Design and Analysis of Algorithms

L24: Prim's Algorithm

Minimum Cost Spanning Tree

Dr. Ram P Rustagi
Sem IV (2019-H1)
Dept of CSE, KSIT/KSSEM
rprustagi@ksit.edu.in

Resources

- Text book 1: Sec 9.1-5.4 - Levitin
- R1: Introduction to Algorithms
  - Cormen et al.
- MIT Open Course Ware
  – https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec11.pdf

Spanning Tree

- Consider N number of villages in a district
- Government would like to ensure that these villages are connected by road
  - reachable from each other, may be via other villages
- The cost of laying road from one village to other villages is known
- Govt would like to incur minimum cost
- Which roads government should lay down
  - How many roads needs to be layed down.
- Answer: **M**inimum Cost **S**panning **T**ree
- Q: Provide other examples:

Application of Spanning Trees

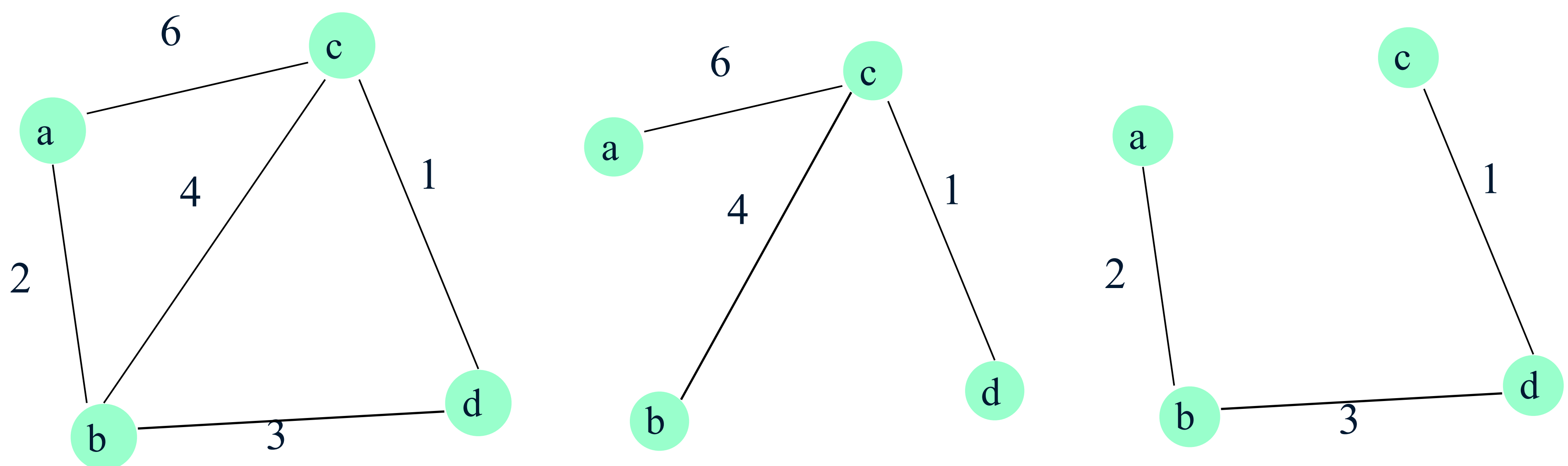
- Laying of utility lines
  – water,
  – electrical,
  – gas,
  – cable TV lines
  – …
- Road distribution network
- Building floor/room corridors with single entry/exit point.

Spanning Tree

- Graph: $G=\{V,E\}$
  - A set of nodes V
  - A set of edges $E=(u,v)$ connecting node u to node v.
- Connected Graph:
  - each node is reachable from any other node via some path.
  - There may exist multiple paths, (have cycles)
- Spanning tree:
  - A subgraph T of G i.e. $T \subseteq G$ such that
    - It contains all the vertices V of G i.e. if $v \in G \Rightarrow v \in T$
    - Between any two nodes u and v, $\exists$ only one path
    - i.e. T is acyclic

Minimum Spanning Tree

- A minimum spanning tree of weighted connected graph `G` is a spanning tree `T` with minimum total weight.
- Examples:

