ZAMAN UNIVERSITY

Data Structures and Algorithms

Chapter 6

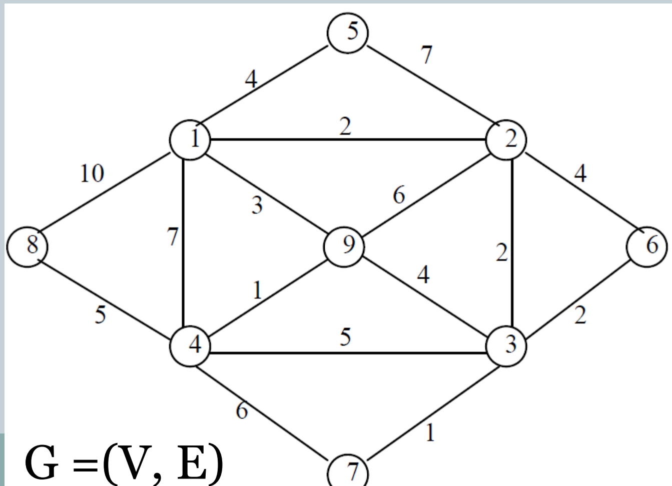
Graphs

Outline

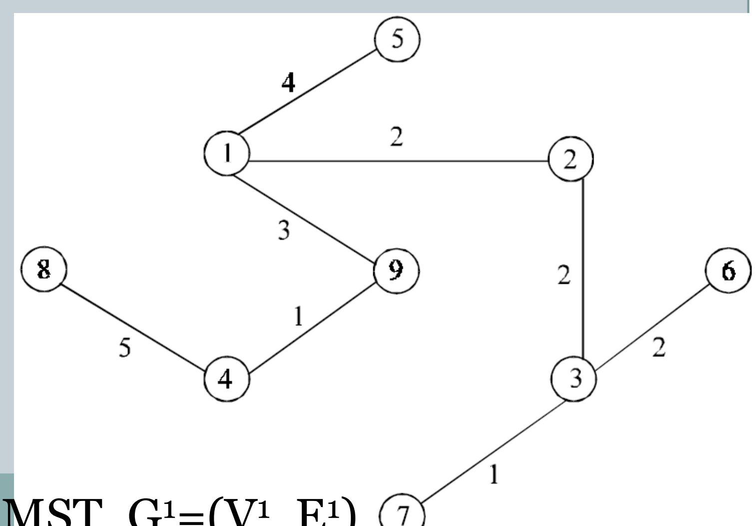
- Terminology and Representations
- Graph Traversals
- Shortest-Paths Problems
- Minimum-Cost Spanning Trees

Minimum Cost Spanning Tree

- A minimum spanning tree (MST) for a graph G = (V, E) is a sub graph $G^1 = (V^1, E^1)$ of G contains all the vertices of G.
  1. The vertex set V^1 is same as that at graph G.
  2. The edge set E^1 is a subset of G.
  3. And there is no cycle.
- It connects all the vertices together with the minimal total weighting for its edges.



G =(V, E)



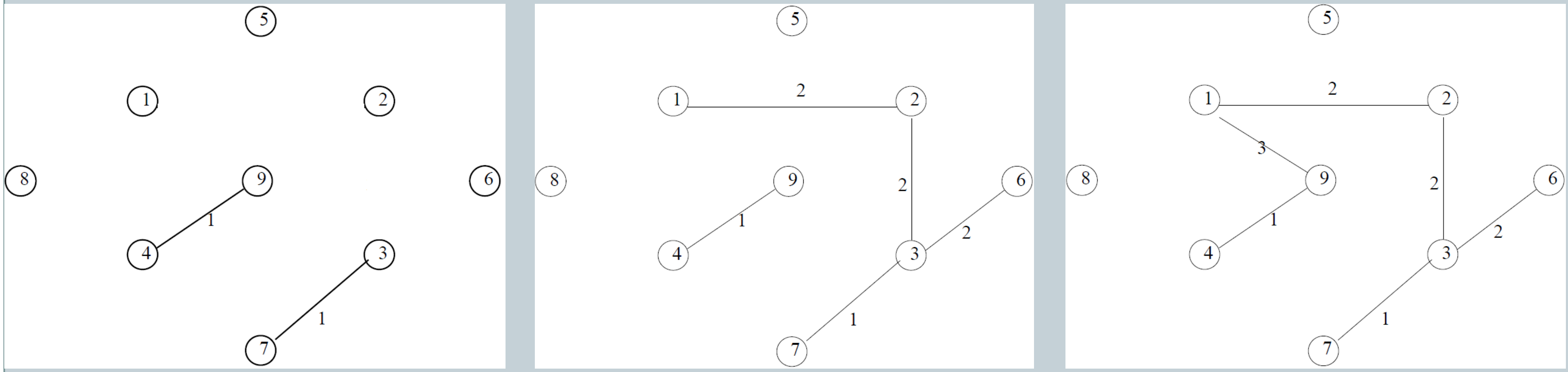
MST, G^1=(V^1, E^1)

