CS 170

Lecture 9

- use iClicker

Sanjam Garg.

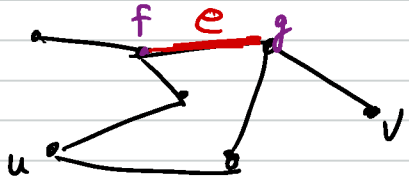
join.iclicker.com/RGWR

Minimum Spanning Tree (MST) — Kruskal and Prims

Definition (Tree): An undirected graph that is (i) connected (ii) acyclic.

Property 1: Removing a cycle edge e in a connected graph does not disconnect it.

Proof:



$\forall u, v \rightarrow G' = (V, E - e) \rightarrow$ it is still connected.

$u \rightsquigarrow v$ in G

Case I: e is not on this path. Then the same path is also in G'.

Case I: e is on this path. $u \rightsquigarrow f \rightsquigarrow g \rightsquigarrow v$ (other side of the cycle)

$\therefore u \rightsquigarrow v$ in G' as well.

Property 2: A tree with n nodes has $n-1$ edges.

Proof:

add this edge → $i = 1$: # of connected components $n-1$

$i = 2$: $n-2$

$\vdots$

$i = n-1$: 1

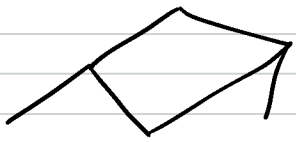
□

Property 3: A connected graph G on n nodes & $n-1$ edges is a tree.

Proof:

→ G has no cycles (to prove)

→ assume, that G has a cycle



By property 1, we can remove one of the edges

- it would still be connected
- $n-2$ edges.

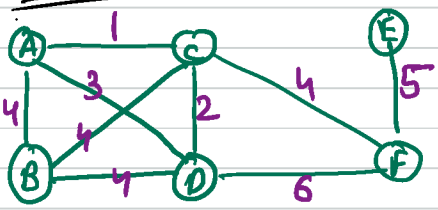
→ contradict property 2.

Minimum Spanning Tree (MST) — Kruskal and Prims

Input: An undirected connected $G=(V,E)$ and weights w_e
Goal: Find tree $T=(V,E')$ s.t. $E' \subseteq E$ s.t.

$$\text{weight}(T) = \sum_{e \in E'} w_e \text{ is minimized.}$$

Example:

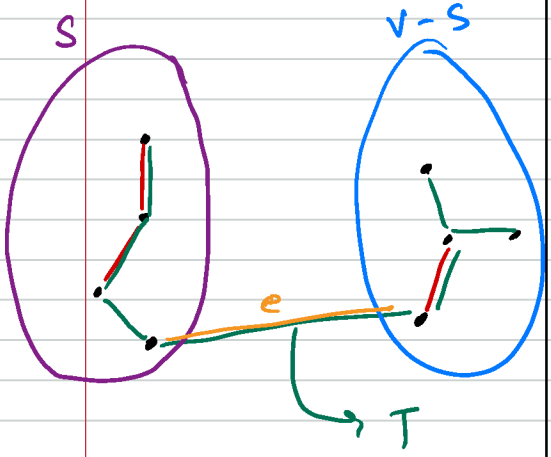


Greedy Approach:
Add the least weight edge not introducing a cycle; and iterate.

Main theorem #

(i) Let $X \subseteq E$ be part of some MST of G → call it T
(ii) $S \subseteq V$ be a set s.t. there are no edges in X from S to $V-S$
(iii) Let $e \in E$ be the lightest edge from S to $V-S$

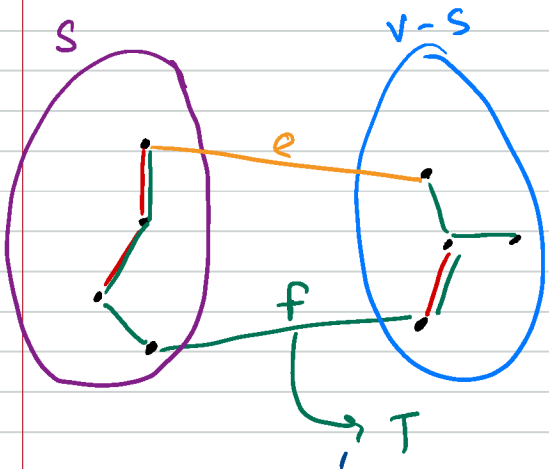
$\Rightarrow$ $X+e$ is part of some MST, of G → will construct



