Lecture 25
Minimal Cost Spanning Tree

Spanning Tree: $G=(V, E)$, $T \subseteq G$

undirected/unweighted graph.

$|V|(V, E') - 1$ edges (Subset)

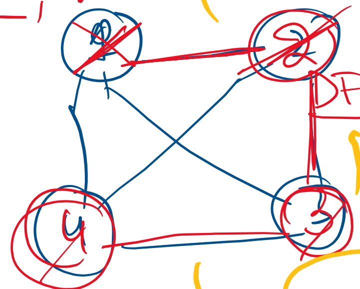
vertices.

Sub graph of G that will contain exactly $|V|-1$ edges and all V vertices of graph.

Spanning trees

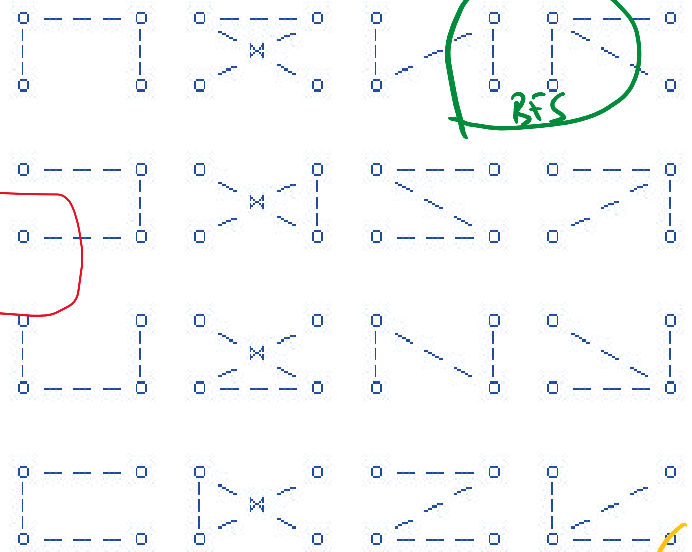
1 → DFS

2 → BFS



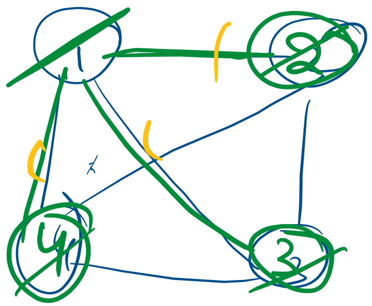
DFS-tree

3

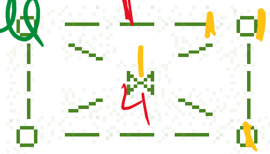


DFS

BFS



BFS



Cost = Sum weights of all edges

Tree cost $= \sum_{i=1}^{|E|} w(u,v)$

Cost = 3

16

Cost = 3

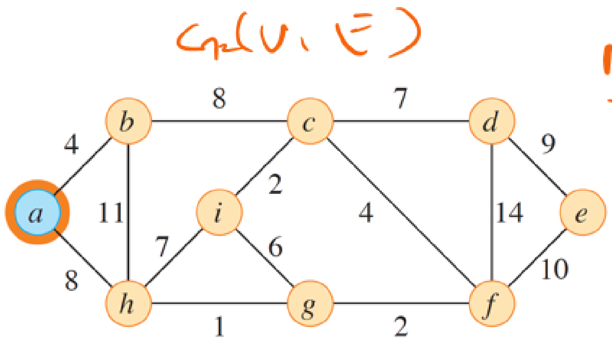
Number of edges

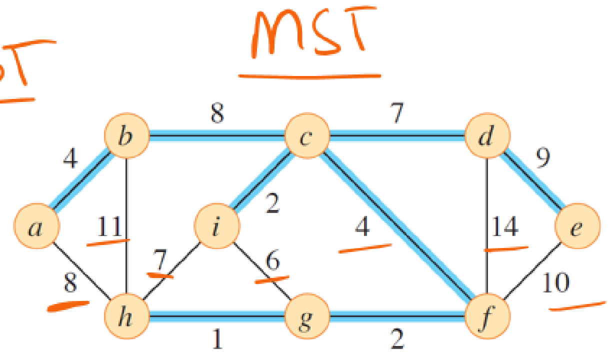
Minimal Cost Spanning Tree (MST):

