That doesn't look at all like the USA."

Rishnak laughed and said, "The regions or faces in this map are indeed the states of the USA, but right, it is not like the normal map of the USA that we see in an atlas. The states are not shaped the same or drawn to scale, but this map of the USA preserves the adjacency relationships between each state. Can you identify the state of Maine?"

Ajur thought for a while, trying to remember how the actual USA map looked. He said, "I think Maine borders only one other state, New Hampshire. So, the small yellow triangular region in the northeast corner must be the state of Maine."

Rishnak nodded and said, "How about the state that borders the most other states? This is actually a tie between two states."

Ajur raised his eyebrows and said, "Wow, that's a tough one. They must be states in the Midwest."

Rishnak said, "Yes, Tennessee and Missouri both border eight other states. They actually border one another, too.[2] Here's another map, this one of India."

Rishnak flashed his hands and a map of India with state names shown appeared [Figure 10.11]. Rishnak said, "Could the two states of Andhra Pradesh and Kerala be colored using the same color?"

Ajur said, "Yes, they could definitely be colored using the same color because they do not share a border."

Rishnak asked, "What about Goa and Kerala?"

Ajur replied instantaneously with a resounding, "Yes, they also do not share a border. We can just convert the map coloring problem into a vertex coloring problem, similar to converting the edge coloring problem to a vertex coloring problem, right?"

Rishnak said, "How might you do that?"

Ajur explained that each region or face of the original map would become a vertex of the new graph, and two vertices in the new graph would have an edge to show they are adjacent if the corresponding regions in the original graph shared a border. He said, "For example, in the map of the USA, all of the states would become vertices of the new graph, with an edge added for each corresponding pair of states that share a border with one another. So the vertex corresponding to Maine would be adjacent only to the vertex corresponding to New Hampshire. Similarly, the vertex corresponding to Washington would be adjacent to vertices corresponding to Oregon and Idaho."

Rishnak nodded. He was very pleased with the way Ajur was able to quickly absorb the material.

[2]Can you find these two states in Figure 10.10?

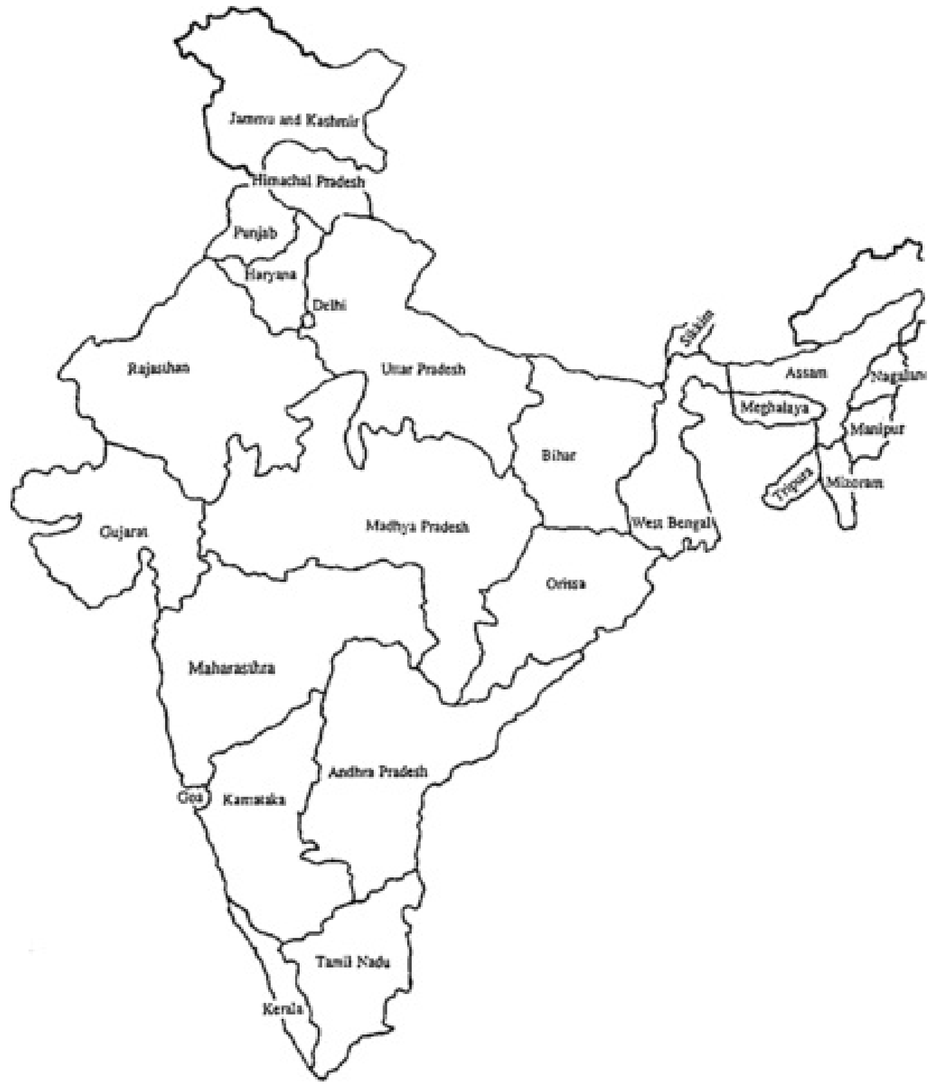


Figure 10.11: An old map of India with states and borders shown

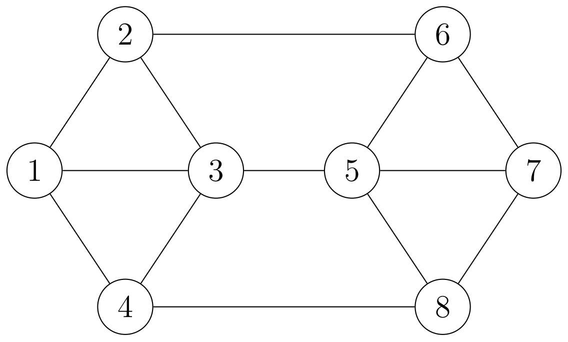


Figure 10.12: Can you vertex color this graph with as few colors as possible?

Question for the eighth day

Rishnak said, "It is time, Ajur, for the question for the eighth day. Can you color the vertices of this graph"—he flashed his hands to show a new graph [Figure 10.12]—"with as few colors as possible."

Before you turn the page, try to come up with an answer of your own!

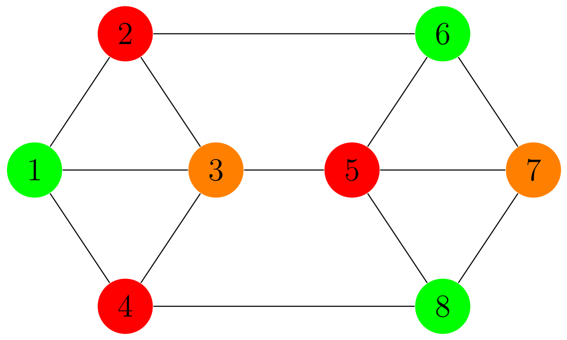


Figure 10.13: Vertex coloring of the graph shown in Figure 10.12 with a minimum number of colors, namely 3

Answer for the eighth day

Ajur had no problem coloring this graph as follows Figure 10.13

Rishnak nodded and did not want to push further, so he called it a day. Ajur and Jura walked home, Ajur smiling over everything he had learned that day.

Chapter 11

Spanning Trees

Rishnak found Ajur and Jura walking along a stretch of road with trees on either side. Recalling his previous talks with Ajur about trees, Rishnak chose a specific kind of tree that is found within a graph for his next session with Ajur.

Rishnak said, "A graph G is connected if there is a path between every pair of vertices in G. So, in connected graph $G = (V, E)$, a *spanning tree* is tree $T(V, E_1)$ that covers all vertices in V and is a subgraph of G that satisfies two conditions:

1. Subgraph T is a tree, meaning it contains no cycles and is connected.

2. Vertex set V of T is the same as that of G."

Rishnak flashed his hands and formed two graphs in the air in front of Ajur [Figure 11.1] [Figure 11.2]. He said, "Here is an example of a graph with a corresponding spanning tree."

Ajur studied both graphs, recognizing that the second graph was a subgraph of the first.

Rishnak asked Ajur if he could construct another spanning tree for the original graph [Figure 11.1].

Ajur was eager to show off and drew a subgraph in the dirt with a stick [Figure 11.3].

Rishnak asked, "How many distinct spanning trees are there?"

Ajur thought about this for a moment. He said, "Do you mean how many labeled non-isomorphic trees there are?"

Rishnak said, "Yes, exactly the question?"

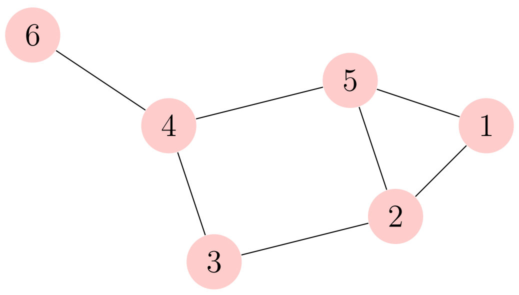


Figure 11.1: Example graph with six vertices and seven edges

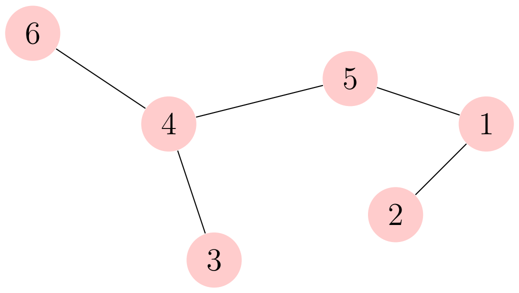


Figure 11.2: A subgraph of the graph shown in Figure 11.1 that forms a spanning tree of the original graph

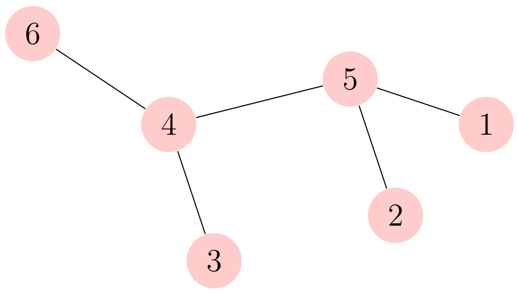


Figure 11.3: Another spanning tree of the graph shown in Figure 11.1

Ajur said, "There are two cycles in this graph, namely $(2, 3, 4, 5)$ and $(1, 2, 5)$, and they share common edge $(2, 5)$. The cycles are of length 4 and length 3, so there are four possible spanning trees to choose from out of the first cycle and three possible spanning trees to choose from out of the other cycle. Multiply these together and we have 12 possibilities, but one of them ends up being cycle $(1, 2, 3, 4, 5)$ if edge $(2, 5)$ is omitted. Therefore, we have 11 spanning trees, each of which must also contain edge $(4, 6)$. Here they are."

Ajur painstakingly wrote out all 11 spanning trees in the dirt as follows:

1. {(4,6),(4,5),(5,2),(2,3),(5,1)}
2. {(4,6),(4,5),(5,2),(2,3),(2,1)}
3. {(4,6),(4,5),(4,3),(5,2),(5,1)}
4. {(4,6),(4,5),(4,3),(5,2),(2,1)}
5. {(4,6),(4,3),(2,3),(5,2),(5,1)}
6. {(4,6),(4,3),(2,3),(5,2),(2,1)}
7. {(4,6),(4,5),(4,3),(2,3),(5,1)}
8. {(4,6),(4,5),(4,3),(2,3),(2,1)}
9. {(4,6),(4,5),(2,3),(5,1),(1,2)}
10. {(4,6),(4,5),(4,3),(5,1),(1,2)}
11. {(4,6),(4,3),(3,2),(2,1),(1,5)}

Rishnak nodded and started to speak, but Ajur asked, "What about the maximum number of labeled spanning trees in a graph with n vertices? For a complete graph, it must be the maximum number of edges in a graph with n vertices."

Rishnak said, "Correct. Complete graphs have the largest possible number of spanning trees. We can find the number of spanning trees in a complete graph using Prüfer codes, which we talked about a few days ago.[1]

Ajur remembered Prüfer codes and said, "For a tree with n vertices, we need a Prüfer code of length $n-2$. Each of the $n-2$ characters in the code

[1]See their discussion in Chapter 9

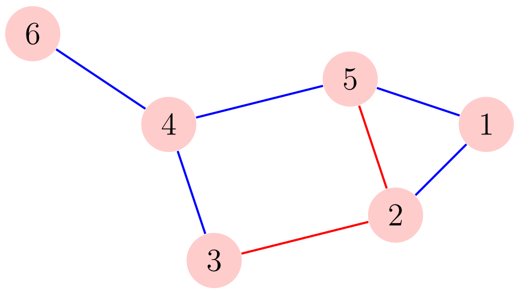


Figure 11.4: Original graph from Figure 11.1 with respect to the spanning tree shown in Figure 11.2; here, spanning tree edges are shown as thick blue lines while non-spanning tree edges are shown in red, the latter forming fundamental cycles $(1, 2, 5)$ and $(3, 4, 5, 1, 2)$

could be any of the n vertices. Therefore, the number of labeled spanning trees in a complete graph with n vertices has to be $n^{(n-2)}$."

Rishnak smiled, then said, "In general, since there are $n-1$ edges in a spanning tree, each of the remaining $e-n+1$ edges, when added to the spanning tree will necessarily create a cycle. Therefore, we will have $e-n+1$ cycles in the given graph. These cycles are called *fundamental cycles* and, as I have just described, each fundamental cycle has exactly one non-spanning tree edge.[2] Let me show you."

Rishnak waved his hands and the original graph appeared with its edges glimmering in red and blue [Figure 11.4]. Rishnak said, "The blue edges are spanning tree edges, whereas the red edges are non-spanning tree edges that each form a fundamental cycle."

Ajur wanted to better understand how to find a spanning tree for a given graph, so he asked Rishnak about this.

Rishnak smiled and said, "First, the graph has to be connected. If that's the case, then it can be done using the following steps:

1. Start from any vertex. Add that vertex to empty set S.
2. From the vertices in S, find a vertex v that is not in S that is adjacent to one of the vertices w in S. Add that edge (v, w) to spanning tree T.
3. Repeat Step 2 until all vertices are in S."

[2]Euler's equation (9.1) back in Chapter 10 uses this fact.

Rishnak formed the original graph again in the air in front of Ajur [Figure 11.1]. He said, "Let me illustrate this. First, add vertex 1 to the set. It has adjacent vertices 2 and 5. Let's say we choose vertex 5 and add it to S. Edge $(1, 5)$ is then the first edge of our spanning tree. From vertices in S, we pick a vertex that is adjacent but not already in S. Choices here are vertices 2 or 4. Let's say we choose vertex 2 and therefore include edge $(1, 2)$ in the spanning tree."

Ajur said, "We could have also included edge $(5, 2)$ instead of $(1, 2)$ in the spanning tree, right?"

Rishnak said, "Yes, the choice is arbitrary. Now, at this point, we have $S = (1, 2, 5)$. So we find a vertex not in S that is adjacent to any of these three vertices. There are two vertices to choose from, namely vertices 3 and 4. Let's say we choose vertex 4 and therefore include edge $(4, 5)$ in the spanning tree."

Ajur chimed in, "Now vertices $(1, 2, 4, 5)$ are in set S. From these vertices, we find a vertex that is adjacent but not in S, so we're looking at vertices 3 and 6. If we choose vertex 3, then we include edge $(4, 3)$ in the spanning tree. Repeating this one more time, we choose vertex 6 and include edge $(4, 6)$ in the spanning tree. And now we're done!"

Rishnak nodded and said, "Right, now set S contains all of the vertices, and we have spanning tree T with edges $(1, 5)$, $(1, 2)$, $(5, 4)$, $(4, 3)$, and $(4, 6)$."—Rishnak waved his hands and the spanning tree from earlier appeared [Figure 11.2]—"This is the same spanning tree we came up with earlier."

Ajur nodded eagerly. He said, "Aha, and if we had made different choices along the way, we would have ended up with a different spanning tree, like this one."—Ajur quickly drew another spanning tree in the dirt [Figure 11.3].

Rishnak smiled, happy to see that Ajur understood. He said, "Here is another interesting problem related to spanning trees. As background, a graph is called a *weighted graph* if there are numeric weights or costs associated with each edge. The weight could represent the distance in miles or the cost of travel, and so on."

Rishnak flashed his hands and the original graph [Figure 11.1] appeared, this time with numeric values next to each edge [Figure 11.5]. He said, "Here is an example weighted graph. See each weighted edge?"

Ajur said, "Yes, but how does this change our spanning tree problem?"

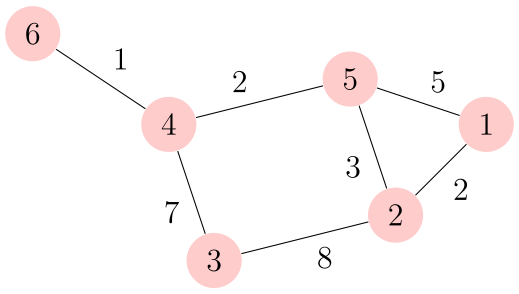


Figure 11.5: The graph shown in Figure 11.1 with edge weights associated with each edge

Rishnak saw Ajur's impatience. He said, "We want to find a *minimum spanning tree* in which the sum of all edge weights in the spanning tree is the smallest possible."

Ajur said, "Aha, we can enumerate all spanning trees and for each such spanning tree, compute the sum of the edge weights, then choose the spanning tree with the smallest sum."

Rishnak smiled and said, "That will work, though that is what we call a *brute force* method. It can be a good strategy for many problems, but we can often find more efficient methods. In this case, we would like to find a faster method that makes certain choices such that we do not have to consider all possible spanning trees."

Ajur frowned and said, "I see. If the graph was much larger, we'd have a lot of work to do if we used a brute force approach."

Rishnak said, "Exactly. Do not despair, though. We can modify Step 2 of the procedure that I already described to better select each edge. Here is the modified approach:

1. Start from any vertex. Add that vertex to empty set S.

2. From the vertices in S, find a vertex v that is not in S that is adjacent to one of the vertices w in S—but make sure that edge (v, w) has the smallest possible edge weight at that decision point. Add that edge (v, w) to minimum spanning tree T.

