CSCI 2200

Foundations of Computer Science

Lecture 11: Graphs I

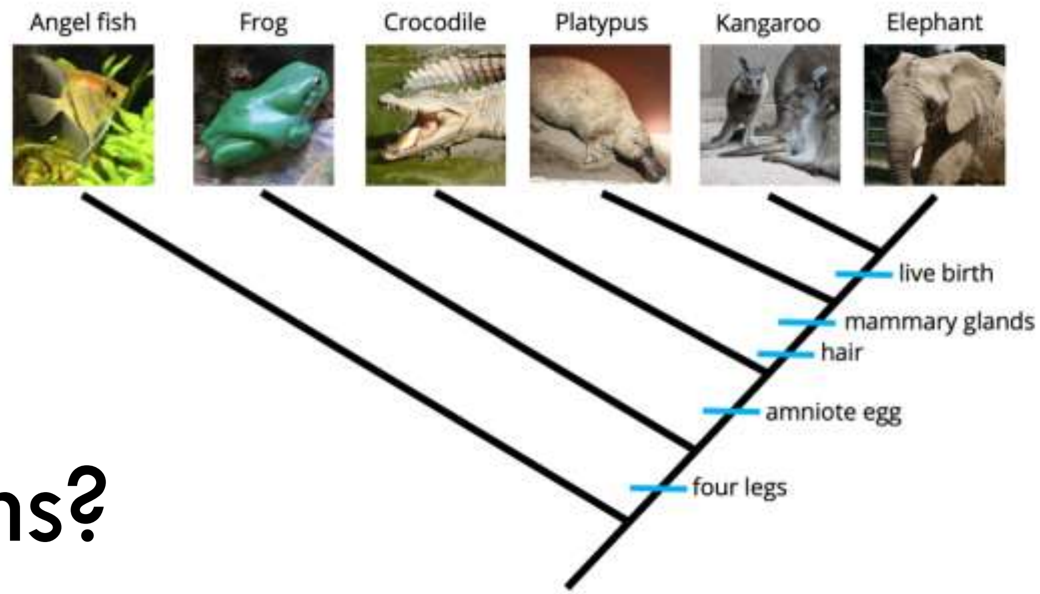
CSCI 2200
Foundations of Computer Science

