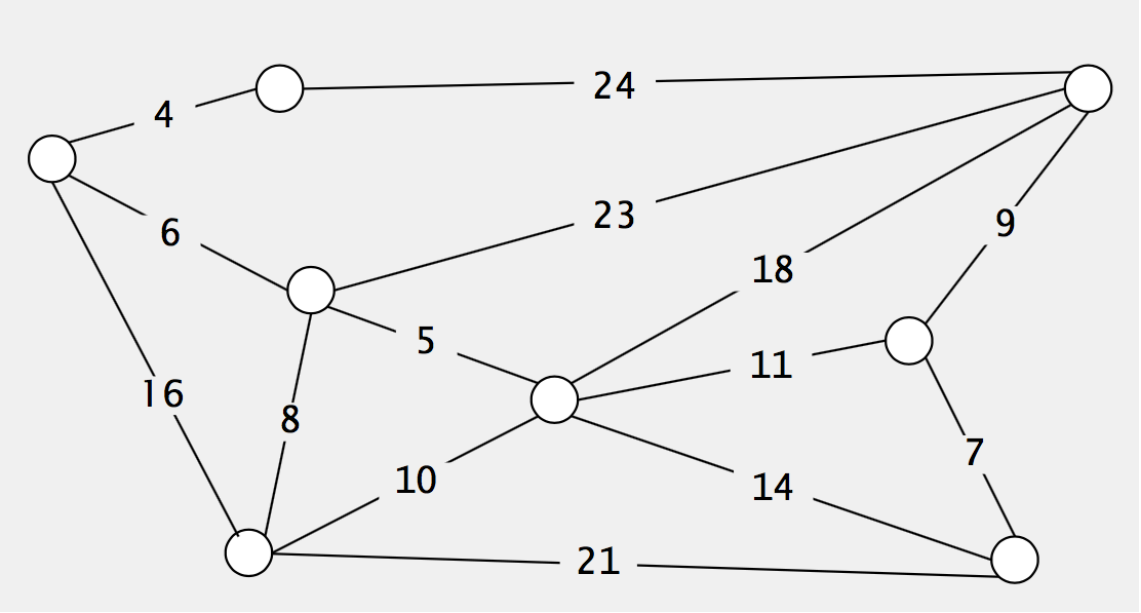
CMSC 132: Object-Oriented Programming II

Minimum Spanning Trees

Minimum spanning tree

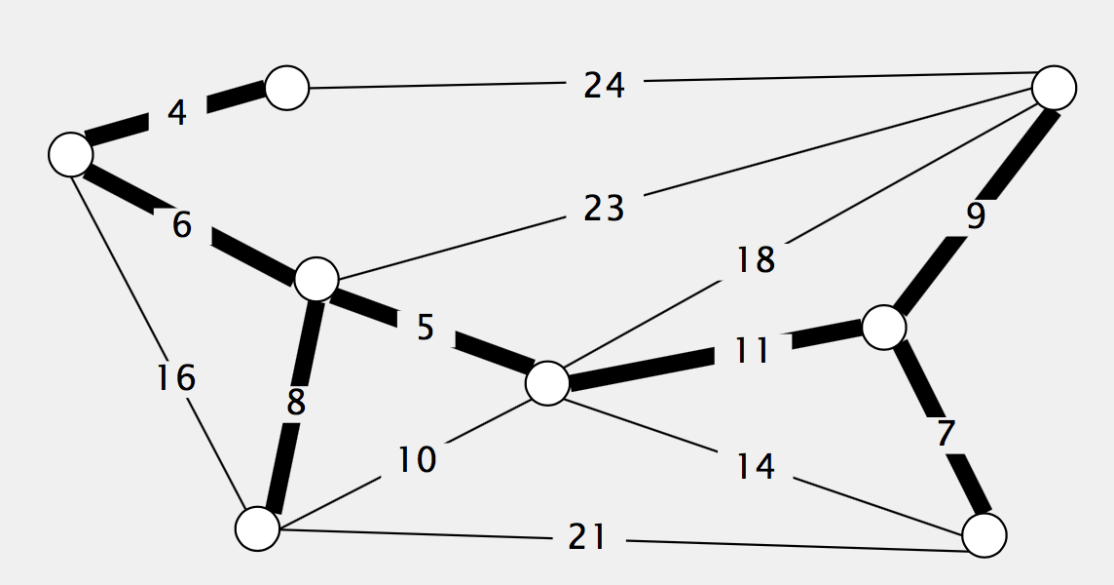
- **Given:** Undirected graph *G* with positive edge weights (connected).
- **Definition:** A spanning tree of *G* is a subgraph *T* that is connected and acyclic.
- **Goal:** Find a min weight spanning tree.



