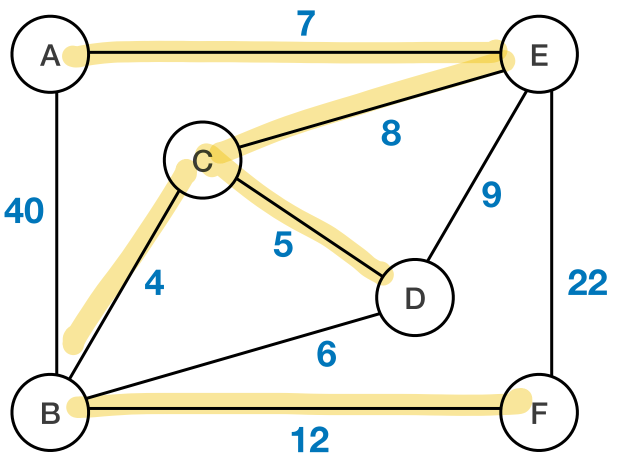
COL 351:
Analysis and Design of Algorithms

Lecture 5

Minimum Spanning Tree

Given: A connected weighted graph $G = (V, E, \text{wt})$ 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)

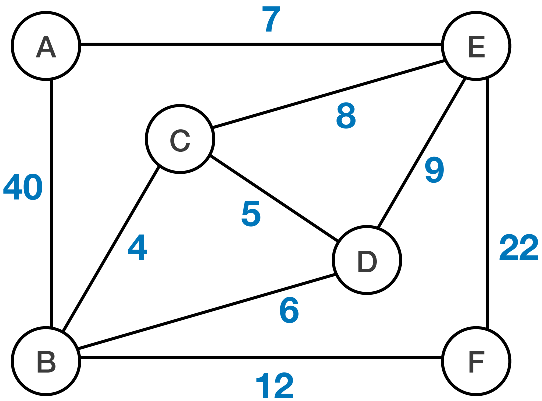
1. Set $H = (V, \varnothing)$.

2. Sort the edges in non-decreasing order of weight, so that $wt(e_1) \leqslant \cdots \leqslant wt(e_m)$.

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?

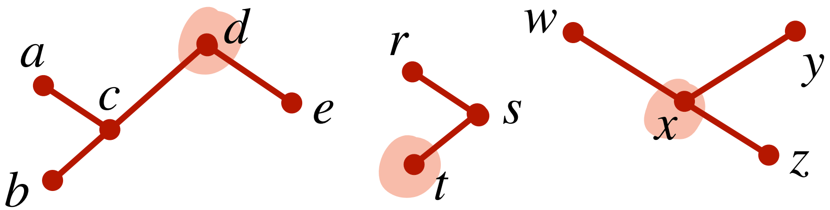


Union-Find Data-structure

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. Find(x) : Pointer to one *representative* vertex in tree.



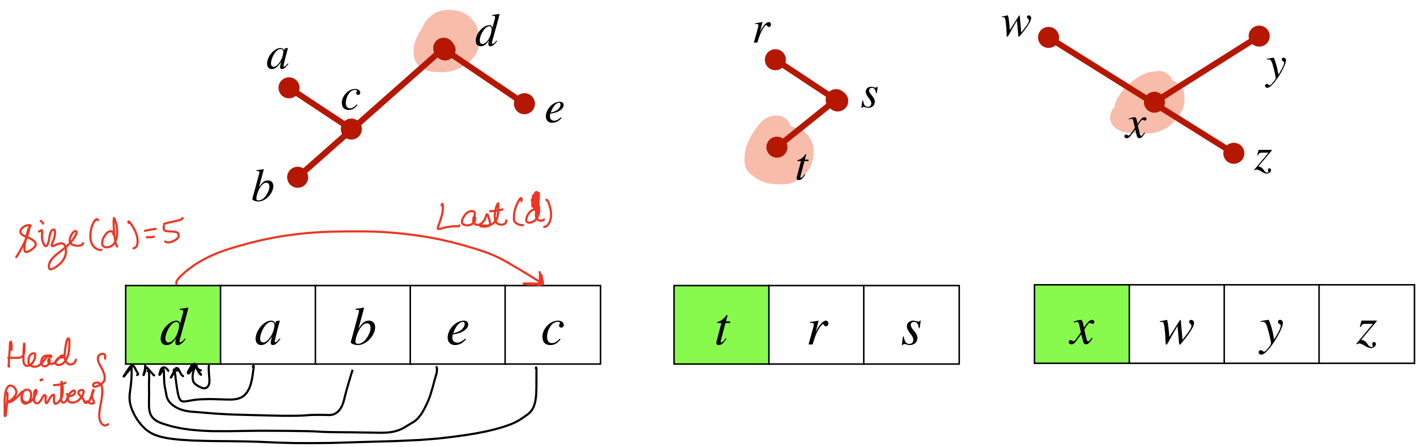
Helps to efficiently check if two vertices are in same tree.

