Graph Algorithms

- **Search Algorithms**
  - **BestFirst algorithms**
    - Minimum Spanning Tree Construction

Spannnig tree of a connected graph is a subgraph
→ spanning means every vertex is included
Minimum Spannnig tree of a connected graph is a subgraph
→ spanning means every vertex is included
→ minimum – net weight is min,

Note. Minimum weight is unique, but the tree may not be unique.

Kruskal's Algorithm
Dijkstra
Prim
Kruskal

Complexity:
Brute Force: Create all spanning trees, determine the one with min weight. This leads to NP solution $\Theta(2^n)$.

There are three **Greedy algorithms**. Greedy algorithms do yield min spanning tree.

Kruskal – start with a min weight/length edge **-- best edge**
Prim **--** start at any vertex, look at local distances between two sets of vertices, determine best edge
Dijkstra – start at any vertex , assign priority tags to vertices, determine best edge based on vertex tag. **Dijkstra min spanning tree and shortest** paths problem are very much a like, with a slight variation.

Why spanning trees: makes search efficient and useful.

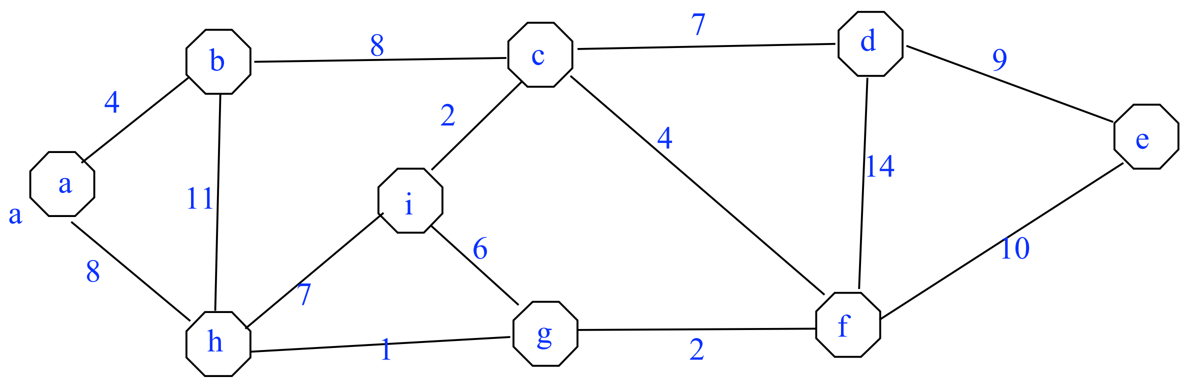
For connected undirected graph G=(V,E),
A spanning **tree T has |V|-1 edges from E.**
A spanning **tree T is an acyclic** subgraph to G.

Weight of the tree T is
weight(T) = sum of the weights of all the edges in the tree T
$= \sum_{(u,v) \in TreeEdges} w(u,v)$

Kruskal's min spanning algorithm is different.

Must contain an edge of min length

Example: Undirected Connected graph.



Greedy algorithm

Kruskal's algorithm

Complexity O(|E| lg |E|)

We want to choose |V|-1 edges from |E| edges so that they do not form a loop.

MST(G,T)

initialization

E_T= null, count=0;

//to begin with T may not be a tree, it is collection of edges to result in the min spanning tree.

//Sort edges in non-decreasing order of weights -- O(|E| lg |E|) insertion sort

//create each vertex as a singleton tree.

For k =1, m

$V_k = \{v_k\}$;

Invariant: E_T is contains edges from minimal spanning tree, $|E_T| \neq |V|-1$.

For j = 1, |E| in sorted order -- O(|E|)

Invariant: E_T is contains edges from minimal spanning tree, $|E_T| \neq |V|-1$.

Let $e_j = (v_{j1}, v_{j2})$;

If v_{j1} and v_{j2} are in different sets say V_x and V_y, O(lg|V|)

replace them with a single subtree with vertices V_x U V_y and insert the edge e_j in the set E_T.

This ensures that the insertion of e_j in E_T does not create a loop.

$E_T = E_T$ U $\{e_j\}$,

count++; If count=|V|-1 return E_T

Invariant: E_T is contains edges from minimal spanning tree, $|E_T| \neq |V|-1$.
Return E_T

Invariant: E_T contains all the edges of the minimal spanning tree.

Complexity $O(|E| (\lg |E| + \lg |V|))$
if ($|E| \geq |V|$), then $O(|E| \lg |E|)$

Once E_T is formed, start any vertex and edges in E_T, to trace the tree.

Definitions.

Cut (S, R) S is a subset of vertices in the minimum spanning tree and R is the remaining set of vertices in the graph.