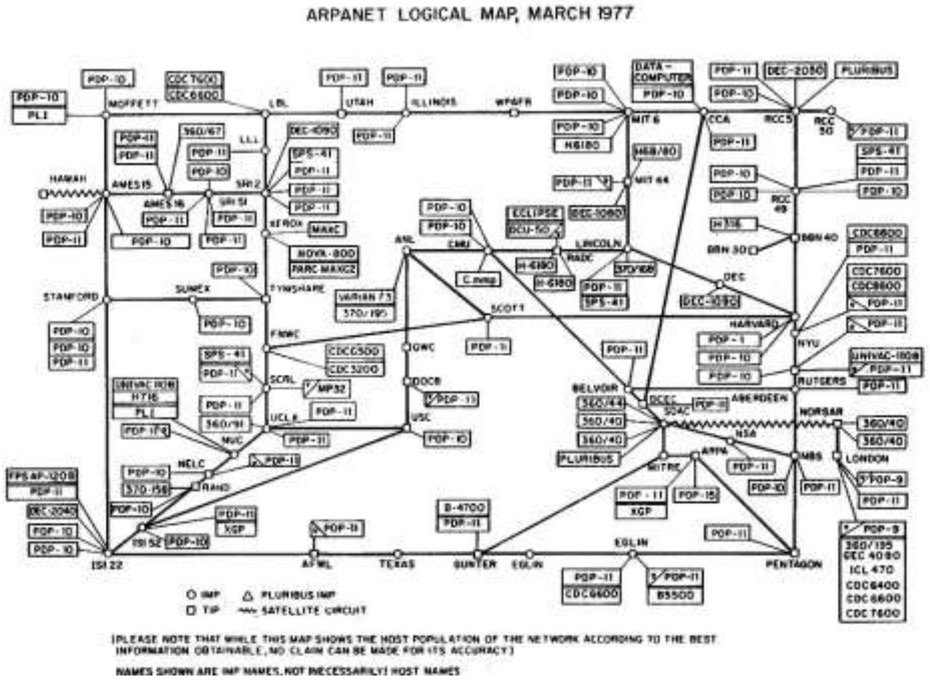
Lecture 11:
Graphs I

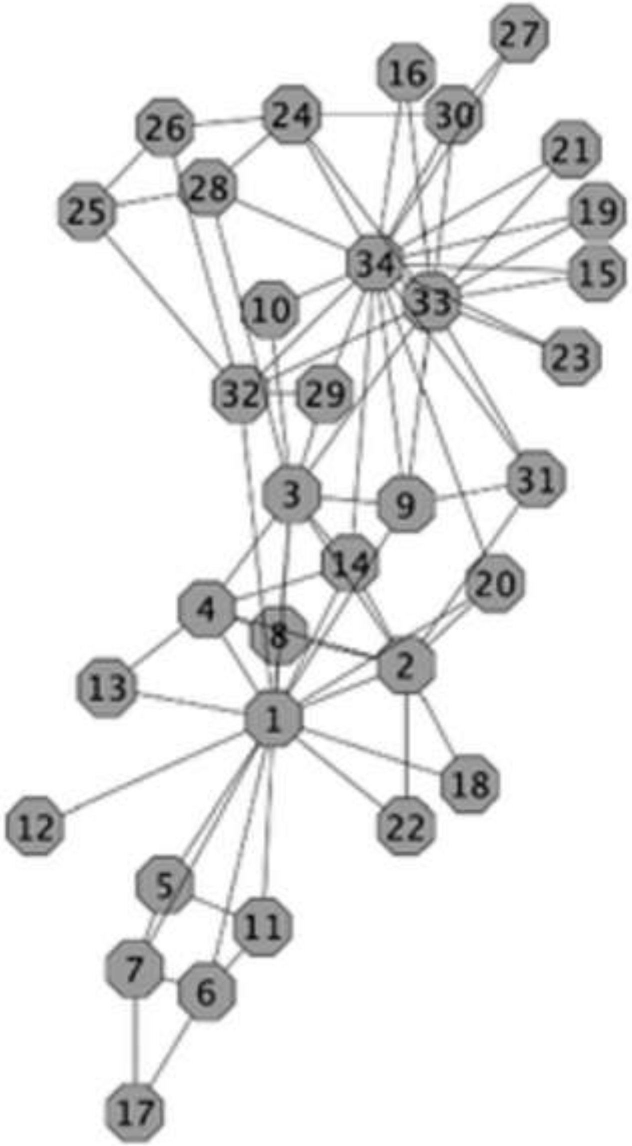
Today's tasks

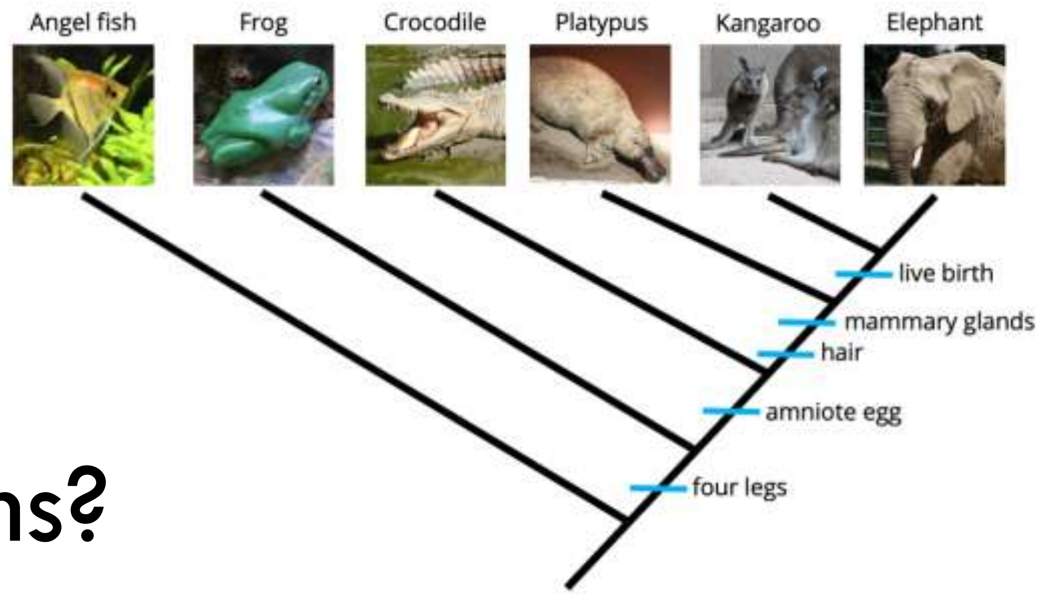
- Gentle reminder: Exam 2 on Wed. 2/26
- More urgent reminder: Next class will be on <u>Tuesday</u> 2/18.
- Graphs
  - Motivation & basic notation / terminology
  - Programmatic representation
  - Connectivity properties
  - Special types of graphs
  - Modeling & problem solving with graphs

Why graphs?

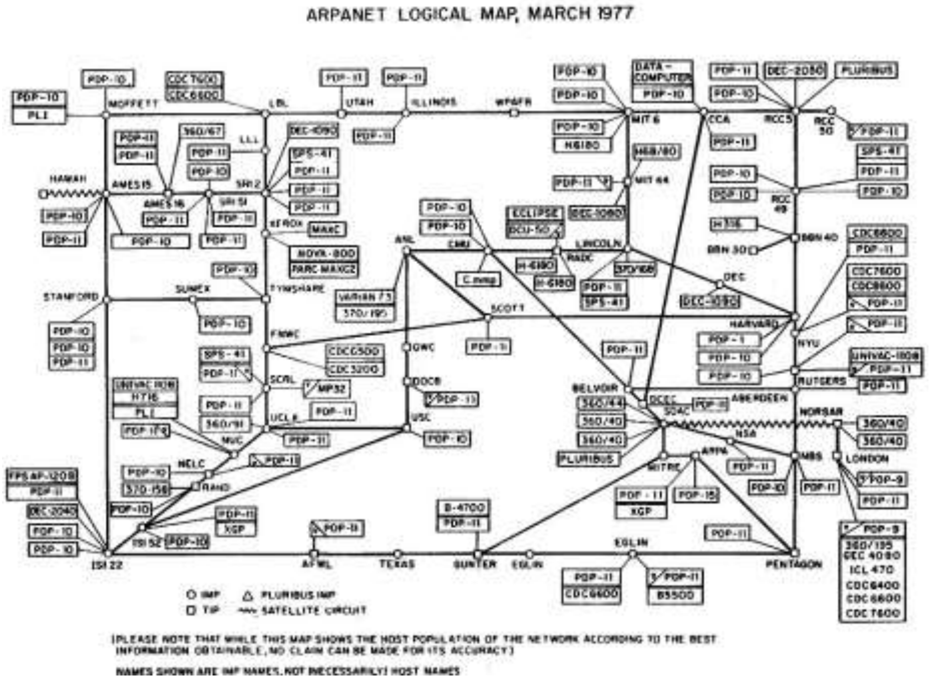

Angel fish
Frog
Crocodile
Platypus
Kangaroo
Elephant
live birth
mammary glands
hair
amniote egg
four legs