3. Repeat Step 2 until all vertices are in S."

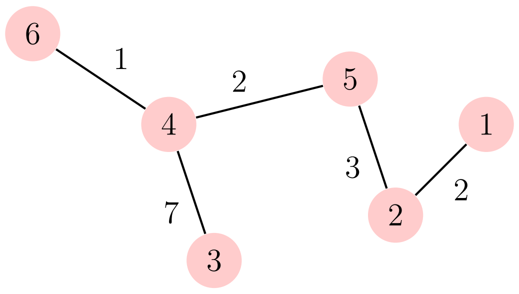


Figure 11.6: A minimum spanning tree with a total weight of 15 for the weighted graph given in Figure 11.5

Ajur decided to work through an example by using the graph that Rishnak had formed in front of him [Figure 11.5]. He first drew the vertices of the graph in the dirt. Next, he chose vertex 1 and added it to set S. From the two possible edges incident at vertex 1, he chose vertex 2 and added it to S since the edge weight for edge $(1, 2)$ was smaller than that of edge $(1, 5)$.

Ajur said, "Aha, I see! Now that vertices 1 and 2 are in S, of the edges leaving S, edge $(2, 5)$ has the smallest edge weight. Therefore, edge $(2, 5)$ is added to the minimum spanning tree and vertex 5 is added to S. Then, from vertices in S, namely 1, 2 and 5, edge $(5, 4)$ has the smallest edge weight. And this continues until we visit each vertex, which we know by adding each vertex in turn to set S."—Ajur drew each edge in the dirt as he followed the algorithm through to its end [Figure 11.6]—"Set S contains all of the vertices, so here's the minimum spanning tree!"

Rishnak laughed as Ajur could barely control his enthusiasm. Ajur exclaimed, "We could also use this algorithm to minimally connect a given set of points in a plane. Vertices would correspond to the points. And the distances between each of these points could simply be the length of the straight lines joining them."

Ajur showed his work in Figure 11.7.

Another ghost named Urpur had been eavesdropping on Rishnak and Ajur. Urpur wanted to show that he was smarter than both Rishnak and Ajur, so he interrupted and said, "I have an even smarter approach. There

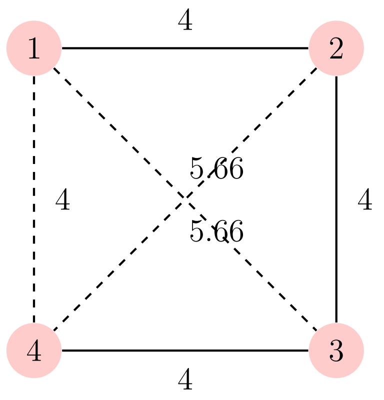


Figure 11.7: A minimum spanning tree representation with a total weight of 12 for a set of four points in the plane; here, edges not part of the minimum spanning tree are shown as dashed lines

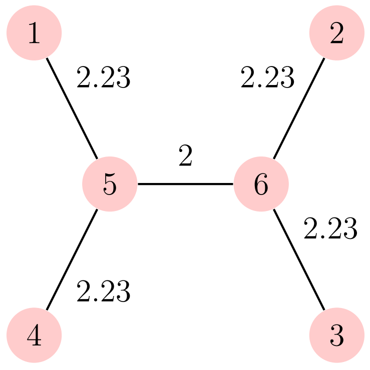


Figure 11.8: A minimum Steiner tree for the set of four points or vertices shown in Figure 11.7; here, Steiner points are vertices 5 and 6

is a spanning tree called the Steiner tree[3] wherein one is allowed to add Steiner points or vertices that were not present in the original problem. With a clever addition of Steiner points, I can find a spanning tree that is better than the one Ajur found."

Urpur clapped his hands and a new graph appeared [Figure 11.8]. Urpur said, "You see?"

Rishnak was impressed with Urpur's solution, but sensing that Ajur was feeling jealous of Urpur, Rishnak said, "Not bad, Urpur. I am sure Ajur would have also come to this solution if I had asked him."

Ajur nodded in agreement.

Question for the ninth day

Rishnak said, "It is time now for Ajur to try and answer the question for the ninth day. It has multiple parts."

Ajur's eyes opened wide in excitement.

Rishnak continued, "Consider a complete graph with with eight vertices and 28 edges. Of these edges, 14 of them have a weight of 1, while the other 14 edges all have a weight of 10. First, how must you assign the weights so as to achieve the smallest possible minimum weight spanning tree? Second, how must you assign the weights so as to achieve the opposite, the largest possible minimum weight spanning tree?"

Before you turn the page, try to come up with an answer of your own!

[3]Named after the Swiss mathematician Jakob Steiner

Answer for the ninth day

Ajur started with the first part. He said, "If there is a cycle that visits all of the vertices—so a cycle of length 8—and each edge weight in the cycle is 1, then the minimum spanning tree has a weight of 7. We can't go smaller than that."

Rishnak smiled and said, "Correct. And for the second part?"

Before Ajur could answer, Urpur said, "The second part is the real question. And the answer is—"

Ajur jumped up and said, "The answer is 25. The largest minimum spanning tree will have a total weight of 25."

Rishnak said, "And how do you know that?"

Ajur said, "All of the edges incident at a vertex must be of weight 10 or else we would select an edge of length 1 instead. Since there are eight vertices, the degree of each vertex must be 7, so two of the graph's vertices can entirely have incident edges of weight 10. That uses up the 14 edge weights of 10, and gives us a minimum spanning tree with a total weight of $10 + 10 + 1 + 1 + 1 + 1 + 1$, which is 25."

Rishnak smiled.

Urpur said, "That's what I was going to say!"

Rishnak laughed and decided to call it a night.

Jura barked. He was happy now to get Ajur's attention and jumping with joy, left with Ajur.

Chapter 12

Shortest Paths

Ajur had found a map of the cemetery. He used it to try to find the best way to go to the water fountain. Then he remembered his recent discussions with Rishnak and excitedly told Jura that by using graph theory methods, they would be able to find a path and maybe even the shortest path to the water fountain.

Rishnak overheard Ajur and realized that a good discussion of paths and shortest paths would be an ideal topic to pursue next.

Rishnak said, “Consider a graph[1] in which you wish to find the shortest path from a specified source vertex to a specified destination vertex. Let’s look at this familiar graph.”—Rishnak waved his hands and an undirected graph appeared in front of Ajur [Figure 12.1]—“What’s the shortest path from vertex 1 to vertex 6?”

Ajur jumped up with excitement and said he could easily draw the shortest path from source vertex 1 to destination vertex 6. He grabbed a stick and drew the graph in the dirt, then he showed the shortest path by thickening edges $(1, 5)$, $(5, 4)$, and $(4, 6)$ [Figure 12.2].

Pleased with Ajur’s enthusiasm, Rishnak wanted to make sure that Ajur understood a general method for finding the shortest path from a source vertex to a destination vertex in any graph.

Rishnak said, “Remember how to find a spanning tree for a given graph?”

Ajur said, “Sure, yes.”

[1]For spanning trees, we consider only undirected graphs; however, for shortest paths we can consider both undirected and directed graphs.

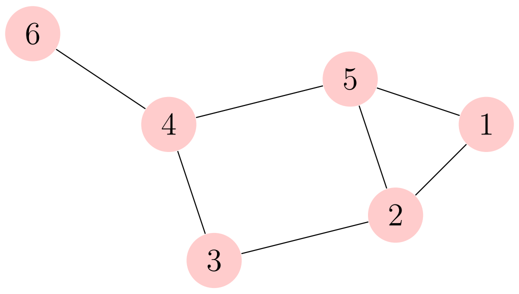


Figure 12.1: Example undirected graph for which we want to find the shortest path from vertex 1 to vertex 6

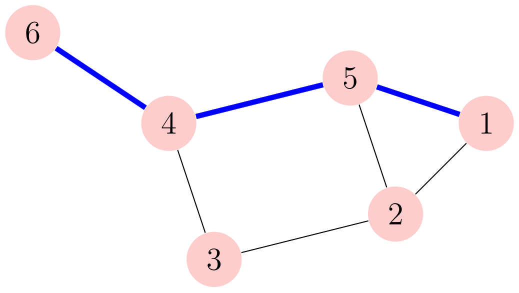


Figure 12.2: The graph from Figure 12.1 with the shortest Path (of length 3) shown from vertex 1 to vertex 6 using thick lines

Rishnak said, "Good. We can come up with a similar algorithm for finding the shortest path from a source vertex to a destination vertex, but first, we need to define distance **dist** of a vertex y as the number of vertices we must visit to get from the source vertex to vertex y. We also define parent **p** of a vertex as the parent vertex[2] that led us to vertex y. Given these definitions, here is the algorithm:

1. Set **dist** for the source vertex to 0 (since the distance from the source vertex to itself is zero). Also set **p** as being undefined.

2. Set both **dist** and **p** for all other vertices as being undefined.

[2]You can also think of this as the *previous* vertex.

3. We start from the source vertex, so add that vertex to a queue.[3]

4. Remove the vertex from the front of the queue, calling this vertex w. Find all vertices with undefined **dist** values that are adjacent to w and put them in set A. Set **dist** for all of these vertices to be one more than the **dist** value for w; also set **p** to be vertex w. Finally, add all vertices from set A to the queue.

5. Repeat Step 4 until the destination vertex has its **dist** value changed, meaning our algorithm has reached the destination vertex. We can then trace the shortest path back to the source vertex by following the **p** vertices until we reach the source vertex."

Ajur knew this was a lot to try to understand. He used the example graph still in front of him [Figure 12.1], tracing the algorithm from source vertex 1. He said, "Okay, so we start by letting the distance **dist** of vertex 1 be 0, then we add vertex 1 to the queue. We remove vertex 1 from the queue. It has adjacent vertices 2 and 5, so the **dist** values of these vertices are both set to 2. And vertices 2 and 5 are added to the queue, say with vertex 2 at the end of the queue. Also, the parent **p** vertices of vertices 2 and 5 are both set to vertex 1."

Rishnak said, "Correct. Keep going."

Ajur went on, "Next, we remove vertex 5 from the queue. The only adjacent vertex with an undefined **dist** value is vertex 4, so we add it to the end of the queue. And we set the **dist** of vertex 4 to be 2 and the parent **p** of vertex 4 to be vertex 5."

Ajur thought for a moment, then said, "The first element of the queue is now vertex 2, which we remove next. Next, we add the only adjacent vertex—vertex 3—to the queue. For vertex 3, we set **dist** to 2 (because it is one more than the (dist) value of vertex 2) and the parent **p** to vertex 2. Finally, we remove vertex 4 from the queue and therefore reach vertex 6, which we add to the queue. We set the **dist** of vertex 6 to be 3 and its parent **p** to vertex 4. And since vertex 6 is the destination vertex, we're done. The distance from vertex 1 to vertex 6 is 3, and we can work backwards to determine the path since $\mathbf{p}(6) = 4$, $\mathbf{p}(4) = 5$, and $\mathbf{p}(5) = 1$, which is the source vertex."

[3]In a queue, we can only add elements to the end and remove elements from the front, just like waiting in line at a grocery store. We can also inspect elements in the queue without changing their order.

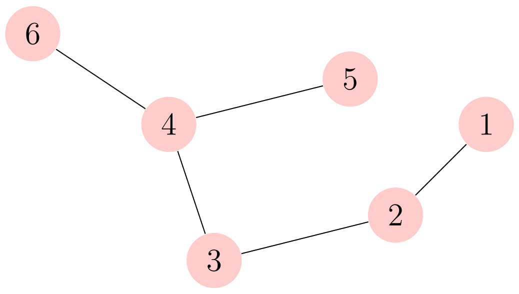


Figure 12.3: An example graph for which we wish to find the diameter

Ajur marveled at the systematic procedure that could be applied to any graph. He also realized that this procedure could be used to find the shortest path from a single source to all other vertices in a given graph. As he thought about this further, he said, "And we could use this same algorithm to find the shortest paths between every pair of vertices in a graph."

Rishnak smiled and said, "There is a graph parameter called the *diameter* of a graph. The diameter is the longest path from among the shortest paths between every pair of vertices. For the example graph we have been using [Figure 12.1], the diameter is 3 since the shortest paths between every pair of vertices are of lengths 1, 2, and 3."

Ajur nodded.

Rishnak flashed his hands and the original graph morphed into a new graph with two fewer edges [Figure 12.3]. He said, "Find the diameter of this graph, Ajur."

Ajur studied the graph and said, "Okay, this is the same graph but without edges $(1, 5)$ and $(2, 5)$. I see that this graph is a tree since there are no cycles in the graph. The longest path among all of the shortest paths is from vertex 1 to vertex 5—no wait, there's also a longest path from vertex 1 to vertex 6. Both of them are of length 4, so that's the diameter of this graph. And therefore, there can be more than one path representing the diameter."

Rishnak nodded and said, "Correct. In general, a simple path with n vertices"—he waved his hands and a new graph appeared [Figure 12.4]—

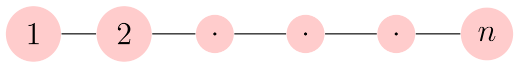


Figure 12.4: A graph with n vertices will always have a diameter of $n-1$

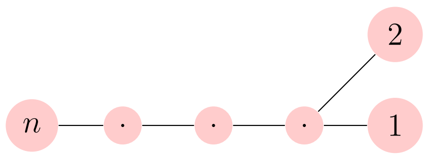


Figure 12.5: A graph with n vertices with a diameter of $n-2$

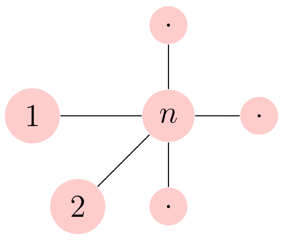


Figure 12.6: A graph with n vertices with a diameter of 2

"will have a diameter of $n-1$. And for a complete graph[4] with n vertices, the diameter is always 1, as the shortest path between every pair of vertices is 1."

Ajur again nodded.

Rishnak asked Ajur, "Can you construct graphs, all with n vertices, that have diameters $n-1, n-2, \ldots, 2,$ and 1?"

Ajur thought about this for a few moments. He said, "Yes, that's not too hard. We already have the graph with a diameter of $n-1$. Here's a graph with a diameter of $n-2$"—he drew a graph in the dirt [Figure 12.5]—"and another with diameter 2."—he quickly drew another graph [Figure 12.6].

Ajur smiled broadly and said, "From these two graphs, you can infer the rest!"

[4]Remember that in a complete graph, there is an edge between every pair of vertices.

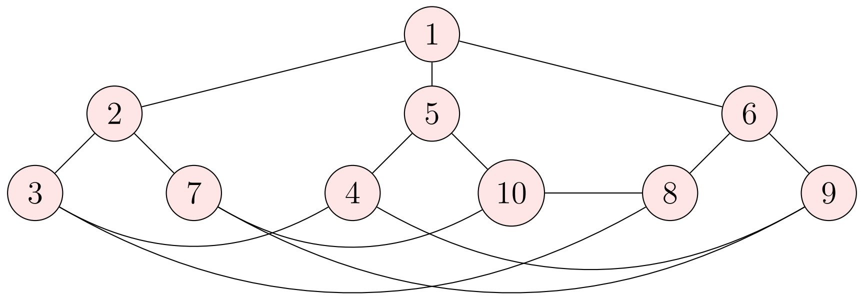


Figure 12.7: The Petersen graph from Figure 7.4 drawn in the shape of a rooted tree with a diameter of 2 and a degree of 3

Rishnak said, “Here is another interesting problem. Can we construct a regular undirected graph with given diameter k?”

Ajur shrugged his shoulders and said, “But how many vertices?”

Rishnak smiled as he continued, “Yes, Ajur, an answer here is a set of graphs called *Moore* graphs. In addition to diameter k, we define degree d for the graph, meaning that each vertex has degree d. Then we can calculate the number of vertices in the graph using this formula.”

Rishnak flashed his hands and the following formula appeared in the air in front of Ajur:

$$\text{number of vertices} = 1 + \sum_{i=1}^{k-1} (d-1)^i$$

Ajur frowned as he studied the formula.

Rishnak said, “Let me explain. We can understood this formula by drawing a graph in the shape of a rooted tree of depth $k - 1$”—Rishnak waved his hands and a new graph appeared [Figure 12.7]—“which we have already seen.[5] In this graph, the root vertex has d children and every other vertex has $d - 1$ children. All of the leaf vertices break the tree properties since they are connected in such a way that every vertex has degree d.”

Ajur studied this graph and said, “Aha, and this graph’s diameter is k.”

Rishnak smiled and said, “Precisely.”

[5]This is the Petersen graph from Figure 7.4.

Ajur thought for a few moments, wondering about weighted graphs. He said, "Is there an algorithm to find shortest paths in weighted graphs?[6]

Rishnak said, "Yes, Ajur, good thinking." Rishnak waved his hands and a weighted graph appeared in the air [Figure 12.8].

Rishnak continued, "We can modify the shortest path algorithm from earlier to work for weighted graphs. Remember from that algorithm that we aim to find the shortest path from a source vertex to a destination vertex. Then we redefine distance **dist** of a vertex y as the minimum sum of the weights along the edges we used to get from the source vertex to vertex y. We still have parent **p** of a vertex as the parent vertex that led us to vertex y. Given these revised definitions, here is the algorithm:

1. Set **dist** for the source vertex to 0 (since the distance from the source vertex to itself is zero). Also set **p** as being undefined. Further, set **explored** to 1.

2. For all other vertices, set **dist** to ∞, set **explored** to 0, and set **p** as being undefined.

3. We start from the source vertex, so add that vertex to a queue.

4. Remove the vertex from the front of the queue, calling this vertex w. Find all vertices with **explored** set to 0 that are adjacent to w and put them in set A. For each vertex $v \in A$, set its **dist** value to the minimum of the **dist** value for v and the sum of the **dist** value for w plus the weight of edge (w, v). If we use the sum here, then also set **p** for vertex v to be w (since we found a shorter path).

5. Add the vertex with the smallest **dist** value from set A to the queue, also setting its **explored** value to 1.

6. Repeat Steps 4 and 5 until the destination vertex has its **dist** value changed, meaning our algorithm has reached the destination vertex. We can then trace the shortest path back to the source vertex by following the **p** vertices until we reach the source vertex."

Ajur studied the graph in front of him, then drew it in the dirt. He said, "Okay, let me try to use the algorithm on this graph. Initially, **dist** of

[6]Remember a weighted graph is a graph with weights or costs associated with each edge.