An edge (u, v) **crosses** the cut if u in S and v is in R or v in S and u is in R

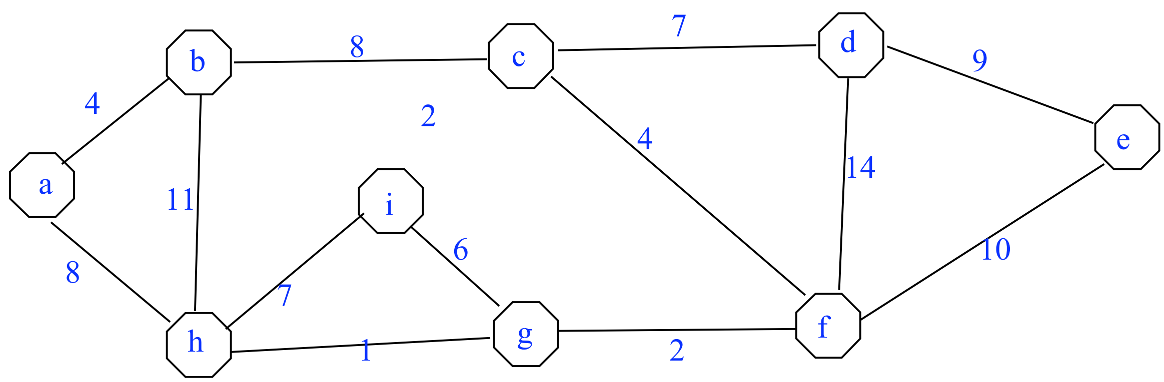
The cut is **compatible** with the tree if no edge in T crosses the cut.
An edge is **light edge** if the weight of this edge is minimum of the weights of all the edges crossing the cut. In the case of multiple min edges, select one arbitrarily.

Start with S =null or ∅, on completion we end with S=V

Prim's algorithm

Min spanning tree of a connected , undirected weighted graph G=(V,E)

First Example



Dijkstra MST-PRIM(G,w,r)
Note. Initial step may be: O(|E|)
Or take an edge with min weight, put it in T and its end points in V_T

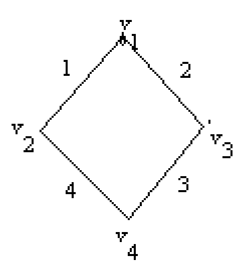
inititalization
For each u V -- O(|V|)
  d(u) = • ;// local distance from tree nodes
  p(u)= null// parent of node
//r -- root is arbitrarily selected vertex
d(r) = 0
E_T = null
V_T = null
H = V(G); min heap with values associated with vertices -- No work is needed here, only r is the root of the heap
Invariant: E_T is contains edges from minimal spanning tree, $|E_T| \neq |V|-1$.
For k=0 to |V|-1 O(|V|) **Invariant: E_T is contains edges from minimal spanning tree, $|E_T| \neq |V|-1$.**
    u = Extract-min(H) -- O(lg |V|) if k>0, $E_T = E_T$ U {(p(u),u)}
  $V_T = V_T$ U {u}
  for each v in adj(u) -- O(|E|)
    if v H and w(u,v) < d(v)
      parent(v) = u
      d[v]=w(u,v)
      adjust heap H from v upward. -- O(lg |V|)

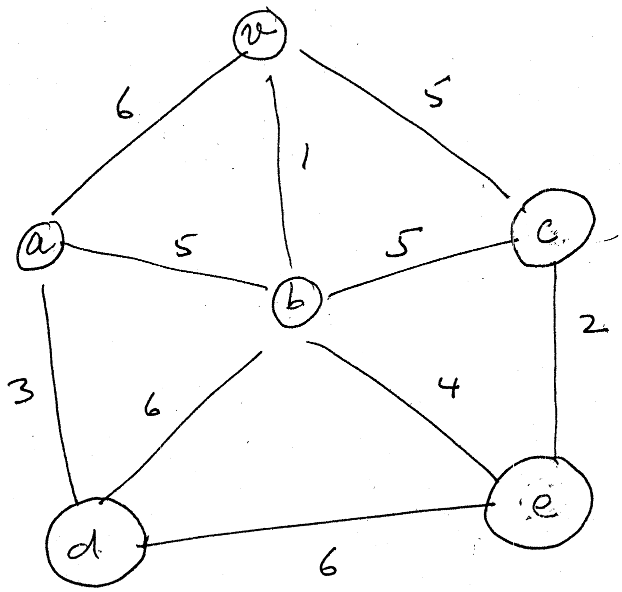
Invariant: E_T is contains all the edges from minimal spanning tree, $|E_T| = |V|-1$.