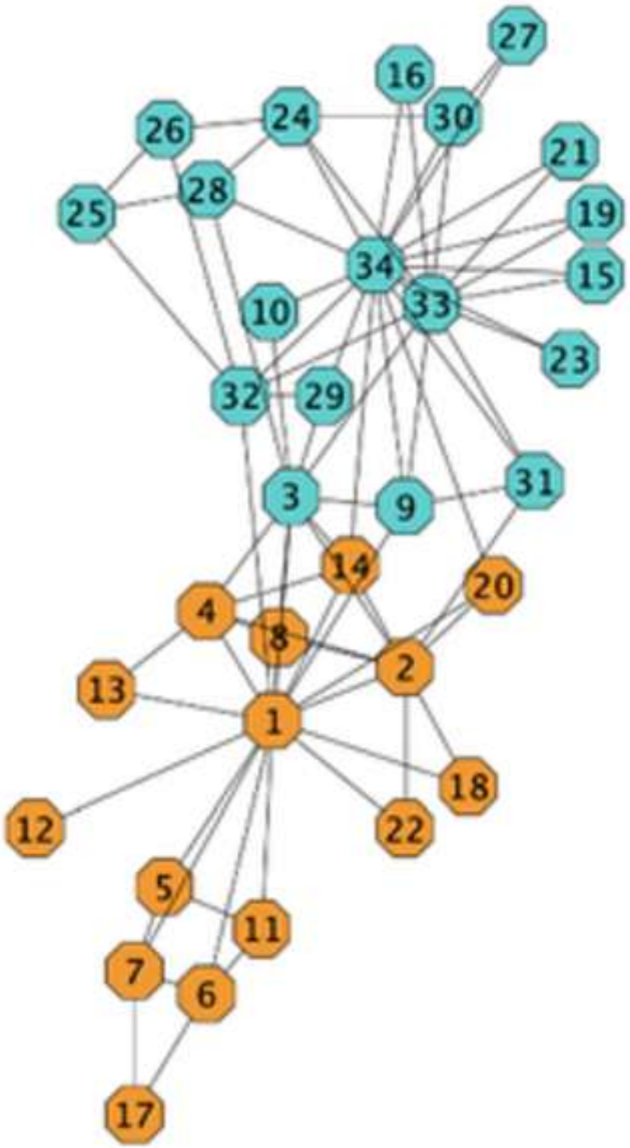

ARPANET LOGICAL MAP, MARCH 1977
○ IMP
△ PLURIBUS IMP
□ TIP
SATELLITE CIRCUIT
(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)
NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES




Angel fish
Frog
Crocodile
Platypus
Kangaroo
Elephant
live birth
mammary glands
hair
amniote egg
four legs

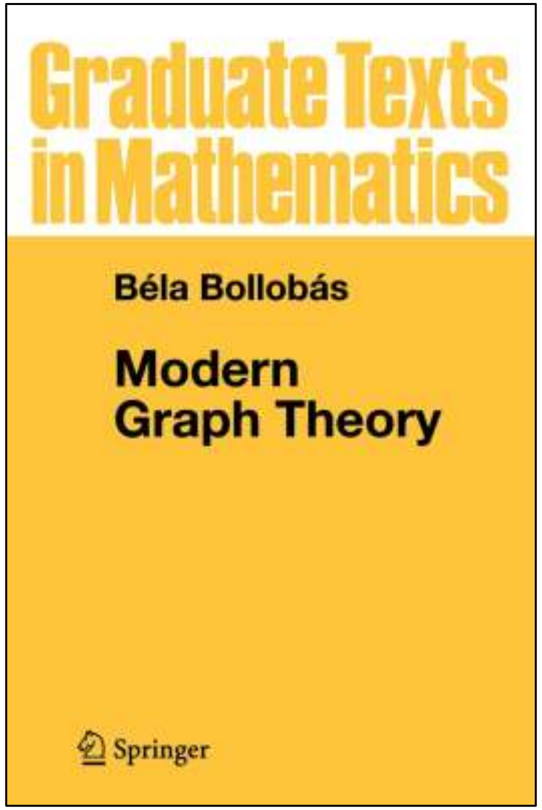
Why graphs?


ARPANET LOGICAL MAP, MARCH 1977
O IMP
□ TIP
△ PLURIBUS IMP
SATELLITE CIRCUIT
(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)
NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES



Graphs model relationships

- Graphs are exceedingly common in computer science because so many different situations call for modeling not just some set of objects, but some kind of relationship between those objects.
- "Mathematics is the art of giving the same name to different things." –Henri Poincaré
    - By distilling a situation down to a graph, we are able to bring our (large) toolbox of graph theorems and properties to bear on the problem.

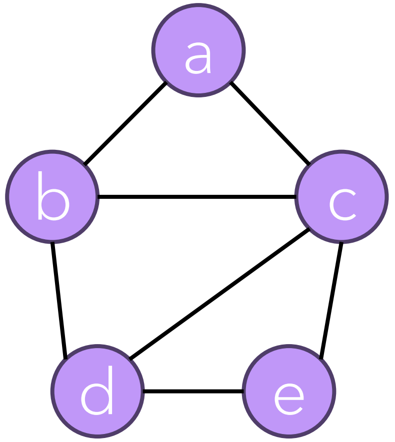


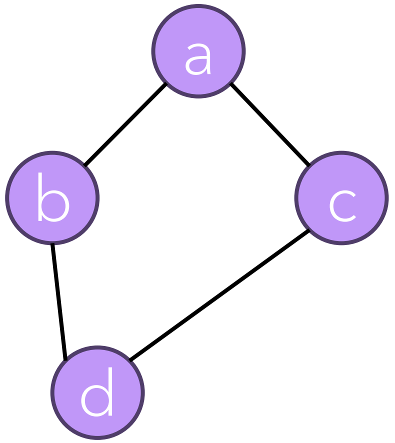
Vocabulary

- You know nodes/vertices and edges/connections. Graphs can be *directed* (one-way edges) or *undirected* (two-way).
- *Degree*: The number of edges incident to a node.
  - When graphs are directed, we talk about *indegree* and *outdegree*.
- *Path*: A sequence of edges leading from one vertex to another, where each set of consecutive edges shares one vertex. A *simple path* does not repeat any vertices.
  - The length of a path is the number of edges in it. Often we are interested in the shortest path between two vertices.
  - A *cycle* is a path that starts and ends at the same vertex.

Subgraphs and induced subgraphs

- A *<u>subgraph</u>* S of a graph G is simply any graph in which the vertices and edges of S are subsets of the vertices and edges of G.
  - Naturally, edges in S must refer to vertices that are also in S.
- An *<u>induced subgraph</u>* is a subset of the vertices and <u>all</u> of the edges between those vertices.
  - The bottom graph is a subgraph of the top one, but it is not an induced subgraph because the edge (b, c) is missing.