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

2. Union(x, y) : Merge trees of nodes x and y.

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

Greedy Approach for Union Find



Approach:

— Represent Trees as **link-list**.

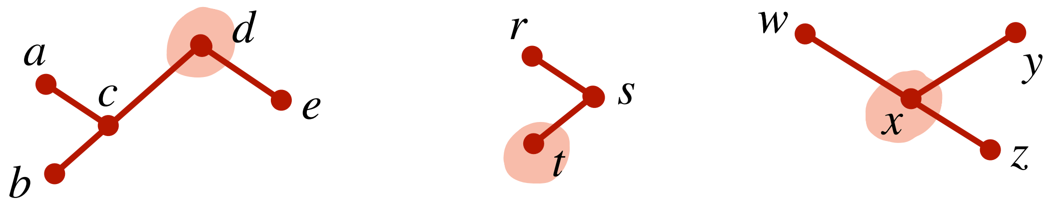
— Each vertex x stores in Head(x) : Pointer to first element of link list.

— 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 Union(x_1, x_2) is called.

① Compute $y_1 = head(x_1)$ and $y_2 = head(x_2)$

② Suppose $size(y_1) \geq size(y_2)$:

(i) $z = LAST(y_1)$. Set $NEXT(z) = y_2$ ← $O(1)$

(ii) $\forall\ v \in LIST(y_2)$, we set $HEAD(v) = y_1$ ← $O(|LIST(y_2)|)$

(iii) $SIZE(y_1) = SIZE(y_1) + SIZE(y_2)$ ← $O(1)$

(iv) $LAST(\ _1) = LAST(y_2)$. ← $O(1)$

Remark

Time taken is

$= O$(# of changes in HEAD pointer)

Greedy Approach for Union Find

Question: Can we bound the number of times Head(v) changes for a vertex v?

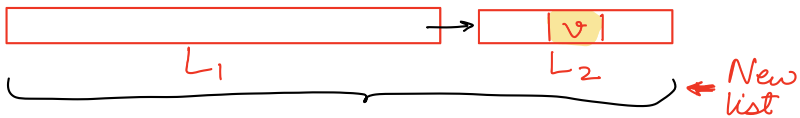
<u>Claim:</u> $\forall\, v \in V(G)$, No of changes in HEAD(v) $= O(\log_2 n)$

<u>Proof:</u> Consider a call of "Union" function in which 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 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 $H = (V, \emptyset)$.

SORTING 2. Sort the edges in non-decreasing order of weight, so that $wt(e_1) \leqslant \cdots \leqslant wt(e_m)$.

$O(n)$ 3. For $v \in V$:

— Create a link-list containing v of size 1.

— Head(v), Last(v) $\leftarrow v$, and size(v) $\leftarrow$ 1.

$O(m)$ 4. For $i = 1$ to m:

— Let x_i, y_i be endpoints of e_i.

— If Find(x_i) $\neq$ Find(y_i) : **Add** e_i to H, and perform **Union(**x_i, y_i**)**.

This step in total takes $O(n \log n)$ time.

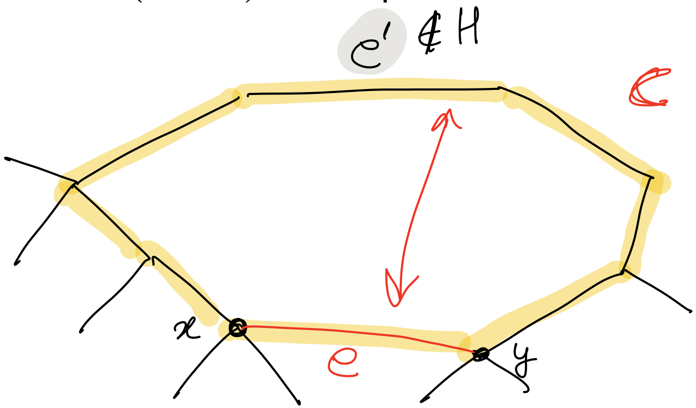
5. Return H.

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

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 $C \backslash (H + e)$.

  (HW: *Prove that such an edge exists*.)

