CS 124 Lecture 7 — Feb 17, 2020

Minimum Spanning Trees!

Cut property

Let $X \subseteq T$ where T is a MST.

Let $S \subset V$ w/ no edge in X crossing between S and $V-S$.

Let e be a minimum weight edge crossing between S and $V-S$.

Then $X \cup \{e\} \subseteq T'$ for an MST T'.

$X = \{\}$

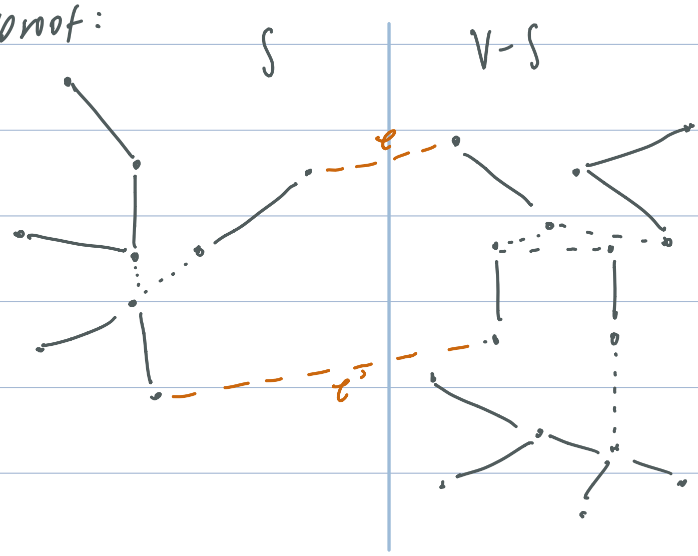
repeat until $|X| = n-1$

pick $S \subseteq V$ w/ no edge crossing, S, $V-S$

find lightest edge e crossing S, $V-S$

$X := X \cup \{e\}$

proof:



e = lightest edge

Pf by contradiction.

- suppose $e \notin T$
- Take $T \cup \{e\}$, we get a cycle.
- Tree has an edge e' crossing S, $V-S$ on the cycle

- Let $T' = T \cup \{e\} - \{e'\}$ spanning tree, same # edges by disconnecting and reconnecting cycle

- T is a tree, connected $n-1$ edges

$W(T') = W(T) - W(e') + W(e)$ $\qquad$ $W(e) \le W(e')$

$\le W(T)$ $\qquad$ $=$ if $W(e') = W(e)$

① If $W(e) < W(e')$: $W(T') < W(T)$

So $e \in T$, But T is a MST → contradiction.

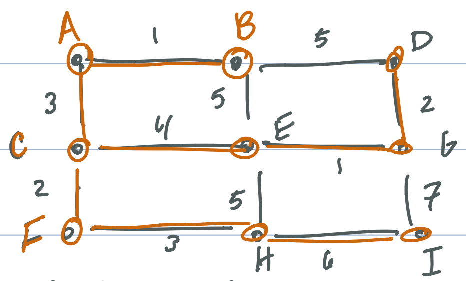
② If $W(e) = W(e')$: $W(T') = W(T)$

So T is also a MST. ∎

Prim's Alg

Grow tree one vertex at a time

Prim's Tree = single growing component



A	$S - A$
A, B	$S - \{A, B\}$
A, B, C	$S - \{A, B, C\}$
⋮	

~~B:1~~, ~~C:3~~

D: ~~5~~ 2, ~~E:5 4~~ ~~G:1~~

~~F:2~~ ~~H:3~~ ~~I:6~~

Algo.

$H = \{s:0\}$

for $v \in V$:

$\quad$ $dist[v] := \infty$, $prev[v] := nil$ $\qquad$ init array for each vertex

$dist[s] := 0$

while $H \ne \emptyset$ $\qquad$ satisfies the cut property

$\quad$ each vertex will be deleted

$\quad$ $v :=$ deletemin, $S := S \cup \{v\}$

$\quad$ for $(v, w) \in E$, $w \in V - S$:

if (dist[w] > length(v,w)):

dist[w] = length(v,w), prev(w) := v

(insert[w], dist[w], H)

runtime: analysis is the same as Dijkstra's Algorithm

Bin $\longrightarrow$ $O(|E| \cdot \text{insert} + |V| \cdot \text{deletemin})$

$O((|E| + |V|)\log v) = O(|E| \cdot \log v)$

List $\longrightarrow$ $O(|E| + |V|^2) = O(|V|^2)$

Kruskal's Algorithm

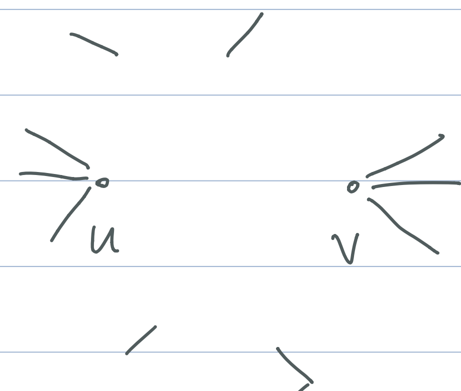
Sort all edges   implicit set

sorting m edges

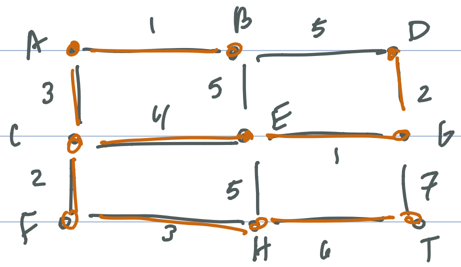
Go through from smallest $\rightarrow$ largest