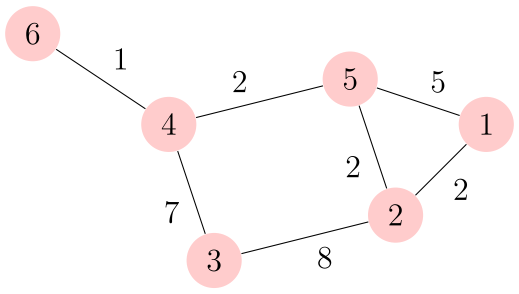


Figure 12.8: A weighted graph with six vertices and seven edges for which we wish to find the shortest path from vertex 1 to vertex 6

vertex 1 is set to 0, and for the rest of the vertices, **dist** is set to ∞. Also, **explore** of vertex 1 is 1 since we have explored that vertex and 0 for the rest of the vertices. Scanning from vertex 1, **dist** of vertex 5 is 5 and **dist** of vertex 2 is 2. So the **p** labels of vertices 2 and 5 are both set to refer back to vertex 1."

Rishnak said, "Good, Ajur, that is a good start, yes."

Ajur continued, "Then we have to choose which vertex to explore next by selecting the adjacent vertex with the smallest **dist** value. That would be vertex 2, so we set **explore** for vertex 2 to 1. Next, we update **dist** of vertex 5 to 4 (since the sum $2+2$ is less than 5) and **dist** of vertex 3 to 10. We also set their **p** values to both be vertex 2. We then select vertex 5 and set its **explore** value to 1. From vertex 5, **dist** of vertex 4 is set to 6 and its parent label is set to vertex 5."—Ajur sighed as this was getting quite tedious—"Okay, then we go to vertex 4 so we set its **explore** label to 1. From vertex 4, **dist** of vertex 6 becomes 7 and its **p** value refers to vertex 4. In this case, **dist** of vertex 3 does not change. So finally, we set **explore** of vertex 6 to 1 and since it is the destination vertex, the algorithm stops."

Ajur stepped back and looked again at the graph he had drawn [Figure 12.9]. He said, "We then know that the shortest path from vertex 1 to vertex 6 is of length 7. The path can be traced backward from vertex 6 through the **p** values, so we have $\mathbf{p}(6) = 4$, $\mathbf{p}(4) = 5$, $\mathbf{p}(5) = 2$, and $\mathbf{p}(2) = 1$, the source vertex."

Rishnak smiled as Ajur appreciated the power of the algorithm, knowing now that he could use this algorithm to compute the shortest route from the cemetery entrance to the water fountain—or anywhere else!

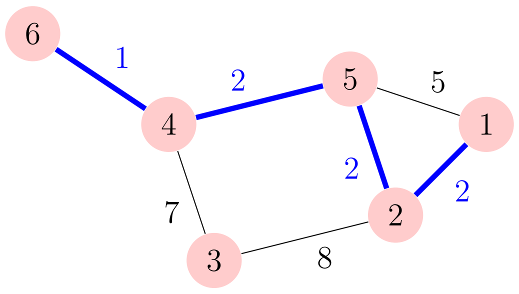


Figure 12.9: The weighted graph from Figure 12.8 with the shortest path from vertex 1 to vertex 6 shown using thick lines

Question for the tenth day

Rishnak said, "It is now time for the question for the tenth day, Ajur. There are two parts to this question. Using the graph you have just seen [Figure 12.9], first, how can you change the weight of a single edge such that the shortest path from vertex 1 to vertex 6 must go through vertex 3? Second, what is the minimum edge weight for the changed edge to still have the shortest path go through vertex 3?"

Before you turn the page, try to come up with answers of your own!

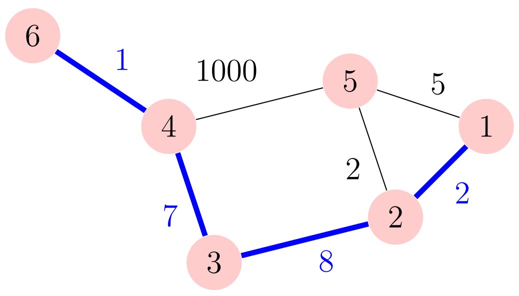


Figure 12.10: The weighted graph from Figure 12.8 with the shortest path from vertex 1 to vertex 6 shown using thick lines, this time going through vertex 3 given the increased edge weight of edge $(5, 4)$

Answer for the tenth day

Ajur thought about changing an edge incident on vertex 1 but quickly found that the shortest path would then simply follow the other vertex (i.e., vertex 2 or vertex 5). After a few moments of pondering, Ajur erased the edge weight for edge $(5, 4)$ and replaced it with a new edge weight of 1000.

Ajur said, "If the weight of edge $(5, 4)$ is increased to 1000, then the shortest path is forced to go through vertex 3." He drew the new shortest path using thick lines to emphasize the path [Figure 12.10].

Rishnak said, "Good, and what could the minimum edge weight be for edge $(5, 4)$ to still have the shortest path go through vertex 3?"

Ajur said quickly, "Easy, the shortest path through vertex 3 has a length of 18, so edge $(5, 4)$ would need to have a weight of 14 for the second-shortest path to be 19."

Rishnak smiled and said, "Yes, Ajur, very good."

Ajur wanted to learn more, but he was getting tired and wanted to find the shortest path from where he stood to a bench on which he could lie down and take a nap. So, along with Jura, Ajur walked off in that direction.

Chapter 13

Cliques, Independent Sets, Vertex Covers

Rishnak was eager to discuss other interesting problems with Ajur and spotted Ajur and Jura walking along the shore of a pond. Rishnak asked Ajur about cliques in his school.

Ajur readily shared his pet peeve with Rishnak. He said, "Yes, students in my school belong to cliques, essentially groups of friends, but they don't always let others join."

Rishnak said, "A clique is also a graph-theoretic term. Here, a *clique* is a subgraph of a graph that is complete, which remember means that every pair of vertices is connected by an edge. Normally we only consider a maximal complete subgraph to be a clique, which is often called a *maximal clique*. We will use that definition for clique, which means that if H is a maximal complete subgraph of graph G, then there is no complete subgraph of G that also contains H as a subgraph."

As Ajur thought about that, Rishnak flashed his hands and a graph appeared in front of Ajur [Figure 13.1]. Rishnak said, "Can you list the cliques—maximal complete subgraphs—in this graph?"

Ajur was unsure of his answer, as he was struggling with all of the definitions. He said, "If I understand correctly, there are three cliques:

1. The induced subgraph containing vertices 4 and 6.
2. The induced subgraph containing vertices 2, 3, 4, and 5.
3. The induced subgraph containing vertices 1, 2, and 5.

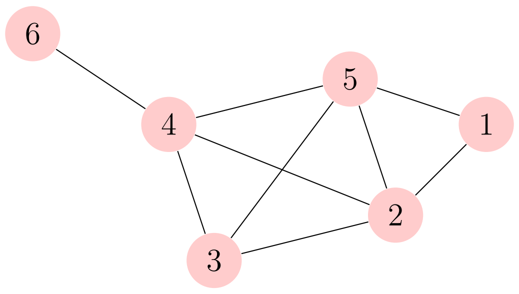


Figure 13.1: A graph with six vertices and nine edges for which we wish to identify the cliques, i.e., maximal complete subgraphs (there are three)

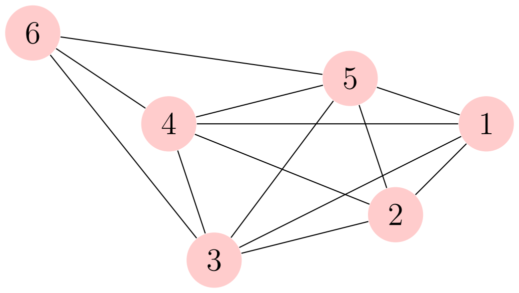


Figure 13.2: The graph shown in Figure 13.1 with four additional edges for which we again wish to identify the cliques (there are two)

Rishnak smiled. He was happy that Ajur understood. He said, “Good, now how about for this graph?” He waved his hands and four edges were added to the original graph to form a new graph [Figure 13.2].

Ajur studied the graph for a few seconds, then answered, “I think there are two cliques:

1. The induced subgraph containing vertices 3, 4, 5, and 6.

2. The induced subgraph containing vertices 1, 2, 3, 4, and 5.”

Rishnak said, “Exactly. Graphs are usually used as models for physical or social processes. For example, when we consider cliques in a school, we can model students as vertices and two students belong to the same group as an edge between the associated vertices, which also note forms a binary relation. Imagine a graph of everyone in your school, with a vertex for each student and an edge for each friendship. Finding cliques or, as in the field of data science, finding clusters is an important task."

Ajur thought about this for a few moments, trying to imagine a graph that would include all of his classmates at school.

Rishnak said, "Let us move on to another related topic. An *independent set* in a graph is a set of vertices that are mutually nonadjacent, which means that we have a subset of vertices that does not include any adjacent vertices. Can you see how an independent set is related to a clique?"

Ajur frowned, frustrated with the lesson so far. At length, Ajur said, "No, I don't understand."

Rishnak also frowned, thinking, "Oh no, if Ajur does not understand and then does not answer my question, I will remain trapped as a ghost forever."

Rishnak said, "Let me give you a hint. A *complement* of a graph $G = (V, E)$ is another graph $H = (V, E_1)$. Both have the same vertex set, but if two vertices are adjacent in G, they are not adjacent in H—and if two vertices are not adjacent in G, then they are adjacent in H. Can you draw the complement of the original graph"—he waved his hands and the four added edges disappeared to form the original graph [Figure 13.1]—"in front of you?"

Ajur looked at the graph. He grabbed a stick and, after scratching his head a few times, drew a new graph in the dirt [Figure 13.3]. He said, "Here is the complement of your graph."

Rishnak said, "Yes, and how many edges will be in edge set E_1 if the original graph has n vertices and e edges?"

Ajur said, "The maximum number of edges a graph with n vertices can have is $\frac{1}{2}n(n-1)$, so then the number of edges in the complement of the graph is simply $\frac{1}{2}n(n-1) - e$."

Rishnak said, "Go on."

Ajur thought for a minute. What was he missing? How did all of these concepts relate to one another?

Rishnak said, "Well, Ajur, do you see it?"

At last, Ajur said, "I do, yes, I see! A clique in a graph G will be an independent set in the complement of G. In your original graph"—he pointed to the graph in the air in front of him [Figure 13.1]—"the maximal independent sets are vertex sets $\{2, 3, 4, 5\}$, $\{1, 2, 5\}$, and $\{4, 6\}$. These are also the cliques!"

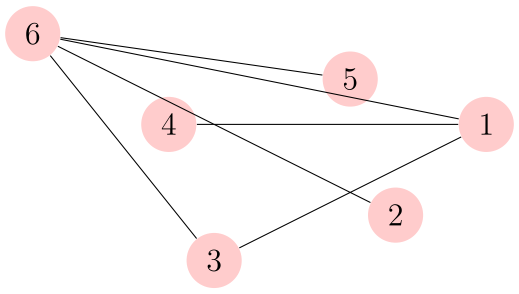


Figure 13.3: The complement of the graph shown in Figure 13.1

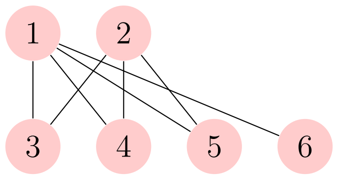


Figure 13.4: A bipartite graph with six vertices and seven edges for which we wish to find all of the maximal independent sets (there are three)

Rishnak smiled. He flashed his hands and a new graph appeared in the air in front of Ajur [Figure 13.4]. Rishnak said, “Here is a bipartite graph, Ajur. Can you figure out what the maximal independent sets are in this graph?”

Ajur studied the graph, then said, “The maximal independent sets are vertex sets $\{1, 2\}$, $\{3, 4, 5, 6\}$”—he hesitated, trying to find if there were any others–“and $\{2, 6\}$. I think that’s all.”

Rishnak nodded. He said, “Finding a maximal independent set is not that hard, but finding a *maximum independent set* is very hard, similar to finding a maximum clique. And what I mean here is this. A maximum independent set refers to the maximal independent set with the largest size. For the graph you still see in front of you [Figure 13.4], the maximum independent set is $\{3, 4, 5, 6\}$. And for this other graph”—Rishnak waved his hands and the previous graph appeared [Figure 13.3]—”the maximum independent set is $\{2, 3, 4, 5\}$.”

Ajur said, “Can we simplify the problem by just looking at a tree?”

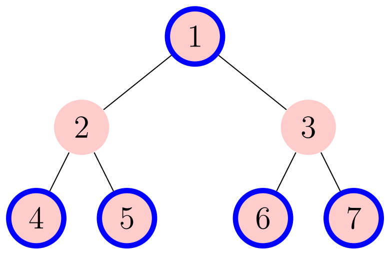


Figure 13.5: A tree for which the maximum independent set is vertex set $\{4, 5, 6, 7, 1\}$, shown as circled vertices

Rishnak raised his eyebrows, pleased with Ajur's question. He said, "Yes, it is much easier to find the maximum independent set in a tree. Can you come up with a procedure to do this?"

Ajur jumped up and thought for a few moments. Before long, he had a solution in mind. He said, "First, the input is rooted tree T and the output is maximum independent set X. The algorithm is:

1. Let $X = \emptyset$.[1]
2. Add all leaf vertices of T to set X.
3. Delete all leaf vertices from T, meaning we also delete all edges incident to each of these leaf vertices.
4. Next, delete all of the newly created leaf vertices, if any.
5. If tree T is not empty, go back to Step 2.
6. X is the maximum independent set of tree T."

Before Rishnak could say a word, Ajur excitedly drew a tree in the dirt [Figure 13.5] and said, "In this tree, the maximum independent set is vertex set $\{4, 5, 6, 7, 1\}$."

Rishnak said, "A problem that is closely related to finding an independent set is finding what is called a *vertex cover*. A vertex cover of a graph is a minimal subset of vertices that are incident to (or cover) all edges of the graph. In other words, if you delete all incident edges in a vertex cover,

[1]Here, $\emptyset$ is the symbol for an empty set, also written as $\{\}$.

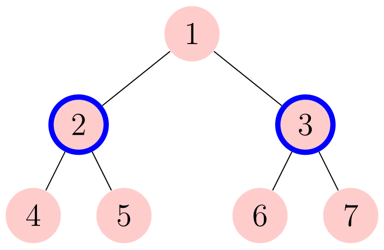


Figure 13.6: The tree from Figure 13.5 for which the minimal vertex cover is vertex set $\{2, 3\}$ (shown as circled vertices), as all edges are incident on either vertex 2 or vertex 3

the remaining graph is always an empty graph, meaning a graph with no edges whatsoever."

Ajur nodded and said, "How is that useful?"

Rishnak said, "Intuitively, if we place police officers at all vertices of the vertex cover, then they will be able to watch all of the edges, which may represent roadways in a city or corridors in a building. And since we want to employ as few police officers as possible, we want a minimal vertex cover, which here means that no subset of it is also a vertex cover."

Ajur said, "Vertex cover and independent set seem to be related. Are they?"

Rishnak smiled and replied, "If X is a maximal independent set in a graph with V as its vertex set, then $V - X$ is a minimal vertex cover. Similarly, if Y is a maximum independent set, then $V - Y$ is a minimum vertex cover."

Ajur smiled, seeing the elegance and simplicity. He also was impressed at how all of these seemingly unrelated sets were actually related. Ajur was lost in thought for awhile, then said, "Maximal independent set X contains no edges, as per its definition. And this implies that vertex set $V - X$ covers all edges of the graph and that every edge is incident in some vertex of $V - X$."

Rishnak said, "In days long ago, parks were constructed at the junctions of roads in a village. Here is an old road map of Royt." Rishnak flashed his hands and a new graph appeared [Figure 13.7]. He continued, "What is the smallest number of parks that would need to be constructed in Royt such that there is at least a park at either end of each roadway or edge?"

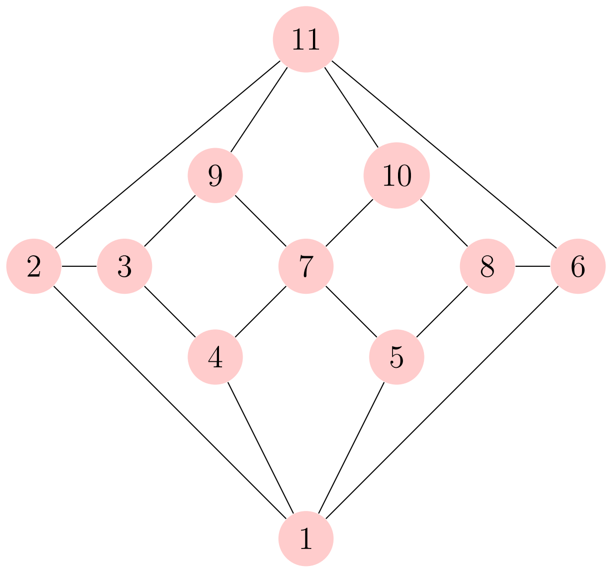


Figure 13.7: An old village road map of Royt

Ajur was mesmerized by the almost ornate layout and symmetry of Royt. He studied the graph and said, "I would construct five parks at vertices 1, 3, 7, 8, and 11."

Rishnak raised his eyebrows and said, "How did you come to that answer so quickly?"

Ajur said simply that he tried to find a minimum vertex cover to solve the problem.

Question for the eleventh day

Rishnak smiled. He said, "You are ready for the question for the eleventh day, Ajur. Here it is. What is the size of the maximum independent set in a complete binary tree of height 5?"

Before you turn the page, try to come up with answers of your own!

Answer for the eleventh day

Ajur thought about the question, repeating it in his head. He said, "Okay, a complete binary tree of height 5 will have 32 leaf vertices. From those leaf vertices, we go up the tree level by level, only counting vertices at alternating levels. So, the size of the maximum independent set will be $32+8+2 = 42$."

Rishnak was impressed, but before he could say good night, Ajur said, "And the minimum vertex cover size is $1 + 4 + 16 = 21$."

Rishnak laughed.

It was getting dark and both of them called it a night.

Chapter 14

Bridges, Cut Vertices, Cut Sets

Remembering that he had not told Ajur about many concepts related to graph connectivity for both undirected and directed graphs, Rishnak went looking for Ajur. He found Ajur and Jura walking across a bridge that spanned a small brook.

Rishnak said, “Today let us talk about graph connectivity.”

Ajur said, “Okay, but already did, and I remember that for a graph to be connected, there has to be a path between every pair of vertices.”

Rishnak smiled and said that was correct. He continued, “A *bridge* is an edge in a connected graph that, when removed from that graph, causes the graph to become disconnected. Have a look at this graph”—Rishnak flashed his hands and a graph appeared in the air [Figure 14.1]—“and you will find that every edge is bridge.”