Outline

- Terminology and Representations
- Graph Traversals
- Shortest-Paths Problems
- Minimum-Cost Spanning Trees
  - Kruskal's Algorithm
  - Jarnik-Prim's Algorithm
  - Boruvka's Algorithm

Kruskal's Algorithm

- This is a one of the popular algorithm and was developed by Joseph Kruskal
- To create a minimum cost spanning trees, using Kruskalls, we begin by choosing the edge with the minimum cost (if there are several edges with the same minimum cost, select any one of them) and add it to the spanning tree.
- In the next step, select the edge with next lowest cost, and so on, until we have selected $(n - 1)^*$ edges to form the complete spanning tee.



Step 1 *Step 2* *Step 3* *Step ….*

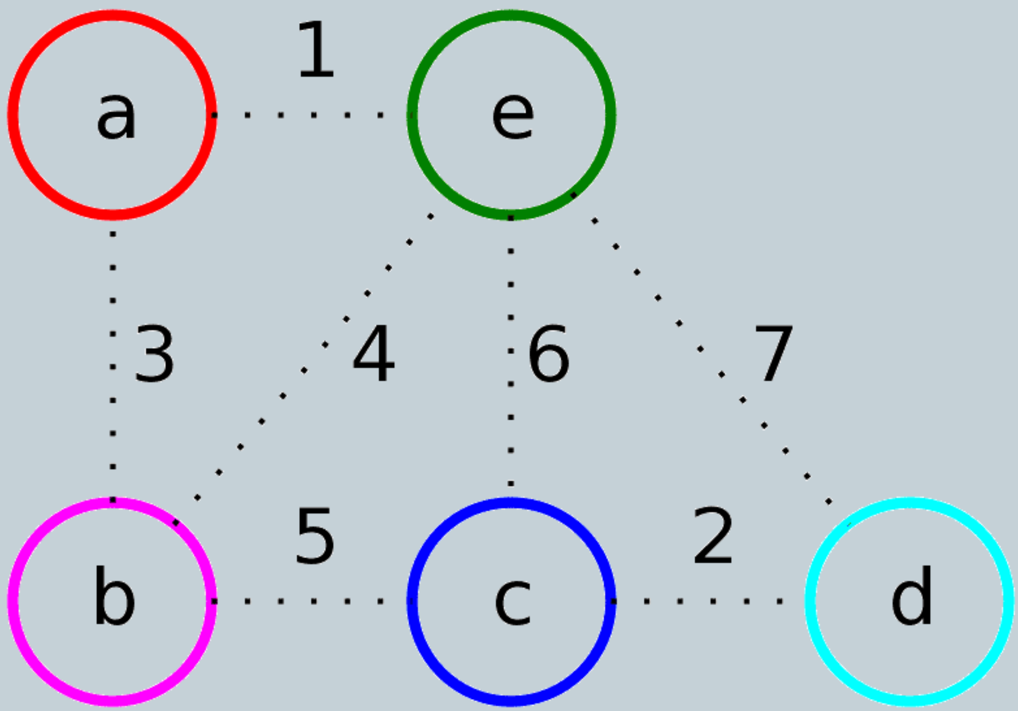
** - In case in graph, there are n edges*

Kruskal's Pseudo-Code

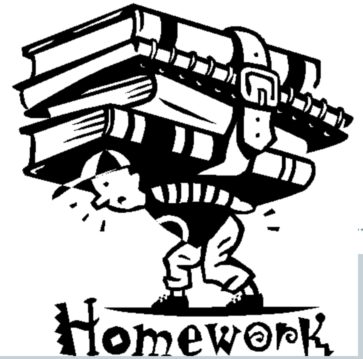
Suppose G = (V, E) is a graph, and *T* is a minimum spanning tree of graph G.

