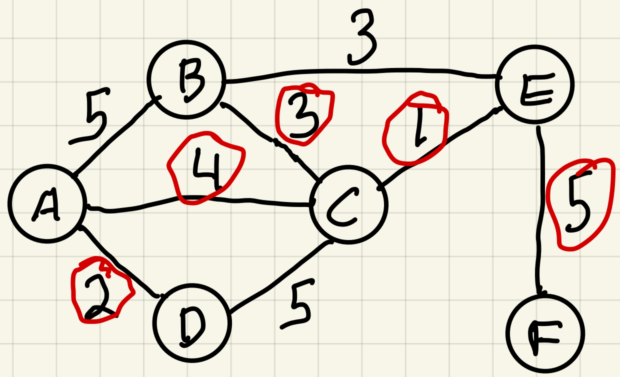
Lecture #9

CS 170

Spring 2021

Introduction to Minimum Spanning Trees (MSTs)



Given an undirected graph $G=(V,E)$ with edge weights $w(e)>0$

Find a subset $T \subseteq E$ such that

① (V,T) connected

② sum of weights of $T = \sum_{e\in T} w(e)$ minimized

Fact: T has no cycles, i.e a tree

What would be a greedy algorithm?

add cheapest edge to T as long as no cycle

|

Properties of Trees

Def: An undirected graph $T(V, E)$ is a tree if

1) T is connected, and 2) T has no cycles

Note: Any vertex could be root

Claim: Any 2 of following 3 properties implies the 3rd:

1) T is connected
2) T has no cycles
3) $|E| = |V| - 1$

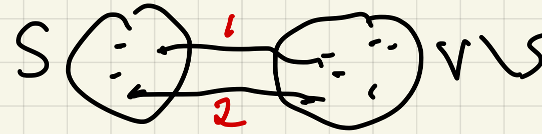
Proof: 1) and 2) $\Rightarrow$ 3)

pick any vertex of to be root, run DFS,
every vertex has one parent, except root

$\Rightarrow |E| = |V| - 1$

2) + 3) $\Rightarrow$ 1) start with no edges, $|V|$ disconnected vertices $\Rightarrow$ $|V|$ connected comps. Add an edge either # comp drops by 1, or got a cycle $\Rightarrow$ $|E| = |V| - 1$ 2

Cuts in a graph

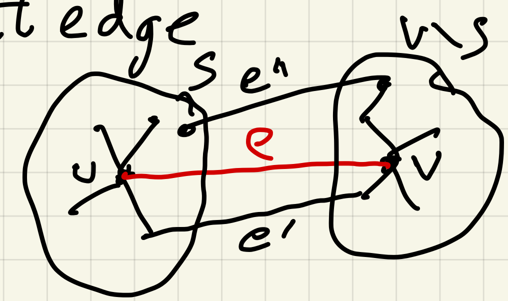


Def: A cut in $G(V,E)$ is a partition $V = S \cup (V\setminus S)$

Also refers to edges connecting S and $V\setminus S$

Claim: Lightest edge (smallest $w(e)$) in a cut appears in some MST

Proof: Let T be a MST, $e=(u,v)$ be light edge connecting S and $V\setminus S$



$e' \in T$ connects S and $V\setminus S$

but $e' \neq e$ (e' not unique)

Want MST containing e: $T \cup \{e\} \Rightarrow$ too many edges $\Rightarrow$ cycle

e' = be edge in cycle in T connecting S and $V\setminus S$

$T' = T \cup \{e\} \setminus \{e'\}$: claim T' is a tree (right # edges, connected)