Graph G

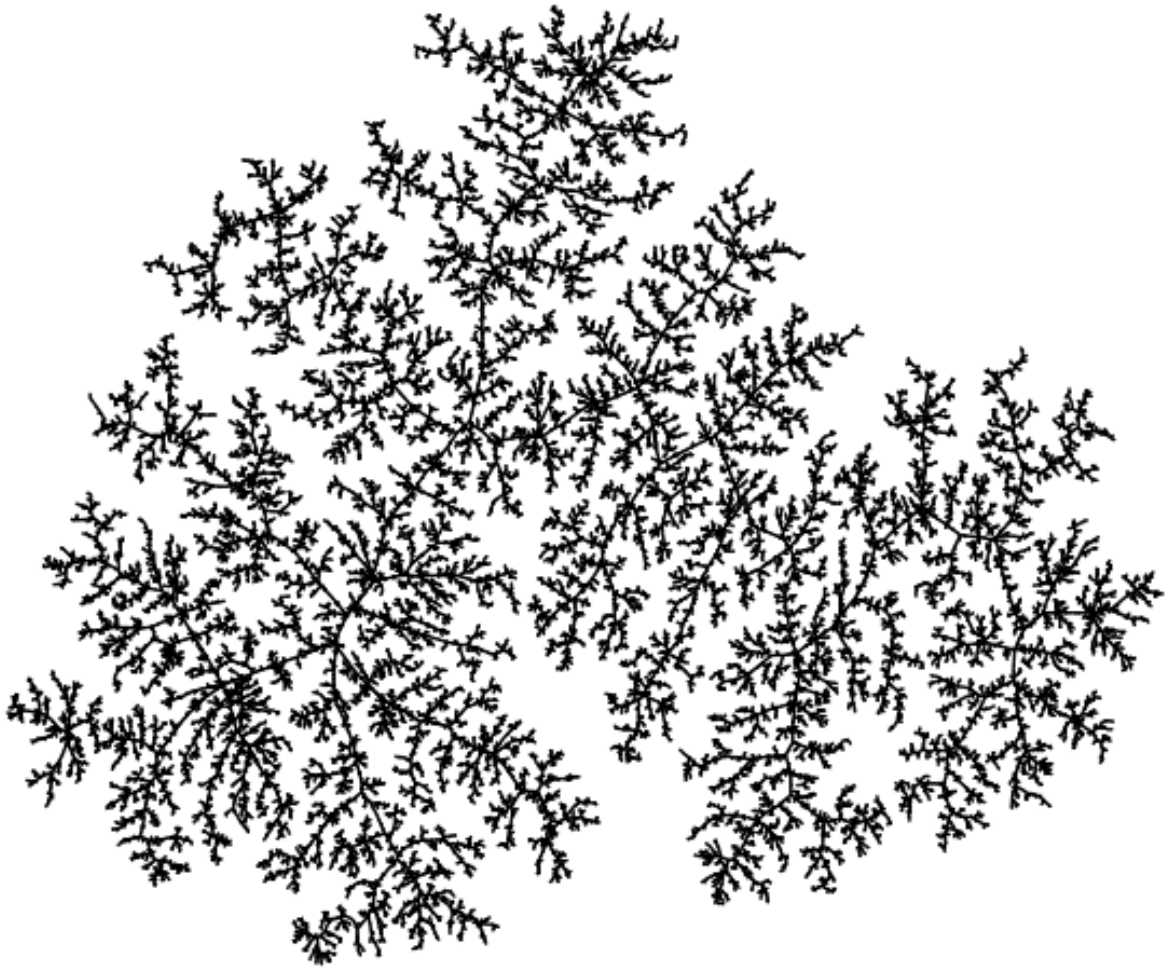
Minimum spanning tree

- **Given:** Undirected graph *G* with positive edge weights (connected).
- **Definition:** A spanning tree of *G* is a subgraph *T* that is connected and acyclic.
- **Goal:** Find a min weight spanning tree.



