

COL 351:

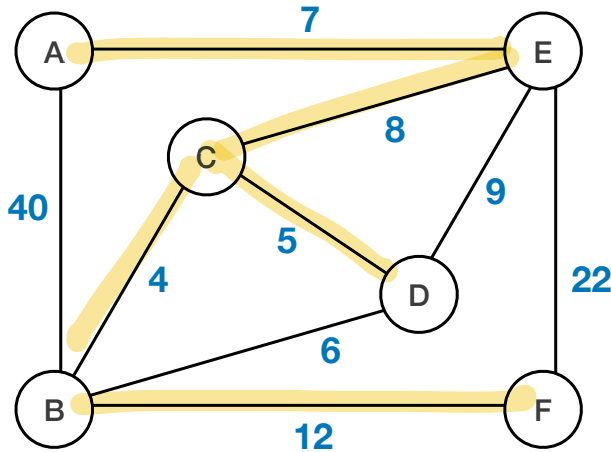
Analysis and Design of Algorithms

Lecture 5

Minimum Spanning Tree

Given: A connected weighted graph $G = (V, E, wt)$ with n vertices.

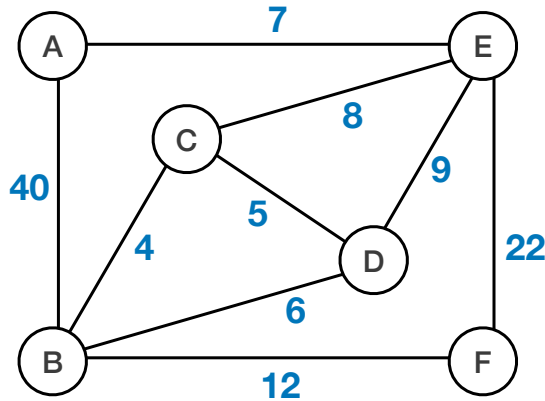
Find: A spanning tree $T = (V, E_T \subseteq E)$ of graph G such that $\sum_{e \in E_T} wt(e)$ is minimized.



Greedy Algorithm (Incremental)

1. Set $H = (V, \emptyset)$.
2. Sort the edges in non-decreasing order of weight, so that $wt(e_1) \leq \dots \leq wt(e_m)$.
3. For $i = 1$ to m :

If endpoints of e_i are in two different components in H , then **add** e_i to H .



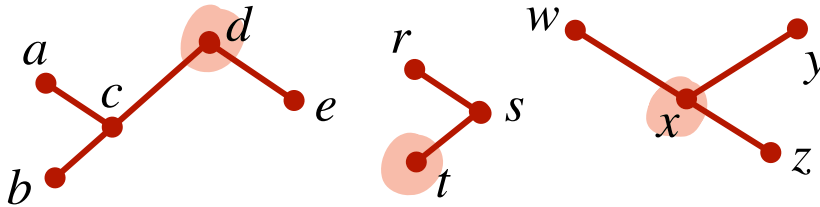
How to efficiently
check this?

Union-Find Data-structure

Given: A forest $H = (V, E_H)$ in which edges are added one at time.

Goal: Design a data-structure with following two functionalities:

1. $\text{Find}(x)$: Pointer to one *representative* vertex in tree.



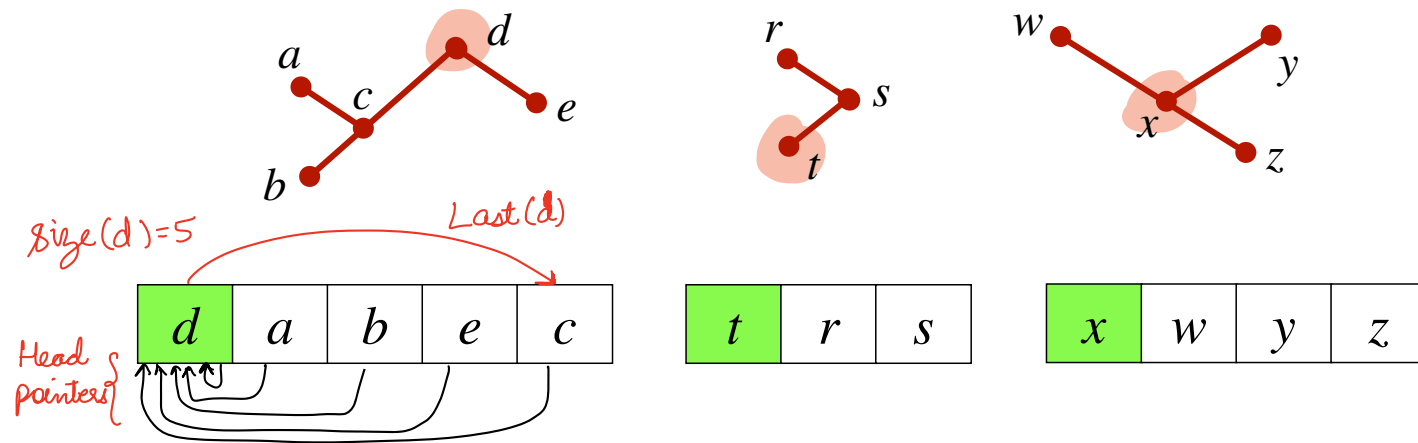
Helps to efficiently
check if two
vertices are in same
tree.