Ajur said, “I see. Every edge is a bridge because that graph is a tree.”

Rishnak flashed his hands again and a second graph appeared [Fig-

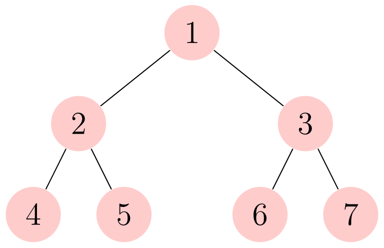


Figure 14.1: In a tree, every edge is a bridge

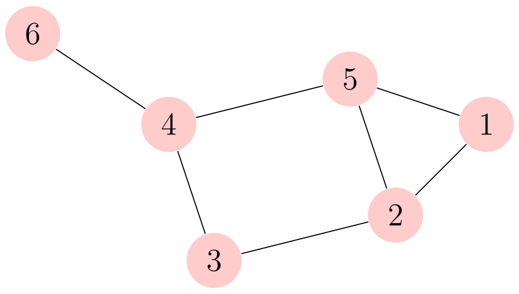


Figure 14.2: A graph in which only one edge is a bridge, i.e., removing edge $(4, 6)$ would cause the graph to be disconnected

ure 14.2]. He said, "And in this familiar graph, only one edge is a bridge."

Rishnak said, "A bridge is an important edge in a graph. As you have seen, if a bridge is removed—in other words, if that edge fails—then the graph becomes disconnected. It is certainly possible for a graph to not contain any bridges. For example, in a complete graph, there are no bridges."

Ajur asked Rishnak whether there was a concept similar to a bridge for vertices.

Rishnak said, "Yes, a *cut vertex* is a vertex that, when removed from a graph[1] causes the graph to be disconnected. In the second graph [Figure 14.2], vertex 4 is a cut vertex."

Ajur studied the graph and said, "I see."

Rishnak continued, "And if you consider the map of the continental United States with states represented as vertices and borders defined by edges, New Hampshire is a cut vertex. If you remove New Hampshire from the graph, it is no longer possible to go from Maine to the other states. Similarly, in India, West Bengal is a cut vertex since its removal from the graph would cause northeastern states such as Assam to be no longer accessible from Kerala!"

Ajur thought for a moment, then said, "In a tree, all vertices other than leaf vertices are cut vertices. And in a complete graph, there are no cut vertices. So if a graph has a cut vertex, that graph is vulnerable and may become disconnected if the cut vertex fails. Like a network with a single point of failure."

[1]Remember that removing a vertex means that all edges incident on that vertex are also removed from the graph.

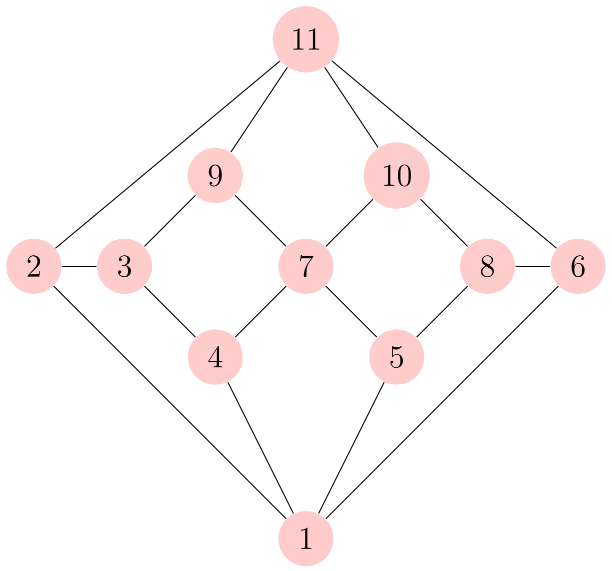


Figure 14.3: The old village road map of Royt for which we wish to determine how many vertices should be removed to cause the graph to be disconnected and also how many edges should be removed to cause the graph to be disconnected

Rishnak said, “Precisely. And further, a graph is k-connected if there is a set of k vertices that, when removed from the graph, causes the graph to become disconnected.”

Ajur realized that the concept of a graph being k-connected was a generalization of a graph having a cut vertex since in that case, $k = 1$.

Rishnak continued, “And similarly, a cut set is a set of edges that, when removed from the graph, causes it to be disconnected. How many vertices need to be removed from this graph”—Rishnak flashed his hands and a new graph appeared [Figure 14.3]—“for it to become disconnected? And what is a cut set for this graph?”

Ajur studied the graph and said, “Removing vertices 1, 3, and 11 would cause the graph to be disconnected since that would isolate vertex 2. And similarly, a cut set is edges $(2, 1)$, $(2, 3)$, and $(2, 1)$.”

Question for the twelfth day

Rishnak said, "Very good, Ajur. Let us see now what the question is for the twelfth day. There are two parts to it. First, what is the largest number of bridges that a connected graph with n vertices, can have? And second, can you draw a graph with six vertices and exactly two bridges.?"

Before you turn the page, try to come up with answers of your own!

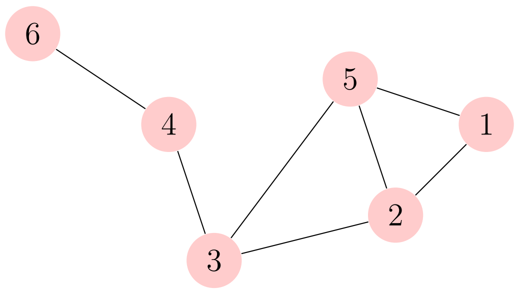


Figure 14.4: A graph with six vertices and two bridges, i.e., edges $(6, 4)$ and $(4, 3)$

Answer for the twelfth day

Ajur thought again about a tree, remembering that a tree is a minimally connected graph. He said, "Since every edge in a tree is in a unique path, all of the edges of a tree are bridges. Therefore, the maximum number of bridges in a connected graph is $n-1$ since this would be a tree of n vertices. And if we instead had n or more edges, then we would have a cycle, which means we would not be able to have the number of bridges be greater than or equal to n."

Rishnak nodded.

Ajur then grabbed a stick and drew a graph in the dirt [Figure 14.4]. He said, "This graph has exactly six vertices and two bridges."

Rishnak smiled and said, "Good work, Ajur."

Chapter 15

Strongly Connected Directed Graphs

Rishnak wanted to continue his discussion with Ajur on the topic of directed graphs[1] and caught up with Ajur and Jura.

Rishnak asked Ajur if he remembered what relations were.

Ajur said, "I think so, yes. A relation describes how two things interact with one another, so a relation could be symmetric like a friendship—if person A is a friend of person B, then person B is also a friend of person A. This would be an undirected graph."

Rishnak nodded and said, "Right, and a relation can also be asymmetric, like a follower on social media—if person A follows the feed of person B, it might not be the case that person B follows person A—and this would be a directed graph. There are many relations that are asymmetric, for example less than or precedence relations, as well as parent or winner relations. In each case, we can model the asymmetric relation as a directed graph."

Ajur nodded, understanding thus far.

Rishnak continued, "Similar to a path or a walk in an undirected graph, we can also define a path or a walk in directed graph. Remember an undirected graph is connected if there is a path between every pair of vertices. A directed graph is said to be *strongly connected* if there is directed path between every pair of vertices. Here"—Rishnak flashed his hands and a graph appeared [Figure 15.1]—"look at this graph."

[1]Sometimes directed graphs are called *digraphs*

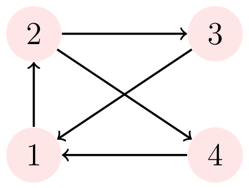


Figure 15.1: Strongly connected directed graph for which there is a path between every pair of vertices

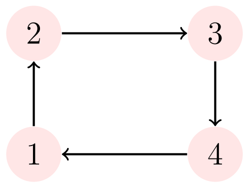


Figure 15.2: Strongly connected directed graph with a minimal number of edges for which there is a path between every pair of vertices

Ajur said, “From any vertex, we can follow one or more directed edges to get to any other vertex in the graph.”

Rishnak said, “Exactly. Can you construct a directed graph with four vertices that is strongly connected with a minimal number of edges?”

Ajur thought for a moment, grabbed a stick, and drew a graph in the dirt [Figure 15.2]. He said, “Sure, this graph is strongly connected with only four edges.”

Pleased, Rishnak said, “Precisely. A strongly connected directed graph must have at least n edges, but we can have a directed graph that is not strongly connected even with $\frac{1}{2}n(n-1)$ edges.”—Rishnak waved his hands and a new graph appeared [Figure 15.3]—“How many edges do we need to add to this graph to make it strongly connected?”

Ajur thought for a minute, then said, “Just one edge. If we add an edge from vertex 4 to vertex 1, then the directed graph would become strongly connected since we could get from any vertex to any other vertex.”

Rishnak smiled. Probing further, he asked, “In this first graph [Figure 15.1], how many edges must we remove to make it not strongly connected?”

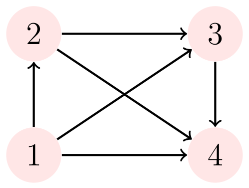


Figure 15.3: A directed graph that is not strongly connected

Ajur quickly said, "Again just one edge. If the edge from vertex 1 to vertex 2 was removed, then the directed graph would no longer be strongly connected since no vertex is reachable from vertex 1."

Rishnak's smile broadened. He said, "Good. This is an important concept to master, since edge removal problems relate to edge failures in a wide variety of problems that we can model using graphs."

Rishnak paused for a moment, then said, "The *transitive closure* of directed graph $G = (V, E)$ is directed graph $H = (V, E_1)$ such that there is a directed edge in H between two vertices u and v if there is a directed path from vertex u to vertex v in G."

Ajur could not contain his excitement as he said, "I see. And the transitive closure of a strongly connected graph will always be a complete directed graph. Let me draw the graph."—Ajur hurriedly drew a new graph in the dirt [Figure 15.4]—"Here is the transitive closure of both of the graphs you have shown me."

Rishnak said, "And how do you reason that?"

Ajur said, "Two key points:

1. In the original directed graph, there is a directed path between every pair of vertices. We know this because the graph is strongly connected.

2. In the transitive closure graph, there is an edge between every pair of vertices since there is a path between every pair of vertices in the original graph."

Rishnak nodded.

Ajur continued, "For example, on a social media page, if I post a message, it's visible to my friends. And if my friends share that post, then that post will become visible to all friends of that friend. That's how transitive

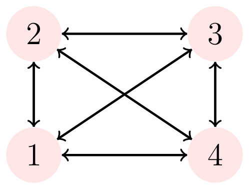


Figure 15.4: The transitive closure of the directed graphs shown in Figure 15.1 and Figure 15.2

Game Results				
Country Name	Played	Won	Lost	Drawn
Scotland	3	3	0	0
England	3	1	1	1
Wales	3	1	1	1
Ireland	3	0	3	0

Table 15.1: Results of six football (soccer) matches

closure works. If someone posts a message, it quickly will be seen by almost everyone!"

Rishnak smiled and said, "Yes, exactly. Here is a similar problem, this one from one of my favorite books by Henry Ernest Dudeney.[2] Problem 452 states that we have four teams, Scotland, England, Wales, and Ireland. This table of data"—Rishnak flashed his hands and the data appeared in front of Ajur [Table 15.1]—"shows the results of their football matches, well soccer I mean. Can you model this problem using a directed graph?"

Ajur thought for a bit and was able to draw a directed graph corresponding to this table [Figure 15.5]. He said, "Scotland won all of its matches, while Ireland lost all of its matches. And therefore, Wales and England must have come to a draw."

Rishnak said, "Scotland defeated England with a 3-0 record. The goals for and against are shown in this table"—Rishnak waved his hands and a new table of data appeared [Table 15.2]—"Can you figure out all of the scores of the rest of the five games?"

[2]The book is "536 Puzzles and Curious Problems."

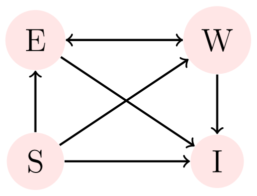


Figure 15.5: A directed graph corresponding to the football (soccer) matches shown in Table 15.1, with S representing Scotland, E representing England, W representing Wales, and I representing Ireland

Goals		
Country Name	For	Against
Scotland	7	1
England	2	3
Wales	3	3
Ireland	1	6

Table 15.2: Goals for and against for each team

Ajur frowned. He could figure out the scores for England immediately, seeing that England defeated Ireland by a score of 2-0 and came to a 0-0 draw with Wales. He spent many long minutes trying various combinations until he finally found a solution. He said, “Aha, Scotland defeated Wales by a score of 2-1 and defeated Ireland by a score of 2-0. These matched up with the goals for and against Scotland and England. So then Wales defeated Ireland by a score of 2-1. Here, let me write down the results.” He drew the scores of the six games in the dirt [Table 15.3].

Rishnak said, “Good, Ajur. And you should see if you can relate these data to a directed graph. For now, here is one more new concept. A directed graph is acyclic if there are no directed cycles in the graph. Such a graph is called a *directed acyclic graph* or *DAG* for short. Can you draw one?”

Ajur drew a DAG in the dirt [Figure 15.6].

Rishnak said, “A DAG may contain what look like undirected cycles, but it remains a DAG as long as there are no directed cycles.” Rishnak waved his hands and the graph that Ajur drew appeared in the air. Rishnak said, “I can add more directed edges”—Rishnak added three more edges

Game Scores		
Country Name	Country Name	Score
Scotland	England	3-0
Scotland	Wales	2-1
Scotland	Ireland	2-0
England	Wales	0-0
England	Ireland	2-0
Wales	Ireland	2-1

Table 15.3: Scores of the six football (soccer) games

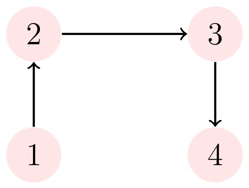


Figure 15.6: A directed acyclic graph with four vertices and three edges

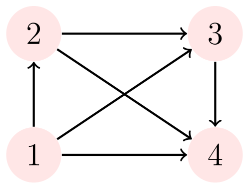


Figure 15.7: Another directed acyclic graph

[Figure 15.7]—"and still this graph is a DAG since it is still acyclic."

Question for the thirteenth day

Rishnak said, "Let me now ask you the question for the thirteenth day. Can you add one edge to make the directed graph you drew in the dirt [Figure 15.6] strongly connected?"

Before you turn the page, try to come up with an answer of your own!

Answer for the thirteenth day

Ajur responded immediately, “Sure, that’s easy. We can just add a directed edge from vertex 4 to vertex 1 to make the directed graph strongly connected.”

It was getting dark and Jura and Ajur wanted to finish their stroll. Rishnak bade them good night.

Chapter 16

Matching and Assignment Problems

Hearing footsteps behind them, Ajur and Jura turned and saw Rishnak rushing to catch up with them.

Rishnak wasted no time and said, "The edges analogue of a maximal independent set, which remember is a set of non-adjacent vertices, is called a *maximal matching*. A maximal matching is a set of edges that do not share any common end vertices. Here is an example with maximal matching edges shown in red."

Rishnak flashed his hands and a new graph appeared in front of Ajur [Figure 16.1], four of its edges glimmering in red.

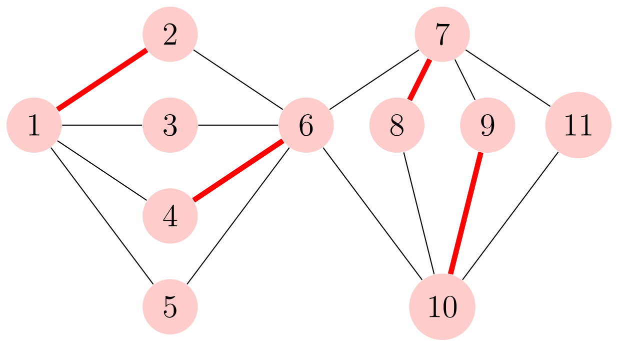


Figure 16.1: A graph for which maximal matching edges are shown in red

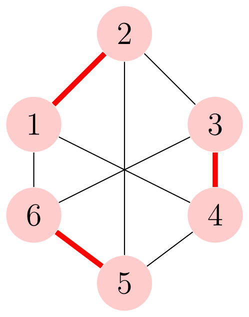


Figure 16.2: A graph with a perfect matching with edges of the perfect matching shown in red

Rishnak continued, “This is a maximal matching. Every edge is incident at one of the end vertices of the edges in the maximal set. Therefore, vertices $\{1, 2, 4, 6, 7, 8, 9, 10\}$ form a vertex cover.”

Ajur said, “But that vertex cover is not the minimum vertex cover.”

Rishnak said, “You are sharp today, Ajur. Right, the minimum vertex cover is $\{1, 6, 7, 10\}$. If we find a maximal matching, we at least know the upper bound on the size of the minimum vertex cover for that graph.”

Ajur nodded.

Rishnak said, “A *perfect matching* is a maximal matching in which all of the vertices are the end vertices of the edges in the matching. For example, the graph already in front of you [Figure 16.1] does not have a perfect matching, but this graph”—Rishnak waved his hands a new graph appeared [Figure 16.2]—“does have a perfect matching since all of the vertices in the graph are end vertices of the edges in the perfect matching.”

Rishnak smiled as he went on. He said, “Here is an interesting problem related to matching that actually earned Professors Lloyd Shapley and Alvin Roth the Nobel prize in Economics in 2012. First, in the 1960s, Professors David Gale and Shapely introduced a problem called the stable marriage problem, which I will describe in a moment. Professor Roth applied this to a variety of applications, earning them the Nobel.”

Ajur raised his eyebrows, eager to hear more.

Rishnak said, “The stable marriage problem may be stated as follows. Given n males and n females and a list of preferences for each male and female as to whom they would like to marry, we want to find stable marriages

Male Name	First	Second	Third
Al	Carla	Ada	Bea
Bob	Ada	Bea	Carla
Caleb	Ada	Carla	Bea

Table 16.1: Male preferences in the stable marriage problem

Female Name	First	Second	Third
Ada	Al	Bob	Caleb
Bea	Caleb	Al	Bob
Carla	Al	Caleb	Bob

Table 16.2: Female preferences in the stable marriage problem

for everyone in the entire group. A marriage is unstable if there is a pair of individuals who are not married to one another but prefer one another over their respective spouses. Otherwise, the marriage is stable."

Ajur rolled his eyes and said, "Wow, what a weird problem."

Rishnak continued, "Okay, let us consider the preferences for males in this table"—he flashed his hands and showed a table of data [Table 16.1]—"and for females in this table"—with a wave of his hand, Rishnak produced a second table [Table 16.2].

