Lecture 9
CS 170

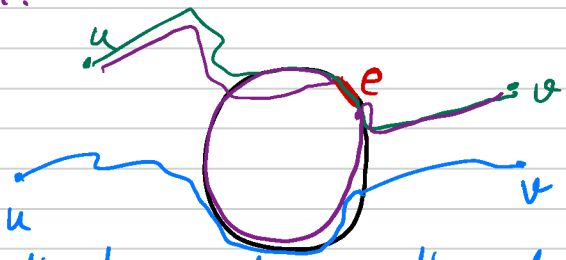
Sanjam Garg.

Minimum Spanning Tree (MST) — Kruskal and Prims

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

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

Proof:

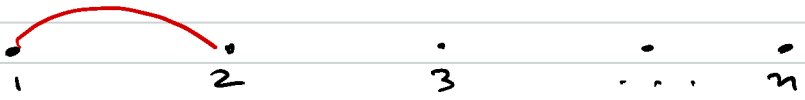


Case I → $u \to v$ path does not use the edge e (done)

Case II → $u \to e \to v$ → Constructed an alternate path □

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

Proof:



$t=0 \rightarrow n$ components

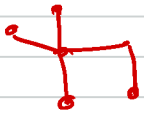
1st edge $\rightarrow n-1$ components

$\vdots$

$i \rightarrow$

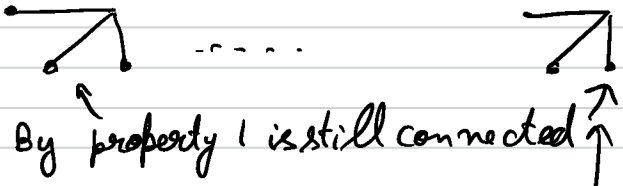
$n-i$ disconnected components

$n-1$ edges $\rightarrow$



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

remove cycle edge $\rightarrow$



Property 2 implies that this graph has $n-1$ edges.

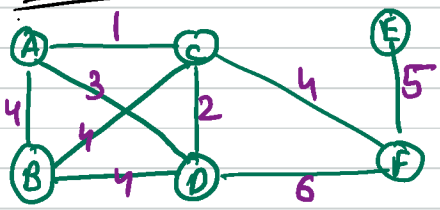
There were no edges to remove to begin with!

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:



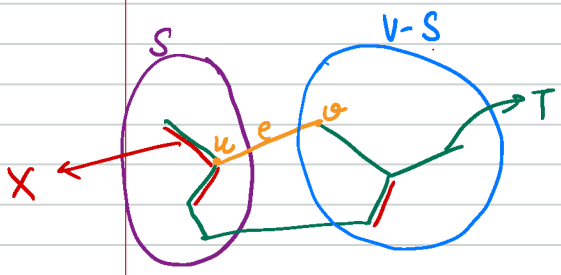
Greedy Approach:

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

Main theorem #

(i) Let $X \subseteq E$ be part of some MST (call it T) of G
(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 $V-S$

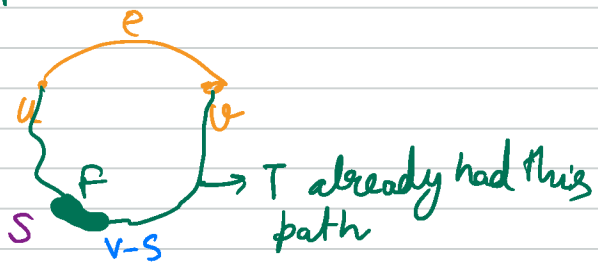
$\Rightarrow$ $X+e$ is part of some MST, of G (will need to construct)



$T+e$ is a MST ← lowest weigh edge from S to $V-S$

Case I: $e \in T \Rightarrow X+e \in T$ (done)

Case II: $e \notin T$



(1) ——— $w_f \geq w_e$ → given that e is the lightest edge.

$T' = T + e - f$ (i) By property 1, T' is connected

(ii) $n-1+1-1 = n-1$ $\xrightarrow{\text{property 3}}$ T' is a tree.

$$cost(T') = cost(T) + w_e - w_f \quad \text{by (1)}$$

$$\Rightarrow cost(T') \leq cost(T)$$

$$\Rightarrow cost(T') = cost(T)$$

$X \subseteq T$, $f \notin X$, $e \in T'$

$X + e \subseteq T'$ → a MST

Kruskal: ① Go over edges in order of increasing weights

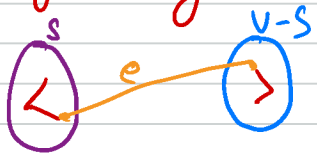
② Add it if doesn't introduce a cycle and skip otherwise.