Find a tree T of a given graph G that contains all the vertices of G and has the minimum total weight of the edges of G over all other such trees

$$\sum_{i=1}^{|E|} w(u,v)$$

MST





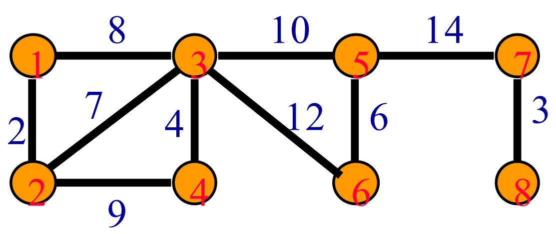
Cost = 37

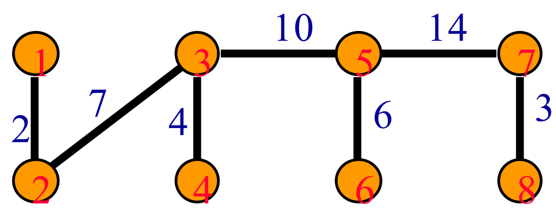
MST Algorithms:

1. Prim's Algorithm
2. Kruskal's Algorithm → Select edges with cheapest/minimum cost + Not creating the cycle.

→ Greedy Approach

Prim’s Method



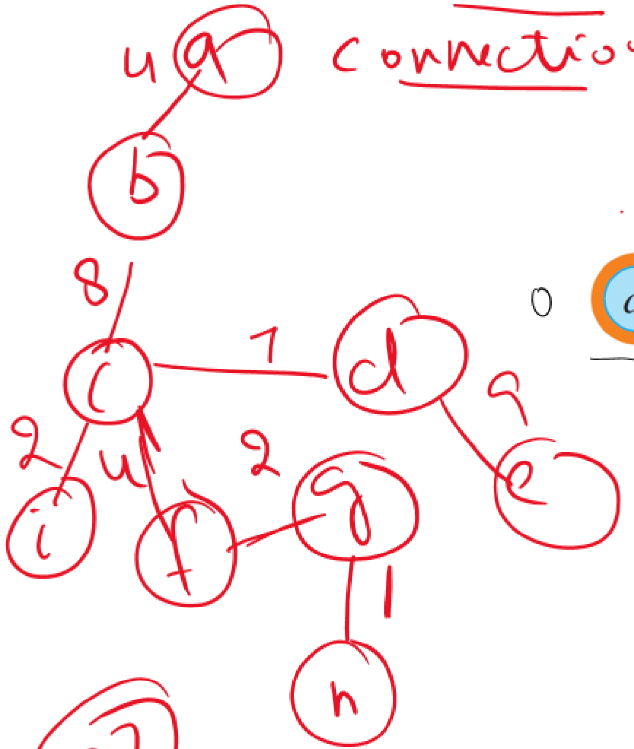


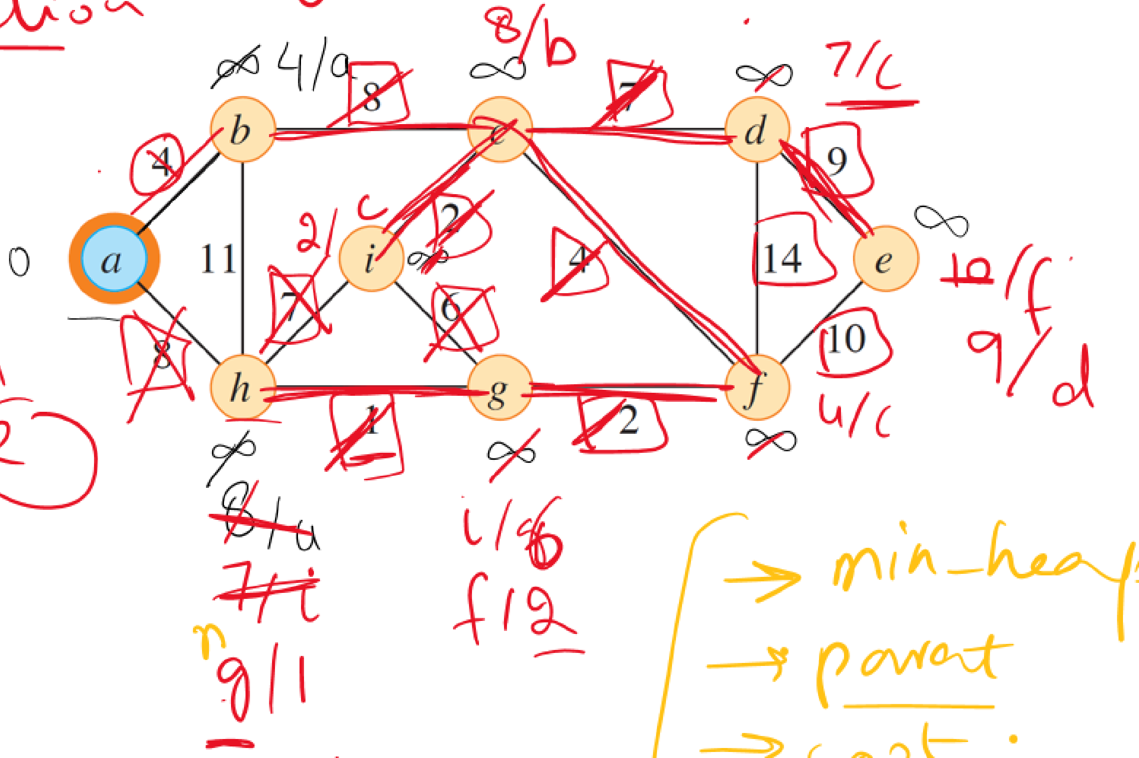
- Start with any single vertex tree.
- Get a 2-vertex tree by adding a cheapest edge.
- Get a 3-vertex tree by adding a cheapest edge.
- Grow the tree one edge at a time until the tree has n - 1 edges (and hence has all n vertices).