Rishnak said, "Given these preferences, we can determine that the following marriages are stable: (Al, Carla), (Bob, Ada), and (Caleb, Bea). We know this because there are no unstable pairs; however, marriages (Al, Carla), (Bob, Bea), and (Caleb, Ada) are unstable marriages since Bob prefers Ada over Bea and Ada prefers Bob over Caleb."

Ajur said, "I see."

Rishnak continued, "Gale and Shapley showed that there is always a stable marriage pair for any set of preferences."

Ajur raised his eyebrows again. He asked, "How?"

Rishnak said, "The algorithm they developed is rather elegant. Here it is:

1. In the first round, each male proposes to the female he prefers the most.

2. Each female replies with a 'maybe' to the suitor she most prefers and a 'no' to all other suitors. She is temporarily engaged to the male she most prefers thus far.

3. In the subsequent round, each non-engaged male proposes to the most-preferred female[1] to whom he has not yet proposed, regardless of whether this female is already engaged.

4. Each female replies 'maybe' if she is currently not engaged or if she prefers this male over her current provisional partner. In this case, she rejects her current provisional partner who then becomes non-engaged. The provisional nature of engagements preserves the right of an already engaged female to essentially trade up and, in the process, jilt her until-then partner.

5. We repeat Steps 3 and 4 until everyone is engaged."

Ajur frowned as this was a lot to understand. He decided to apply this procedure to the data still in front of him [Table 16.1] [Table 16.2]. He said, "Okay, after the first round, Al is temporarily engaged to Carla and Bob is temporarily engaged to Ada since both Bob and Caleb opt for Ada, but Ada prefers Bob over Caleb. In the next round, Caleb proposes to Carla, who is temporarily engaged to Al. Since Carla prefers Al over Caleb, Caleb's offer is rejected. Next, Caleb proposes to Bea, his last choice, and Bea accepts Caleb since she does not have a suitor thus far. Then the algorithm terminates with the stable marriage pairs of (Al, Carla), (Bob, Ada), and (Caleb, Bea)."

Rishnak smiled and said, "Precisely."

Ajur said, "This algorithm sounds like a messy game of musical chairs."

Rishnak laughed and said, "Another application of the matching problem is the optimal assignment problem. Here, we need to assign jobs to a group of workers, with the cost required for each worker to do each job as a given. We wish to find an assignment of workers to jobs so as to minimize the total cost."

Ajur thought of one of his favorite types of graphs. He said, "We can use a bipartite graph with workers in one partition and jobs in the other partition."

[1]Each male starts from his most preferred female and proceeds through to his least preferred female.

	Salesman	Tutor	Chef
Al	17	40	45
Bob	23	60	35
Caleb	21	40	25

Table 16.3: Cost matrix in which rows represent workers and columns represent the jobs they can do; the numeric values show the cost for each worker to do each job

Rishnak said, "Yes, an edge from worker i to job j has the cost of worker i doing job j. Let us assume that the number of workers is equal to the number of jobs—this can be n."

Rishnak flashed his hands and a new table of data appeared in front of Ajur [Table 16.3]. Rishnak said, "Here is a cost matrix showing how much each job would cost for our three workers, Al, Bob, and Caleb."

Ajur studied the data.

Rishnak said, "For the optimal assignment problem, we want to assign jobs to minimize the total overall cost. Here is an algorithm we can use:

1. For each row of the matrix, circle the smallest element and subtract that from every other element in the same row.

2. Also perform Step 1 for all of the columns.

3. Cover all zeros in the matrix using a minimum number of horizontal and vertical lines. In other words, cross these zeros out, which might also cover non-zero values.

4. If the minimum number of lines we draw in Step 3 is n, an optimal assignment is possible and we are done. Otherwise, proceed to Step 5.

5. Determine the smallest entry not yet covered by any line. Subtract this entry from each uncovered row, then add it to each covered column. Next, erase all of the lines and return to Step 3 to perform the crossing out step again."

Ajur again studied the data. He said, "Let me try this out. And $n = 3$ for this problem." He copied the table in the dirt and performed the first

$$\begin{bmatrix} 0 & 23 & 28 \\ 0 & 37 & 12 \\ 0 & 19 & 4 \end{bmatrix}$$

Table 16.4: The table for the optimal assignment problem after Step 1

$$\begin{bmatrix} 0 & 4 & 24 \\ 0 & 18 & 8 \\ 0 & 0 & 0 \end{bmatrix}$$

Table 16.5: The table for the optimal assignment problem after Step 2

$$\left[\begin{array}{c|cc} 0 & 4 & 24 \\ 0 & 18 & 8 \\ \hline 0 & 0 & 0 \end{array}\right]$$

Table 16.6: The table for the optimal assignment problem after Step 3

step, subtracting the smallest element from every other element in the same row [Table 16.4]. Ajur quickly went on to Step 2, subtracting out the smallest value in each column [Table 16.5].

Ajur said, “Okay, let me now cross out the zeros in as few horizontal and vertical lines as possible.” He drew two lines [Table 16.6].

Ajur frowned and said, “Since there are two lines and $n = 3$, we go from Step 4 to Step 5, taking the smallest entry from among the uncovered entries—so that would be 4—and subtracting it from each uncovered row.” Ajur scribbled in the dirt, updating the first two rows [Table 16.7].

He said, “Wait, now I need to also add it to each covered column”—he did so [Table 16.8]—“and erase all of the lines, then go back to Step 3.”

Ajur drew three vertical lines [Table 16.9]. He said, “Now we have exactly three lines covering all of the zeros, so we are done. Ignoring the lines, we can find the assignments to be Caleb as Chef, Al as Tutor, and

$$\begin{bmatrix} -4 & 0 & 20 \\ -4 & 14 & 4 \\ 0 & 0 & 0 \end{bmatrix}$$

Table 16.7: The table for the optimal assignment problem after the subtraction in Step 5

$$\begin{bmatrix} 0 & 0 & 20 \\ 0 & 14 & 4 \\ 4 & 0 & 0 \end{bmatrix}$$

Table 16.8: The table for the optimal assignment problem after Step 5

$$\begin{bmatrix} 0 & 0 & 20 \\ 0 & 14 & 4 \\ 4 & 0 & 0 \end{bmatrix}$$

Table 16.9: The table for the optimal assignment problem after Step 4 in which the algorithm stops

Bob as Salesman. This gives us a total cost of $25 + 40 + 23 = 88$."

Rishnak said, "Good, Ajur."

Ajur said, "But how is this problem related to the perfect matching problem?"

Rishnak was patient and explained the relation between these two problems. He said, "The jobs and workers can be thought of as vertices of a bipartite graph, with one set of vertices being the workers and the other set being the jobs. If we assume that both sets of vertices have the same size, we can construct a complete bipartite graph, meaning there is an edge from every worker to every job. Remember that if a job j has cost w for worker i, then the corresponding edge (i, j) gets assigned a weight of w."

Ajur followed Rishnak so far.

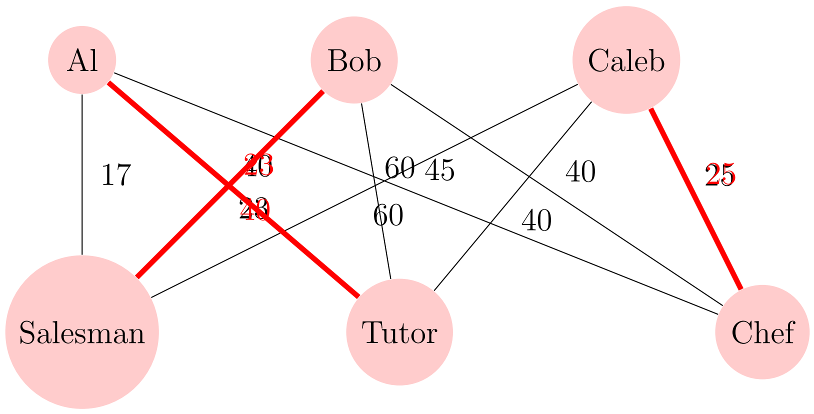


Figure 16.3: Complete bipartite graph corresponding to Table 16.3 with weights representing the costs and the minimum perfect matching edges shown in red

Rishnak continued, “For example, the table still in front of you [Table 16.3] can be represented by this complete weighted bipartite graph.” Rishnak flashed his hands and a new graph appeared [Figure 16.3].

Ajur studied the graph, seeing how it related to Table 16.3]. He said, “And the perfect matching problem applies to this graph then?”

Rishnak said, “Yes, to solve the optimal assignment problem, we can simply find the minimum perfect matching in this bipartite graph representation.”

Ajur smiled, satisfied that all of these problems had interesting connections.

Male Name	First	Second	Third
M1	W1	W3	W2
M2	W3	W2	W1
M3	W3	W1	W2

Table 16.10: Male preferences

Female Name	First	Second	Third
W1	M2	M1	M3
W2	M2	M3	M1
W3	M1	M2	M3

Table 16.11: Female preferences

Question for the fourteenth day

Seeing Ajur was tired, Rishnak said, "Here we are, Ajur, ready for the questions for the fourteenth day." Rishnak flashed his hands and two tables appeared [Table 16.10] [Table 16.11].

Rishnak said, "These two tables show the preferences for males and females. Assume already that $M2$ and $W1$ are temporarily engaged. If $M3$ approached $W1$ with a proposal, what would $W1$ do? Next, flipping this around, assume $W1$ is married to $M2$. Who would $W2$ approach then?"

Before you turn the page, try to come up with answers of your own!

Answer for the fourteenth day

Ajur studied the data and walked through Rishnak's questions in his head. At length, he said, "$W1$ will reject the proposal from $M3$ since $W1$ prefers $M2$ over $M3$. And if women are proposing, $W2$ will approach $M2$ first."

Rishnak nodded.

Ajur was deep in thought, more interested in why and how these procedures work.

Seeing Ajur's puzzled look, Rishnak said, "Do not fret, Ajur. I myself had not understood these things very well, but you will learn this all in college."

This seemed to be okay with Ajur. He nodded and left with Jura.

Chapter 17

Graph Operations

Rishnak found Ajur walking with Jura. Not wasting a moment, Rishnak started the session. He said, "Today we will discuss various graph operations."

Eager to learn, Ajur asked, "Are they similar to arithmetic operations like addition, subtraction, and multiplication that operate on numbers? Or set operations such as union, intersection, and complement?"

Rishnak said, "There are many binary operations[1] and since graphs are represented as sets, many of the operations are similar to set operations. Each graph operation typically generates another graph. We can also see how graphs evolve. Let us walk through a few of these operations."

Rishnak described the following binary graph operations, flashing his hands to form figures as he went on:

1. *Graph union:* Let graph $G_3 = (V_3, E_3)$ denote the union of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Then $V_3 = V_1 \cup V_2$ and $E_3 = E_1 \cup E_2$. Naturally, graph G_3 is not connected.

2. *Graph complement:* The complement of graph $G = (V_1, E_1)$ [Figure 17.1] is Graph $H = (V_2, E_2)$ [Figure 17.2] if $V_2 = V_1$ and $E_2 = \{e | e \notin E_1\}$.

3. *Vertex addition:* Vertex addition applied to graph $G_1 = (V_1, E_1)$ [Figure 17.1] produces graph $G_2 = (V_2, E_2)$ [Figure 17.3], where $V_2 = V_1 \cup \{x | x \notin V_1\}$ and $E_2 = E_1 \cup \{(x, y) | x \in V_2 \text{ and } y \in Y \subset V_1\}$.[2]

[1]Here, binary means operating on two operands.

[2]Barabassi-Albert used a version of this to generate graphs. In the Barabassi-Albert

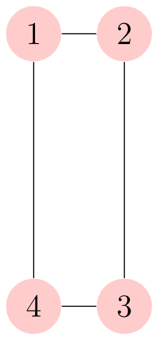


Figure 17.1: Example graph for which we wish to apply the union and complement operations, as well as vertex addition

4. *Cartesian product:* The Cartesian product $G_1 \square G_2$ is graph G_3 such that the vertex set of G_3 is Cartesian product $V(G_1) \times V(G_2)$ and two vertices (u, u') and (v, v') are adjacent in G_3 if and only if either $u = v$ and u' is adjacent to v' in G_2 or $u' = v'$ and u is adjacent to v in G_1. [Figure 17.4] [Figure 17.5] [Figure 17.6]

5. *Line graph:* Given graph G, the line graph of G, denoted by $L(G)$, is a graph with each vertex of $L(G)$ representing an edge of G and two vertices in $L(G)$ being adjacent if the corresponding edges share a common end vertex. [Figure 17.7] [Figure 17.8]

Ajur listened intently at these graph operations, at times asking Rishnak to repeat the definition.

After some time, Rishnak said, “The Petersen graph they we discussed before is actually the complement of the line graph of complete graph K_5.”

Ajur tried to keep up. He knew that graph K_5 was a complete graph with 10 edges. He also knew that the Petersen graph had 10 edges. Did they match? Ajur calculated that the line graph of K_5 was a regular graph of degree 6 and therefore its complement would be a regular graph of degree 3, so that much matched. But still, he had to verify Rishnak’s assertion in full, so he decided to work on it later.

model, the new vertex adds an edge to an existing vertex with a probability related to the degree of that vertex to generate graphs.

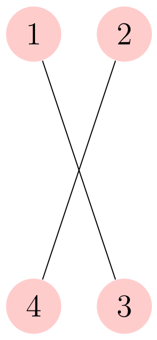


Figure 17.2: Complement of the graph shown in Figure 17.1

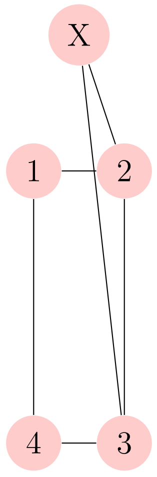


Figure 17.3: Adding a vertex and edges from that vertex to some subset of vertices in the graph shown in Figure 17.1

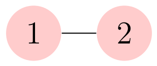


Figure 17.4: Example graph G_1 for which we wish to apply the Cartesian product

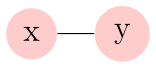


Figure 17.5: Example graph G_2 for which we wish to apply the Cartesian product

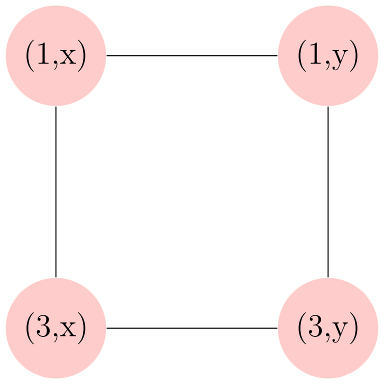


Figure 17.6: The Cartesian product G_3 of graphs G_1 [Figure 17.4] and G_2 [Figure 17.5]

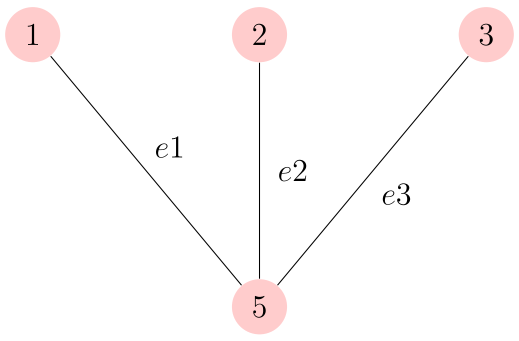


Figure 17.7: Example graph for which we wish to apply the line graph operation

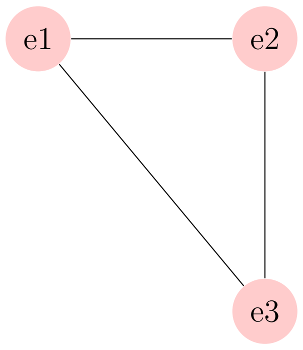


Figure 17.8: A line graph for the graph shown in Figure 17.7

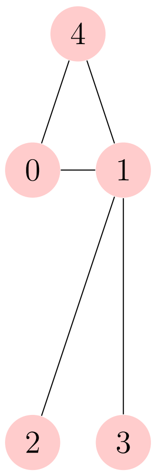


Figure 17.9: Randomly generated graph using the Erdős–Rényi method with $n = 5$ vertices, $e = 5$ edges, and edge probability 0.5

Realizing that it was already getting late, Rishnak said, “Let me also talk about graph construction. There is a method called the Erdős–Rényi model[3] for constructing graphs. In this method, for a graph with n vertices and e edges, for every unordered pair of vertices, an edge is present with the following probability:

$$\frac{e}{\frac{1}{2}n(n-1)}$$

Rishnak flashed his hands and a new graph appeared [Figure 17.9]. He said, “This graph is generated using the Erdős–Rényi model with $n = 5$ and $e = 5$, so our edge probability is $\frac{5}{10} = 0.5$.[4]

Question for the fifteenth day

Rishnak said, “For the question for the fifteenth day, can you draw the complement of the line graph of complete graph K_5?”

Before you turn the page, try to come up with an answer of your own!

[3]Named after mathematicians Paul Erdős and Alfréd Rényi.

[4]When Rishnak wrote this down later, he generated the graph using Sage Math (CoCALC) at `https://cocalc.com/`. You should try to generate a few graphs using this method.

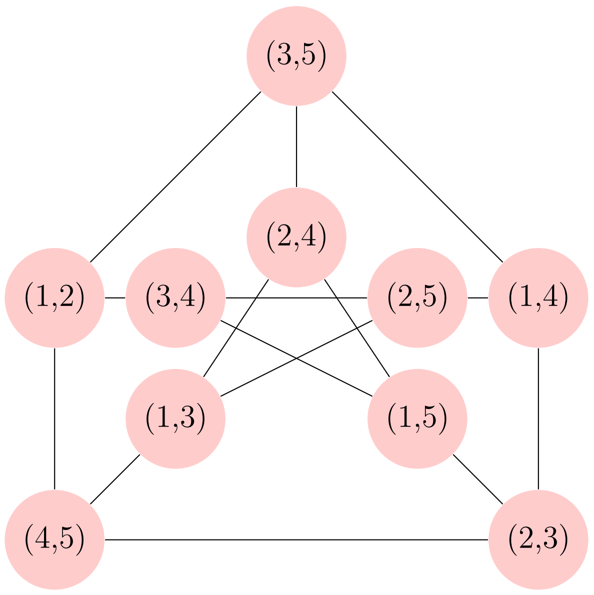


Figure 17.10: The graph representing the complement of the line graph of complete graph K_5; this graph is the Petersen graph

Answer for the fifteenth day

Ajur remembered from earlier that he reasoned out that there would be 10 vertices in the line graph since K_5 has 10 edges. He said, "The complement will also have 10 vertices and will be a regular graph of degree 3 because the line graph will be a regular graph of degree 6."