Case I: $e \in T$

$X + e \subseteq T$ → implies (iii)

Case II: $e \notin T$ $\quad w_e \le w_f$ — ①



Part I: Let's consider $T' = T + e - f$

$$cost(T') = cost(T) + w_e - w_f$$

From ① $cost(T') \le cost(T)$

T is an MST. Thus, $cost(T') = cost(T)$

Part II: T' is a tree.

(1) By Property 1, removing f from $T+e$, T' is still connected.

(2) T' has $n-1$ edges
$\Rightarrow$ Property 3 $\rightarrow$ T' is a tree.

Part III: $X + e \subseteq T'$

$f \notin X$

$X + e \subseteq T + e - f$

$= T'$

Kruskal: ① Go over edges in order of increasing weights
② Add it if doesn't introduce a cycle and skip otherwise.

Claim: Kruskal finds a MST.

Induction over the number of added edges.

(Base) $X = \phi \rightarrow$ part of every MST

(IS) $X \rightarrow X + e$

Implementation
- Need to keep track of connected components
- Find if an edge introduces a cycle.

Union-Find Data Structure

Makeset (x) : makes singleton set containing element x

Find (x) : find the set x belongs to.

Union (x,y) : take the union of sets containing x & y

Kruskal (G, w)

```
For all v ∈ V, makeset(v)
X = ∅
Sort edges in E by w
∀ (u,v) ∈ E in that order
    if find(u) ≠ find(v)
        X = X ∪ {(u,v)}
        union(u,v)
return X
```

$|V|-1 \leq |E| \leq \binom{|V|}{2}$

$|E| \log |E|$ sorting

$|V| \times$ makeset $\rightarrow O(1)$

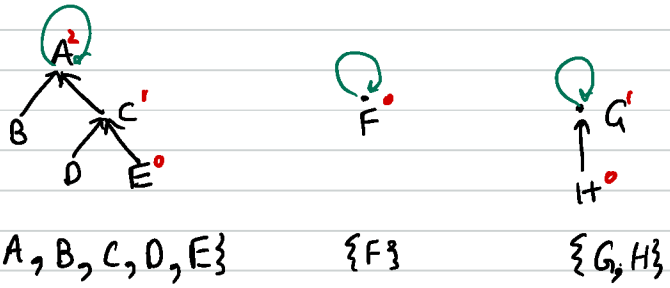
$2|E| \times$ Find $\rightarrow O(\log |V|)$

$|V|-1 \times$ unions $\rightarrow O(\log |V|)$

$O((|V| + |E|) \log |V|)$

Union Find Data Structure

Idea: Each set stored as a tree and labeled by its root.