Spanning Tree T: cost = 4+6+8+5+11+9+7 = 50

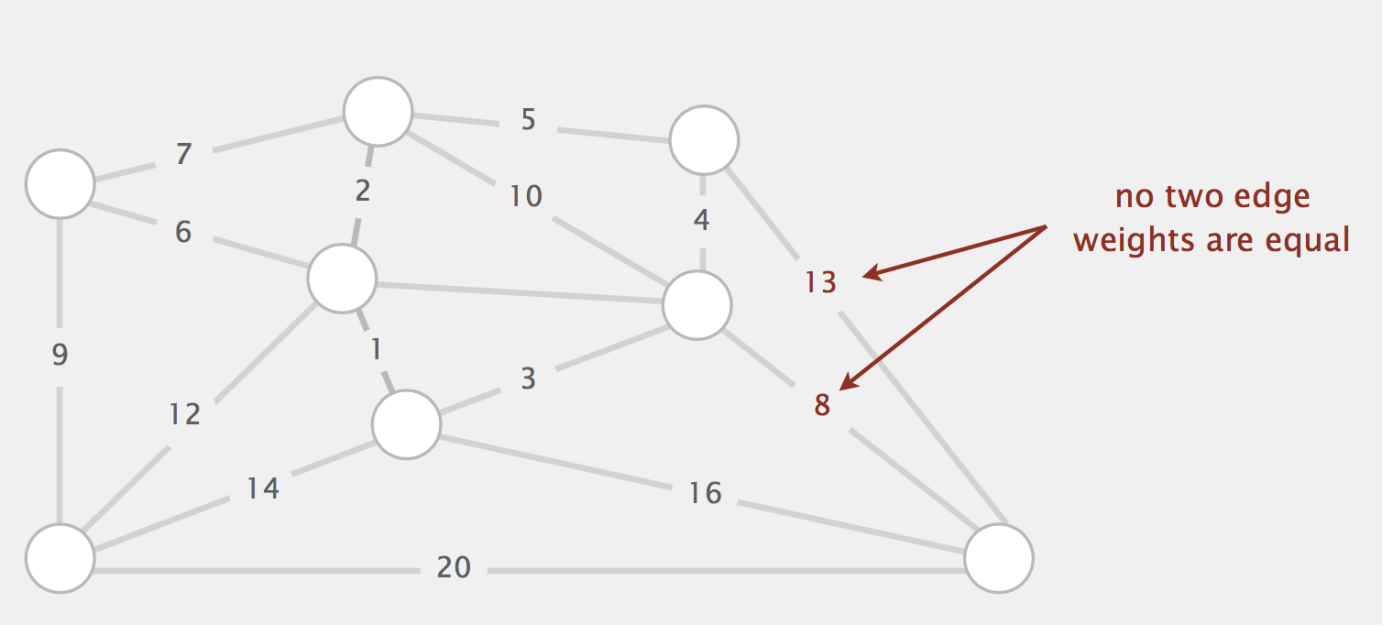
MST of random graph



https://who.rocq.inria.fr/Nicolas.Broutin/gallery.html

Simplifying assumptions

- Simplifying assumptions.
  - Edge weights are distinct.
  - Graph is connected.
- Consequence. MST exists and is unique.



MST Algorithms

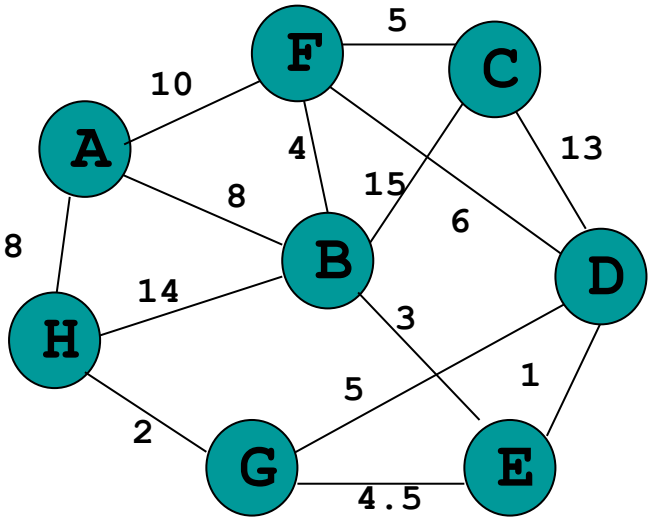
- Greedy Algorithms: Prim's and Kruskal's.
- Both Prim's and Kruskal's Algorithms work with undirected graphs
- Both work with weighted and unweighted graphs but are more interesting when edges are weighted
- Both are greedy algorithms that produce optimal solutions

Kruskal's algorithm

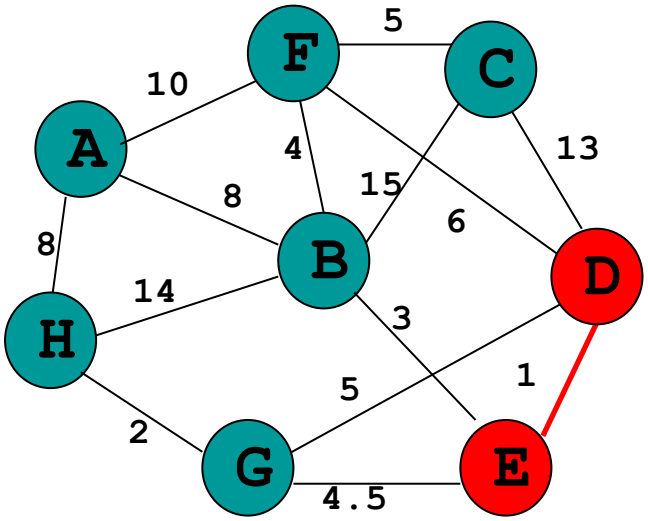
- Minimum-spanning-tree algorithm
  - Consider edges in ascending order of weight.
  - Add next edge to tree *T* unless doing so would create a cycle.
  - If the graph is not connected, then it finds a minimum spanning forest

