COL 351: Analysis and Design of Algorithms

Friday, 13 August 2021

Today's Lecture

- Graph Notations
- Spanning Tree Problem
- **Minimum Spanning Tree Problem**
  (uses Greedy approach)

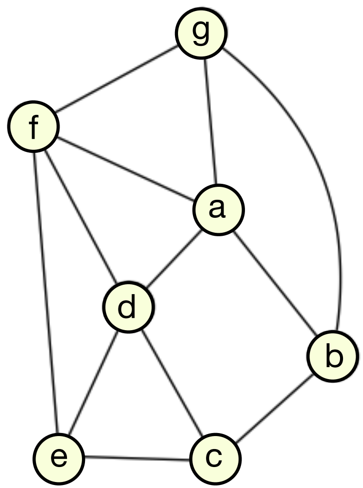
What are graphs?

- A graph is a pair $G = (V, E)$ where V is a set of vertices and $E \subseteq V \times V$ is a set of edges.

- $n = |V|$

$$|E| \leq \frac{|V| * (|V|-1)}{2}$$

- $m = |E|$



$V = \{a, b, c, d, e, f, g\}$

of G   of G

- **Subgraph:** A graph $H = (V', E')$ is subgraph of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$.

H

What is tree?

(Un-rooted tree) — No edge directions

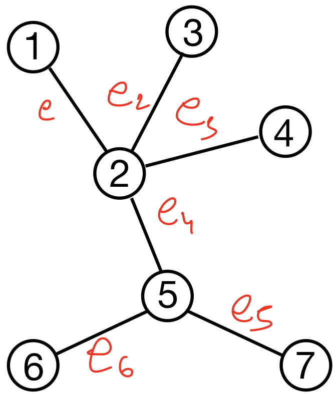
Examples of Tree

Binary tree

R. B. tree

AVL tree

- A tree $T = (V, E)$ is a connected acyclic undirected graph.



← tree

$n = |V| = 7$

$m = |E| = 6$

Rooted tree

Any arbitrary vertex is set to root

Equivalent definition of trees:

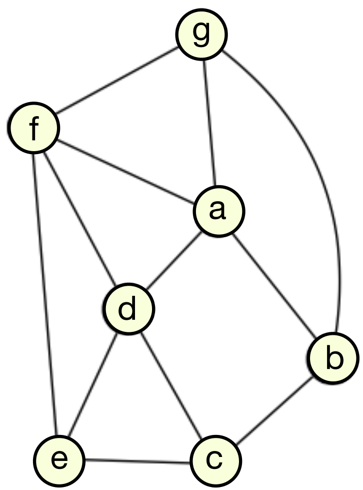
- G is a connected graph with n vertices and $(n-1)$ edges.
- G is a acyclic graph with n vertices and $(n-1)$ edges.
- G satisfies that each pair of vertices has a unique path between them.

H.W.

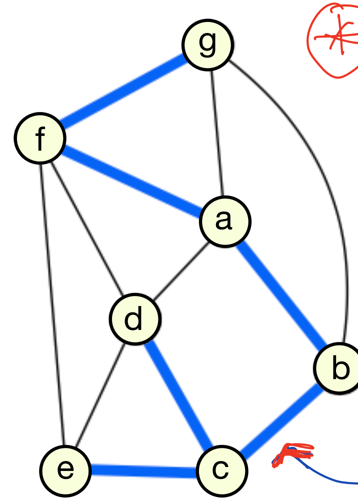
Motivational Problem: Metro Layout

There are n locations in a city connected by roads.

GOAL : Compute a “metro-network” on top of road map such that each pair of location is connected by metro. with minimum no. of edges.