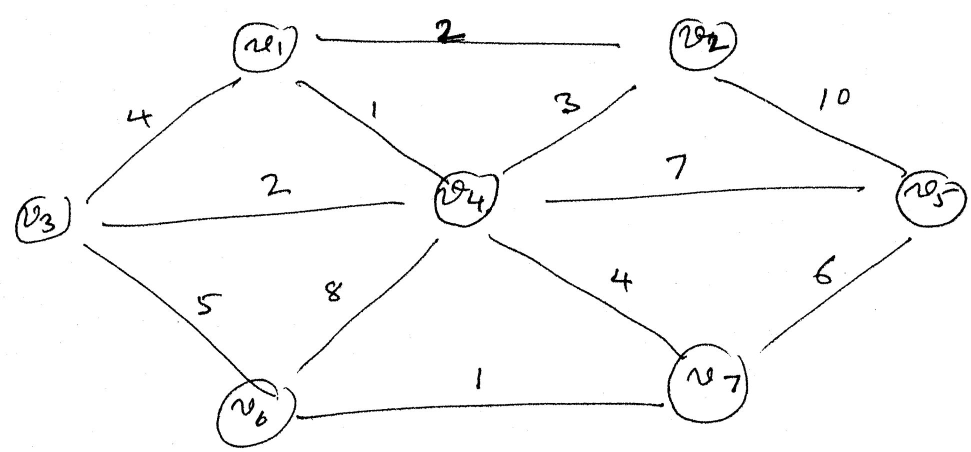
Complexity O ((|E|+|V|)lg |V|)) same as O ((|E|)lg |V|))
Once $\mathbf{E_T}$ is complete, start at any vertex and $\mathbf{E_T}$ as the edges.

<u>Note</u>: Minimum Cost tree must include a min cost edge in the graph.

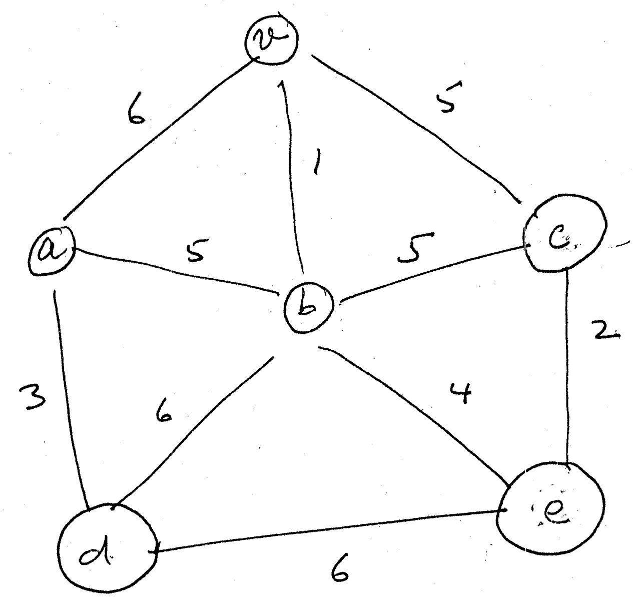
Examples Create minimum spanning trees for the following graphs.

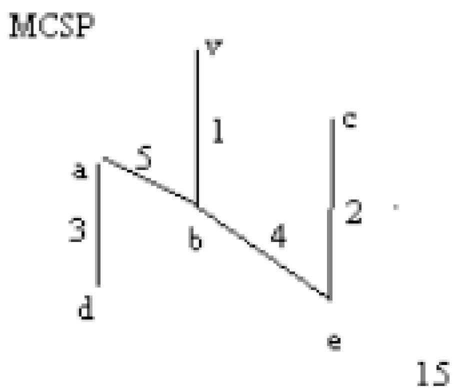


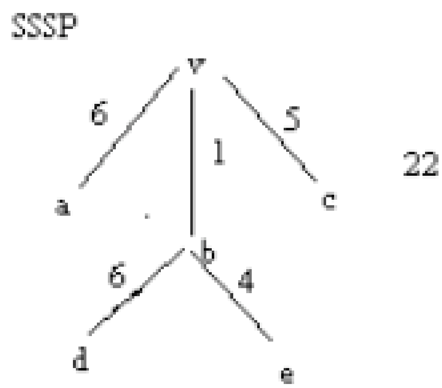




|E| - updates
|V| - removals
(|E|+|V|) updates & removal each heap balancing lg|V|

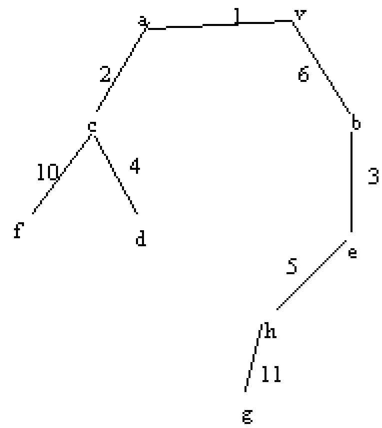






Find the min spanning tree for the graph (show all steps)

every time add an edge of min cost



cost from f to g
= 38
cost from h to f = 27
cost of min spanning tree = 42