Ajur drew his answer in the dirt [Figure 17.10].

Ajur needed time to digest all of the new material from the day, so he and Jura drifted away as Rishnak proceeded to meet his other ghost friends.

Chapter 18

Dominating Set

Rishnak was pleased to find Ajur and Jura, who were on their way to the water fountain. Rishnak told them that placing a water fountain is an interesting graph-theoretic problem.

Ajur shrugged his shoulders and said, "How so?"

Rishnak said, "Let me explain by starting with a new concept. A *dominating set* of graph $G = (V, E)$ is a subset of vertices, say D, such that every vertex in $V - D$—this means a vertex $x \in V$ and $x \notin D$—is adjacent to some vertex in D."

Ajur frowned, not yet understanding.

Realizing his definition was a bit dense, Rishnak waved his hands and a graph appeared [Figure 18.1]. He said, "The vertices in dominating set D are shown in orange."

Ajur studied the graph.

Rishnak continued, "You might think that the dominating set and vertex cover problems seem similar, but they are not the same. In a vertex cover, every edge is incident on one of the vertices in the vertex cover, but in a dominating set, not every vertex is involved. So in general, we want to find the minimum dominating set."

Ajur started to see the connection between placing a water fountain and the dominating set.[1]

Rishnak flashed his hands an a familiar graph appeared [Figure 18.2]. He said, "Here is a street map of Royt. Where would you place water

[1]Do you see the connection yet?

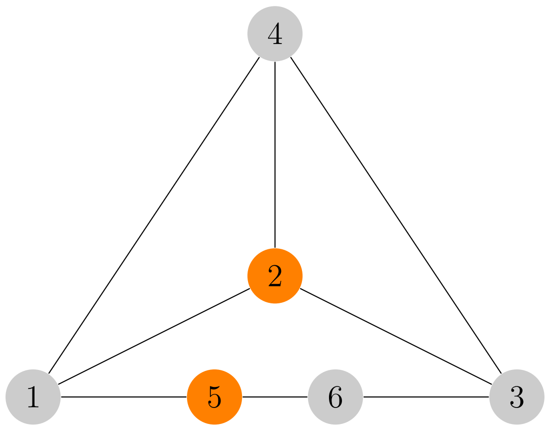


Figure 18.1: A graph in which vertices in dominating set D are shown in orange; each vertex that is not orange is adjacent to one of the orange vertices

fountains with the condition that every junction or corner either has to have a water fountain or has to be adjacent to a water fountain?"

Ajur studied the graph. He rolled the problem around in his, saying out loud, "So we assume people will always walk at most one edge to reach a water fountain."

At length, Ajur drew the graph in the dirt and marked the vertices with petals from an orange flower [Figure 18.3]. He said, "This is the same as finding the dominating set."

Rishnak smiled. He said, "Excellent work. Now let me introduce the concept of a *total dominating set*, which is a variation of the dominating set problem. Total dominating set D_T of graph $G = (V, E)$ is a set of vertices for which every vertex $v \in V$ is adjacent to some vertex in D_T. The dominating set $\{2, 6, 7\}$ that you came up with for our map of Royt is not a total dominating set because vertex 2 is not adjacent to any of vertices in $\{2, 6, 7\}$.[2] What do you think the total dominating set is for Royt?"

Ajur had to think hard about this, studying the graph in front of him. Finally, he said, "I think vertices 3, 4, 7, 8, and 10 form a total dominating set."

Rishnak said, "Not bad, Ajur. We can remove vertex 7 from your answer

[2]A vertex is not considered to be adjacent to itself.

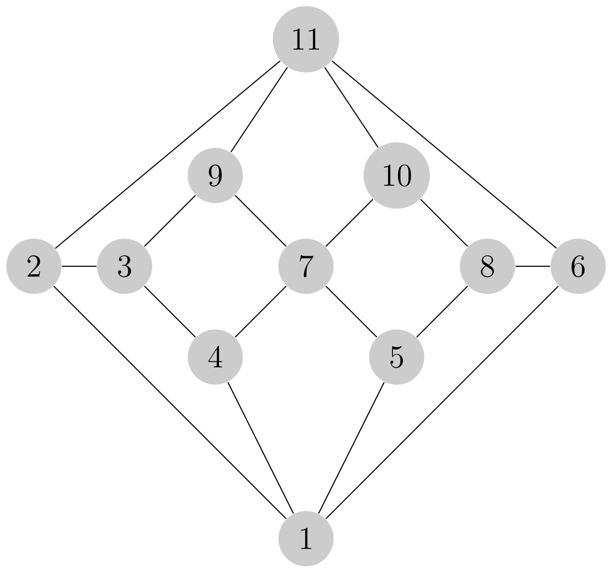


Figure 18.2: A graph representing the streets of Royt for which we wish to place a minimal number of water fountains such that every junction (vertex) either has a water fountain or is adjacent to a water fountain

and we have a minimum total dominating set." He flashed his hands and some of the vertices of the graph of Royt changed colors [Figure 18.4].

Ajur nodded and said, "I see."

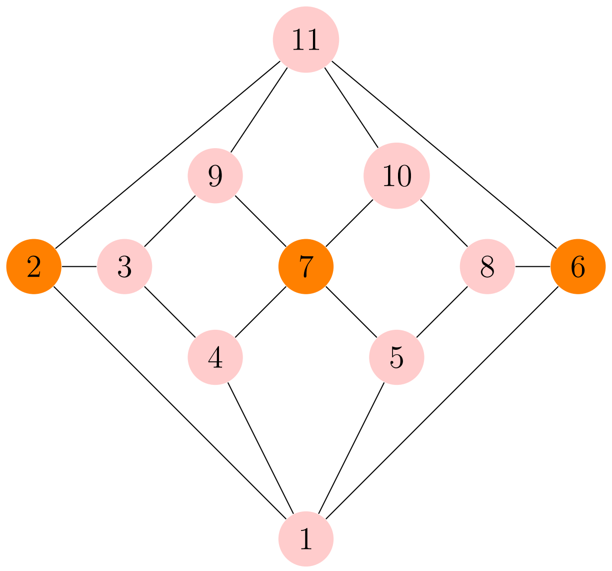


Figure 18.3: The graph from Figure 18.2 with water fountains placed at vertices 2, 6, and 7, which form a dominating set

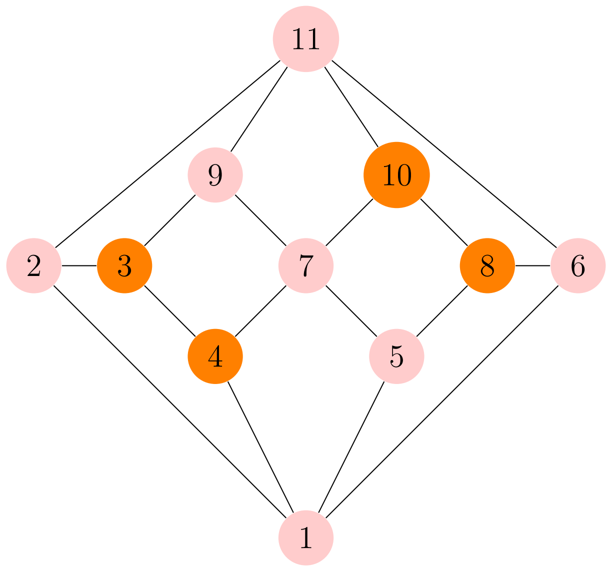


Figure 18.4: The graph from Figure 18.2 with total dominating set vertices 3, 4, 8, and 10 shown in orange

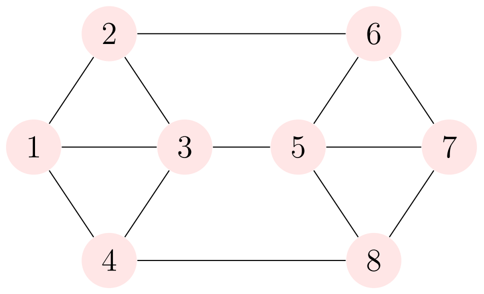


Figure 18.5: For this graph, where can we place water fountains such that each vertex either has a water fountain or is adjacent to a water fountain?

Question for the sixteenth day

Rishnak said, "And with that, let us move on to the question for the sixteenth day. For this new graph"—Rishnak flashed his hands and a new graph materialized in the air [Figure 18.5]—"can you identify the vertices at which we can place water fountains such that every vertex either has a water fountain or has one adjacent to it?"

Before you turn the page, try to come up with an answer of your own!

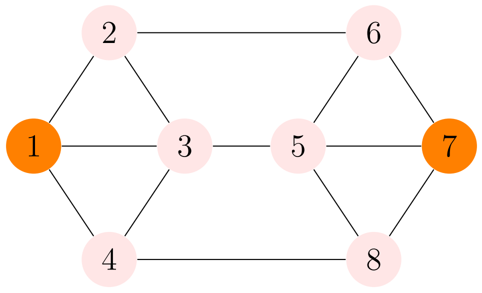


Figure 18.6: The graph from Figure 18.5 with water fountains, placed at vertices 1 and 7, shown in orange

Answer for the sixteenth day

Ajur copied the graph in the dirt. Within seconds, he had no trouble in placing water fountains at vertices 1 and 7 [Figure 18.6].

By this time, they had walked all the way to the water fountain. Ajur and Jura went to drink from the water fountain, then headed home.

Chapter 19

Graceful Numbering

Rishnak found Ajur and Jura playing near the water fountain. Continuing the conversation from the previous day, Rishnak reminded Ajur about vertex coloring and edge coloring as types of graph labeling schemes with constraints defined such that adjacent vertices and edges are not colored the same.

Rishnak said, "There are other labeling schemes. One such scheme in particular is called the *graceful numbering* of a graph. Given graph $G = (V, E)$ with n vertices and e edges, we assign distinct numbers to the vertices from $0, 1, 2, \ldots, e$ such that all edge labels are distinct from $1, 2, \ldots, e$. The edge label is the absolute value of the difference between the numbers associated with the end vertices."

Ajur scratched his head, so Rishnak waved his hands and a graph appeared [Figure 19.1].

Rishnak said, "Here is an example of a star tree. It consists of a center vertex connected to many leaves. Each edge weight is the absolute value of the difference between the two endpoint vertex labels. Can you draw such a graph that is a simple path of six vertices?"

Ajur picked up a stick and tried to draw such a graph in the dirt [Figure 19.2]. He smiled and said, "Yes, here is a gracefully numbered graph for a path of length 5."

Rishnak smiled and said, "Good, Ajur. And like a star tree, your snake graph can also be generalized to an arbitrary number of vertices.[1] Next, try a binary tree with seven vertices."

[1]Can you generalize each of these problems?

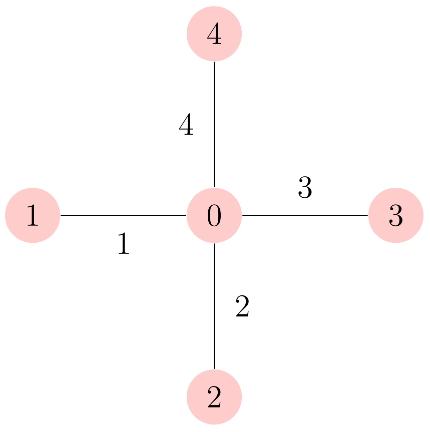


Figure 19.1: A star tree that is gracefully numbered since each of the edges has a distinct weight that is the absolute value of the difference between the endpoint vertex labels

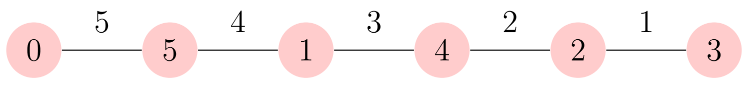


Figure 19.2: A gracefully numbered path (snake) in which each of the edges has a distinct number that is the absolute value of the difference between the labels of the endpoint vertices

Ajur drew a binary tree with seven vertices in the dirt, then slowly labeled each vertex and each edge [Figure 19.3]. He said, “Here is a graceful numbering!”

Rishnak said, “It is not known whether all binary trees have a graceful numbering. This is one of many open questions in graph theory. Attempts have been made to find counterexamples—meaning trees that do not have a graceful numbering—but these attempts have failed. And efforts to find a proof to show that all binary trees have a graceful numbering have also failed.”

Ajur wondered at this, then said, “Are there graphs for which we know there is no graceful numbering?”

Rishnak said, “There are graphs that do not have a graceful numbering”—

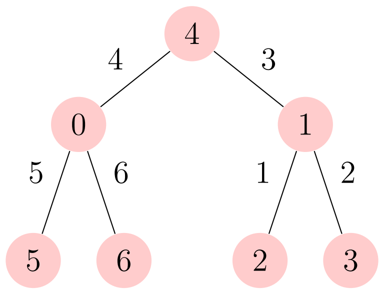


Figure 19.3: A gracefully numbered binary tree with seven vertices

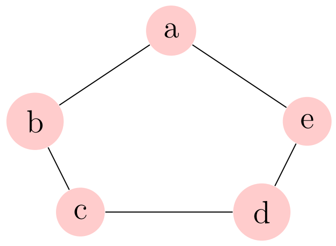


Figure 19.4: The C_5 graph (cycle of length 5) for which we cannot come up with a graceful numbering

he flashed his hands and a new graph appeared [Figure 19.4]—"such as this graph with a cycle of length 5."

Rishnak continued, "Complete bipartite graph $K_{m,m}$ can be gracefully numbered (or colored) by numbering vertices in one partition $0, 1, \dots, m-1$ and numbering vertices in the other partition $m, 2m, 3m, \dots, m^2$. With this numbering all of the edges get distinct numbers that meet the requirements of a gracefully numbered graph. Here, Ajur, try this for $m = 3$."

Ajur thought about this for a minute. He knew the $K_{3,3}$ graph must have six vertices, with three vertices in each partition, but how should they be numbered? He tried a few possibilities in his head, then drew a bipartite graph in the dirt and labeled its vertices and edges [Figure 19.5].

Rishnak smiled, pleased to see Ajur's work. Rishnak said, "One last new concept for you, Ajur. Closely related to graceful numbering, a *Golomb ruler* is an imaginary ruler that has a set of marks at integer positions such that no two pairs of marks are the same distance apart."

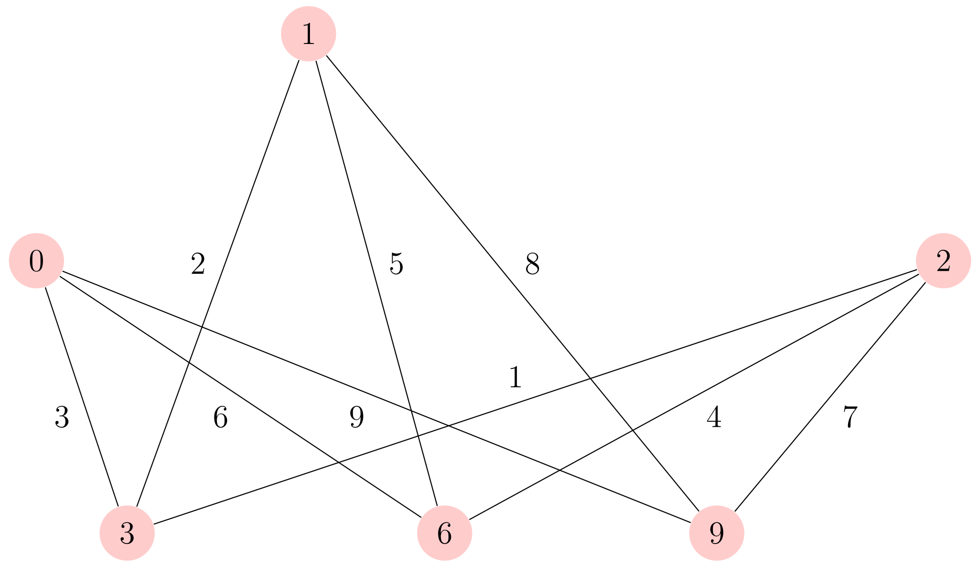


Figure 19.5: A gracefully numbered $K_{3,3}$ graph

Ajur tried to understand this, repeating it to himself.

After a pause, Rishnak said, "The number of marks on a Golomb ruler is its *order*, and the largest distance between two of its marks is its *length*. So a Golomb ruler of order 3 has marks $\{0, 1, 3\}$ and can be used to measure all lengths up to length 3. A Golomb ruler of order 4 has marks $\{0, 1, 4, 6\}$ and can measure all lengths up to length 6."

Ajur hurriedly said, "I get it."

Rishnak said, "The Golomb ruler has applications in coding theory and, it turns out, can be used to gracefully number complete graph K_4, which"—Rishnak flashed his hands and a new graph appeared [Figure 19.6]—"I present to you here."

Ajur studied the graph, understanding after some trial and error that the vertices and edges corresponded to the Golomb ruler lengths.

Question for the seventeenth day

Seeing that Ajur understood, Rishnak said, "The questions for the seventeenth day comes from my favorite author, Henry Ernest Dudeney and is Problem 453. It goes something like this. A young studious pupil finds

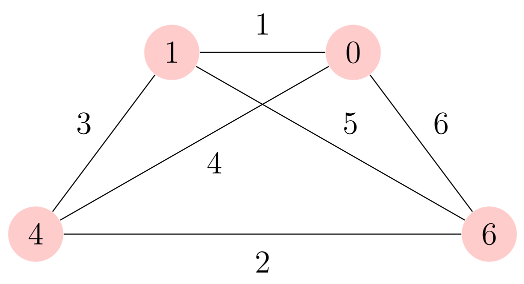


Figure 19.6: Complete graph K_4 gracefully numbered using a Golomb ruler

an old yardstick—which you might not know, a yardstick is 36 inches long. This yardstick is broken in that 3 inches have been chopped clean off, leaving a yardstick that is only 33 inches in length, no longer a yard."

Ajur laughed as Rishnak continued, "Some of the graduation marks are also obliterated, leaving only eight such legible marks. Still, this studious pupil is able to measure any length, in inches, from 1 inch to 33 inches. How many ways can you place these eight marks on the broken yardstick?"

Ajur said, "That doesn't sound too hard."

Rishnak smiled and said, "Be careful with this one. It is more difficult than it seems."

Before you turn the page, try to come up with answers of your own!

Answer for the seventeenth day

Ajur quickly saw that this problem was quite similar to the Golomb ruler concept. Excited, Ajur started working on the problem, but an hour passed as Ajur worked. This was more difficult than he had thought.

