CLASS RECORDING: 60% attendance (≥ 80)

Tutorials: M / T / Th / Fri 1 – 1:50 PM

Minimum Spang Tree

Given: A connected graph $G = (V, E, wt)$ where $wt: E \to \mathbb{R}$

Find: A spang tree T of G such that

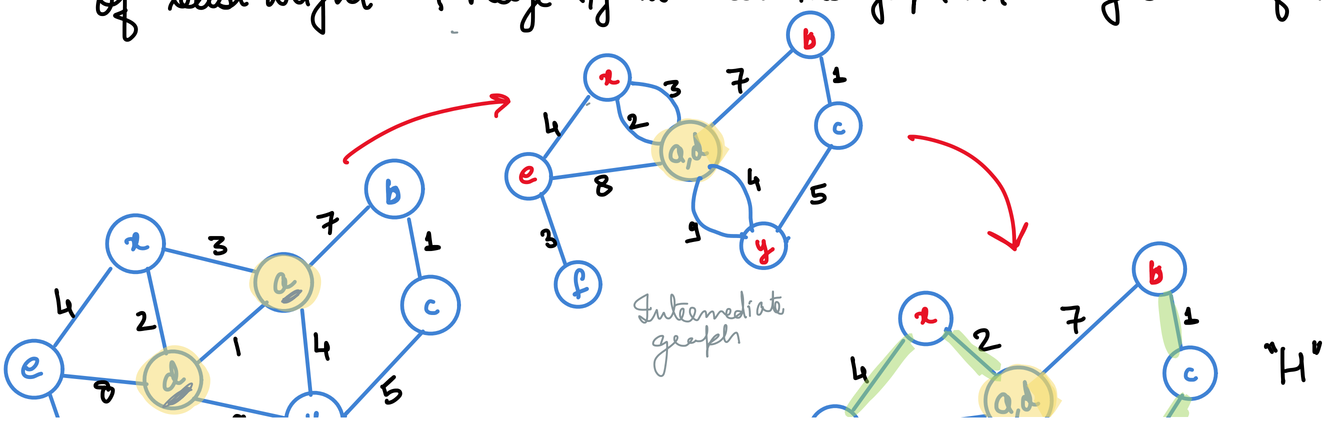
$$\text{“}\sum_{e \in T} wt(e)\text{”} \text{ is minimized.}$$

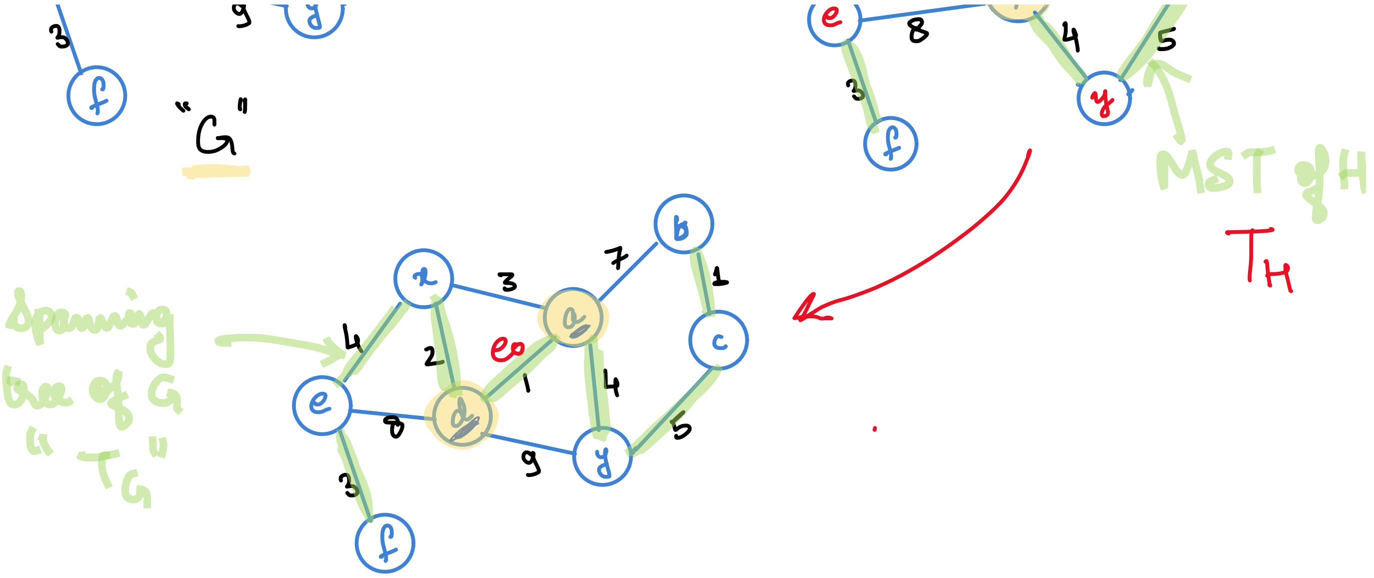
Observation 1 (Last class):

If $e_0 = (x, y)$ is edge of least weight, then there exists at least one Minimum Spanning Tree (MST) of G that contains "e_0".