** $\text{Find}(x) \neq \text{Find}(x')$ iff x, x' are in different trees*

2. $\text{Union}(x, y)$: Merge trees of nodes x and y .

we perform union(x, y) only if $\text{Find}(x) \neq \text{Find}(y)$.

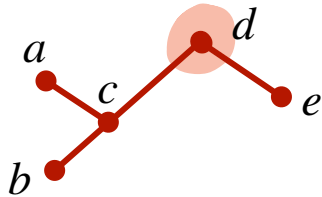
Greedy Approach for Union Find



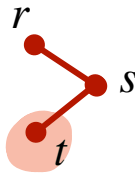
Approach:

- Represent Trees as **link-list**.
- Each vertex x stores in **Head(x)** : *Pointer to first element of link list.*
- A representative vertex y stores:
 - Size(y)** = Size of the link-list
 - Last(y)** = Pointer to last element of list

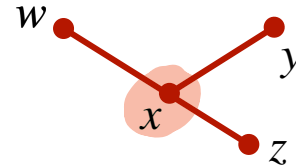
Greedy Approach for Union Find



d	a	b	e	c
---	---	---	---	---



t	r	s
---	---	---



x	w	y	z
---	---	---	---

Question: How to efficiently perform **Union** (i.e. **Merge**) operation?

Suppose $\text{Union}(x_1, x_2)$ is called.

- ① Compute $y_1 = \text{head}(x_1)$ and $y_2 = \text{head}(x_2)$
- ② Suppose $\text{size}(y_1) \geq \text{size}(y_2)$:
 - ① $z = \text{LAST}(y_1)$. Set $\text{NEXT}(z) = y_2 \leftarrow O(1)$
 - ② $\forall v \in \text{LIST}(y_2)$, we set $\text{HEAD}(v) = y_1 \leftarrow O(|\text{LIST}(y_2)|)$
 - ③ $\text{SIZE}(y_1) = \text{SIZE}(y_1) + \text{SIZE}(y_2) \leftarrow O(1)$
 - ④ $\text{LAST}(y_1) = \text{LAST}(y_2) \leftarrow O(1)$

Remark

Time taken is

$= O \left(\begin{matrix} \# \text{ of changes} \\ \text{in HEAD} \\ \text{Pointer} \end{matrix} \right)$

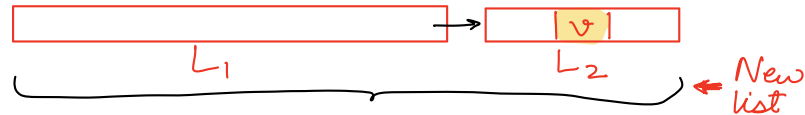
Greedy Approach for Union Find

Question: Can we bound the number of times $\text{Head}(v)$ changes for a vertex v ?

Claim: $\forall v \in V(G)$, No of changes in $\text{HEAD}(v) = O(\log_2 n)$

Proof: Consider a call of "Union" function in which $\text{HEAD}(v)$ changes.
Suppose in this call L_2 list is appended at end of list L_1 .

- Then
- (i) $v \in L_2$ and
 - (ii) $|L_1| \geq |L_2|$.



So, if α = size of older list in which v belonged, then the size of new list of v is $\geq 2\alpha$.

Thus, whenever $\text{HEAD}(v)$ changes, size of list of v DOUBLES.

This can happen only $\log_2(n)$ times.