G



connecting each pair of nodes

Sol^n should be tree

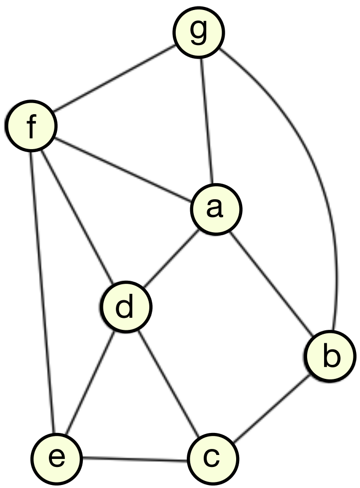
use simple BFS/DFS trees

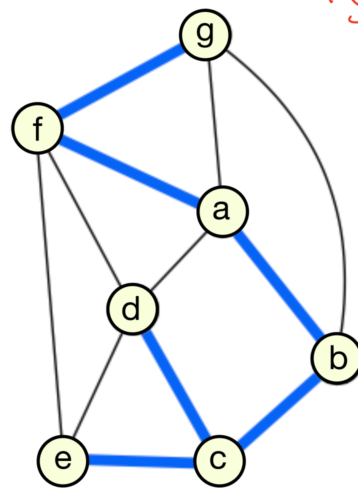
Mathematical Formulation: Spanning Tree

GIVEN: An n vertex, m edges connected undirected graph $G = (V, E)$.

GOAL: Find a connected subgraph $H = (V, E_H \subseteq E)$ of G with minimum edges.

Sol^n: Any Spanning tree





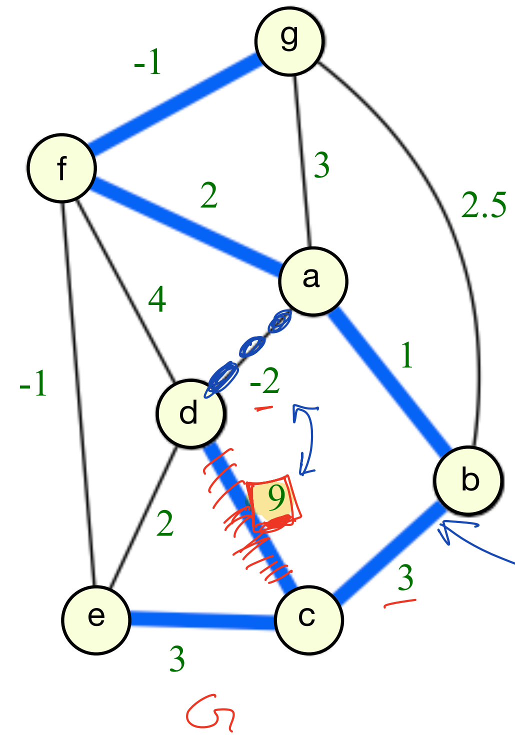
Minimum Spanning Tree: If G is edge-weighted, find a sol^n with least weight

What if the edges in G are weighted?

Goal: Find a sol^n with least weight.

Will the same Spanning Tree be optimal?

weights of tree = sum of edge weights in it.