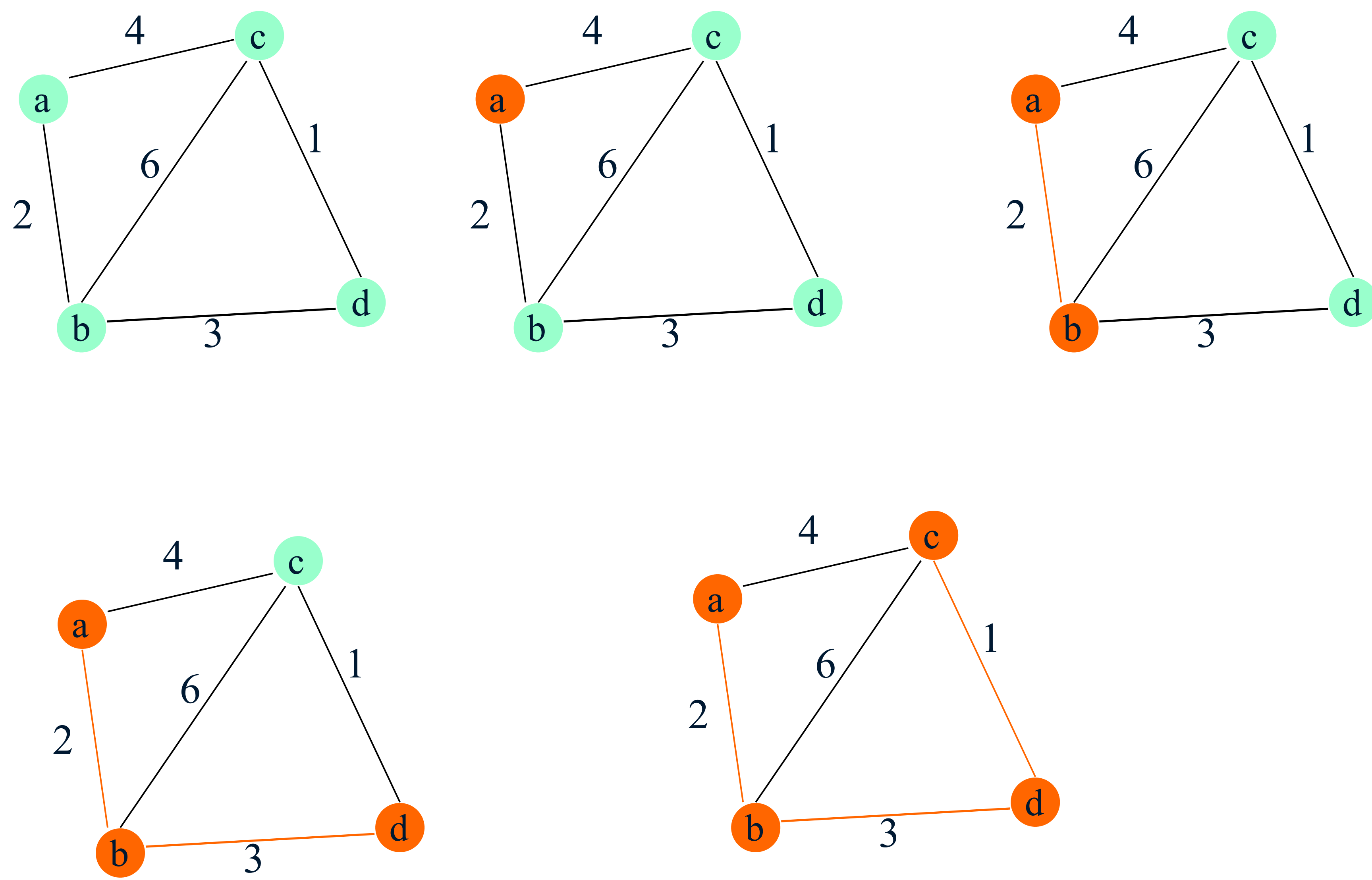


- Q: Are other spanning trees possible?
- Q: What happens when all edges have same weight?