1. Initialize the spanning tree *T* to contain all the vertices in the graph G but no edges.
2. Choose the edge *e* with lowest weight from graph G.
3. Check if both vertices from *e* are within the same set in the tree *T*, for all such sets of *T*. If it is not present, add the edge *e* to the tree *T*, and replace the two sets that this edge connects.
4. Delete the edge e from the graph G and repeat the step 2 and 3 until there is no more edge to add or until the panning tree T contains (n-1) vertices.
5. Exit

Edge	ab	ae	bc	be	cd	ed	ec
Weight	3	1	5	4	2	7	6



More on -- http://weierstrass.is.tokushima-u.ac.jp/ikeda/suuri/kruskal/KruskalApp.shtml?demo1



Write functions of Kruskal (Minimum Spanning Tree) algorithm base on pseudo-code in the slide.

Outline

- Terminology and Representations
- Graph Traversals
- Shortest-Paths Problems
- Minimum-Cost Spanning Trees
  - Kruskal's Algorithm
  - Jarnik-Prim's Algorithm
  - Boruvka's Algorithm

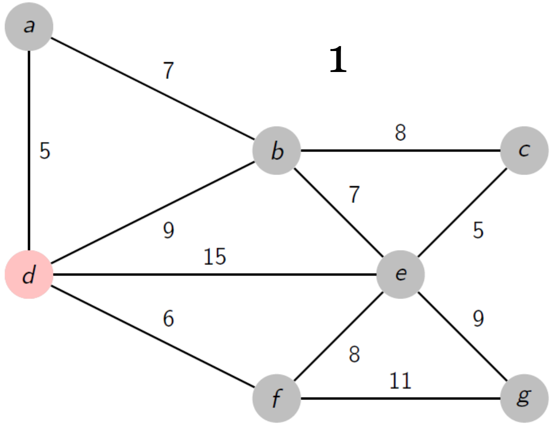
Jarnik-Prim's Algorithm Pseudo-Code

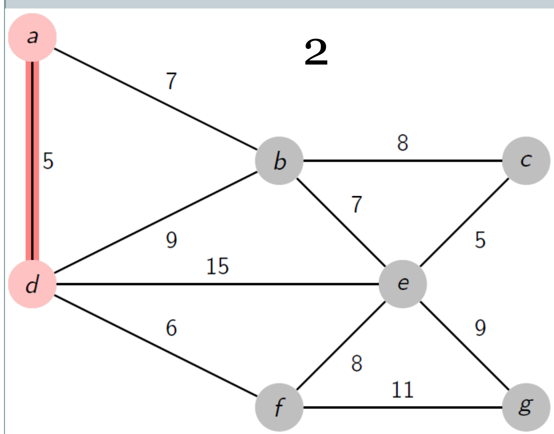
Suppose $G = (V, E)$ is a graph, which V – Set of vertex and E– Set of Edge.

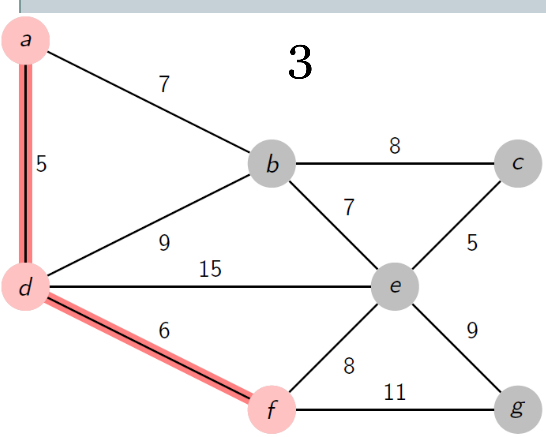
1. Choose a start vertex s in G, $N = \{s\}$ and $A = \{\}$
2. Add all edges connected from s to PQ (Priority Queue) by increased weight
3. Select a smallest edge from PQ which a vertex in N and another on in $V \backslash N$;