Kruskal's algorithm Demo

Consider an undirected, weight graph

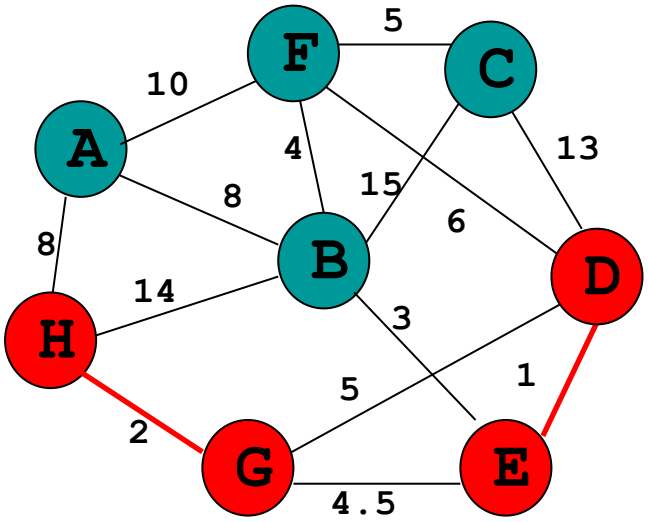


Kruskal's algorithm Demo



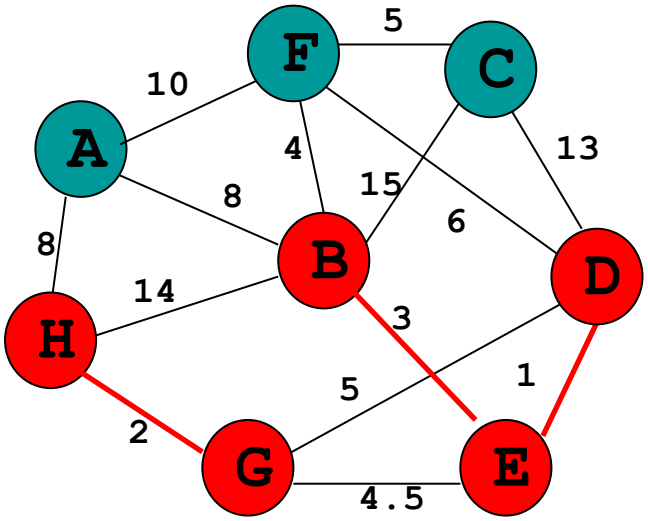
Add Edge (E,1,D)

Kruskal's algorithm Demo



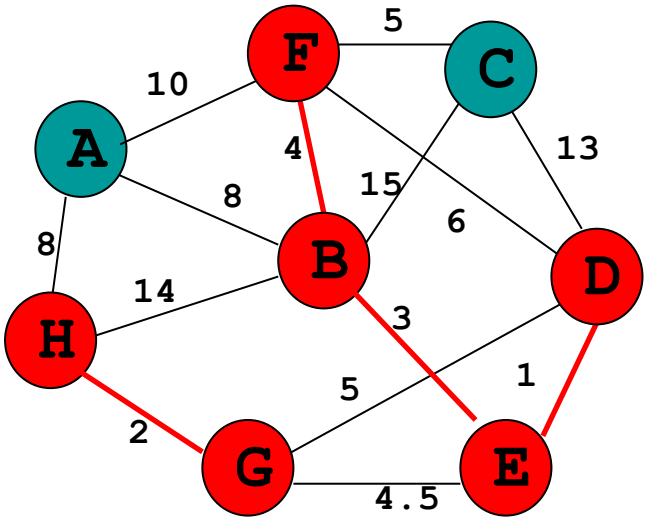
Add Edge (H,2,G)

Kruskal's algorithm Demo



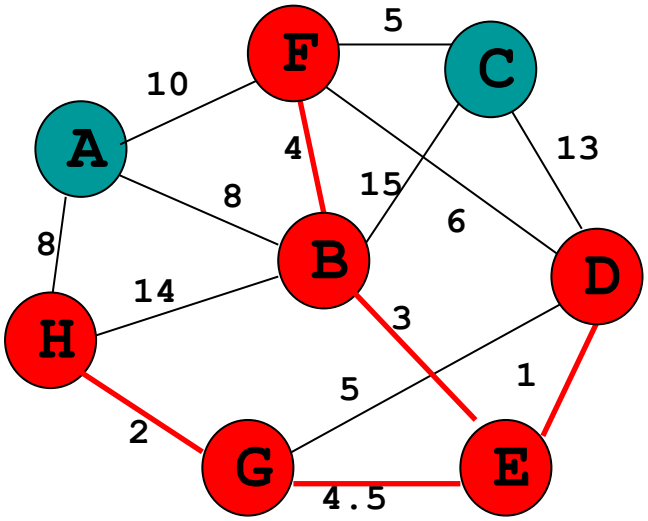
Add Edge (E,3,B)