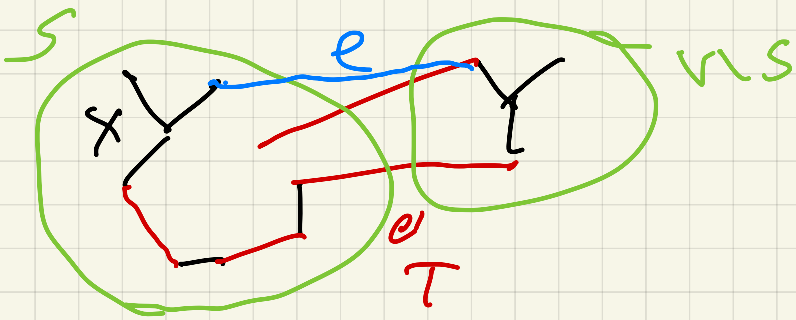
$$w(T') = \sum_{e \in T'} w(e) = w(T) + w(e) - w(e') \le w(T) \text{ since } w(e) \le w(e')$$

$\Rightarrow T'$ a MST

How to add one more edge to a partial MST

Claim: Suppose $X \subseteq E$ and $X \subseteq T$ where T is some MST.
Suppose X has no edges connecting S and $V \setminus S$
and e is lightest edge connecting S and $V \setminus S$.
Then $X \cup \{e\} \subseteq T'$ where T' is some MST.

Proof:



$$T' = \underbrace{T \cup \{e\}}_{\text{has cycle}} \setminus \underbrace{\{e'\}}_{\text{breaks cycle}}$$

is MST by same argument

MST Algorithm

Meta-Algorithm:

$X = \emptyset$
repeat
  pick cut $(S, V \setminus S)$ s.t. X doesn't cross cut
  add edge e with smallest weight in cut to X
until $|V|-1$ edges added (or graph connected)

Kruskal

$X = \emptyset$
sort all e by $w(e)$
for all e in increasing order
  if $X \cup \{e\}$ has no cycle, $X = X \cup \{e\}$

Kruskal's Algorithm is Correct

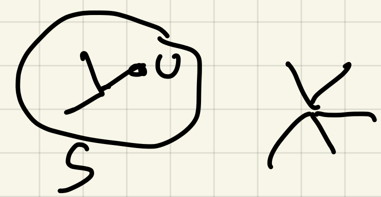
$X = \emptyset$
sort all $e \in E$ by $w(e)$
for all $e \in E$ in increasing order
  if $X \cup \{e\}$ has no cycle, $X = X \cup \{e\}$

Claim: At any point X is a subset of some MST T

Proof: Induction base case: adding first edge OK (Slide 3)

If Kruskal adds $e = (u, v)$ to X,
  no cycle in $X \cup \{e\}$

$S =$ connected comp. of u in X



Can there be a lighter edge e' connecting S and $V \setminus S$?
If there were, would have been considered already and not chosen it, contradiction, because it also would not have created cycle $\Rightarrow w(e') \geq w(e)$

by Slide 4, $X \cup \{e\} \subseteq$ some MST T

Implementing Kruskal (1)

Cost

$X=\phi$, sort all $e\in E$ by $w(e)$ $\qquad O(|E|\log|E|)=O(|E|\log|V|)$

for all $e\in E$ in increasing order

 if $X\cup\{e\}$ has no cycle, $\qquad$ Naive: DFS $O(|V|)$ per loop

  $X=X\cup\{e\}$ $\qquad \Rightarrow O(|E|\cdot|V|)=O(|V|^3)$

$X=\phi$, sort e

for all $v\in V$, makeset(v) ... each set is a conn. comp

for all e in order ... $e=(u,v)$

 if find$(u)\neq$ find(v) ... find(u) = name of u's conn. comp

  union(u,v) ... merge conn. comps of u and v

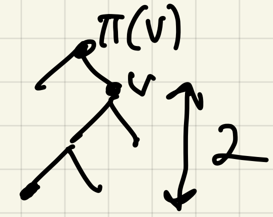
Cost: $|V|\cdot\text{cost(makeset)}=O(|V|)$

$\left.\begin{array}{l} |E|\cdot\text{cost(find)}=O(|E|\log|V|) \\ |V|\cdot\text{cost(union)}=O(|V|\log|V|) \end{array}\right\}$ or $\underbrace{O(|E|\cdot\log^*|V|)}_{\text{dominated by sorting}}$