Claim: Kruskal finds a MST.

Induction over the number of added edges.

(Base) $X = \phi$ → part of every MST

(IS) $X \to X + e$



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
```

$|E| \log |V|$ sorting

$|V|$ × makeset → $O(1)$

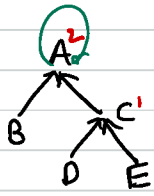
$2|E|$ × Find → $O(\log |V|)$

$|V|-1$ × unions → $O(\log |V|)$

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

Union Find Data Structure

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



$\pi(x)$ = parent of x

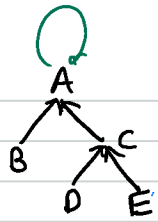
rank(x) = height of tree under x

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

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

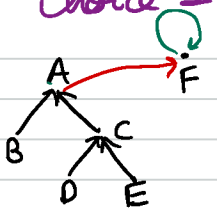
find (x)
- if $\pi(x) \neq x$
  - find($\pi(x)$)
- else return x

Union

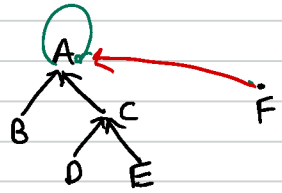


union →

Choice I



Choice II



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

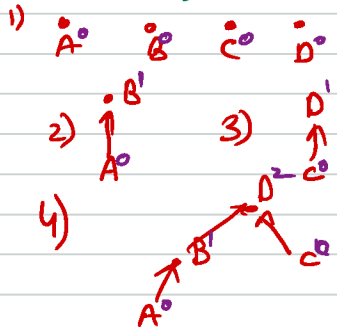
$\{A, B, C, D, E, F\}$

Observation: Shallow tree ⇒ faster find.

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

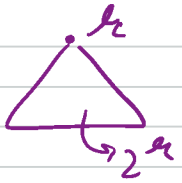
Example:

1) makeset(A) ... makeset(D)
2) union(A, B)
3) union(C, D)
4) union(A, D)



Running time :

makeset → O(1)
find → O(rank of root)
union → O(rank of the roots)



Claim : If rank(x) = r then x has $\geq 2^r$ nodes in tree rooted at r. $r \leq \log n$

Base ($r = 0$) # of node is $1 \geq 2^0 = 1$

IS ($r \to r+1$) # of nodes from first tree + # of nodes from second tree

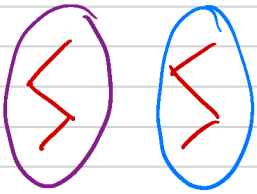
$\geq 2^r + 2^r = 2^{r+1}$

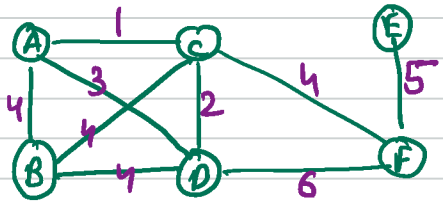
Prims Algorithm

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)

$\forall\ u \in V$ set $cost(u) = \infty$, $prev(u) = nil$

pick any initial node u_0
$cost(u_0) = 0$

$\forall\ v \in V$ insert (v, $cost(v)$) // priority queue.
while queue is non-empty
  v = Delete Min()
  $\forall\ (u,v) \in E$
    if $cost(u) > w(u,v)$
      $cost(u) = w(u,v)$
      $prev(u) = v$
      Decrease Key(u)

$O(n)$ inserts & deletes & $O(m)$ decrease keys
$O((n+m)\log n)$

Horn Formulas

$x_1, x_2 \ldots x_n$ are n boolean variables (can be TRUE/FALSE)

Input: $C_1 \ldots C_m$ s.t. $\forall i \in [m]$ C_i is either $(\bar{x}_1 \vee \bar{x}_2 \ldots)$] "pure negative"
or $(\bar{x}_1 \vee \bar{x}_2 \ldots \vee x)$ "implication"
$\hookrightarrow (x_1 \wedge x_2 \wedge \ldots) \Rightarrow x$

Goal: Is $F = C_1 \wedge C_2 \ldots C_m$ satisfiable?

Special case $(\Rightarrow x)$ $\rightarrow$ (x)

Example:
$(w \wedge y \wedge z) \Rightarrow x$
$(x \wedge z) \Rightarrow w$
$x \Rightarrow y$
$\Rightarrow x$
$(x \wedge y) \Rightarrow w$

$(\bar{w} \vee \bar{x} \vee \bar{y})$
$(\bar{z})$