At last, Ajur said, “I have 11 solutions and here they are!” He had sketched them in the dirt:

- 1, 2, 3, 4, 10, 16, 22, 28
- 1, 2, 3, 4, 10, 17, 22, 28
- 1, 2, 3, 8, 14, 20, 24, 29
- 1, 2, 3, 10, 15, 22, 27, 31
- 1, 2, 3, 10, 16, 21, 25, 29
- 1, 2, 3, 11, 17, 21, 26, 30
- 1, 2, 3, 14, 18, 23, 25, 29
- 1, 2, 3, 14, 18, 23, 26, 29
- 1, 2, 3, 16, 21, 22, 26, 30
- 1, 2, 3, 16, 21, 24, 27, 31
- 1, 2, 4, 11, 18, 25, 30, 31

Rishnak raised his eyebrows and said, “Good, Ajur. I was worried you were stumped. There are more than 50 solutions, but let us call it a day.”

Rishnak noticed that Ajur had used a systematic “brute force” technique to solve this problem. Therefore, Rishnak decided to talk about this and more efficient search techniques the next day.

Ajur was tired but pleased to have learned a new type of ruler, one that was connected to graceful numbering.

Chapter 20

Brute Force Searching

Trying to locate Ajur to talk to him about different search techniques, Rishnak followed his own search method, an approach called depth first search, much like a strategy used in solving mazes. The search led Rishnak to Ajur, who was sitting under a tree.

Rishnak said, “There are many different search methods, and come to think of it, search is a trillion-dollar business.”

Ajur laughed and said, “Yes, fast search results from a search engine is a good business to be in.”

Rishnak said that most search engines use some kind of a table lookup. He said, “The more interesting issues to consider here are how the tables are stored and how each lookup is done. An index-based lookup and a hash-based lookup are two common techniques. Hashing involves finding a seemingly randomized location based on the given search keywords. And binary search on a sorted list works if the size of the list is not prohibitively large. All of these search techniques work well for unstructured data, but a graph is structured, so we need a different type of search technique.”

Ajur did his best to keep up. As per usual, Rishnak was overflowing with different ideas that he wanted to share.

Rishnak continued, “When searching a maze, you can use what is called a *depth first search.* This strategy essentially divides the vertices into sets of visited and non-visited vertices. When you visit a vertex, you can do any operation associated with that vertex. Here are the steps:

1. Place the start vertex on top of a *stack.*[1]

[1] A stack is a list but we can only access the top element, much like a stack of plates.

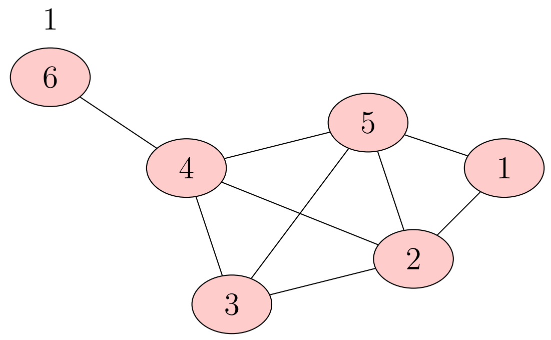


Figure 20.1: A depth first traversal—at start vertex 6

2. Remove the top vertex from the stack and mark the vertex as visited.
3. Create a list of that vertex's adjacent vertices. Add each of these adjacent vertices to the top of the stack as long as the vertex is marked non-visited.
4. Repeat Steps 2 and 3 until the stack is empty."

Ajur said, "I think I understand. So you go as far or as deep as you can before tracing your steps backward."

Rishnak said, "Precisely. Let me take you through the depth first search algorithm with you for this graph." He flashed his hands and a familiar graph appeared [Figure 20.1].

Ajur followed each step as Rishnak visited subsequent vertices [Figure 20.2] [Figure 20.3] [Figure 20.4] [Figure 20.5] [Figure 20.6]. Rishnak also labeled each edge that he traversed in red.

Rishnak said, "At the end of this algorithm, we have what is called a *depth first traversal*."

Rishnak said, "There is another search technique called *breadth first search*. It also classifies vertices as either visited or non-visited. The algorithm is:

1. Place a start vertex at the front of a queue.
2. Remove the vertex from the front of the queue and mark it as visited.

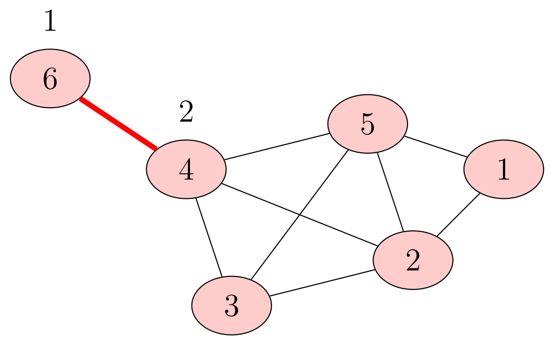


Figure 20.2: A depth first traversal—at Step 1

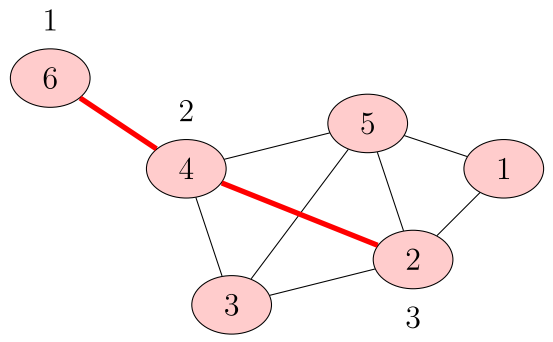


Figure 20.3: A depth first traversal—at Step 2

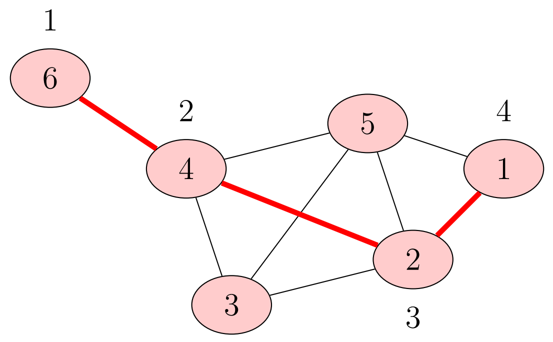


Figure 20.4: A depth first traversal—at Step 3

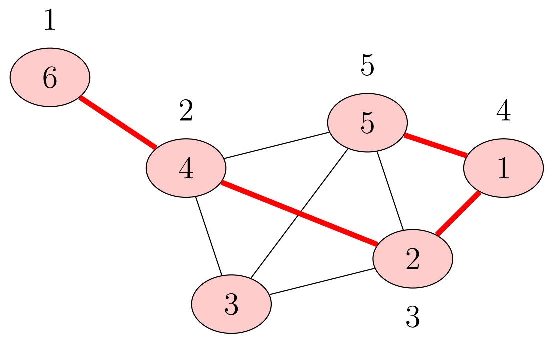


Figure 20.5: A depth first traversal—at Step 4

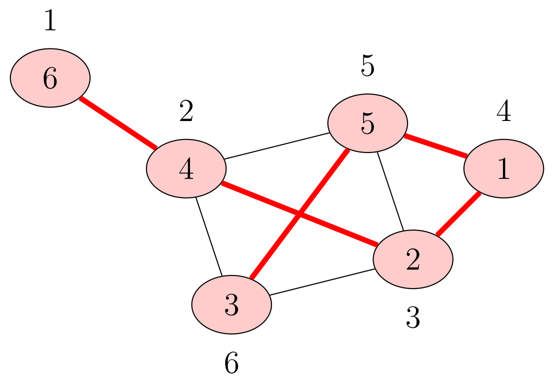


Figure 20.6: A depth first traversal—at Step 5

3. Create a list of that vertex's adjacent vertices. Add each vertex that is not yet visited to the end of the queue.

4. Repeat Steps 2 and 3 until the queue is empty."

Ajur noticed that by using a queue instead of a stack, the search would visit vertices in a different order than that of depth first search.

Rishnak said, "Here is our graph again, and"—he waved his hands and the original graph appeared [Figure 20.7]—"I will number the vertices in the order in which we visit them."

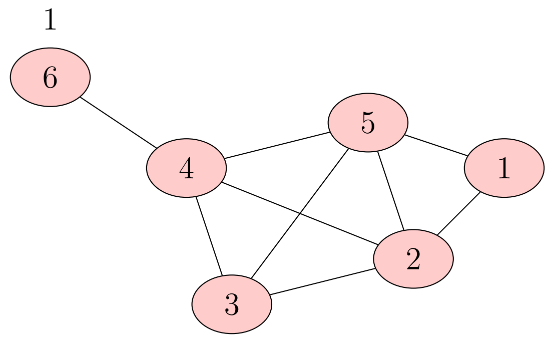


Figure 20.7: A breadth first traversal—at start vertex 6

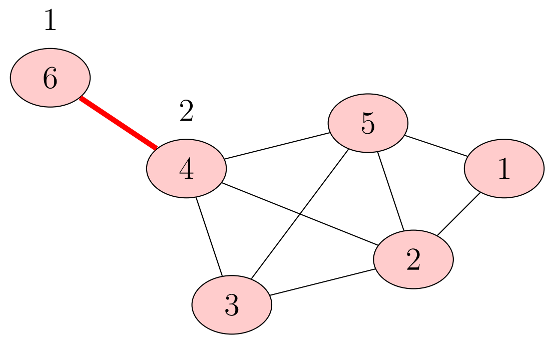


Figure 20.8: A breadth first traversal—at Step 1

Ajur watched as the graph changed from step to step [Figure 20.7] [Figure 20.8] [Figure 20.9] [Figure 20.10].

Ajur smiled, wondering about how each different type of search could be useful. He asked, "Do both search techniques always work?"

Rishnak said, "Yes, both depth first search and breadth first search can be used for searching the state space of all solutions. If we can enumerate all possible states—for the Hamiltonian cycle problem and the isomorphism problem, all possible permutations constitute all possible states—then how we visit all possible states could be done either using depth first search or breadth first search."

Ajur tried to understand. He asked, "We don't need to figure out all

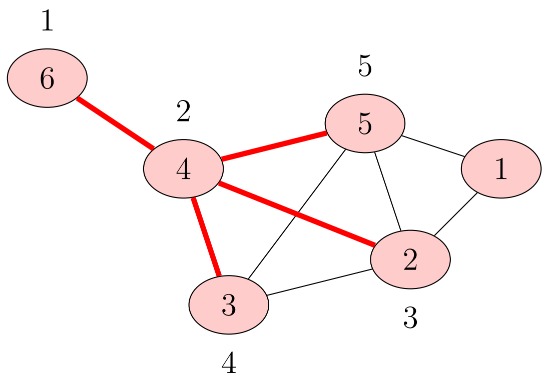


Figure 20.9: A breadth first traversal—at Step 2

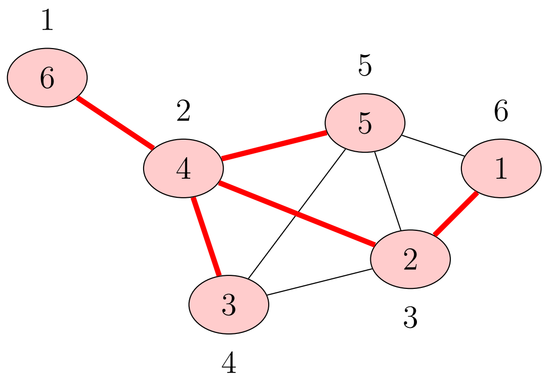


Figure 20.10: A breadth first traversal—at Step 3

permutations or combinations ahead of time, do we?"

Rishnak said, "No, there are different methods for generating permutations and combinations. In general, we generate a permutation, test it out, then go on to the next permutation. And there are efficient methods for generating the next permutation. You will learn all of this during your college years."

Ajur understood. He said, "The generation of all permutations or combinations is a brute force approach, right? This is how I solved the problem with the Golomb ruler yesterday."

Rishnak laughed and said, "Yes, brute force approaches are typically slow and tedious. There are often more efficient techniques for solving a problem involving what is called *backtracking*, which is a far more efficient means of searching the solution space."

Ajur smiled and said, "And I'll learn this in college?"

Rishnak laughed again and nodded.

Ajur realized there was so much more to learn but he did not get discouraged. Instead, he felt invigorated. He said, "Could we use backtracking to solve a Sudoku puzzle?"

Rishnak smiled and said, "Yes, that is one way to solve such puzzles. We can also think of solving a Sudoku puzzle as fully coloring all vertices of a graph given a partial coloring of the vertices."

Ajur thought about this and was going to ask another question, but Rishnak was too quick.

Question for the eighteenth day

Rishnak said, "Here is our question for the eighteenth day, Ajur. Can you draw a connected graph containing six vertices for which depth first search and breadth first search both produce the same numbering or labeling of vertices?"

Before you turn the page, try to come up with answers of your own!

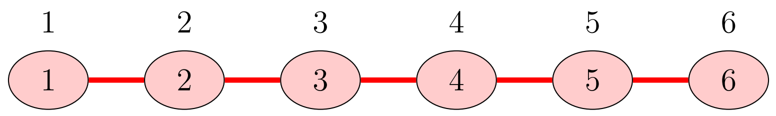


Figure 20.11: A graph for which breadth first search and depth first search, starting from vertex 1, both produce the same vertex numbering

Answer for the eighteenth day

Ajur quickly drew a graph in the dirt [Figure 20.11] and said, "Start at vertex 1 in this graph and both depth first search and breadth first search produce the same numbering of vertices."

Rishnak smiled.

After this interesting and exciting discussion, Ajur and Jura happily called it a day.

Chapter 21

Social Networks

The next day, Rishnak was talking with his ghost friends about his mathematical adventures with Ajur. Seeing Ajur and Jura approach, Rishnak cut short his conversation and joined Ajur and Jura.

Rishnak started the session right away. He said, "Over the last century, applications of graph theory abound in such fields as engineering, physics, and anywhere there is a need for optimization. More recently, though, with the prolific use of social network applications, including Facebook, Twitter, and LinkedIn, sociologists and psychologists have also found graph theory methods quite useful in their work"

Ajur nodded his head and said, "Sure, like how we are all so well-connected. Groups of friends. I remember we talked about cliques awhile ago."

Rishnak said, "Exactly. Stanley Milgram, a famous psychologist, wanted to show that people were well-connected. He devised an experiment in which he selected a collection of people living in the Midwestern United States. He asked them to send postcards to a single person P in Boston. And if they did not know person P then they could send a postcard to a person that knows someone who knows person P. The result of this experiment showed that most of the letters reached person P in five or six steps. Does that remind you of a graph theory concept, Ajur?"

Ajur frowned and thought for a few moments. Then it came to him and he said, "Aha, yes, this experiment shows that the diameter of this network of people is very small, just about five or six, as you said."

Rishnak said, "Precisely. The diameter of this social network tells us the longest of all of the shortest distances between any pair of vertices. Social networks have also been used in studying connections within a network of actors and actresses. Actors and actresses form the vertices, and we add edges between vertices if the corresponding actors or actresses have acted in the same film. Have you ever heard of Kevin Bacon?"

Ajur said, "Kevin who?"

Rishnak rolled his eyes, then said, "There is an actor by the name of Kevin Bacon who has acted with many other actors and actresses over the years. Many would ask what their Kevin Bacon distance was! It is essentially the length of the path from the person asking the question to Kevin Bacon in the co-actor network. There is even a dedicated website for finding these paths.[1] And a play about this concept of *six degrees of separation*. Just like Milgram's experiment, this Kevin Bacon network shows that the diameter of the network is very small and there is a Kevin Bacon vertex with a large degree."

Ajur laughed at this, though was also intrigued by the resulting diameter being very small in comparison to the vast number of vertices in the network or graph.

Rishnak continued, "Social networks have also been identified in co-author networks for authors of scientific and mathematical publications. Remember Paul Erdős? He was a famous twentieth century mathematician and a prolific contributor to the field of graph theory. He wrote so many papers with so many different co-authors that the concept of an Erdős distance was defined. Here, authors are vertices and two vertices (or authors) are connected by an edge if they have written a paper together. The American Mathematical Society has a website that calculates this Erdős distance to other authors.[2]"

Ajur said, "Is the diameter of that graph also very small?"

Rishnak nodded and said, "Usually the Erdős distance to other graph theorists is a very small number because of the collaborative nature of the work and of course an author like Paul Erdős wrote such a large number of papers with a large number of co-authors." Ajur said, "So the vertex corresponding to Erdős has a very large degree."

Rishnak smiled and said, "Exactly."

Ajur asked, "What other social networks are there?"

[1]Have a look at `http://oracleofbacon.org/movielinks.php`.

[2]See `https://mathscinet.ams.org/mathscinet/collaborationDistance.html`.

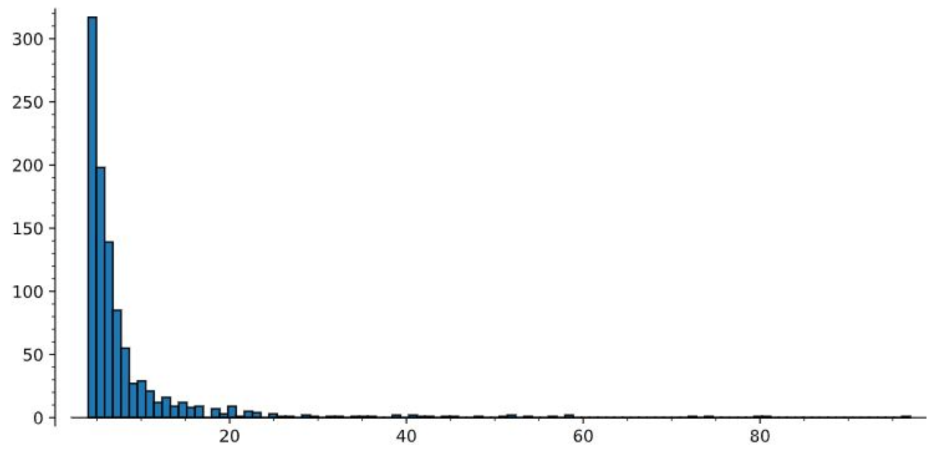


Figure 21.1: Sample degree distribution of a social network

Rishnak frowned and said, “Well, while social networks have been used to spread real news, they have also been abused to spread rumors and disinformation called fake news. The news—both real and fake—about individuals or political parties can be spread or shared very easily with one’s friends and then one’s friends of friends until eventually it covers more or all of the social network. Most fake news items are generated by bots, an abbreviated term for a computer program. And advances in image and audio manipulation make it nearly impossible to distinguish between what is real and what is fake.”

