Ch 11.5 Minimum Spanning Tree

- Problem:
  - A town has a set of houses and a set of roads
  - A road connects 2 and only 2 houses.
  - A road connecting houses u and v has a repair cost $w(u, v)$
- Goal:
  - Repair enough (and no more) roads such that

    a) Everyone stays connected.

    - Can reach every house from all other houses.

    b) Total repair cost is minimum.

UTD

Model As A Graph

- Undirected graph $G = (V, E)$.
- *Weight* $w(u, v)$ on each edge $(u, v) \in E$.
- Find $T \subseteq E$ such that

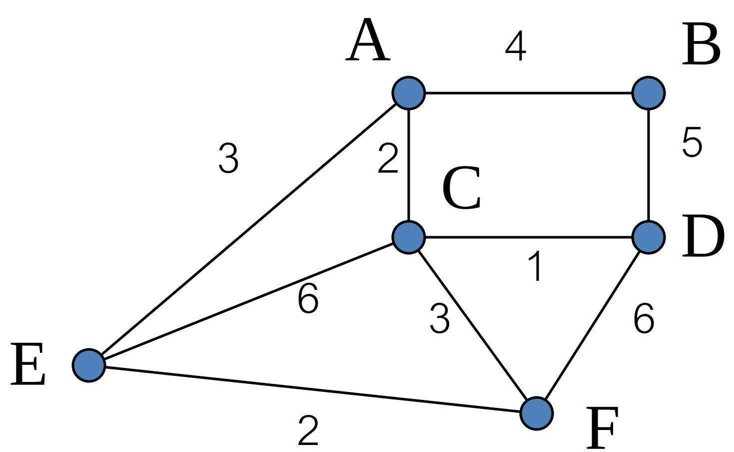
1. T connects all vertices (T is a ***spanning tree***), and
2. $w(T) = \sum_{(u,v) \in T} w(u, v)$ is minimized.

A spanning tree whose weight is minimum over all spanning trees is called a ***minimum spanning tree***, or ***MST***.

UTD

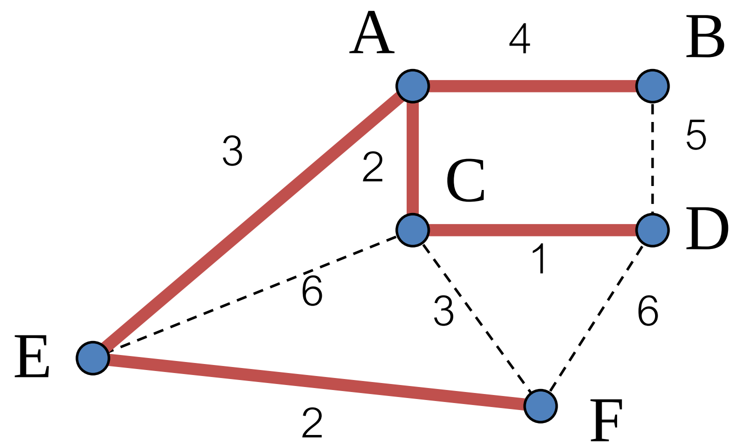
minimum Spanning Tree

- A minimum spanning tree T is a subgraph of a *weighted* graph G which contains all the verticies of G *and* whose edges have the minimum summed weight.
- Example weighted graph G:

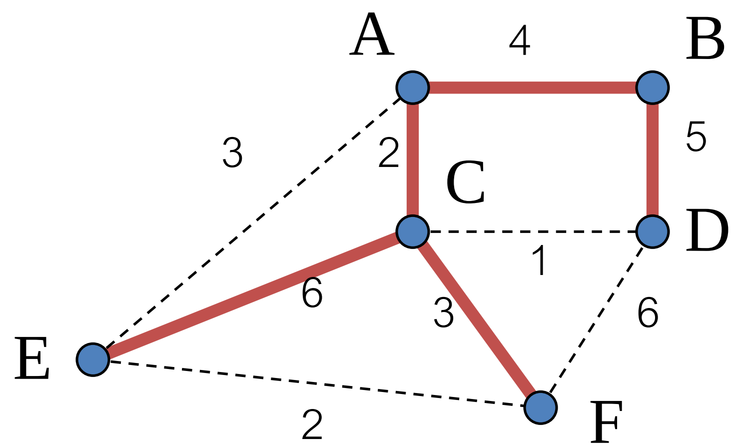


minimum Spanning Tree

- A *minimum* spanning tree (weight = 12):



- A *non-minimum* spanning tree (weight = 20):



minimum Spanning Tree

Some Properties of MST

- It has $|V| - 1$ edges.
- It has no cycles.
- It might not be unique.

minimum Spanning Tree

- Two algorithms can be used:
  - Prim's Algorithm
    - Robert Prim, 1957
  - Kruskal's Algorithm
    - Joseph Kruskal, 1956

Prim's Algorithm

- Prim's algorithm finds a minimum spanning tree T by iteratively adding edges to T.