spanning tree

spanning tree of weight 17 = -1+2+1+3+9+3

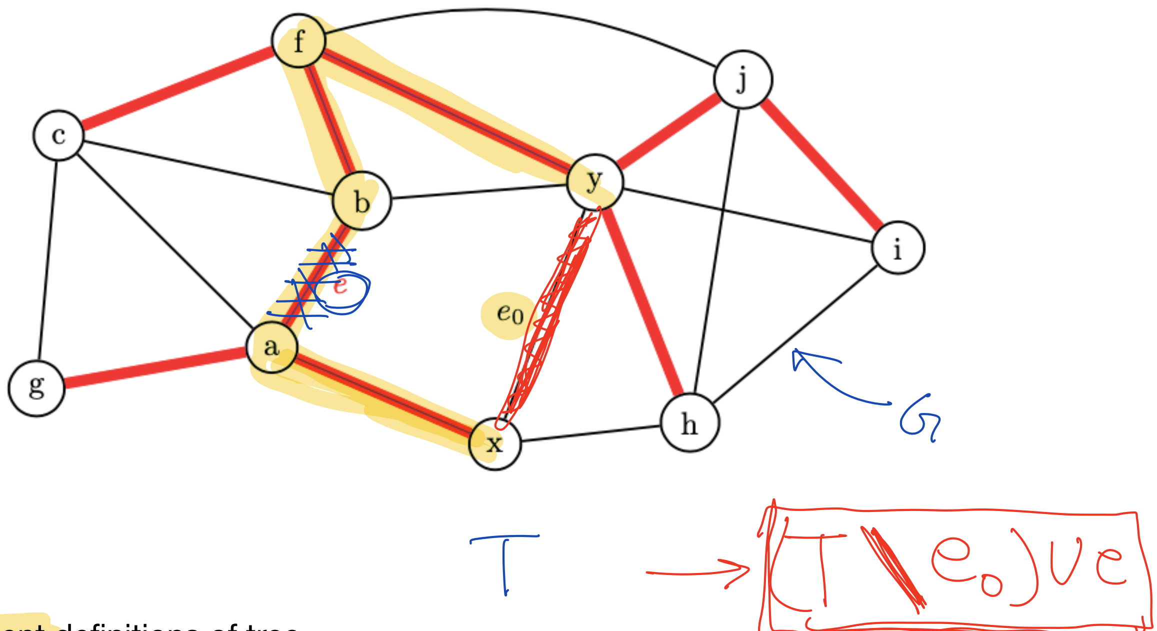
An Important Property of Trees

Simple

Property 1: Let T be a spanning tree of G and $e_0 = (x, y)$ be an edge not lying in T.

Let e be any edge on the *unique* path from x to y in T.

Then on swapping e with e_0 in T we get another spanning tree of G.



H. W.

Proof Hint: Use equivalent definitions of tree.

H.W.

Ques. given : $G = (V, E)$

and a spang tree of G

Repeat { Swap edge of larger weight with smaller weight edge }

Invariant

T is spang.

Prove

Process will terminate

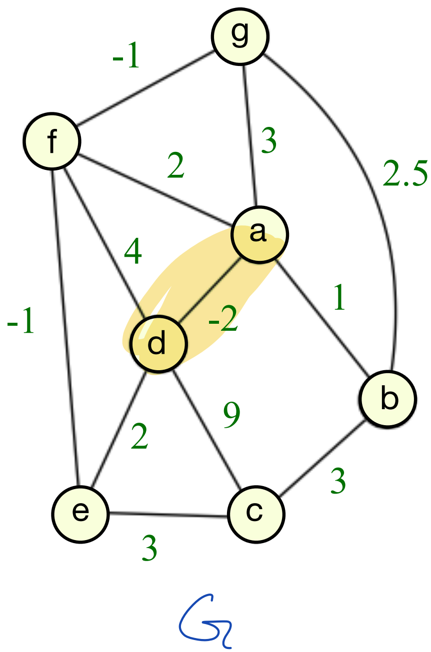
Final tree will have minimum possible weight.

Main Problem: Minimum Spanning Tree (MST)

GIVEN: An n vertex, m edges connected undirected graph $G = (V, E)$.

Weight function $wt : E \to \mathbb{R}$.

GOAL: Find a spanning tree $T = (V, E_T \subseteq E)$ of G such that $\underbrace{\sum_{e \in E_T} wt(e)}_{\text{weight of tree}}$ is minimum.



least weight edge
= (a, d)
weight = -2

An Important Property of MST

Property 2: Let $e_0 = (x, y)$ be an edge in G of least weight.

Then there exists at least one MST of G containing e_0.

Proof: Consider an MST T of G. Let us suppose $e_0 \notin T$.

Let e be any edge on the *unique* path from x to y in T.

Swap e with e_0 to compute a new tree T_0.

By Property 1, T_0 is also a spanning tree of G.

Since $\underbrace{wt(T_0) \leq wt(T)}$, tree T_0 is an MST of G .

bcoz e_0 is least weight edge.

An Important Property of MST

Minimum Spanning Tree

Property 2: Let $e_0 = (x, y)$ be an edge in G of least weight.

Then there exists at least one MST of G containing e_0.