Add e unless it creates a cycle

(stop at n-1 edges)



claim to draw cut where $u \leftrightarrow v$ is smallest cut



runtime:

sort(m) + $O(m \log^* n)$

$\log^* n$ = number of repeated $\log_2$ till you get to a number ≤ 1

$\log^* 1 = \log 1 = 0$

$\log^* 2 = 1$ $\log^* \underline{16} = 3$

$\log^* 4 = 2$ $\log^* \underline{2^{16}} = 4$

$\log^* \underline{2^{2^{16}}} = 5$

largest value

Alg. Checking if edge creates cycle is hard

X = { E, sorted by weight }

for $u \in V$:

makeset (u)

for $(u,v) \in E$ in increasing order:

if find(u) $\neq$ find(v): check

$X = X \cup \{(u,v)\}$

union (u,v) merge

disjoint set data structure

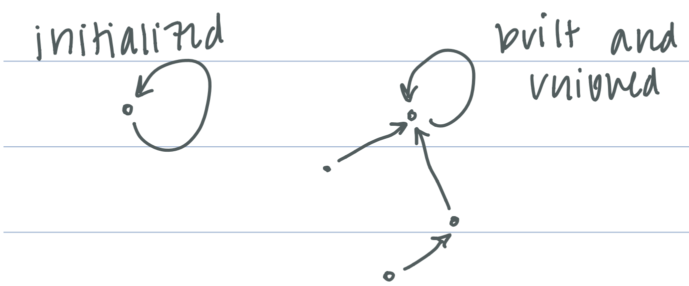
makeset (x): new set containing x

find(x) = give me name of set containing x

union(x,y) = replace sets containing x, y with union

using an array: O(1), O(1), O(n) union (x, y) = link (find(x), find(y))

implement sets as directed trees:



link(x, y) = x, y are roots, join one root to another to form single tree

Alg. rank is like "height"

PROC MAKESET (x)

p(x) := x

rank (x) := 0

PROC FIND (x)

if (x ≠ p(x)):

PROC LINK (x, y):

If rank(x) > rank(y):

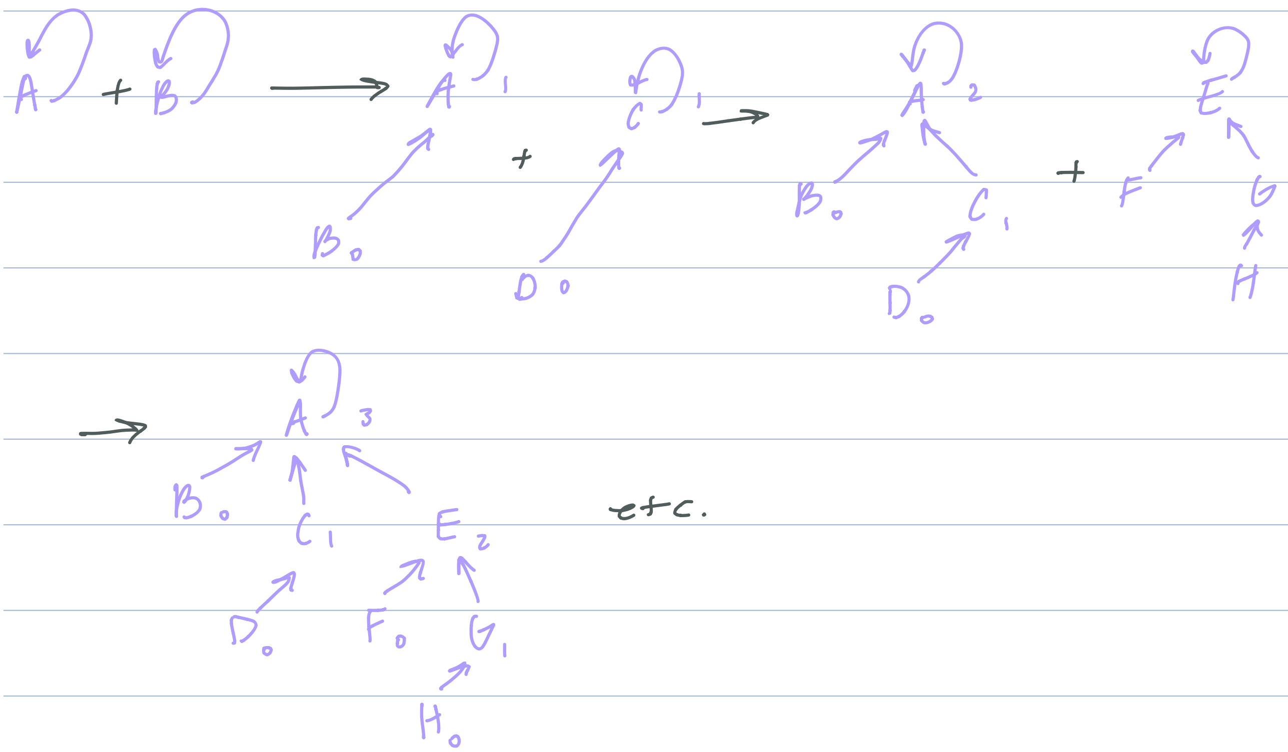
switch (x, y)

if rank(x) = rank(y):

rank(y) := rank(y) + 1

$p(x) := \text{FIND}(p(x))$ $\quad\quad$ $p(x) := y$

RETURN $(p(x))$



properties

0. Rank fixed when you're not a root.
1. If $v \neq p(v)$ then $\text{rank}(p(v)) > \text{rank}(v)$
2. If $p(v)$ is updated for v, $\text{rank}(p(v))$ increases.
3. # of elements of $\text{rank}(k) \leq n/2^k$

   induction on k, repeated pairing is best you can do
4. # of elements of rank $\geq k$ is at most $n/2^{k-1}$

   follows from 3, geometric sequence of rate $1/2$