Kruskal's algorithm Demo



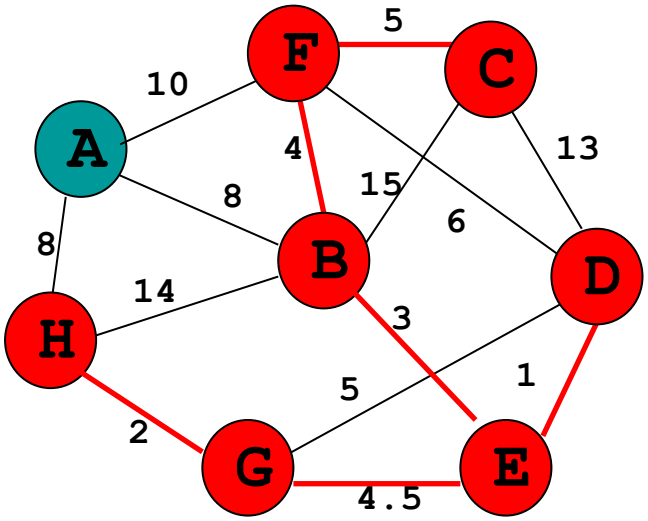
Add Edge (F,4,B)

Kruskal's algorithm Demo



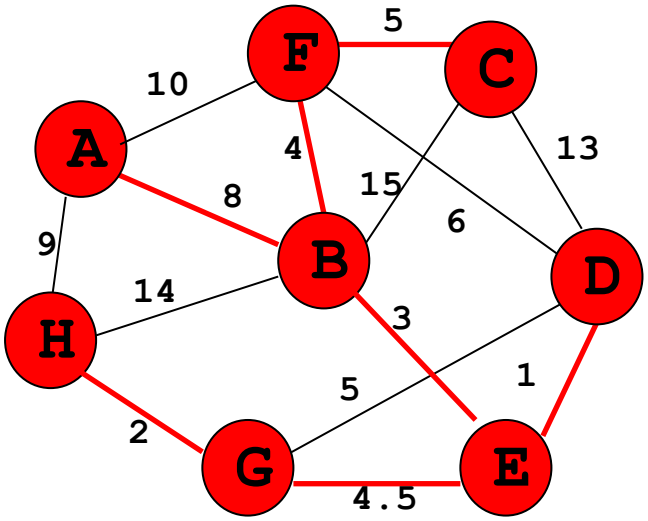
Add Edge (G,4,E)

Kruskal's algorithm Demo



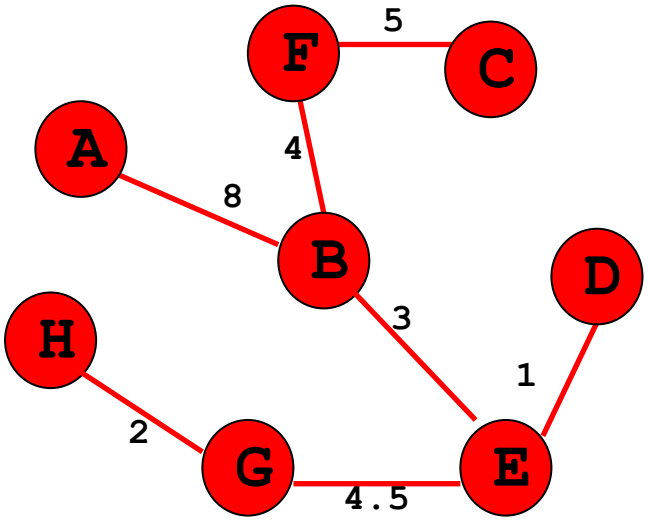
Add Edge (F,5,C)

Kruskal's algorithm Demo



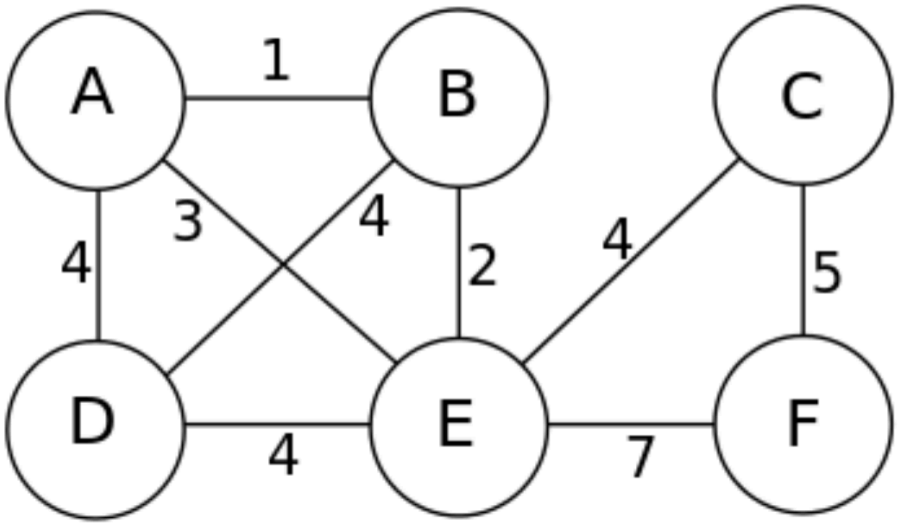
Add Edge (B,8,A)