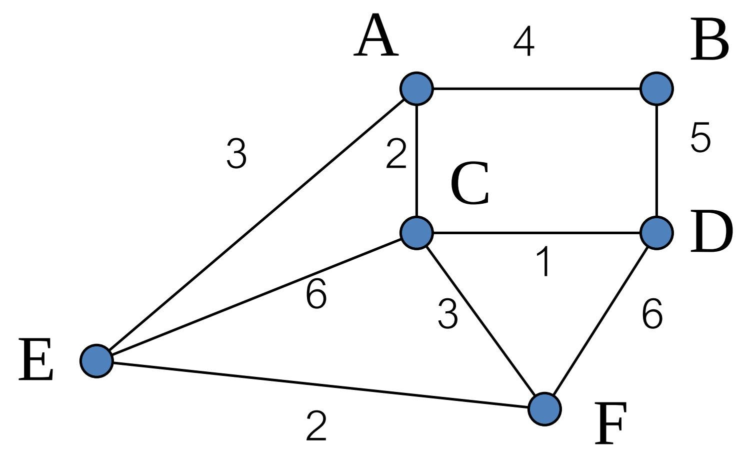
- At each iteration, a minimum-weight edge is added that does not create a cycle in the current T.

Prim's Algorithm

- There may be more than one minimum spanning tree for a given connected weighted simple graph.

- If there are two edges with similar smallest weight, chose either one.

Informal Algorithm



- For the graph G.
- 1) Add any vertex to T
  - e.g A, T = {A}
- 2) Examine all the edges leaving {A} and add the vertex with the smallest weight.
  - edge weight

    (A,B) 4

    (A,C) 2

    (A,E) 3
  - add edge (A,C), T becomes {A,C}

UTD

Informal Algorithm

- 3) Examine all the edges leaving {A,C} and add the vertex with the smallest weight.
edge	weight
(A,B)	4
(A,E)	3
(C,F)	3

edge	weight
(C,D)	1
(C,E)	6
  - add edge (C,D), T becomes {A,C,D}

Informal Algorithm

- 4) Examine all the edges leaving {A,C,D} and add the vertex with the smallest weight.
edge	weight	edge	weight
(A,B)	4	(D,B)	5
(A,E)	3	(C,E)	6
(C,F)	3	(D,F)	6
  - add edge (A,E) or (C,F), it does not matter
  - add edge (A,E), T becomes {A,C,D,E}

Informal Algorithm

- 5) Examine all the edges leaving {A,C,D,E} and add the vertex with the smallest weight.
  - edge weight

edge	weight	edge	weight
(A,B)	4	(D,B)	5
(C,F)	3	(D,F)	6
(E,F)	2		

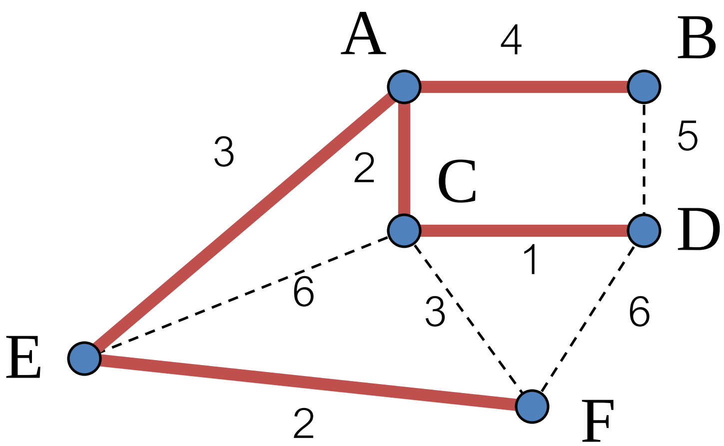
  - add edge (E,F), T becomes {A,C,D,E,F}