$\{A, B, C, D, E\}$ $\{F\}$ $\{G, H\}$

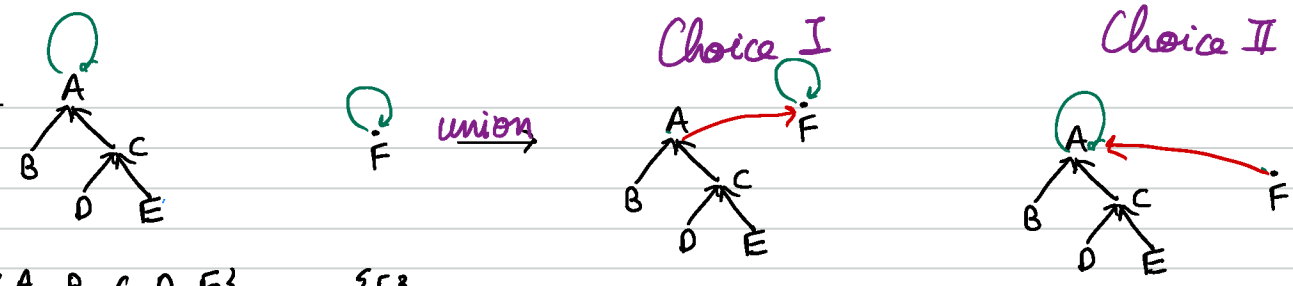
$\pi(x)$ = parent of x

rank(x) = height of tree under x

makeset(x)
- Set $\pi(x) = x$
- rank$(x) = 0$

find (x)
- if $\pi(x) \neq x$
  - find $(\pi(x))$
- else return x

Union



$\{A, B, C, D, E\}$ $\{F\}$

$\{A, B, C, D, E, F\}$

Observation: Shallow tree $\Rightarrow$ faster find.

union (x, y)
- r_x = Find (x) r_y = Find (y)
- if rank$(r_x) \leq$ rank(r_y)
  - $\pi(r_x) = r_y$
  - if rank(r_x) = rank(r_y) then rank(r_y) = rank(r_y) + 1
- else $\pi(r_y) = r_x$

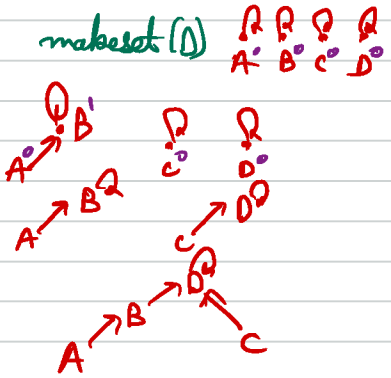
Example:

makeset(A) ... makeset(D)

union (A, B)

union (C, D)

union (A, D)



Running time :

makeset	$\rightarrow O(1)$
find	$\rightarrow O(\text{rank of root})$
union	$\rightarrow O(\text{rank of the roots})$

Claim : If $\text{rank}(x) = r$ then x has $\geq 2^r$ nodes in tree rooted at r.

Base : $r = 0$ # of nodes $1 \rightarrow 2^0 \geq 1$

$r \rightarrow r+1$ # of nodes from first tree + # " " " second "

$\geq 2^r + 2^r \geq 2^{r+1}$

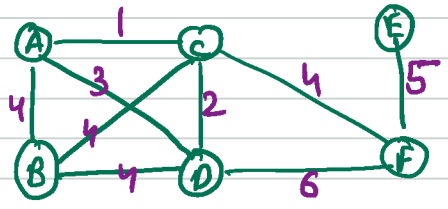
$\therefore$ max rank of any root $\leq \log n$

Prim's Algorithm

General process based on the theorem (proved few slides back)

$X = \phi$
Repeat till $|X| = n-1$
Pick $S \subseteq V$ s.t there are no edges in X between S & $V-S$
Let e be the minimum weight edge from S to $V-S$
Set $X = X \cup \{e\}$

Prims works analogous to Dijkstra's



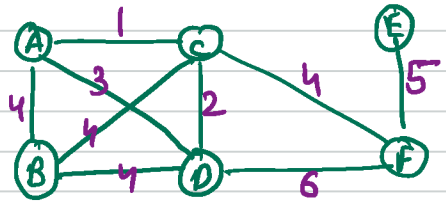
Prim (G, w)

```
∀ u ∈ V set cost(u) = ∞ , prev(u) = nil

pick any initial node u_0
cost(u_0) = 0

∀ v ∈ V   insert ( v, cost(v) )        // priority queue.
while queue is non-empty
    v = Delete Min()
    ∀ (u,v) ∈ E
        if cost(u) > w(u,v)
           cost(u) = w(u,v)
           prev(u) = v
           Decrease Key(u)
```

$O(|V|)$ inserts & deletes / $O(|E|)$ decrease keys



S	A	B	C	D	E	F
{}	0	∞	∞	∞	∞	∞
{A}	0	4	1	3	∞	∞
{A, C}	0	4	1	2	∞	4
⋮						