Proof : Take any MST "T" of G

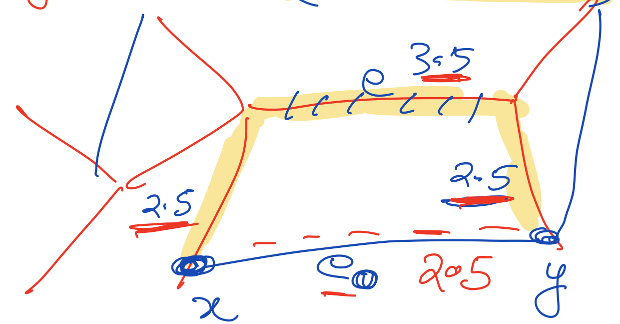
Idea	e_0 lies in T	$e_0 \notin T$
	We are done.	By Property 1, $(T \setminus e) \cup e_0$ = T_0

HOW.

CLAIM : All edges on x to y path in T have weight = wt(e_0)

(By contradiction)

Arya



Algorithm sketch

- Step 1: Reduce Problem to Smaller Instance with $n-1$ vertices.

n vertices

- Step 2: Compute MST of Smaller Instance H. (n-1) vertices

- Step 3: Extract back MST of G.

MST of H ⇒ MST of G

What we know?

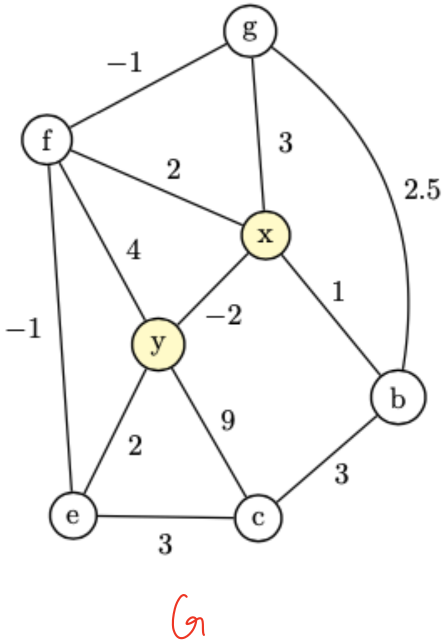
If e_0 has least weight

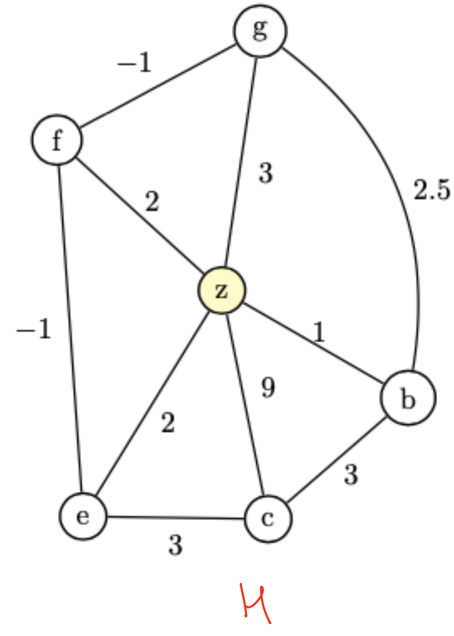
⇒ ∃ MST of weight e_0

Step 1: Reduce Problem to Smaller Instance

Transformation: Let $e_0 = (x, y)$ be an edge in G with least weight. Compute H as below:

- Remove vertices x and y from G, and add a new vertex z.
- For each v neighbour of x or y in G, add (v, z) to H and set $wt(v, z) := \min\big(wt(v, x), wt(v, y)\big)$