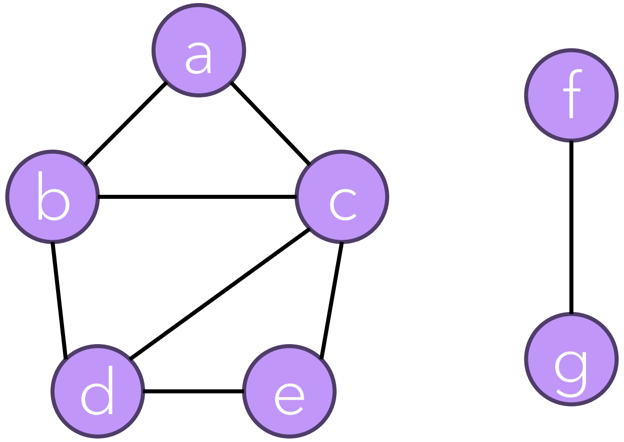


Graph variations

- A *weighted graph* involves attaching a numerical value to each edge. These can represent **many** different things: distances, strength of attachment, capacity / bandwidth, etc.
- Generally, we insist that our graphs contain no self-edges (i.e. edges like (a, a)), and no more than one edge between any given pair of vertices. These are *simple graphs*.
  - If we relax these rules, we get a *multigraph* – we saw one example of this with the Bridges of Konigsberg problem.
- Unless specifically stated, all of our graphs will be simple, unweighted, undirected graphs.

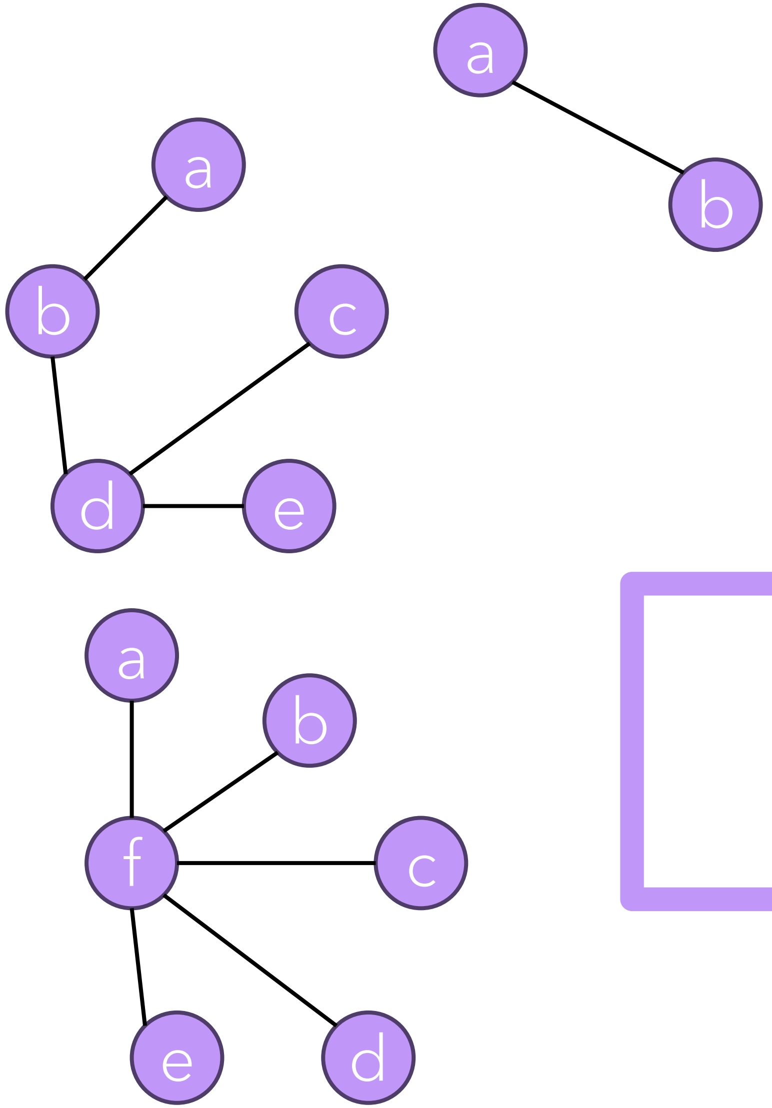
Connectivity

- *Connected*: An undirected graph is connected if there is a path from every vertex to every other vertex.
  - Connectivity in directed graphs can get a bit trickier, and we won't address it here.
  - Even when a graph is not connected, we can partition it into *connected components*. The graph at right has two of them.
  - A connected component must have all of the connected vertices. We would not refer to $\{a, b, c, d\}$ as a connected component because e has been left out.



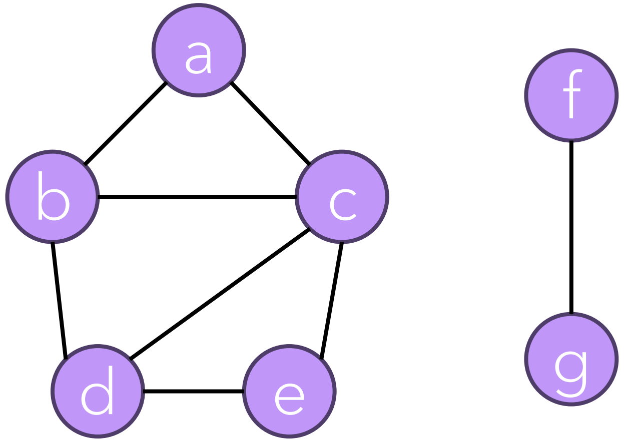
Trees

- A tree is a connected graph with no cycles.
  - Note that we usually talk about *rooted* trees in programming, but there's no presumption of that here. All of the graphs at right are trees.
- Count the edges of each of the trees at right. What do you notice?
- Theorem: Every tree has $|V| - 1$ edges.



Adjacency matrix vs. adjacency list

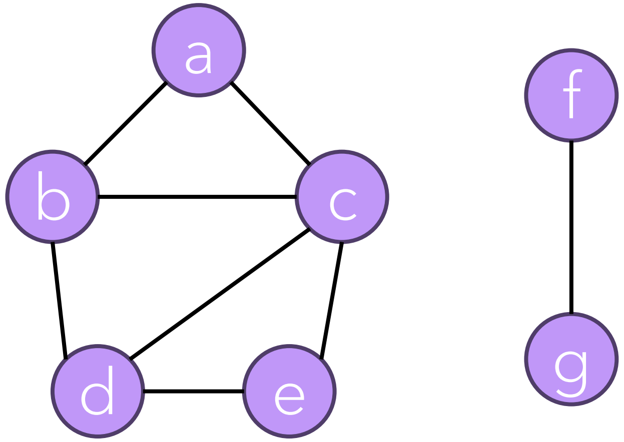
- There are two primary methods for representing a graph within a computer program.
- If you have a *dense* graph (i.e. lots of edges), an adjacency matrix can make sense.
  - Symmetric in undirected graphs.
  - Observe that it requires $O(|V|^2)$ memory, and so it only makes sense when you have $O(|V|^2)$ edges. But access $O(1)$!