Kruskal's algorithm Demo

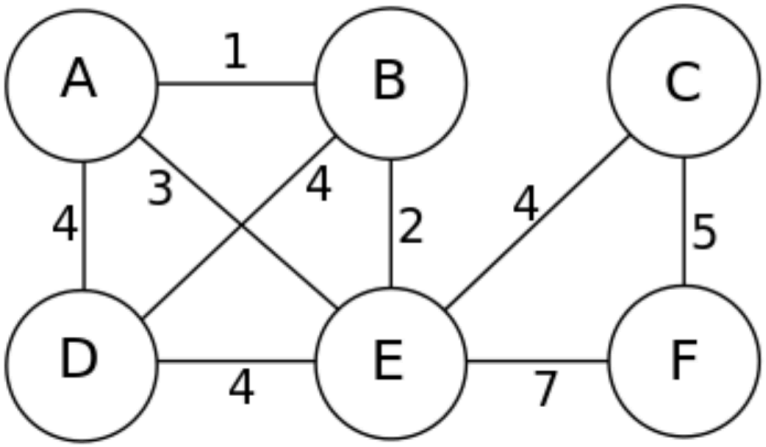


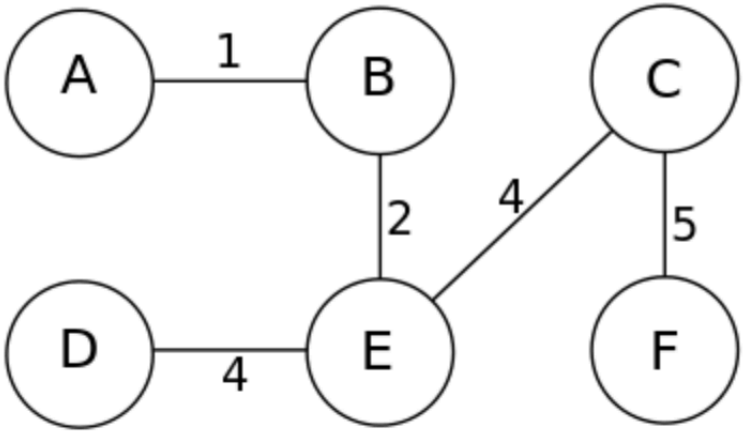
Cost: 3+4+1+4+8+4+5 = 29

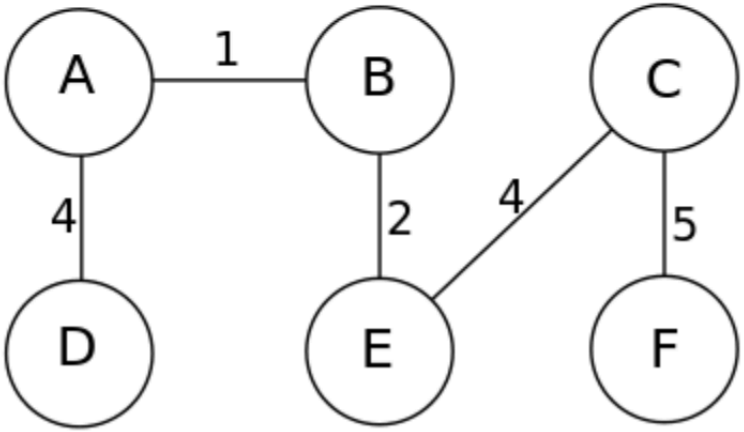
Kruskal's Algorithm Demo



Kruskal's Algorithm Demo





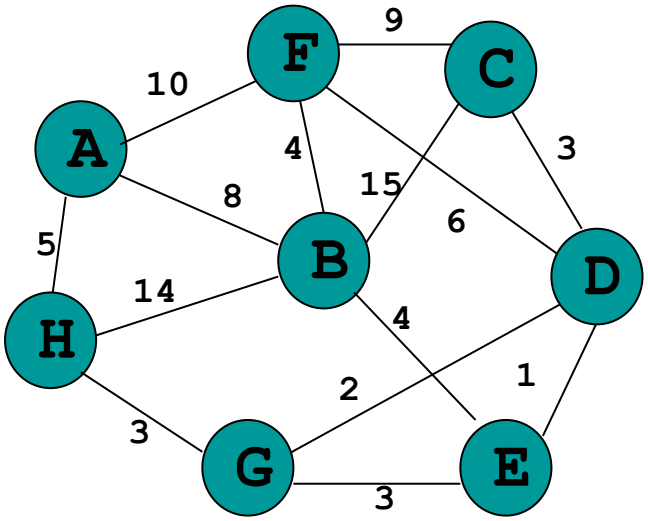


Prim's algorithm

- Builds the tree one vertex at a time
- Starts from an arbitrary starting vertex
- Each step adds the cheapest possible connection from the tree to another vertex.