Ajur thought about this for a few seconds, an uneasy feeling coming over him.

Rishnak sighed and said, “Here’s something a bit more positive and rather interesting. The degree distribution of social networks tend to follow what is called a *power law*. In this chart”—Rishnak waved his hands and a chart appeared [Figure 21.1]—“we count the degrees of each vertex in a social network. There are a few vertices with large degrees (on the left) and a very large number of vertices with small degrees that form a long tail on the right.”

Ajur thought further about this and said, “And the vertices with large degrees have many adjacent vertices, so they’re like the popular students in school with many friends.”

Rishnak nodded and said, “Yes, these individuals are sometimes referred to as hubs.”

Thinking about Erdős again, Ajur remembered the Erdős–Rényi model for generating random graphs. He asked, “Do the Erdős–Rényi graphs have a uniform degree distribution—I mean all of the vertices have roughly the same degrees?”

Rishnak smiled and said, “Let me try to explore that further with an example using Facebook. A Facebook graph consists of users as vertices and edges between two users if they are friends of one another—and remember, in Facebook, friendship is a symmetric relationship. It has been found that the average number of Facebook friends per user, which is therefore the average degree of a vertex, is 300 and the median degree is 200.”

Ajur asked, “How big is the Facebook graph? It must be humongous.”

Rishnak said, “The Facebook graph has approximately two billion vertices—many of these vertices could be fake users or groups. With a median degree of 200, that means that one billion users have less than 200 friends.”

Ajur raised his eyebrows. He said, “Wow, that’s quite a long tail.”

Rishnak smiled, happy to see Ajur put the various pieces of the puzzle together. He continued, “And Facebook restricts the maximum number of friends one can have to 5000. According to sociologists, a person can actually be close to at most 150 friends, which tells you something about how close friendships are in Facebook.”

Ajur laughed.

Rishnak continued, “Also, the Facebook graph has an average separation or diameter of only 3.74.[3] There is a well-known Facebook paradox that states that, on average, most people have fewer friends than their friends have. We can see this by studying this graph.” Rishnak flashed his hands and a graph appeared [Figure 21.2].

Rishnak said, “In this graph, the average number of friends—in other words, the average degree—is $\frac{2+2+3+1}{4} = 2$. Person A sees that D has two friends and B has three friends. Person B sees that A and D both have two friends and C has only one friend. Person C sees that B has three friends. And person D sees that A has two friends and B has three friends. Therefore, the average number of friends one sees is $\frac{2+3+2+2+1+3+2+3}{8} = 2.25$.”

Ajur frowned at this result.

[3]`https://research.fb.com/blog/2016/02/three-and-a-half-degrees-of-separation/`

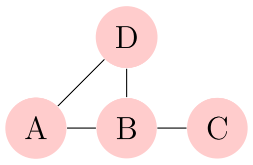


Figure 21.2: A graph that explains the Facebook friends paradox

Rishnak continued, “This is in contradiction to the common belief that one has more friends than their friends have!”

Ajur protested, saying, “That doesn’t make any sense.”

Rishnak laughed and mentioned that there was a nice mathematical explanation for this phenomenon—and that Ajur should discover it on his own.

Kinaja, a glowing white ghost, suddenly appeared. She had been listening to Rishnak go on and on about social networks. She glided in to join them and said, “I think that virtual social networks, being unregulated, cause a lot of harm—and here are three reasons why:

1. Users can too easily be bullied.

2. Bots and trolls pretend to be genuine users, but they propagate misinformation.

3. One’s private information too easily gets stolen or sold to advertisers.”

Rishnak and Ajur both nodded in agreement.

Question for the nineteenth day

Rishnak said, “Ajur, we have come to the nineteenth day. Here is your question. Does the Facebook paradox still hold for a regular graph of degree 3 with 10 vertices?”

Before you turn the page, try to come up with an answer of your own!

Answer for the nineteenth day

Ajur worked this problem out by thinking aloud. He said, "If the degree is 3, then each person will have three friends, but the average number of friends one sees is $\frac{(3+3+3)*10}{30} = 3$. Therefore, for this graph, there will be no paradox."

Rishnak said, "Good, Ajur. And note that most real social networks are not regular, so most social networks do see this paradox."

That was a good place to stop the discussion for the night.

Chapter 22

Instant Insanity Puzzle

Rishnak was examining the headstones of some of the graves and came across a headstone with the name Schossow[1] on a grave. Memories of this puzzle flooded Rishnak's senses. Schossow's puzzle had card suits (i.e., hearts, diamonds, clubs, and spades) marked on each face. In 1967, Frank Armbruster created a variation of this puzzle with colors instead of suits, which was published by Parker Brothers—that puzzle was called the "instant insanity" puzzle.

Rishnak thought this puzzle, with its graph-theoretic solution, would certainly appeal to Ajur. It did not take long for Rishnak to find Ajur and Jura.

Excitedly, Rishnak said, "For our last day, we are going to talk about one of my favorite puzzles. This puzzle has a long solution that uses search techniques and a short solution that we can explore using some clever graph theory techniques."

Ajur, seeing Rishnak's enthusiasm, said, "Sure, what's the puzzle?"

Rishnak continued, "The puzzle itself consists of four cubes numbered $1, 2, 3, 4$, with each face colored as shown in front of you." Rishnak waved his hands and a colorful display of cubes floated in front of Ajur [Figure 22.1].

[1]Frederick Schossow received a patent in 1900 for proposing this puzzle in 1899, as described here: `https://patents.google.com/patent/US646463A/en`.

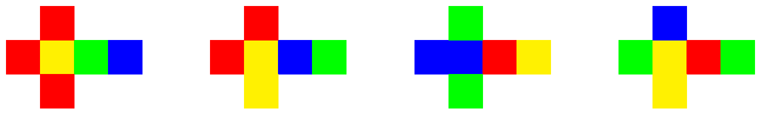


Figure 22.1: The four cubes that make up the instant insanity puzzle

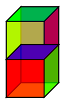


Figure 22.2: Cube 4 placed on top of cube 1 such that all faces have distinct colors

Rishnak said, "The problem is to stack these four cubes vertically in a column such that all of the cubes on each side of the column (left, right, front, and back) have distinct colors. Here"—he flashed his hands and two cubes came together [Figure 22.2]—"we have cube 4 sitting on top of cube 1. See that all faces have distinct colors?"

Ajur nodded. He asked, "We need to do that with all four cubes?"

Rishnak nodded, unable to stop himself from smiling. He said, "If we start with a brute force solution, also known as an exhaustive search technique, then for each cube, we have to choose which face should be at the bottom of the cube. There are six possibilities to choose from. And once we have decided on the color of the bottom face, there are four rotations for the sides of the cube. Therefore, we have 24 possibilities for each cube."

Ajur caught on quickly. He said, "And with four cubes to be stacked, that means we have $24 \times 24 \times 24 \times 24$ possibilities."

Rishnak said, "Correct, or 24^4, which equals $331,776$ possibilities. By searching each of these $331,776$ possibilities, we will find a solution and—"

Ajur said, "Wait, for the bottom cube, we don't need to consider the four rotations. Instead, we can fix that cube in place."

Rishnak smiled, impressed with Ajur's quickness. Rishnak continued, "And if we find one solution, then we can also permute the cube positions without altering the constraints to find other solutions. We could consider three opposite faces of each cube. There are four colors—red, yellow, green, and blue—so these form the vertices. Two vertices (or colors) are adjacent if they form the opposite sides of the same cube."

Ajur was puzzled, so Rishnak waved his hands and showed an example graph [Figure 22.3]. Rishnak said, "This graph represents the first cube." He waved his hands again and again, forming three other graphs [Figure 22.4] [Figure 22.5] [Figure 22.6], then said, "Similarly, we can draw graphs for each of the other cubes."

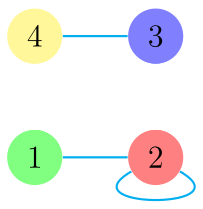


Figure 22.3: The graph representing cube 1

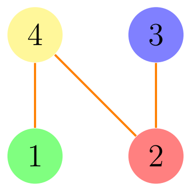


Figure 22.4: The graph representing cube 2

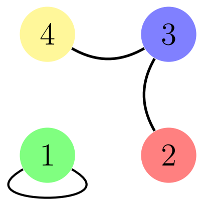


Figure 22.5: The graph representing cube 3

Ajur was mesmerized by the graphs in front of him. He said, "Now we need to combine these graphs, right?"

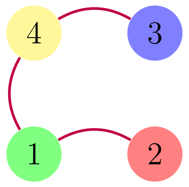


Figure 22.6: The graph representing cube 4

Rishnak smiled broadly. He flashed his hands and the graphs came together to form one giant graph [Figure 22.7].

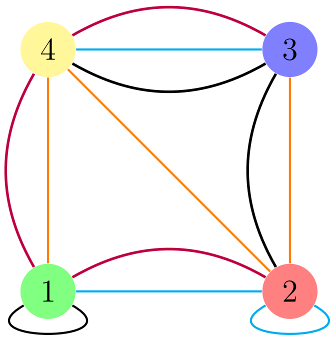


Figure 22.7: The graph representing all four of the cubes

Rishnak said, "From this graph, we need to figure out which faces of the cubes are selected as sides of the column. Essentially, there are four sides—front, back, left, and right[2]—and the graph in front of you has this information as a subgraph. Let us try to obtain two subgraphs with orientation so that we know the order of the sides. The first subgraph will give us the front and back faces, while the second subgraph will give us the left and right faces."

Ajur scratched his head but continued to follow along.

Rishnak continued, "The constraints on the subgraphs are:

1. The two subgraphs should have no edge in common since an edge represents front-to-back or left-to-right, and we cannot use them for

[2]For this problem, the top and bottom faces are not relevant.

both.

2. Each subgraph should contain an edge from each cube, which will make sure that we use all four of the cubes. In the graph, this means all of the edges should be different colors.

3. Each vertex should have a degree of 2 in the subgraph since the colors cannot repeat."

Ajur tried to keep up.

Excitedly, Rishnak went on, waving his hands to form the two subgraphs [Figure 22.8] [Figure 22.9]. He said, "From the original graph, we can extract these two subgraphs."

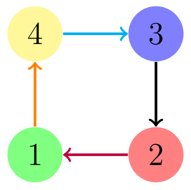


Figure 22.8: A subgraph depicting the front and back faces

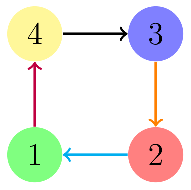


Figure 22.9: A subgraph depicting the left and right faces

Rishnak continued, "From these two subgraphs, we can figure out what colors the four sides must be. Here is the solution for each cube.

1. Cube 1: front is yellow, back is blue, left is red, right is green—and this corresponds to an edge color of cyan in the subgraphs.

2. Cube 2: front is green, back is yellow, left is blue, right is red—and this corresponds to an edge color of orange in the subgraphs.

3. Cube 3: front is blue, back is red, left is yellow, right is blue—and this corresponds to an edge color of black in the subgraphs.

4. Cube 4: front is red m back is green, left is green, right is yellow—and this corresponds to an edge color of purple in the subgraphs."

Ajur asked, "How did you find the subgraphs with the constraints that you mentioned?"

Rishnak grinned and said, "For a small case, we can do a visual inspection and obtain the decomposition, but in general, this is a very hard problem."

Ajur said, "Suppose we have four cubes, with each cube colored on all faces with the same color. So the first cube is all red, the second cube is all yellow, the third cube is all blue, and the fourth cube is all green. Then the two subgraphs will be easy to extract—and of course the solution is trivial."

Rishnak waved his hands and four new graphs appeared [Figure 22.10]. He said, "Yes, here are the original four graphs. And the decomposition into two subgraphs is shown here."

He flashed his hands again and two more figures appeared [Figure 22.11] [Figure 22.12].

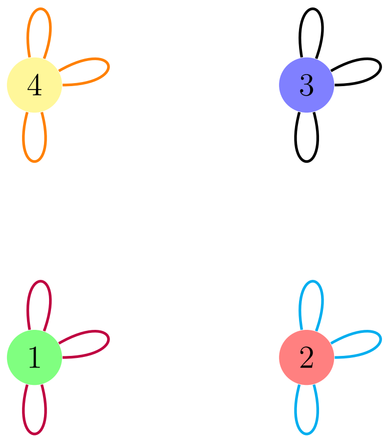


Figure 22.10: A graph for all four of the cubes

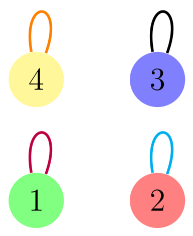


Figure 22.11: A subgraph for the graph shown in Figure 22.10 representing the front and back faces

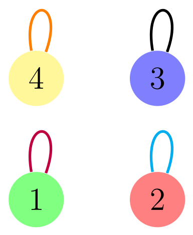


Figure 22.12: A subgraph for the graph shown in Figure 22.10 representing the left and right faces

Before Rishnak could speak further, Ajur quickly said, "Aha, the solution is then this.

1. Cube 1: front is red, back is red, left is red, right is red.
2. Cube 2: front is yellow, back is yellow, left is yellow, right is yellow.
3. Cube 3: front is blue, back is blue, left is blue, right is blue.
4. Cube 4: front is green, back is green, left is green, right is green."

By this time, from the past nineteen days, Ajur felt he had a very clear understanding of all of the concepts of graph theory they had covered, as well as a desire to keep learning more.

He faced Rishnak and said, "Rishnak, thank you for teaching me and helping me understand all of these wonderfully interesting problems."

Rishnak blushed (if that is possible for a ghost).

Question for the twentieth day

After a few moments, Rishnak said, “At long last, we have arrived at the twentieth day. I have one last question for you, Ajur. I hope you can answer it, as it will also set me free.”

Ajur took a deep breath and said, “I will do my best.”

Rishnak said, “Can you decompose this graph”—Rishnak flashed his hands and a final graph appeared [Figure 22.13]—“for all new sets of cubes by drawing the color of one face of one of the cubes?”

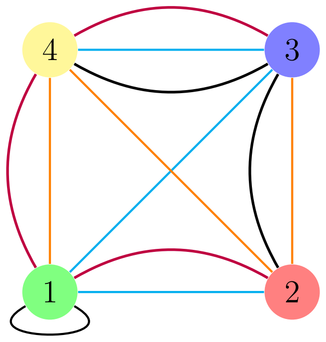


Figure 22.13: A graph for all of the new cubes

Before you turn the page, try to come up with answers of your own!

Answer for the twentieth day

Ajur tried to decompose the graphs using the same approach as before. Before long, he drew two final subgraphs in the dirt [Figure 22.14] [Figure 22.15]. He said, “Here are the two subgraphs, Rishnak.”

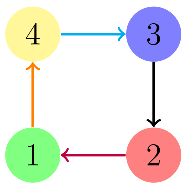


Figure 22.14: The subgraph from the graph in Figure 22.13 depicting the front and back faces

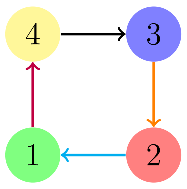


Figure 22.15: The subgraph from the graph in Figure 22.13 depicting the left and right faces

Rishnak smiled like he had never smiled before. He said, “Correct.”

Ajur smiled back and said, “Thank you, Rishnak.”

Ajur and Jura said their goodbyes to Rishnak. Not far away, Kinaja watched, smiling to herself that Ajur not only understood the concepts but also had the decency to thank Rishnak for all of his efforts.

Chapter 23

Conclusion

Rishnak used to teach at a college in Royt. He was such a mean teacher that his students landed a curse on him such that he would remain a ghost until a student whom he teaches appreciates his teaching. Shortly after Ajur thanked Rishnak, Rishnak was released from his curse. He could now join his own ancestors in the other world. Rishnak was also happy that young Ajur was not afraid of failure and instead persevered to understand and solve all of the hard problems he faced. To show his appreciation to Ajur, Rishnak presented Ajur with a pet monkey named Ruele and Ajur was ecstatic.

Chapter 24

Summary

Here at journey's end, we summarize which concepts were covered in each chapter. If some of these concepts are not clear, please go through the chapter again or contact the first author, mskgraph149@gmail.com.

In the **first chapter**, we defined a **graph** in terms of **vertices**, **edges**, and **relations** between them. A few example graphs were given to illustrate the concept.

In the **second chapter**, we introduced the **main characters** of our story.

In the **third chapter**, we explained the **degree** of a vertex and provided some elementary properties of degrees.

In the **fourth chapter**, we discussed **trees** and **rooted trees**, as well as some example applications.

In the **fifth chapter**, we discussed **subgraphs** and the different types of subgraphs.

In the **sixth chapter**, we covered a specific type of **path** and a specific type of **cycle**, i.e., the **Eulerian path** and the **Eulerian cycle**.

In the **seventh chapter**, we introduced another type of path and cycle, i.e., the **Hamiltonian path** and the **Hamiltonian cycle**.

In the **eighth chapter**. we discussed **isomorphism** and **subgraph isomorphism** between graphs.

In the **ninth chapter**, we covered the embedding of graphs in a plane, i.e., **planar graphs**.

In the **tenth chapter**, we introduced various **graph coloring problems** and showed a few examples.

In the **eleventh and twelfth chapters**, we discussed the two closely related problems of **spanning trees** and **shortest paths**. In these chapters, we also provided methods to compute minimum cost spanning trees and shortest paths.

In the **thirteen, fourteenth chapters**, we described special types of subgraphs, including **cliques**, **independent sets**, **vertex covers**, **bridges**, **cut vertices**, and **cut sets**. We also discussed applications of these special subgraphs.

In the **fifteenth chapter**, we covered **connected** and **strongly connected directed graphs**, including a football matches example.

In the **sixteenth chapter**, we discussed both **matching problems** and **assignment problems**, including the **stable marriage problem**.

In the **seventeenth chapter**, we discussed various **graph operations**, including the **graph union**, the **graph complement**, **vertex addition**, the **Cartesian product**, and the **line graph**.

In the **eighteenth chapter**, we introduced the **dominating set** of a graph.

In the **nineteenth chapter**, we covered the **graceful numbering** of a graph.

In the **twentieth chapter**, we discussed **brute force searching** techniques, including **depth first search** and **breadth first search**.

In the **twenty-one chapter**, we discussed **social networks** as an application of graphs.

Finally, in the **twenty-second chapter**, we presented the **instant insanity puzzle** and how graph theoretical techniques can be used to solve it.

Index