	a	b	c	d	e	f	g
a	0	1	1	0	0	0	0
b	1	0	1	1	0	0	0
c	1	1	0	1	1	0	0
d	0	1	1	0	1	0	0
e	0	0	1	1	0	0	0
f	0	0	0	0	0	0	1
g	0	0	0	0	0	1	0

Adjacency matrix vs. adjacency list

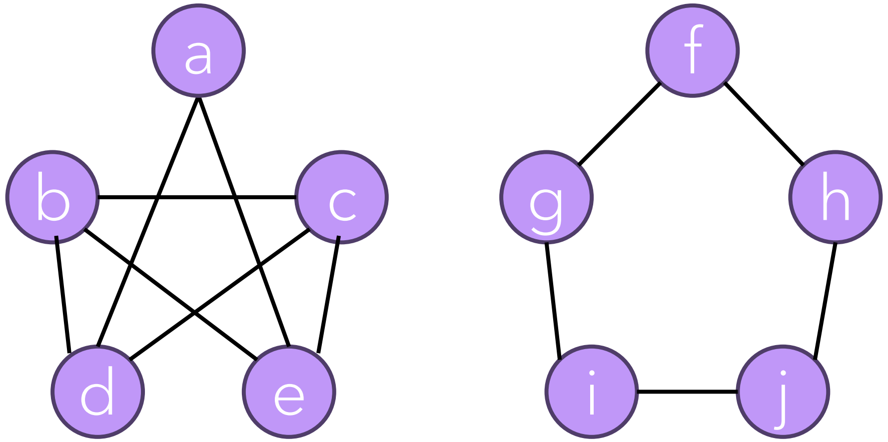
- There are two primary methods for representing a graph within a computer program.
- In most situations, an *adjacency list* representation makes more sense.
  - Requires $O(|V| + |E|)$ memory, which is vastly better for *sparse* graphs.
  - Determining if a pair of vertices is connected is slower.




```
a: b,c
b: a,c,d
c: a,b,d,e
d: b,c,e
e: c,d
f: g
g: f
```

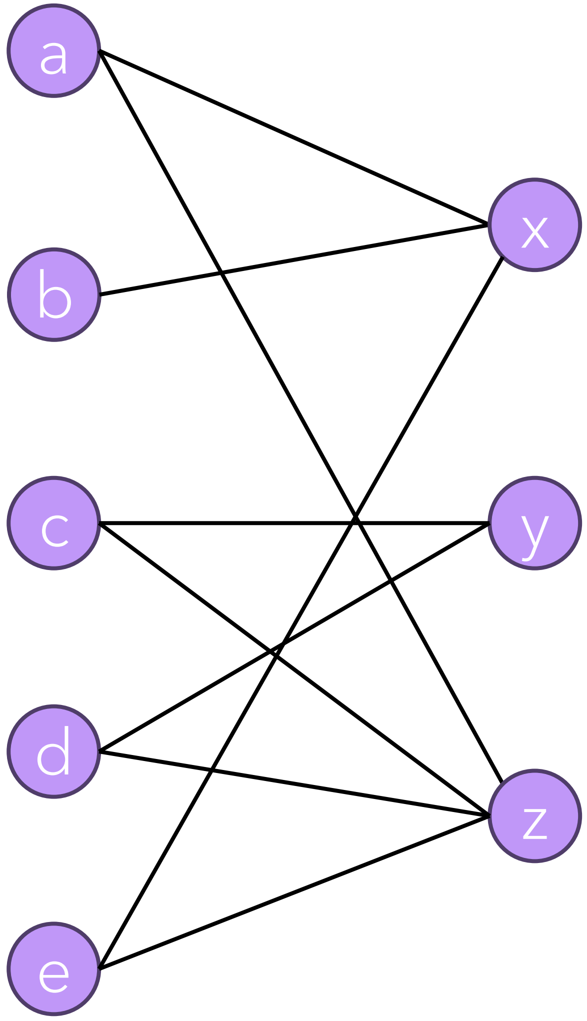
Graph isomorphism

- As previously discussed, it does not matter how one draws a graph – all that matters are what's connected to what.
- In many situations, it doesn't even matter how the vertices are labeled.
- If we can obtain the exact same list of edges from two different graphs, we say they are *isomorphic*.
  - Another way of thinking of this: If you can rearrange rows & columns to get identical adjacency matrices.



Bipartite graphs

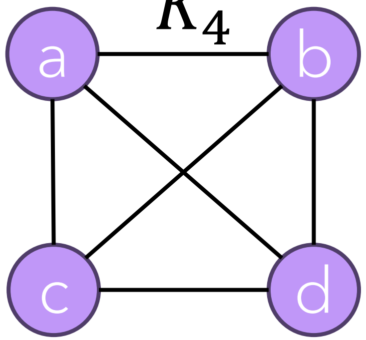
- It is a common situation where we have two different types of object that we want to model with vertices, where connections will only occur between vertices of <u>different</u> types.
  - *Example*: We want a graph to represent which students have taken courses with each professor.
- We call such a graph <u>*bipartite*</u>.



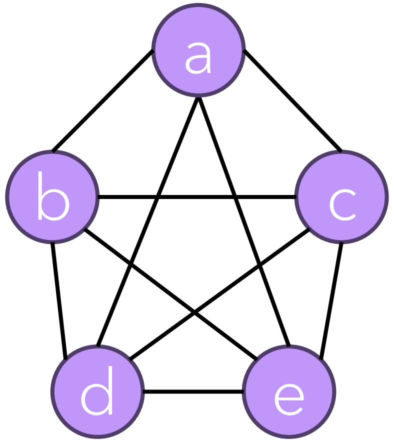
Named graphs

- Complete graphs (K_n) are graphs that contain all possible edges.
  - Complete bipartite graphs $\left(K_{m,n}\right)$ contain all edges between the two sides.
- Line graphs (L_n), star graphs (S_{n+1}), cycle graphs (C_n), and wheel graphs (W_{n+1}) are all what they say on the tin.

