Lecture 8 - Tree algorithms

Friday, February 3, 2023 5:46 PM

Independent set on graph G

set $S \subseteq V(G)$

s.t. $\forall u,v \in S : (u,v) \notin E(G)$

Quiz 3 p 2:

$\forall v \in V(D) : d(v) \geq 1$

$\Rightarrow \exists C_n \subseteq D$

Basis: P(1) ∞

IMPORTANT: any construction we do in an inductive proof must be able to realize all possible configurations that fit our assumptions

Weak → can't just ~~add~~ add an edge or vertex, because not all graphs have a self loop strong → can't just delete an edge or vertex, since we'd need to guarantee we don't break a cycle

---

Weighted Graphs

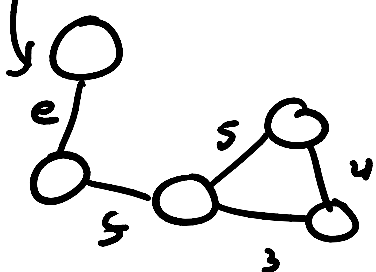
weighted $G = \{V, E, w\}$

$w = w_V$ vertex weights

$w = w_E$ edge weights

$w = \{w_V, w_E\}$ both

$w(e) = 1$



G

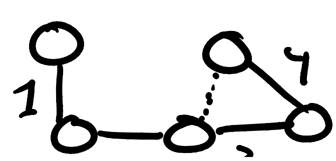
$w(f) = 2$

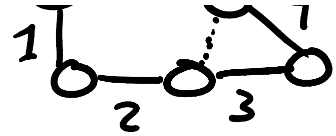
← Edge weighted graph

Minimum Spanning Tree (MST)

a spanning tree on an edge-weighted graph that has a minimum sum of weights

MST of G:



MST of G: 

To determine MST: Krushkal's Algo

output $V(T) \leftarrow V(G)$ input

$E(T) = \emptyset$

sort W, E in nondecreasing order

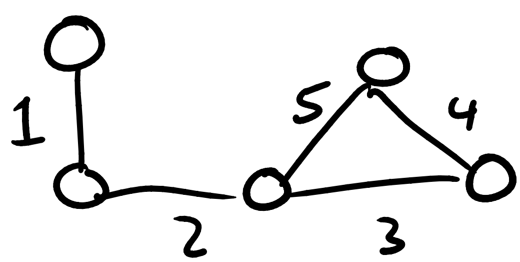
for all $w, e \in W, E$:

  if numComp$(T+e) <$ numComp(T):

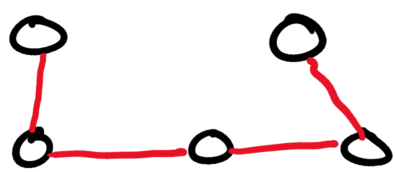
    $E(T) \leftarrow e$

  if numComp$(T) = 1$:

    break



G



MST

Prove Correctness of Krushkal's

To show: prove M, S, T

T: any edge we add will decrease

T: any edge we add will decrease the number of components

→ every edge is a cut edge

→ no edge is on a cycle

S: Assuming G is connected, we go until T is a single component

→ T contains all $v \in V(G)$

$\Rightarrow$ T is a spanning tree ✓

M: psuedo-algorithmic argument

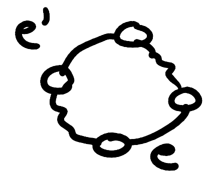
Consider: Krushkal outputs a S.T. that is <u>not</u> minimum, called T

define: T^* = actual M.S.T.

consider some $e \in E(T)$ s.t. $e \notin E(T^*)$ where e is the first such edge chosen by the algorithm

Adding e to T^* creates a cycle C

Consider $e' \in C$, $e' \notin E(T)$



Note: T^* has all edges in T that were selected before e

→ so both e and e' were available for selection by T: $w(e) \leq w(e')$

define $T' = T^* + e - e'$

Note: $w(T') \leq w(T^*)$

sum of weights

⇒ T' has more edges in common with T than T^*

⇒ Repeating this argument for all edges makes $T' \to T$ and therefore $T \to T^*$ □