Informal Algorithm

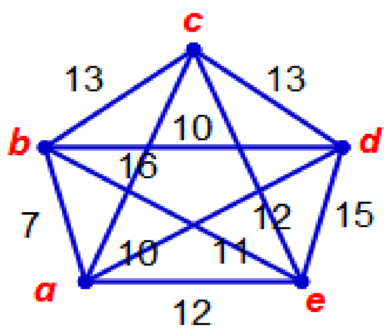
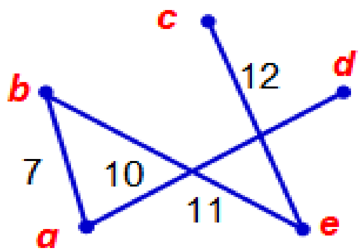
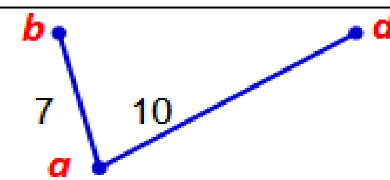
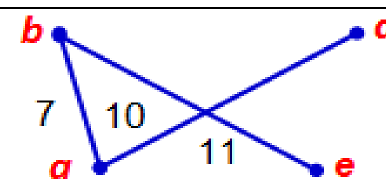
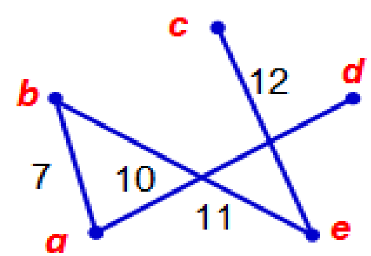
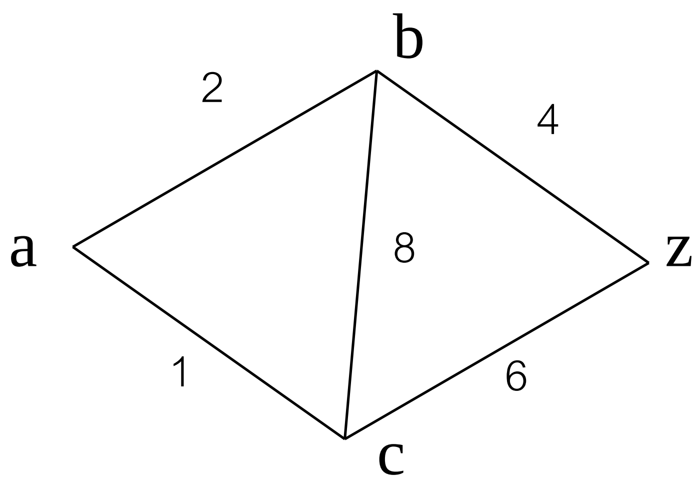
- 6) Examine all the edges leaving {A,C,D,E,F} and add the vertex with the smallest weight.
edge	weight	edge	weight
(A,B)	4	(D,B)	5
  - add edge (A,B), T becomes {A,B,C,D,E,F}
- All the verticies of G are now in T, so we stop.

Prim's Algorithm

- Resulting minimum spanning tree (weight = 12):