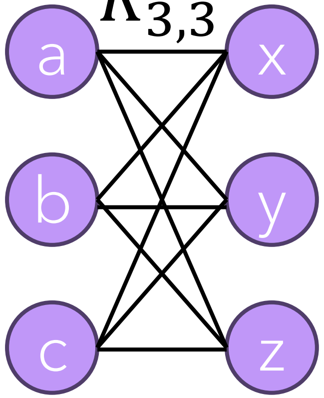
K_4

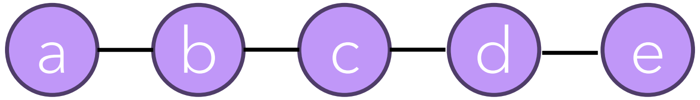


K_5

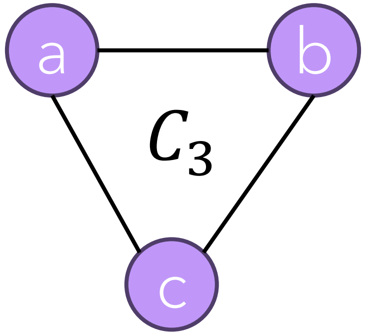


$K_{3,3}$

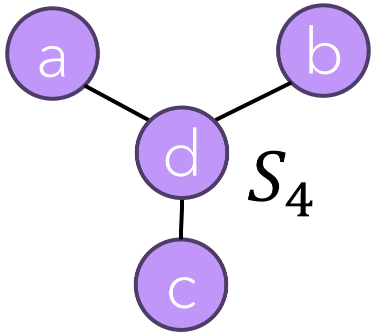




L_5

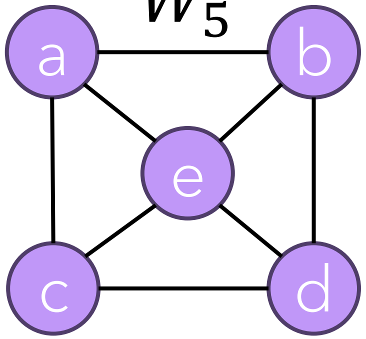


C_3



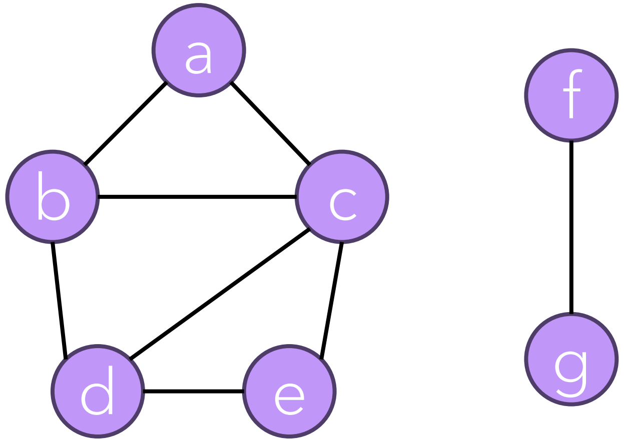
S_4

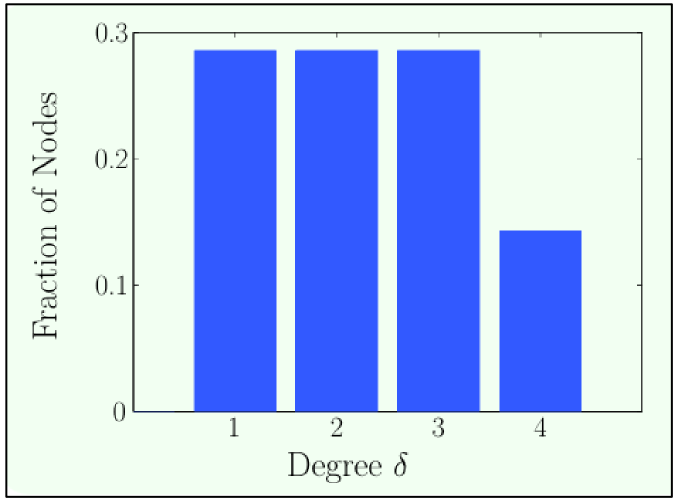
W_5



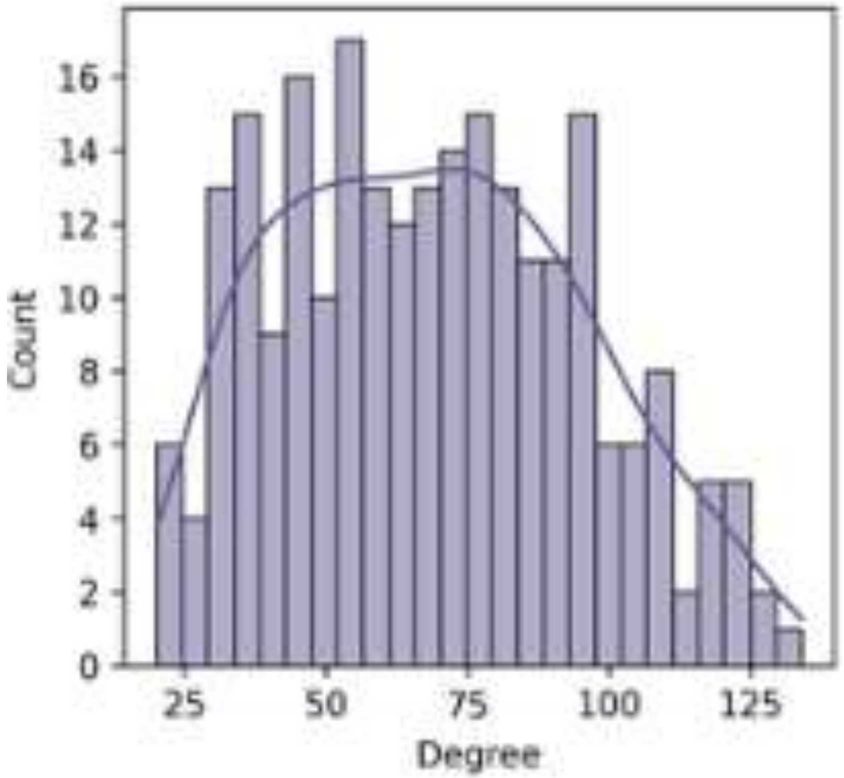
Degree sequences & histograms

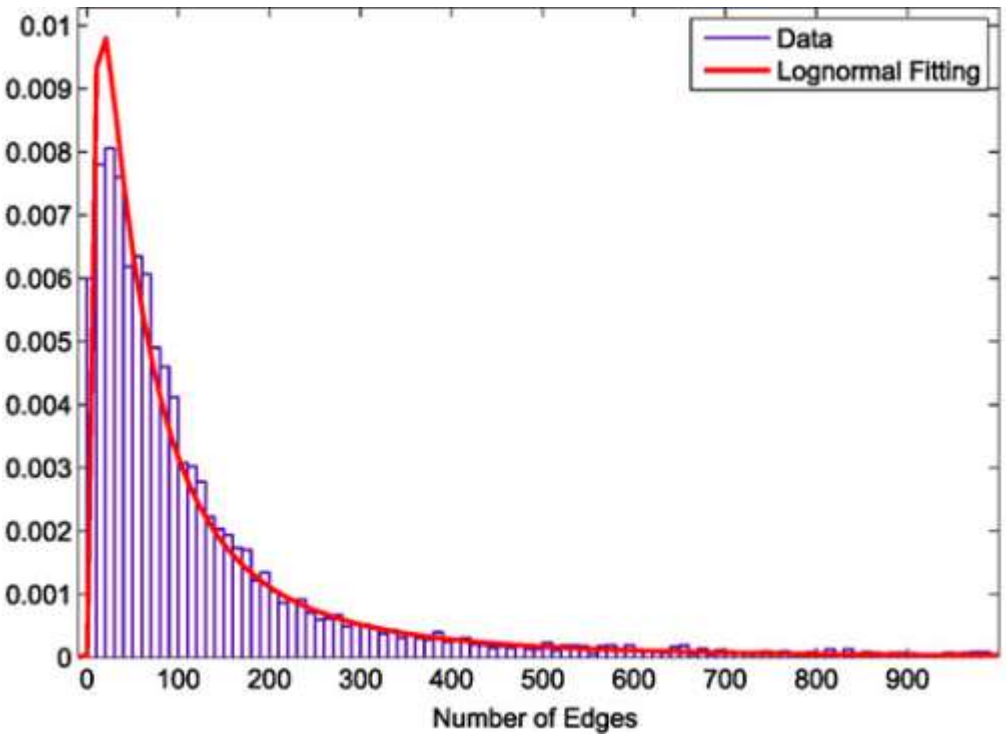
- Sometimes it can be interesting to list the degrees of the vertices in order.
  - This one is [4 3 3 2 2 1 1].
- We can also plot them on a histogram.
  - Again,