Kruskal's MST algorithm

- $O(n)$ { 1. Set $H = (V, \emptyset)$.
- $O(m \log m)$ { 2. Sort the edges in non-decreasing order of weight, so that $wt(e_1) \leq \dots \leq wt(e_m)$.
- $O(n)$ { 3. For $v \in V$:
- Create a link-list containing v of size 1.
 - $\text{Head}(v), \text{Last}(v) \leftarrow v$, and $\text{size}(v) \leftarrow 1$.
- $O(m)$ { 4. For $i = 1$ to m :
- Let x_i, y_i be endpoints of e_i .
 - If $\text{Find}(x_i) \neq \text{Find}(y_i)$: **Add** e_i to H , and perform **Union**(x_i, y_i).
5. Return H .
- This step in total takes $\checkmark O(n \log n)$ time.*

$$\text{Time complexity} = O(\text{Time to Sort}) + O(m + n \log n)$$

This is $O(m)$ for integer weights,

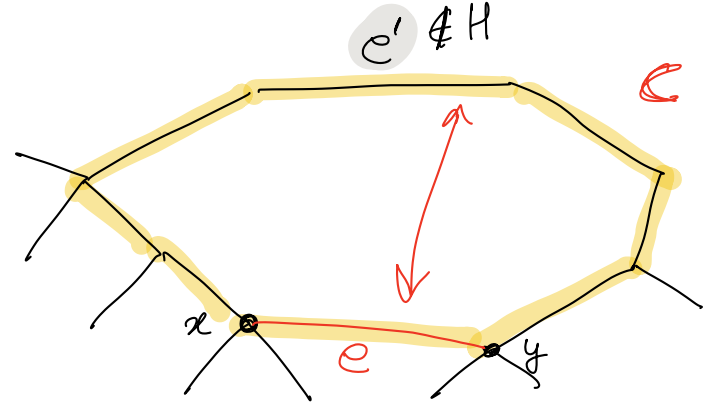
and $O(m \log m)$ for general edge weights.

Correctness

Lemma: Let H be a partial solution to MST of G . Let $e = (x, y)$ be edge of **smallest** weight in G connecting two different components in H . Then $(H + e)$ is also partial solution.

Proof:

- Let T be any MST containing H .
- Consider the case where $e \notin T$.
- Let ' C ' be unique cycle in $T + (x, y)$.
- Let e' be any edge in $C \setminus (H + e)$.
(HW: Prove that such an edge exists.)



As $(H+e)$ is acyclic it can't contain C .

Define $T' := (T \setminus e) + e'$.

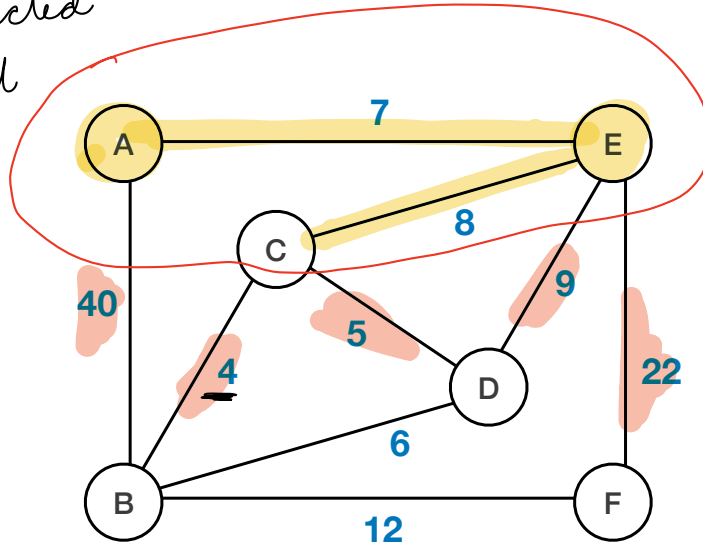
Claim 1: T' is a spanning tree.

Claim 2: $wt(T') = wt(T)$. (for proof use Defⁿ of e)

The two claims together implies there is an MST containing edges in set $(H + e)$.