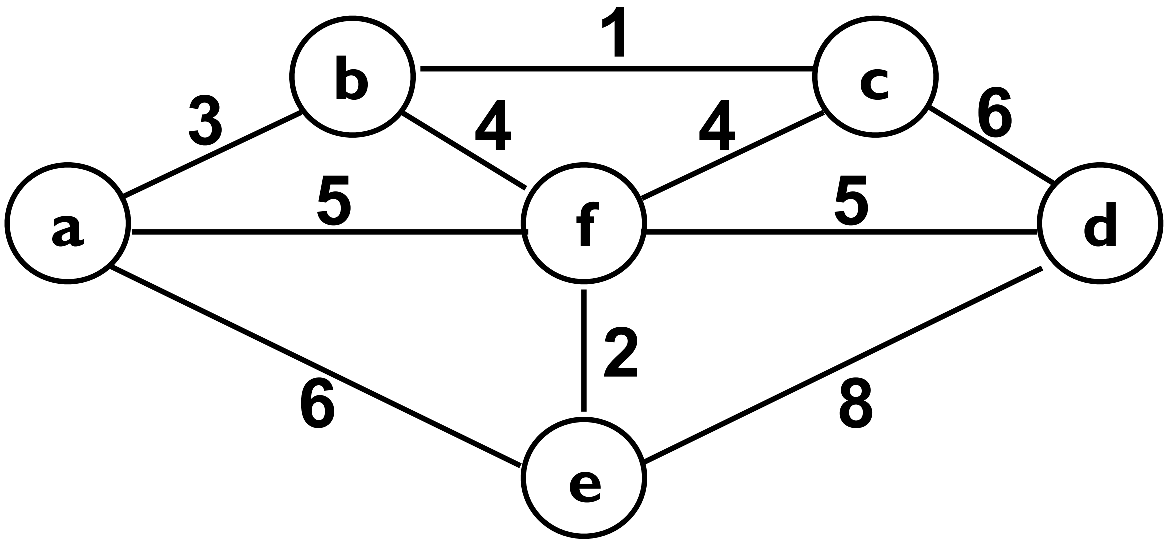
Prim's MST Algorithm

- Approach:
  - Start with tree T_1 consisting of one (any) vertex, and
    - *grow* tree one vertex at a time to produce MST
    - through a series of expanding subtrees $T_1, T_2, \ldots, T_n$
- Greedy Appraoch:
  - On each iteration, construct T_{i+1} from T_i
    - Add a vertex not in T_i which is
      - closest to those already in T_i
      - this is a **greedy** step!
- Stop when all vertices are included.