Greedy Algo 1:

① Compute $e_0 = (x, y)$ of least weight | ② Put $e_0 = (x, y)$ in sol^n. Merge x, y to create new graph H | ③ Find MST of H and use it to get MST of G.





Analysis of Time Complexity

$$T(n) = T(n-1) + \underbrace{O(m)}$$

- finding min weight edge
- Finding H from G
- Finding T_G from T_H

$$\Rightarrow T(n) = O(m \cdot n).$$

Correctness Proof (To show $wt(T_G)$ is optimal)

Observation: $wt(T_G) = wt(T_H) + wt(e_0) = OPT(H) + wt(e_0)$

So to show $wt(T_G) = OPT(G)$, it suffices to sho

"$OPT(G) = opt(H) + wt(e_0)$".

① $opt(G) \leq opt(H) + wt(e_0)$

H.W.

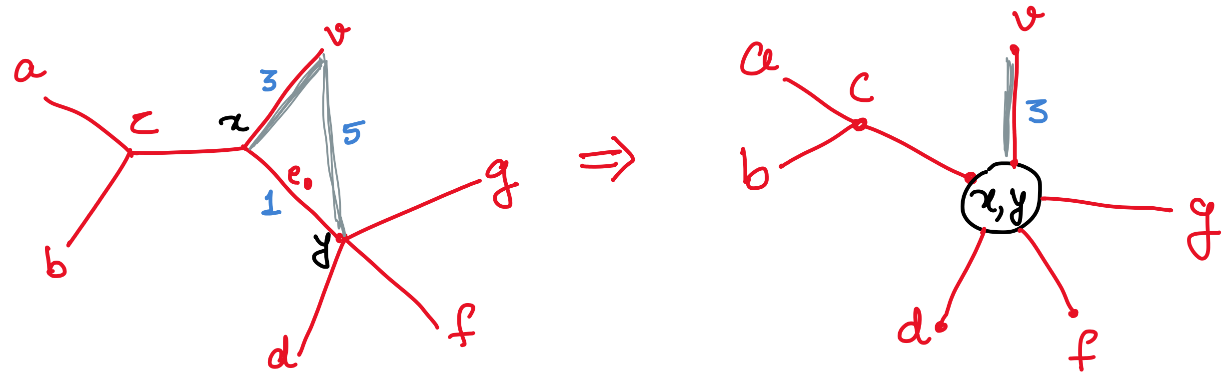
② $opt(H) \leq opt(G) - wt(e_0)$

Let T_0 be opt spanning tree of G having e_0

(It was shown in last class such T_0 exists).

Let T be obtained from T_0 by merging endpoints x and y of e_0.

<u>Claim 1</u>: T is spanning tree of H



MST T_0 of graph G Tree T in H

<u>Claim 2</u>: $wt(T) = wt(T_0) - wt(e_0)$.

Proof:

(i) <u>By Defⁿ of H</u>: For any v neighbor of both x, y in G,

$$wt_H(v, \{x,y\}) = \text{Min}\left(wt_G(v,x),\ wt_G(v,y)\right)$$

(ii) By defⁿ of MST: For any v neighbor of both x, y in G

(ii) ~~of edge of MST~~ : For any v neighbour of both x, y in G,
if (v,x) or (v,y) lie in MST T_0 of G, then

$\boxed{\text{Both } (v,x) \text{ \& } (v,y) \text{ can't lie in } T_0 \text{ (why)?}}$

$$wt_G(v,x) \lneq wt_G(v,y) \Rightarrow (v,x) \in MST$$
$$wt_G(v,y) \lneq wt_G(v,x) \Rightarrow (v,y) \in MST$$

(i) and (ii) $\Rightarrow$ $wt(T) = wt(T_0) - wt(e_0)$

As T is spanning tree of H with weight "$wt(T_0) - wt(e_0)$",
we get, $OPT(H) \leq wt(T_0) - wt(e_0)$

So, (1) and (2) implies $opt(G) = opt(H) + wt(e_0)$

Algorithm 2

Main Idea: Keep edges of smaller weight in the tree.