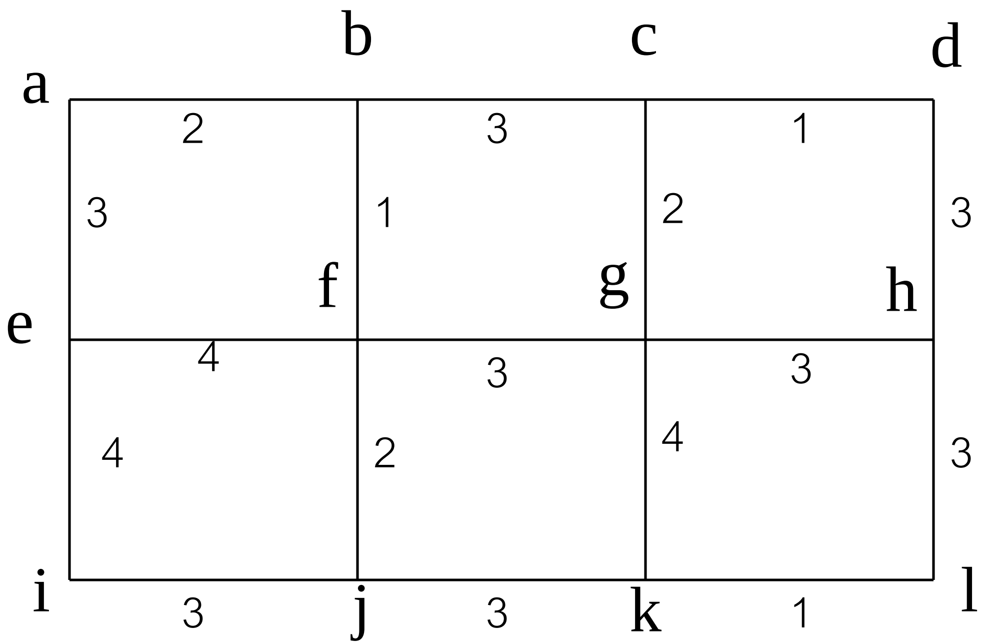
Pseudocode

- ```
  Tree prim(Graph G, int numVerts)
  {
    Tree T = anyVert(G);
    for i = 1 to numVerts-1{
      Edge e = an edge of minimum weight incident to
               a vertex in T and not forming a
               cycle in T when added to T;

      T = T with e added;
    }
    return T
  }
  ```
  ```

Prim's Algorithm



FIGURE 1

The minimum spanning tree is given by:



Total weight = 40

CHOICE	EDGE	WEIGHT	SPANNING TREE
1	$\{a, b\}$	7	
2	$\{a, d\}$	10	
3	$\{b, e\}$	11	
4	$\{e, c\}$	12	

A Greedy Algorithm

- Prim's algorithm is an example of a *greedy algorithm*
  - its choice at each iteration does not depend on any previous choices
- A greedy algorithm does the best thing at this time without considering the past (or the future).

Greed is not Always Best

- An example where greed is not the best approach:
  - a greedy shortest-path algorithm can lead to a non-optimal solution (not the best)
  - the algorithm selects an edge with the minimum weight connected to the last vertex added

Greed is not Always Best

- Find the shortest path a --> z.
- The greedy algorithm produces (a,c,z), but the shortest is (a,b,z)



Kruskal's Minimum Spanning Tree Algorithm

- At the start, the minimum spanning tree T consists of all the verticies of the weighted graph G, but no edges.
- At each iteration, add an edge e to T having minimum weight that does not create a cycle in T.
- When T has n-1 edges, stop.

Kruskal's Minimum Spanning Tree Algorithm

- There may be more than one minimum spanning tree for a given connected weighted simple graph.
- If there are two edges with similar smallest weight, chose either one.

Pseudocode

- ```
  Tree kruskal(Graph G, int numVerts)
  {
    Tree T = allVerts(G);
    for i = 1 to numVerts-1 {
      Edge e = an edge of minimum weight in G
               and not forming a
              cycle in T when added to T;

      T = T with e added;
    }
    return T
  }
  ```
  ```

Example

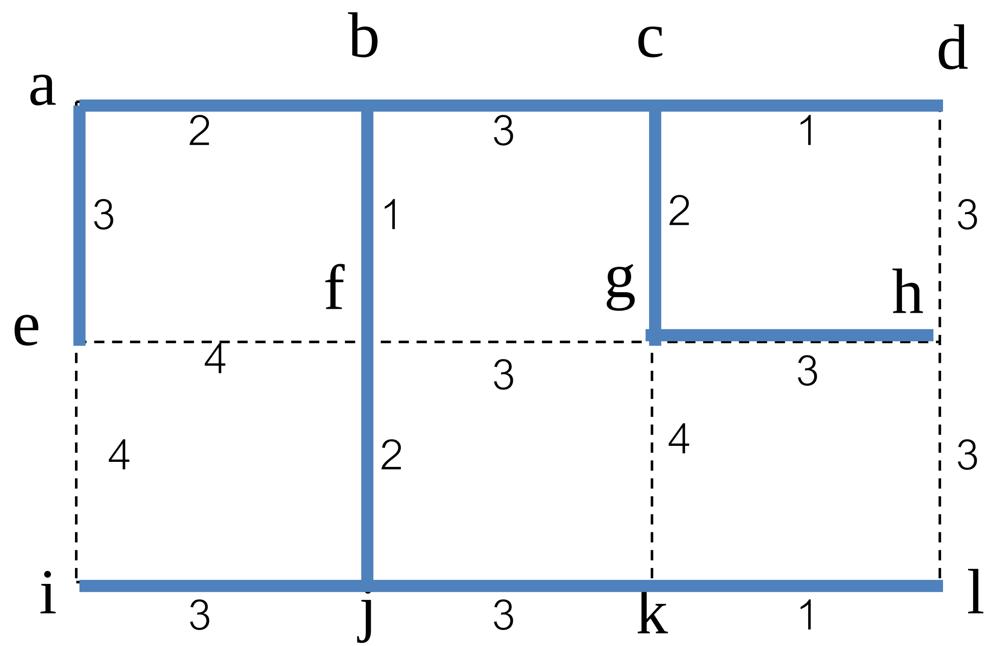
- Graph G:



iter	edge
1	(c,d)
2	(k,l)
3	(b,f)
4	(c,g)
5	(a,b)
6	(f, j)
7	(b,c)
8	(j,k)
9	(g,h)
10	(i, j)
11	(a,e)