- But for large graphs...





Different degree distributions





Handshake theorem

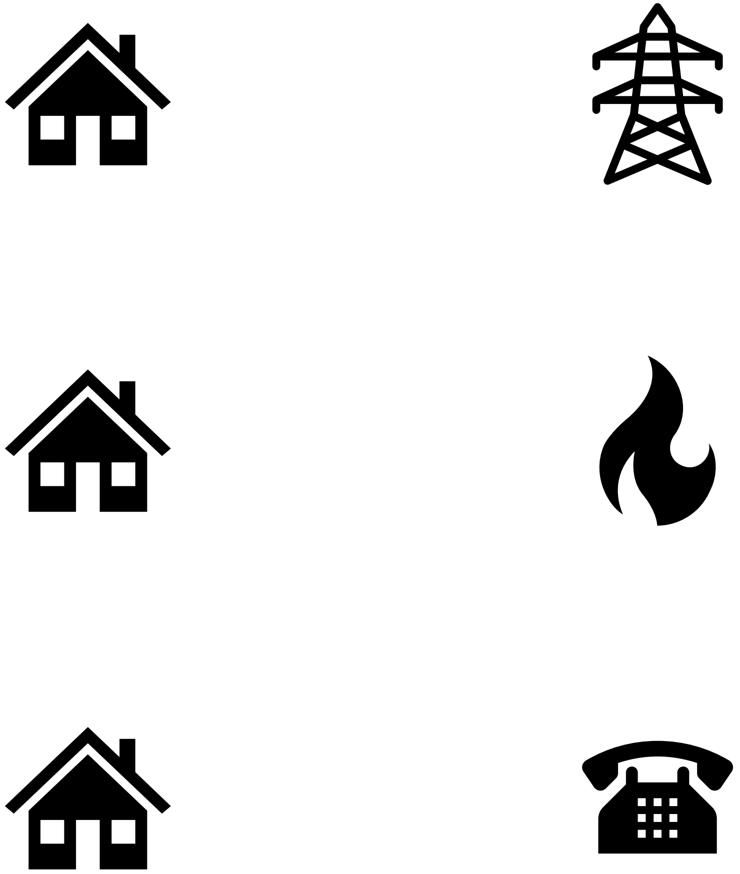
- Is it possible to construct a graph in which the degree sequence is $(3,3,3,2,1,1)$? Try it now.

- What do you get if you add up the degrees of every vertex in a graph? Why?

- Theorem: The sum of the degrees of all vertices in a graph is $2 \cdot |E|$. This holds even in multigraphs.

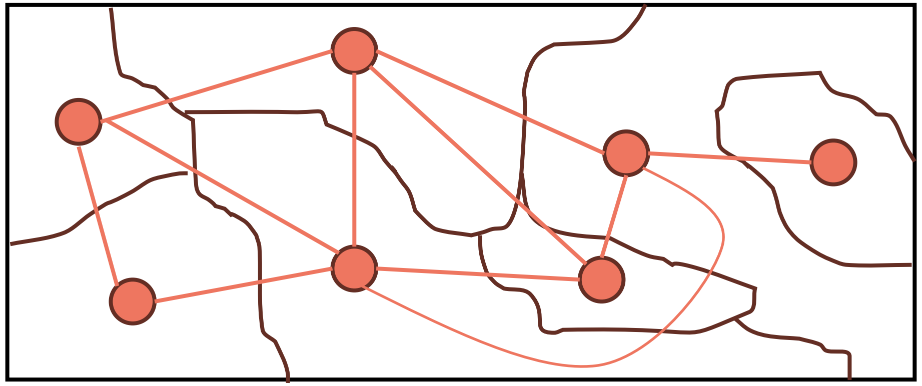
An old riddle

- There are three houses and three utilities (water, gas, phone). Each house needs to be connected to each utility. Do so without crossing any lines. Try it now!
- Any luck?