ALGO

(1) Sort edges of G in non-decreasing order of weight
$$wt(e_1) \leq wt(e_2) \leq \cdots \leq wt(e_m)$$

(2) Set $T = (V, \emptyset)$

$\boxed{\text{Invariant : } T \text{ is acyclic}}$

(3) For $i = 1$ to m:

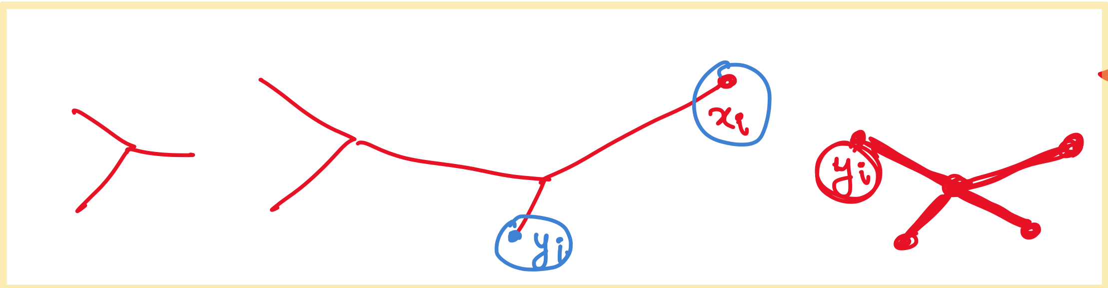
If $(T \cup e_i)$ is acyclic : add e_i to T

(4) Return T.

Time complexity : $O(m \log n + m * n) = O(mn)$

$\uparrow$ |Edges in T|

* Improve this to $\underline{O(m \log n)}$



← current "T"

$e_i = (x_i\, y_i)$

Time of tree traversal from y_i = O(no of edges in graph T)

$= O(n-1)$ $\quad$ (|Edges of T| $\leq n-1$)

Correctness Proof

(1) To show final T is a spanning tree (H.W.)

(2) Claim : T_0 is MST.

Let $\tilde{e}_1 \dots \tilde{e}_{n-1}$ be edges in T_0

$\boxed{H(n-1) \Rightarrow \exists \text{ a MST with edge } \tilde{e}_1 \dots \tilde{e}_{n-1}}$

Hypothesis $H(i)$: $\exists$ a MST of G having edges $\tilde{e}_1 \dots \tilde{e}_i$.

Proof of $H(1)$: Last class, $\exists$ a MST with edge of least weight.

$H(i) \Rightarrow H(i+1)$:

Take a MST say T' having edges $\tilde{e}_1 \dots \tilde{e}_i$

Take a MST, say T', having edges $e_1, \dots, e_i$

Case 1: $\left. e_{i+1} \in T' \right\} \Rightarrow H(i+1)$ is true

Case 2: $e_{i+1} \notin T'$

H.W. : Find a MST T'' from T'
(New class) s.t. $\underbrace{\tilde{e}_1 \dots \tilde{e}_i \, \tilde{e}_{i+1}} \in T''$

ALGO

① Sort edges of G in non-decreasing order of weight

$$wt(e_1) \leq wt(e_2) \leq \dots \leq wt(e_m)$$

② Set $T = (V, \emptyset)$

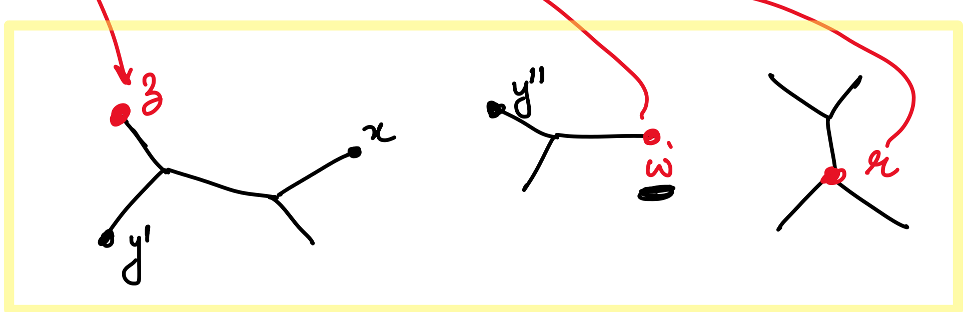
③ For $i = 1$ to m:

$\boxed{\text{If } (T \cup e_i) \text{ is acyclic}}$: add e_i to T

④ Return T.

Checking this efficiently

Representative element



current T

Given 2 vertices x, y tell if x & y are connected.

$x, y'' \rightarrow$ No

$x, y' \rightarrow$ Yes

Think:

$$\text{Find}(x) = z$$
$$\text{Find}(y) = z$$
$$\text{Find}(y'') = w$$

How can representative function - "Find" be used to judge if 2 vertices are connected.