Example

- Minimum spanning tree (weight = 24):



Kruskal's Algorithm

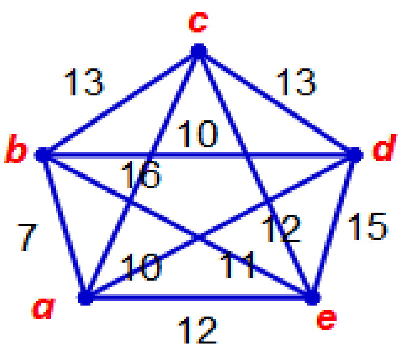
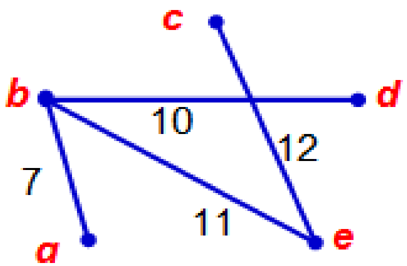
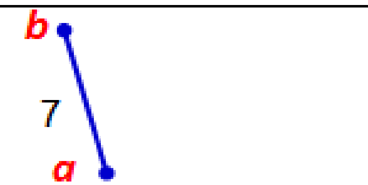
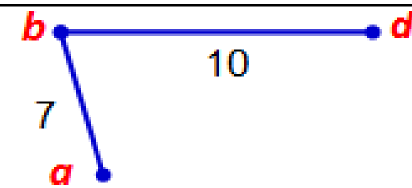
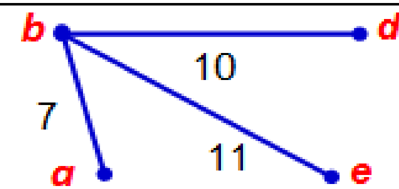
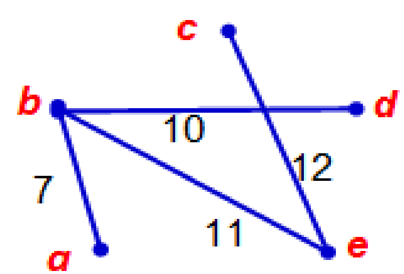


FIGURE 1

The minimum spanning tree is given by:



Total weight = 40

CHOICE	EDGE	WEIGHT	SPANNING TREE
1	$\{a, b\}$	7	
2	$\{b, d\}$	10	
3	$\{b, e\}$	11	
4	$\{e, c\}$	12	

Difference between Prim and Kruskal

- Prim's algorithm chooses an edge that *must* already be connected to a vertex in the minimum spanning tree T.

- Kruskal's algorithm can choose an edge that *may not* already be connected to a vertex in T.

Exercise

1. Construct a minimum spanning tree for each of the following connected weighted graphs using Prim's and Kruskal's algorithm.

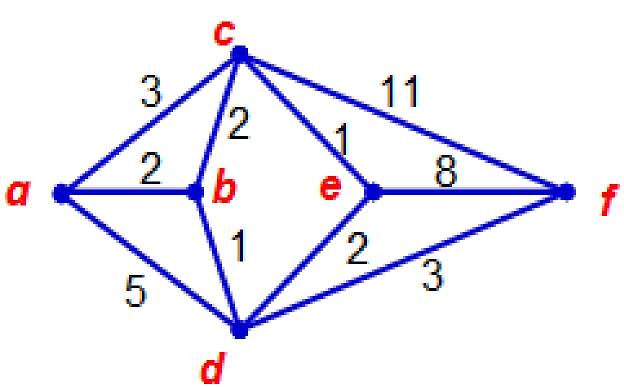


FIGURE 1

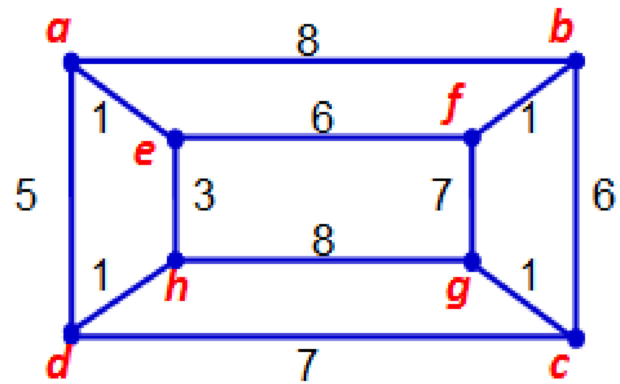


FIGURE 2