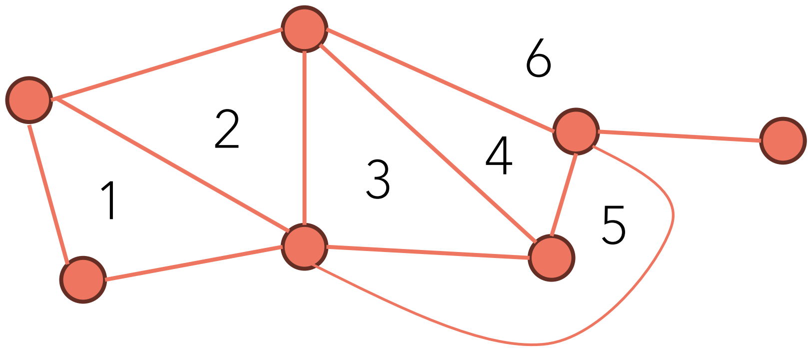
Planar graphs

- A graph that can be drawn without crossing any lines is called a *planar graph*. They come up in a variety of situations.
  - Wiring diagrams: On a circuit board or within one level of a chip, one generally cannot cross wires. (Note: Modern processors are laid out in 3-D, to an extent.)
  - Maps: Every map of territories corresponds to a planar graph in which neighbors share an edge.

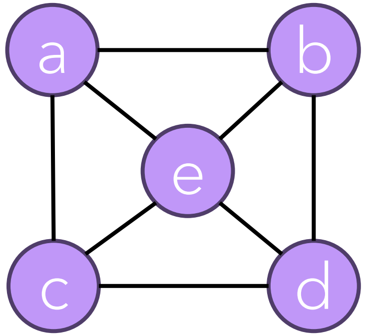


A property of planar graphs

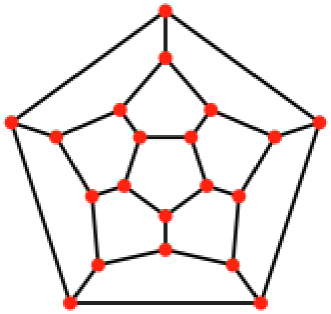
- Leonhard Euler came up with a formula about *Platonic solids* that (because topology) also describes planar graphs.
  - Define the *faces* of a planar graph as the discrete regions enclosed by cycles, plus the exterior.
  - Then $|F| + |V| - |E| = 2$.



6 faces. 6 + 7 - 11 = 2

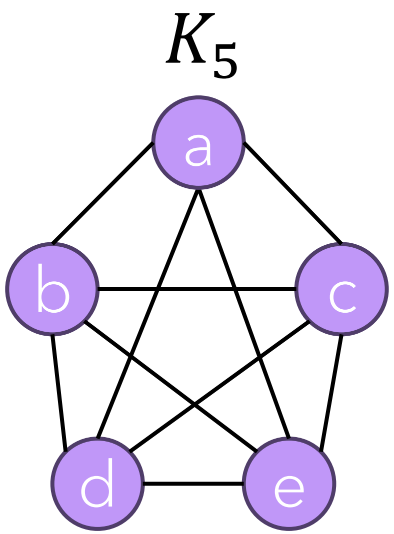


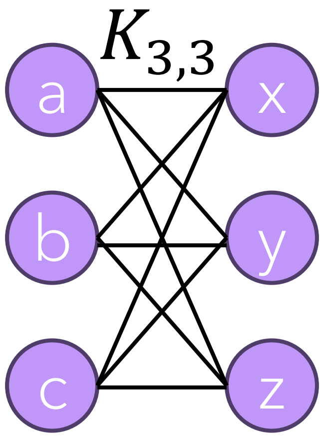
5 faces. 5 + 5 - 8 = 2


Dodecahedron. 12 faces, 20 vertices, 30 edges

Non-planar graphs

- Many graphs cannot be drawn without crossing edges. As you have already seen, $K_{3,3}$ is one of them. K_5 is another.
- In fact, in a sense, these two graphs are **<u>the</u>** non-planar graphs!
- <u>Theorem</u> (not proven here): A graph is non-planar **if and only if** it contains* one of K_5 or $K_{3,3}$.
  - "Contains" is a bit tricky here. A subgraph is certainly sufficient, but there are other non-planar graphs that must be "reduced" in a fashion before one of these shows up. ("Graph minors" is the search term, if you're interested.)

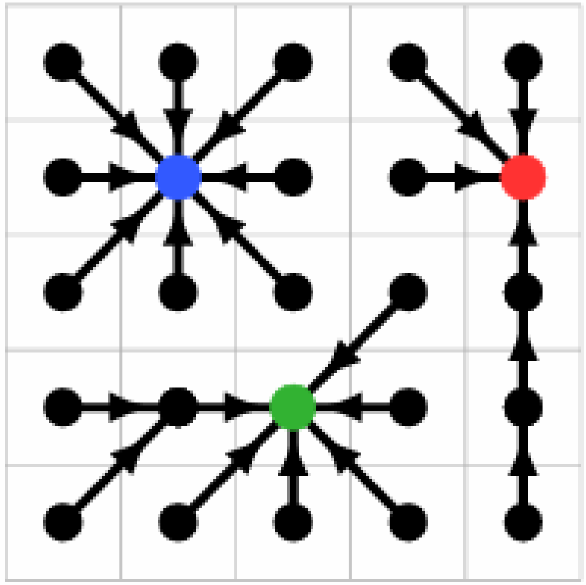




Modeling with graphs

- Consider a park where the elevation of the land is as shown in the table.
- Water, of course, runs downhill. We can roughly assume that it will take the steepest / fastest path downhill; that is, it will flow to the adjacent spot with the lowest elevation.
- Where should we put storm drains, and how much water do they need to handle?
- We need three drains, with capacities 9, 9, and 7. Easy to see once we make a graph.

3	2	7	8	7
4	1	8	9	6
5	6	7	10	8
6	3	2	11	9
8	7	8	10	12



Questions?

Please fill out the survey for today's lecture:

https://forms.gle/Lrd5mcXNRmrM7c9G6