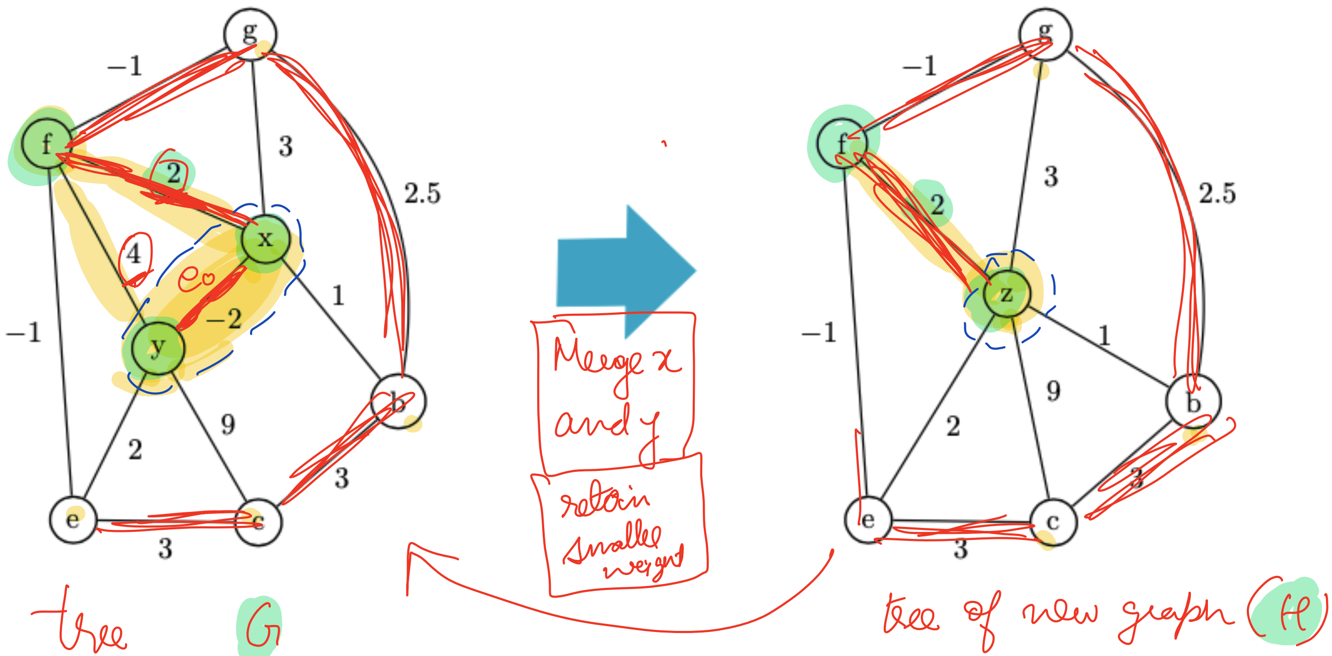


Step 1: Reduce Problem to Smaller Instance

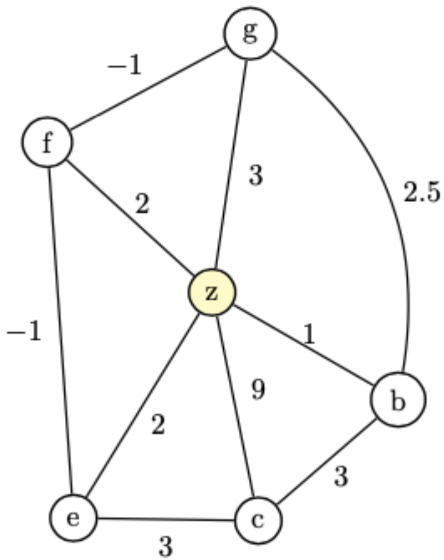
n vertices

Transformation: Let $e_0 = (x, y)$ be an edge in G with least weight. Compute H as below:

- Remove vertices x and y from G, and add a new vertex z.
- For each v neighbour of x or y in G, add (v, z) to H and set $wt(v, z) := \min\big(wt(v, x), wt(v, y)\big)$