$2^{2^{2^{2^{2}}}}\geq n$ $\quad$ #2s $\geq\log^* n$

$\log\cdots\log\log_2 n\leq 1$ $\qquad \log^*(n)=\#\log s$

Implementing Kruskal (2)



For each $v \in V$ add

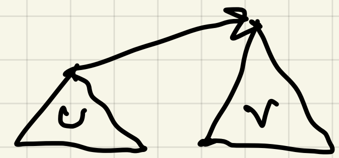
$\pi(v)$ = "parent of v" = pointer to parent in tree defining connected component to which v belongs

rank(v) = height of subtree rooted at v

$\forall v \in V$ makeset(v): $\pi(v) = v$, rank(v) = 0

find(v) while $\pi(v) \neq v$, $v = \pi(\pi)$, return v

union(u,v)



make root of shorter tree point to root of taller tree

$\Rightarrow$ all trees have depth $O(\log |V|)$

$\Rightarrow$ find, union each cost $O(\log |V|)$

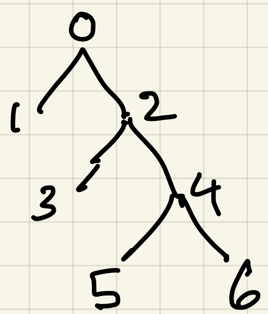
---

ranks all fit in in some set

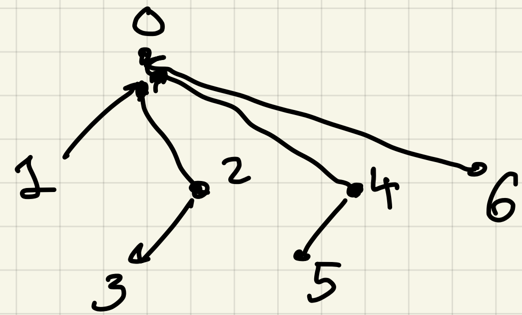
$\{1\}, \{2\}, \{3,4\}, \{5 \ldots 16\}, \{17, \ldots, \underset{65536}{2^{16}}\}, \{2^{16+1}, \ldots 2^{2^{16}}\}, \ldots$

Implementing Kruskal (3)

Even better: when doing find, make all vertices on path to root point to root



find(6) = 0



find(v)
  if $v \neq \pi(v)$ $\pi(v) = \text{find}(\pi(v))$
  return $\pi(v)$

all paths to root keep getting shorter

$\Longrightarrow O(|E| \cdot \log^* |V|)$ cost of all finds and unions

MST Algorithm

Meta-Algorithm:

$X = \emptyset$

repeat

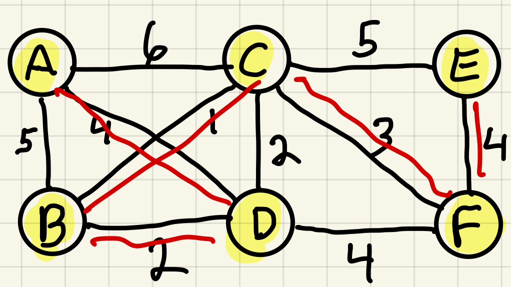
  pickcut$(S, V \setminus S)$ s.t. X doesn't cross cut

  add edge e with smallest weight in cut to X

until $|V|-1$ edges added (or graph connected)

Prim: S = vertices touched by X

{Kruskal: S connected comp of v in X}



S	A	B	C	D	E	F
{ }	0/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
A		5/A	6/A	4/A add	∞/nil	∞/nil
A,D		2/D add	2/D		∞/nil	4/nil D
A,D,B			1/B add		∞/nil	4/nil D

Implementing Prim

```
for all v∈V
    cost(v) = ∞ , prev(v) = nil
pick any initial v0, cost(v0) = 0

H = makequeue(V) ... priority queue, based on cost()
while H ≠ ∅
    v = deletmin(H) ... pick v with lowest cost()
    for each (v,u) ∈ E
        if cost(u) > w(v,u)
            cost(u) = w(v,u)
            prev(u) = v
```

cost(Prim) = cost(Dijkstra)

only difference is value used by priority queue