Example 1: Prim's MST

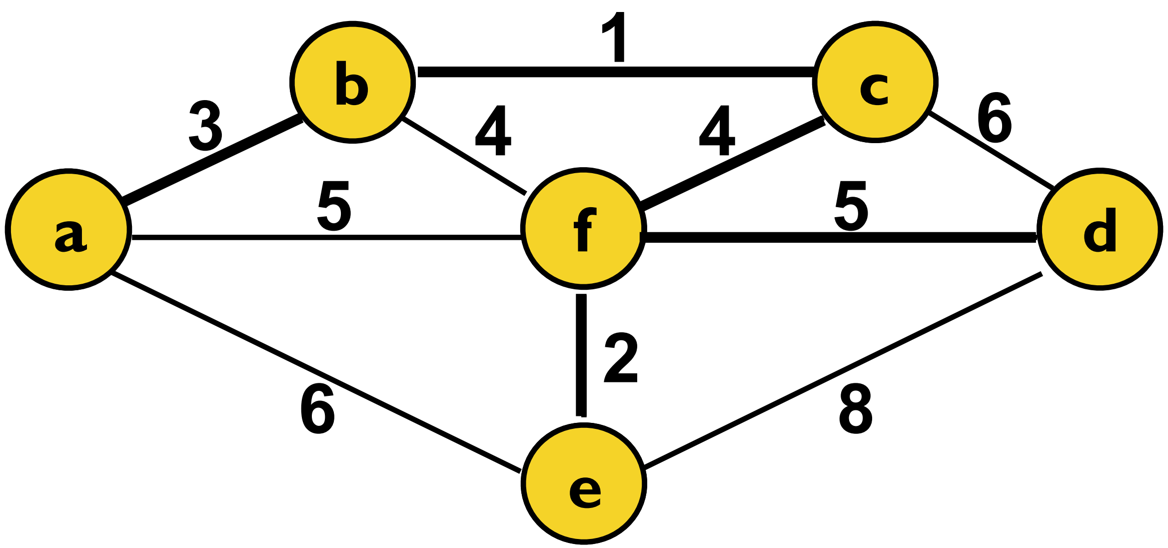


Example 2: Prim's MST



- Q: Construct an MST starting from vertex `a`

Example 2: Prim's MST



w(a):0,w(b):∞,w(c):∞,w(d)=∞,w(e)=∞,w(f)=∞

w(**a**):0,w(b):3,w(c):∞,w(d)=∞,w(e)=6,w(f)=5

w(a):0,w(**b**):3,w(c):1,w(d)=∞,w(e)=6,w(f)=4

w(a):0,w(b):3,w(**c**):1,w(d)=6,w(e)=6,w(f)=4

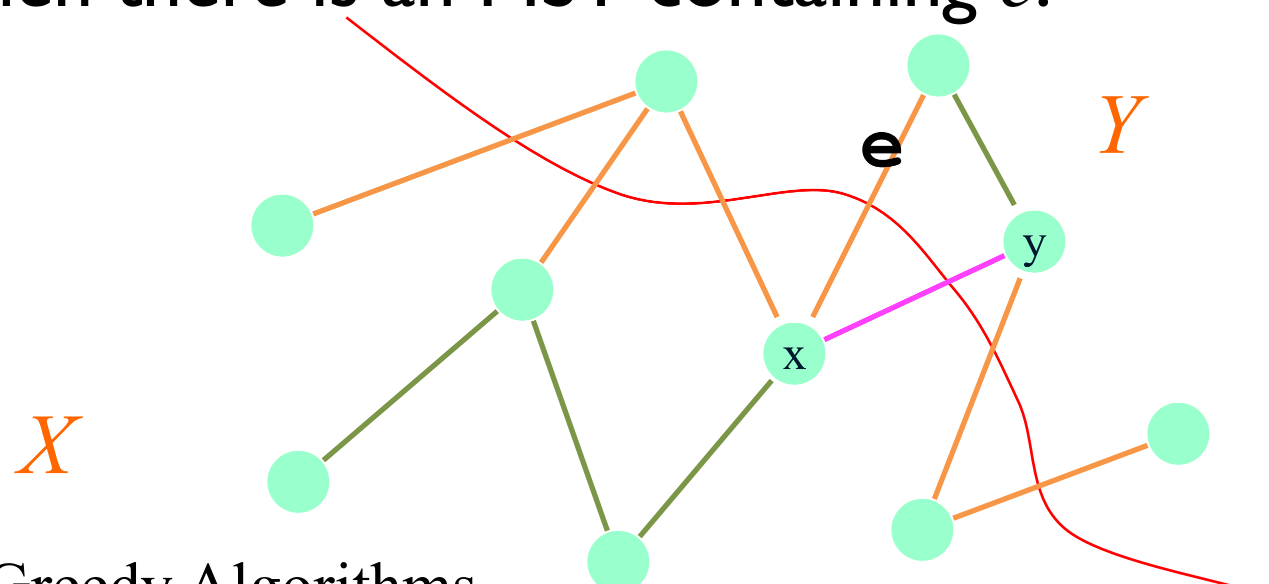
w(a):0,w(b):3,w(c):1,w(d)=5,w(e)=2,w(**f**)=4

w(a):0,w(b):3,w(c):1,w(d)=5,w(**e**)=2,w(f)=4

w(a):0,w(b):3,w(c):1,w(**d**)=5,w(e)=2,w(f)=4

Prim's Algo: Proof by Induction

- Claim: Let $G = (V,E)$ be a weighted graph and (X,Y) be a partition of V (called a *cut*).
- Suppose $e = (x,y)$ is an edge of E across the cut, where
  - x is in X, and
  - y is in Y, and
  - e has the minimum weight among all such crossing edges (called a light edge).
- Then there is an MST containing e.



Prim's Algo

- Needs priority queue for implementation

Algo: $\texttt{Prim(G)}$

// i/p: A weighted connected graph $G=(V,E)$

// o/p: E_T, the set of edges composing an MST of G

$V_T \leftarrow \{v_0\}$ # initialize with any vertex

$E_T \leftarrow \varnothing$

for $i=1$ to $|V|-1$ do

  find a min weight edge $e^*=(v^*,u^*)$ among all edges (v,u) such that $v \in V_T$ and $u \in |V|-V_T$

  $V_T \leftarrow V_T \cup \{v^*\}$

  $E_T \leftarrow E_T \cup \{e^*\}$

return E_T

Prim's Algo: Efficiency

- Efficiency depends upon implementation
- Maintain $V-V_T$ in priority queue
- Initially, assign a weight(value) of ∞ to each vertex
- Weight of each edge is known (given graph G)
- Using Adjacency weight matrix
  - If priority queue is maintained in an unordered array
    - vertex can be accessed by index in the array
  - Picking min vertex u takes $|V|$ time.
    - Requires linear search in array
  - For each edge (u,w), update the weight of w
    - `weight(w)=min(weight(w),weight(u,w))`
  - Total time: $O(|V|^2+|E|)=O(|V|^2)$

Prim's Algo: Efficiency

- Efficiency depends upon implementation
- Maintain $V-V_T$ in priority queue
- Initially, assign a weight(value) of ∞ to each vertex
- Weight of each edge is known (given graph G)
- Using Adjacency weight List
  - Maintain priority queue is `BinSearch` Tree
    - Height of the tree is `lg|V|`
  - Find vertex `u` with min weight is `O(1)` time
  - For each edge `(u,w)`, update the weight of w
    - `weight(w)=min(weight(w),weight(u,w))`
    - TIme taken to adjust `BinSearch` Tree is `O(lg|V|)`
  - Total time: `O(E*lg|V|)`

Summary

- Minimum Spanning Tree
- Prim's algorithm
- Time efficiency
  - Depends upon implementation