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

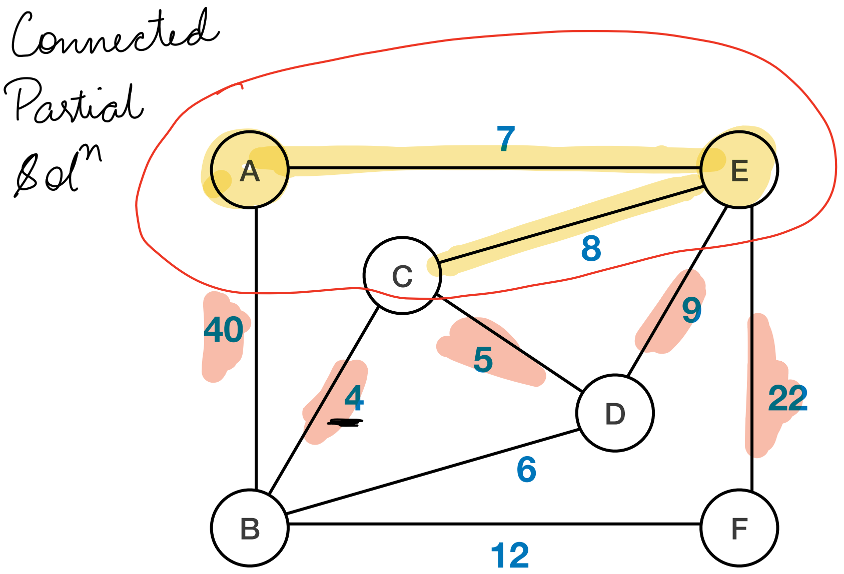
Define $T' := (T \backslash e') + e$.

Claim 1: T' is a spanning tree.

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

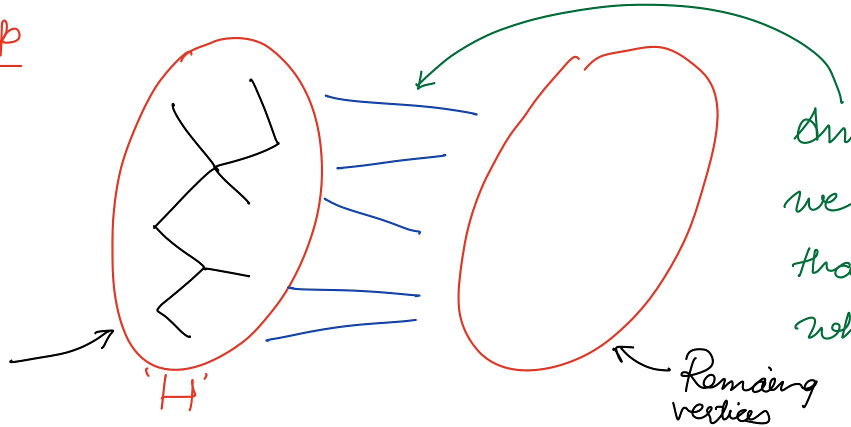
The two claims together implies there is an MST containing edges in set $(H + e)$.

Alternate MST algorithm using Min-Priority Queues



- 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



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 KEY(v) = ∞, for each $v \neq z$.

3. For $y \in N(z)$: Set KEY(y) $\leftarrow$ WEIGHT(y, z) and VALUE(y) $\leftarrow z$.

4. While Q is non-empty:

   — Let x be node with minimum KEY, and let v_x = VALUE(x).

   — Add edge (x, v_x) to H.

   — For $y \in N_G(x)$ satisfying KEY(y) > WEIGHT(x, y):

   Set KEY(y) $\leftarrow$ wt(x, y) and VALUE(y) $\leftarrow x$.

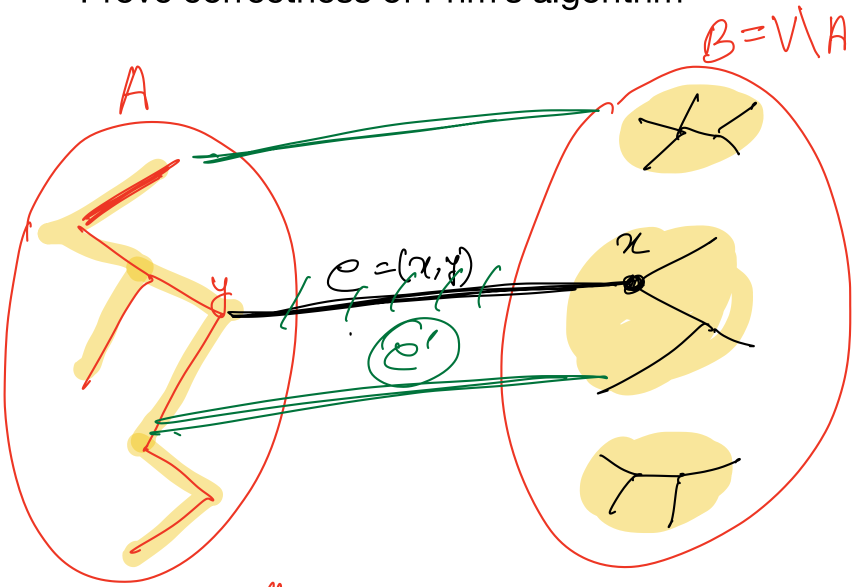
   — Remove x from Q.

5. Return H.

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

Homework Exercises

- Prove correctness of Prim's algorithm



H = Partial sol^n
is tree

CLAIM:
$\exists$ MST contains $(H + e)$.

Proof

$T = MST(G)$ contains H.

Suppose $e \notin T$

CLAIM: swap e & e' in MST.

$wt(e) \leq wt(e')$

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

Argue using components in B