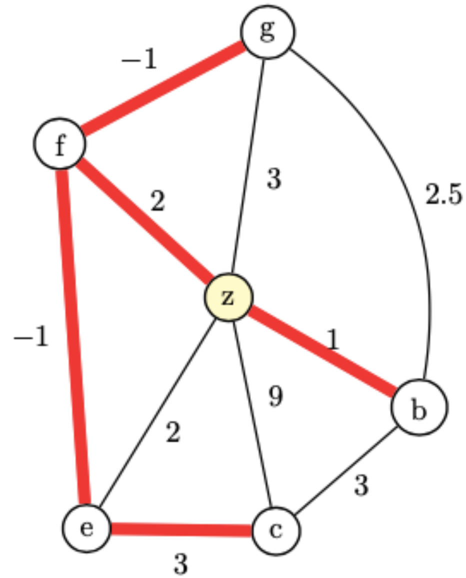


Step 2: Compute MST of Smaller Instance H



H

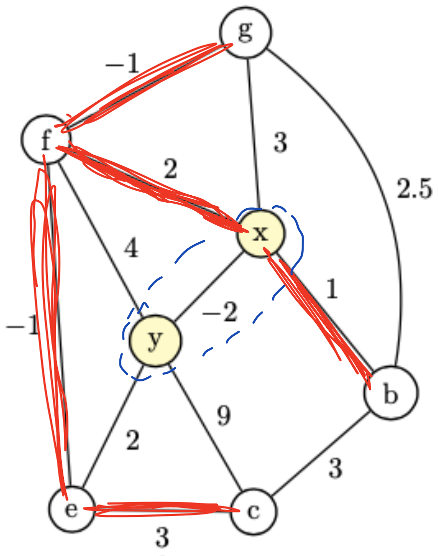
(n-1) vertices



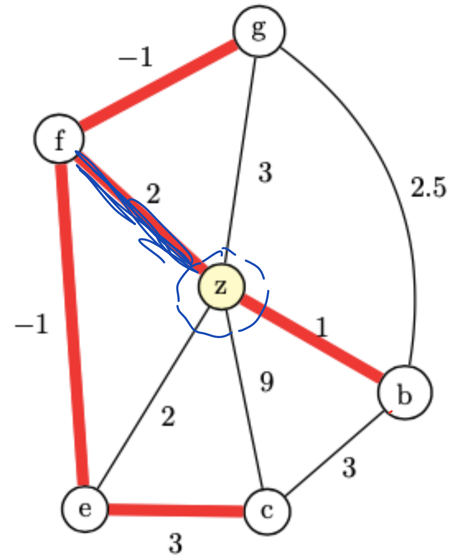
MST of *H*

wt = 4

Step 3: Extract back MST of G



G



MST of H

Algorithm

Let $e_0 = (x, y)$ be edge in G with least weight.

Initialize H to G.

Remove vertices x and y from H, and insert a new vertex z.

foreach v neighbour of x or y in G **do**

> add edge (v, z) to H and set $wt(v, z) := \min\big(wt(v, x),\ wt(v, y)\big)$.
>
> Set $map(v, z) = \begin{cases} (v, x) & \text{if } \ wt(v, x) \le wt(v, y) \\ (v, y) & \text{otherwise.} \end{cases}$

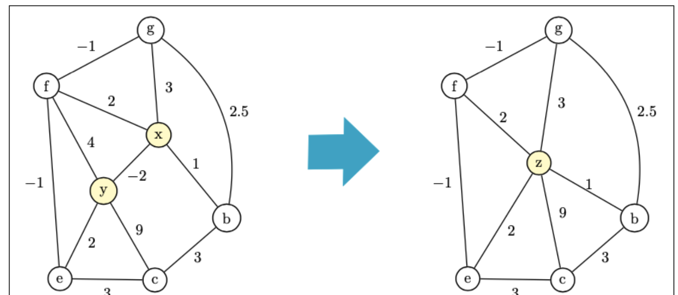
$T_H \leftarrow$ MST of H.

Initialize $T_G := (V, \{e_0\})$.