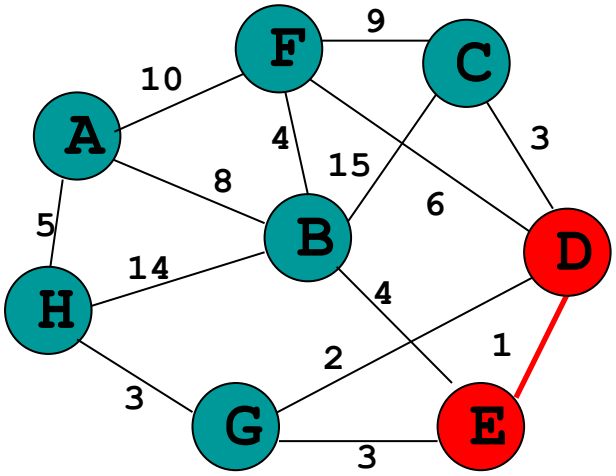
Prim's algorithm Demo

Consider an undirected, weight graph



Prim's algorithm Demo

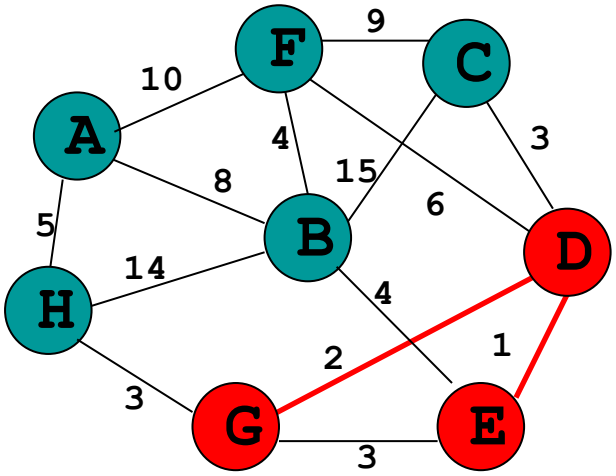
Consider an undirected, weight graph



Add Edge (E,1,D)

Prim's algorithm Demo

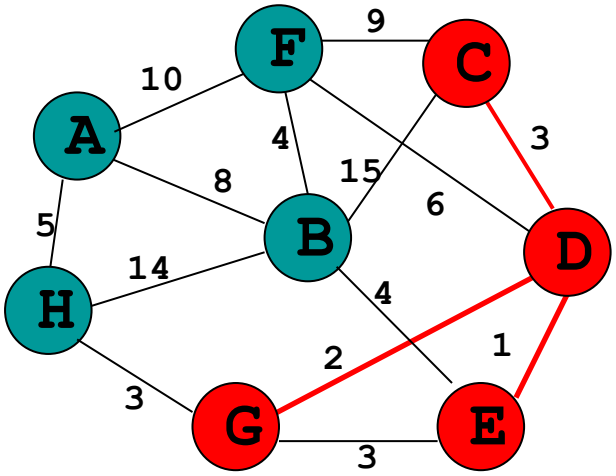
Consider an undirected, weight graph



Add Edge (D,2,G)

Prim's algorithm Demo

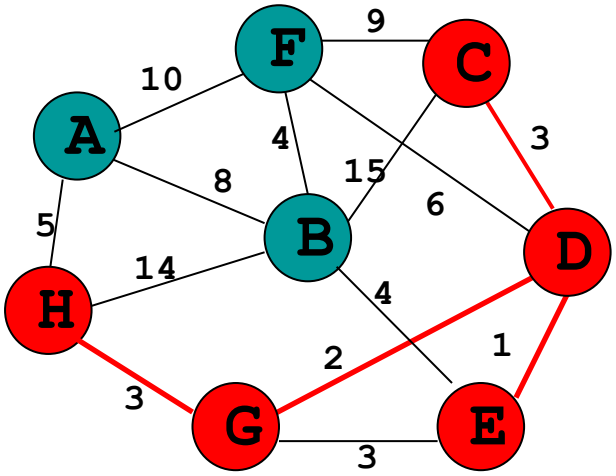
Consider an undirected, weight graph



Add Edge (D,3,C)

Prim's algorithm Demo

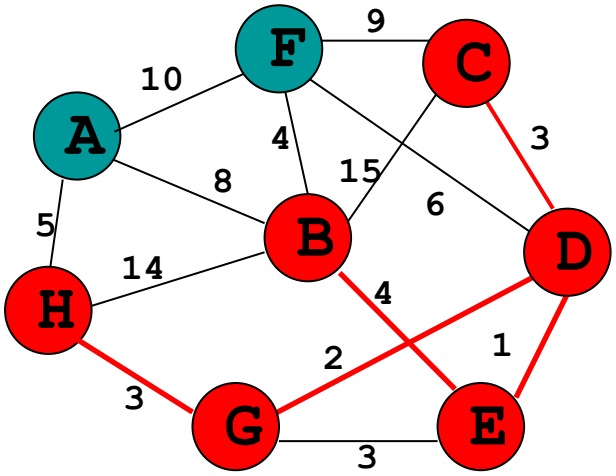
Consider an undirected, weight graph



Add Edge (G,3,H)