Prim's MST Algorithm

Tree → cost

connection → which vertex is connected with other one?





37

mst cost

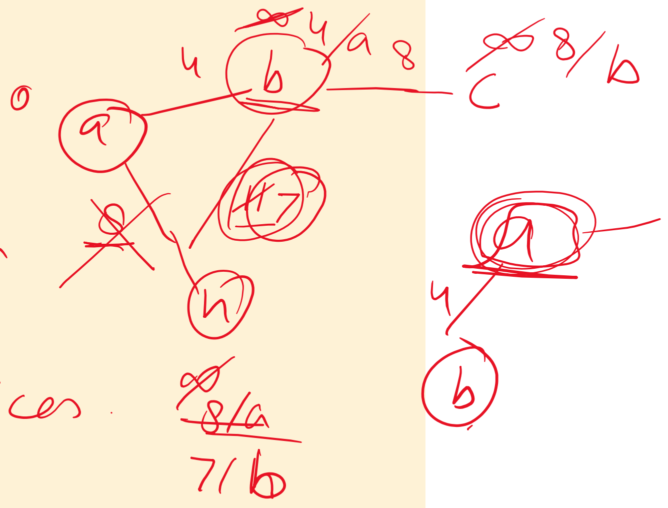
parent

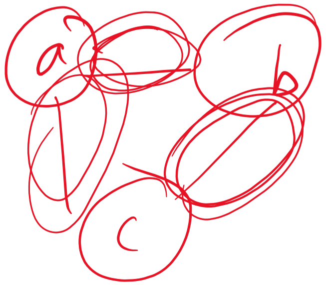
cheapest edge

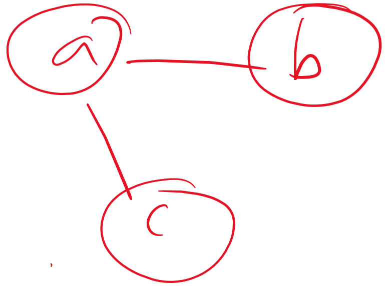
→ min_heap
→ parent
→ cost
→ Tree Store

MST-PRIM(G, w, r)

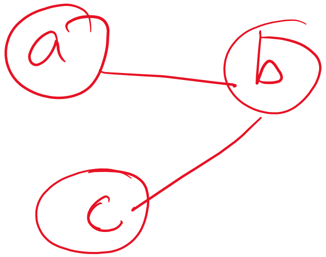
```
for each vertex u ∈ G.V
    u.key = ∞                          cost
    u.π = NIL                          parent / connection
r.key = 0
Q = ∅                                  min heap
for each vertex u ∈ G.V                insert all vertices
    INSERT(Q, u)
while Q ≠ ∅
    u = EXTRACT-MIN(Q)                 // add u to the tree
    for each vertex v in G.Adj[u]      // update keys of u's non-tree neighbors
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
            DECREASE-KEY(Q, v, w(u, v))
```

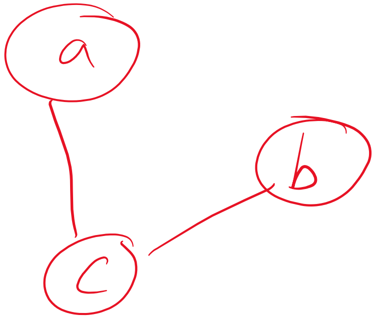


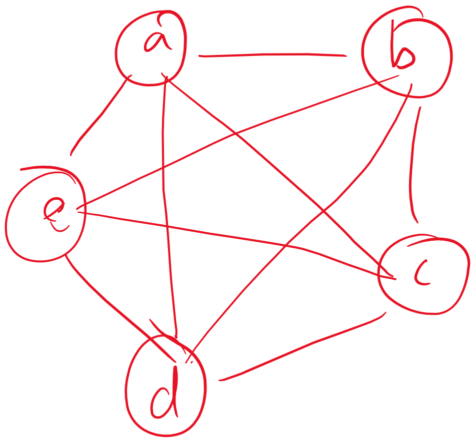




Scale up / 5







2^4 / Vertices

4X4

edges 6