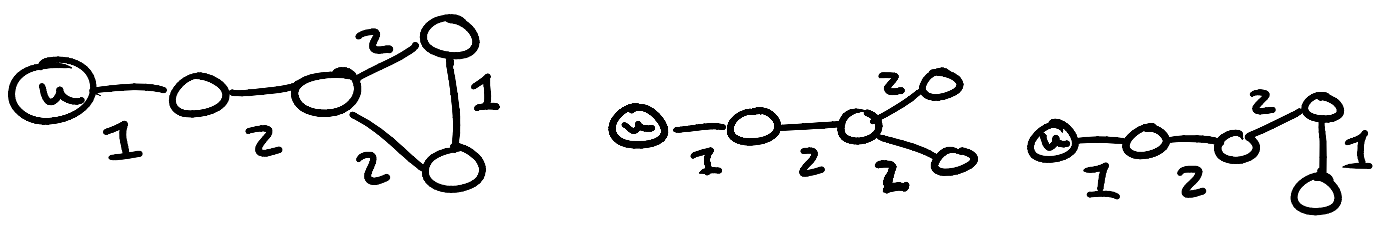
Single Source Shortest Paths (SSSPs)

(SSSPs)

$\in G$

From vertex u, identifying all $d(u,v)$ to all $v \in V(G)$

Also: consider all u,v paths as a shortest paths tree (spanning)



G SSSP tree MST

Note: SSSP tree $\neq$ MST

Dijkstra's Algorithm to solve for SSSP

$\leftarrow$ distance to v

$\forall v \in V(G): D(v) = \infty$

root $D(u) = 0$

$\leftarrow$ unvisited set

$S = V(G)$

[illegible] vertex in S with

$S = V(G)$

while $S \neq \emptyset$:

$\quad w = \min(D, S)$ ← vertex in S with minimum D value

$\quad \forall x \in N(w)$ s.t. $x \in S$:

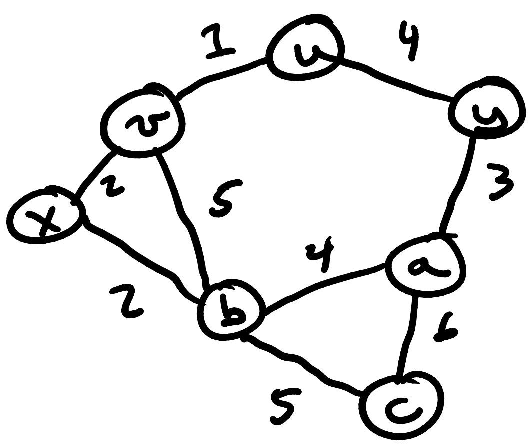
$\quad\quad t = W(x, w)$ ← weight of edge

$\quad\quad$ if $D(w) + t < D(x)$:

$\quad\quad\quad D(x) = D(w) + t$

$\quad S = S - w$

Example of Dijkstra in action



	D_0	D_1	D_2	D_3	D_4
$\boxed{u}$	0	0	0	0	
$\boxed{v}$	∞	1	1	1	
$\boxed{x}$	∞	∞	3	3	
y	∞	4	4	4	
a	∞	∞	∞	∞	
b	∞	∞	6	5	
c	∞	∞	∞	∞	

Exercise 4 reader

Prove Correctness of Dijkstra

At every iteration:

X: visited set of vertices

actual

X: visited set of vertices

1 - $\forall v \in X$ $D(v) = d(u,v)$ ← actual shortest u,v-path

2 - $\forall v \notin X$ $D(v)$ is shortest u,v path from X

We'll do weak induction on $|X|$

Basis $P(1)$: $X = \{u\}$ $D(u) = d(u,u) = 0$

all $v \in N(u)$ take the edge weight from u

$P(k) \Rightarrow |X| = k$

assume via I.H. that the above two conditions holds

$P(k+1) \Rightarrow X' = X + v$

v is selected s.t. $D(v)$ is least for all $v \notin X$

First show: $D(v) = d(u,v)$

By I.H. → shortest path directly

By I.H. → shortest path directly from X to v is $D(v)$, so any other possible path that exits X and reaches v is bounded below by $D(v)$

Secondly show: $D(w)$ is correct for $X' = X + v$, $w \notin X'$

By I.H. → $D(w)$ is shortest u,w-path distance directly from X

→ we update $D(w) = \min\left(\underbrace{D(w)}_{\text{distance from } X}, \underbrace{D(v) + w(v,w)}_{\text{distance from } X'}\right)$

⇒ shortest possible path to w through X', as v is the the only way to get to w through a vertex not originally in X □