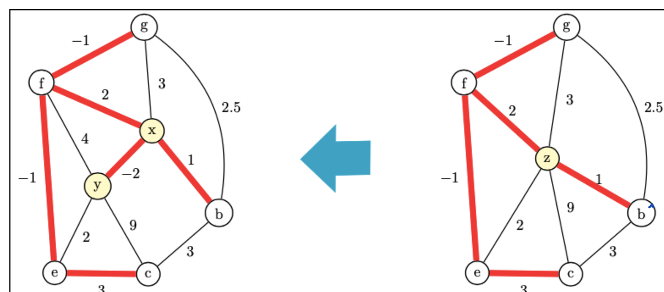
foreach edge e in T_H **do**

> **if** e is not incident to z **then** add e to T_G.
>
> **else** add $map(e)$ to T_G.

Output T_G.



Edges incident to z in H	(z, b)	(z, c)	(z, e)	(z, f)	(z, g)
Mapped edges in G	(x, b)	(y, c)	(y, e)	(x, f)	(x, g)



Algorithm

Let $e_0 = (x, y)$ be edge in G with least weight.

Initialize H to G.

Remove vertices x and y from H, and insert a new vertex z.

foreach v neighbour of x or y in G **do**

  add edge (v, z) to H and set $wt(v, z) := \min\big(wt(v, x),\ wt(v, y)\big)$.

  Set $map(v, z) = \begin{cases} (v, x) & \text{if } \ wt(v, x) \le wt(v, y) \\ (v, y) & \text{otherwise.} \end{cases}$

$T_H \leftarrow$ MST of H.

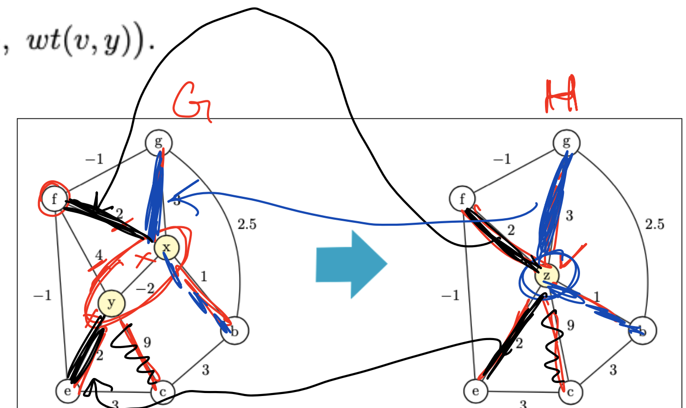
Initialize $T_G := (V, \{e_0\})$.

foreach edge e in T_H **do**

  if e is not incident to z **then** add e to T_G.

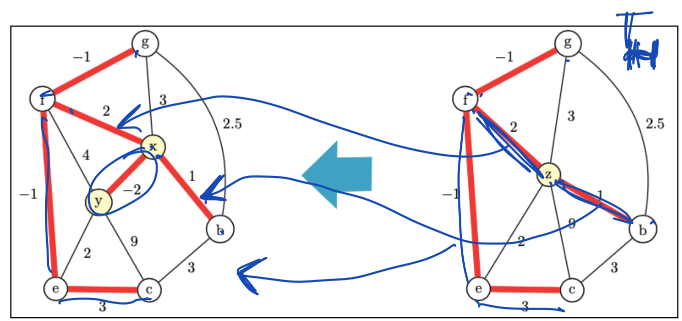
  else add $map(e)$ to T_G.

Output T_G.



Edges incident to z in H	(z, b)	(z, c)	(z, e)	(z, f)	(z, g)
Mapped edges in G	(x, b)	(y, c)	(y, e)	(x, f)	(x, g)

Sol^n of G



Time complexity = ?

correctness = ?

Correctness

Theorem 1: Let $e_0 = (x, y)$ be edge with least weight in G, and H graph computed by Algorithm.

Then $wt(MST(G)) = wt(MST(H)) + wt(e_0)$.

Part 1: $wt(MST(G)) \leq wt(MST(H)) + wt(e_0)$

Part 2: $wt(MST(G)) \geq wt(MST(H)) + wt(e_0)$