Prim's algorithm Demo

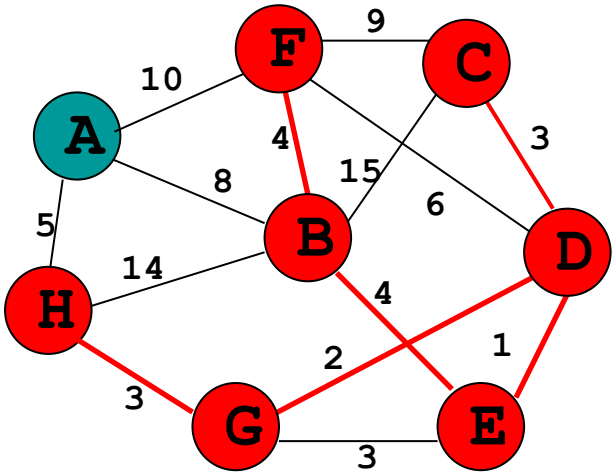
Consider an undirected, weight graph



Add Edge (E,4,B)

Prim's algorithm Demo

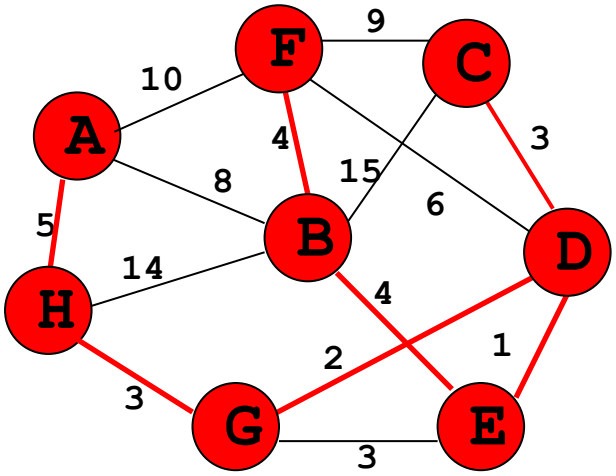
Consider an undirected, weight graph



Add Edge (F,4,B)

Prim's algorithm Demo

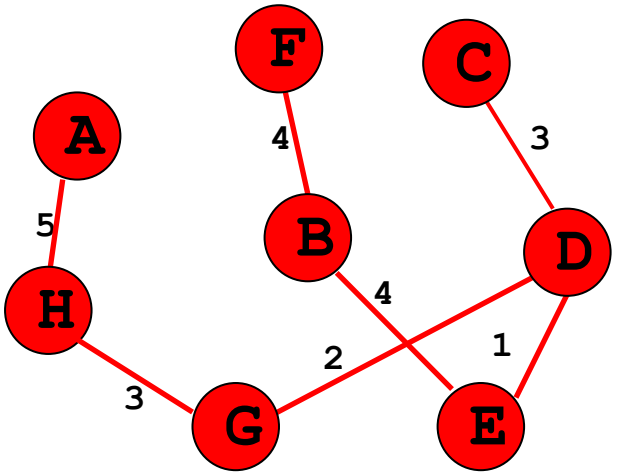
Consider an undirected, weight graph



Add Edge (H,5,A)

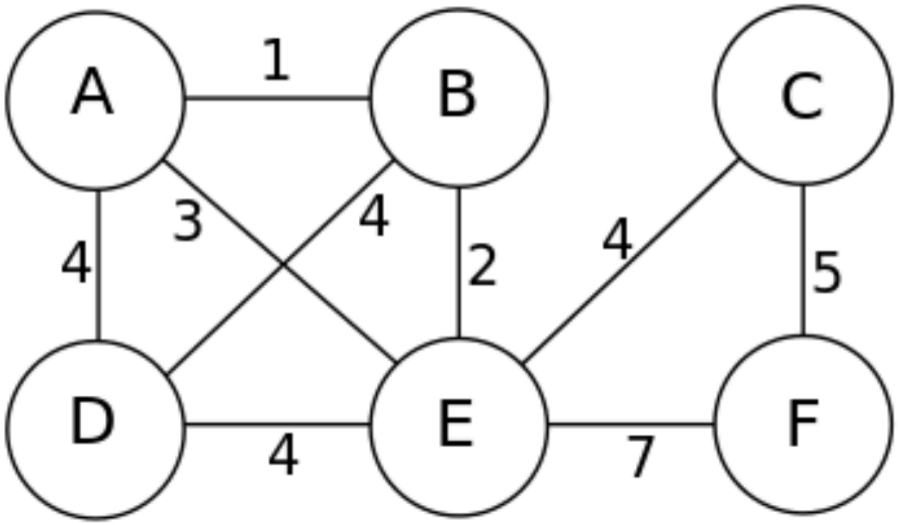
Prim's algorithm Demo

Consider an undirected, weight graph



Cost: 5+3+2+3+1+4+4 = 20

Prim's Algorithm Demo



Prim's Algorithm Demo