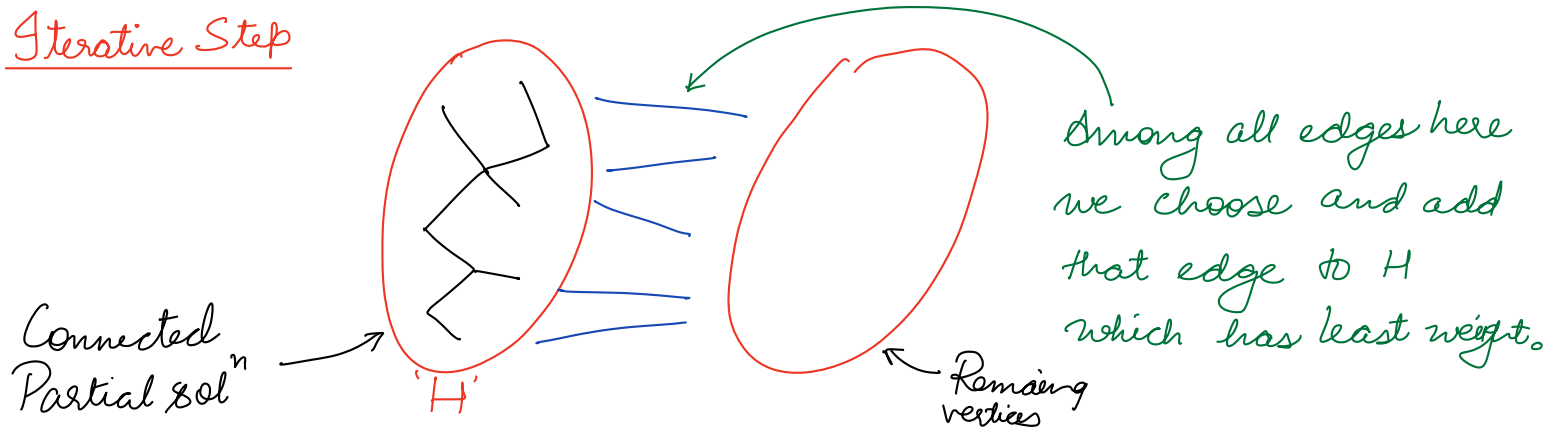
Alternate MST algorithm using Min-Priority Queues

Connected
Partial
 sol^n



- Initialize solution H to an arbitrary vertex, and grow H so that it is always connected.
- Use Min-Priority Queue to find next edge to be added.

Iterative Step



Connected
Partial sol^n

Remaining
vertices

Among all edges here
we choose and add
that edge to H
which has least weight.

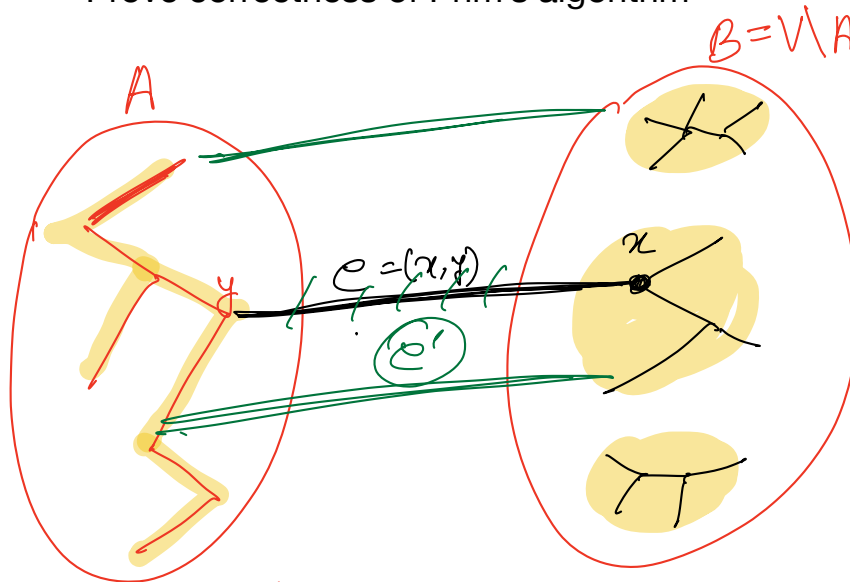
Prim's MST algorithm

1. Set $H = (\{z\}, \emptyset)$.
2. $Q \leftarrow$ a min-priority queue of size $(n - 1)$ storing $\text{KEY}(v) = \infty$, for each $v \neq z$.
3. For $y \in N(z)$: Set $\text{KEY}(y) \leftarrow \text{WEIGHT}(y, z)$ and $\text{VALUE}(y) \leftarrow z$.
4. While Q is non-empty:
 - Let x be node with minimum KEY, and let $v_x = \text{VALUE}(x)$.
 - Add edge (x, v_x) to H .
 - For $y \in N_G(x)$ satisfying $\text{KEY}(y) > \text{WEIGHT}(x, y)$:
Set $\text{KEY}(y) \leftarrow \text{wt}(x, y)$ and $\text{VALUE}(y) \leftarrow x$.
 - Remove x from Q .
5. Return H .

H.W. Prove that algo takes $O(m \log m)$ time.

Homework Exercises

- Prove correctness of Prim's algorithm



$H = \text{Partial sol}^n$
is tree

left over
vertices

CLAIM :

$\exists \text{ MST contains } (H + e)$.

Proof

$T = \text{MST}(G)$ containing H .

Suppose $e \notin T$

CLAIM : swap e & e' in
MST.

$w(e) \leq w(e')$

$T' = (T \setminus e') + e$

Argue using components
in B