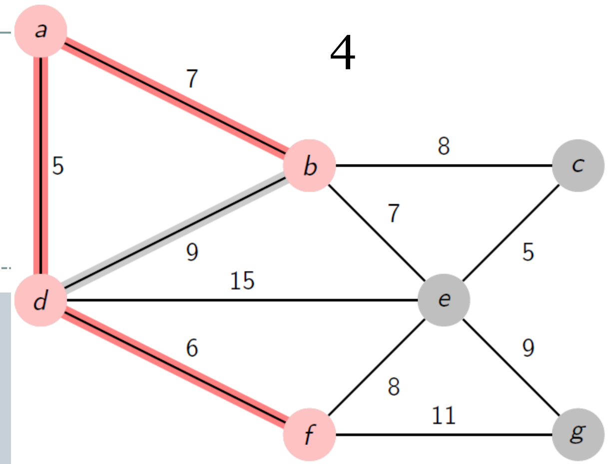
   - Add the selected edge to A and its (other) vertex to N

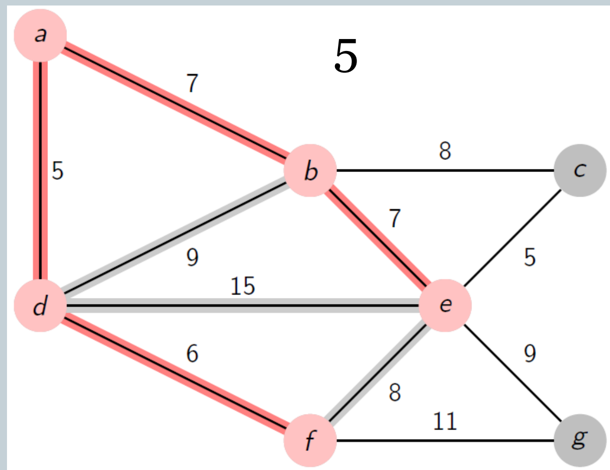
   - Add all edges to PQ which connected from the recently added vertex (to N)
4. If $V \backslash N = o$, then terminate. Otherwise, go to step 3

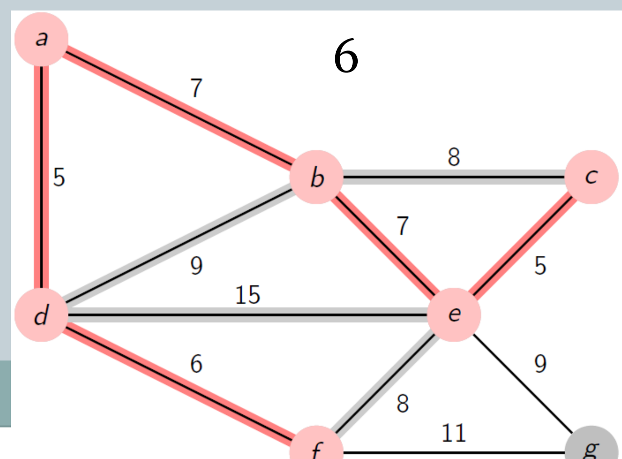


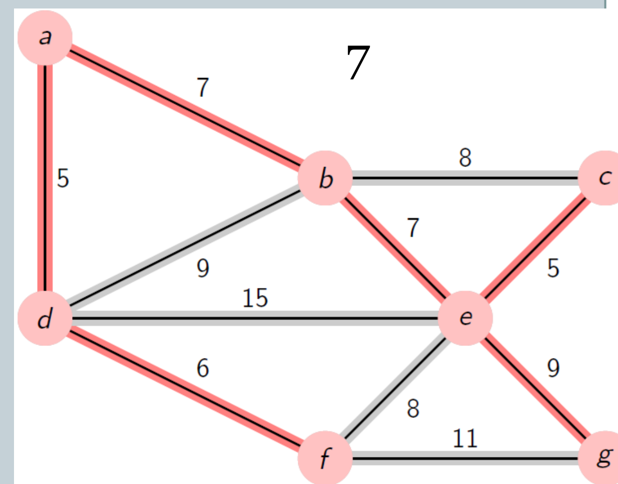




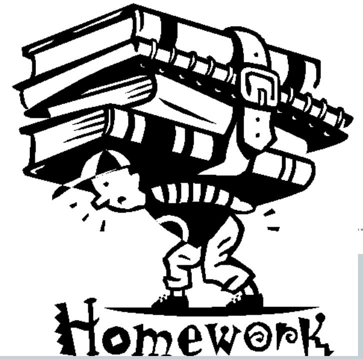








http://www.texample.net



Write functions of Jarnik-Prim's Algorithm (Minimum Spanning Tree) base on pseudo-code in the slide.

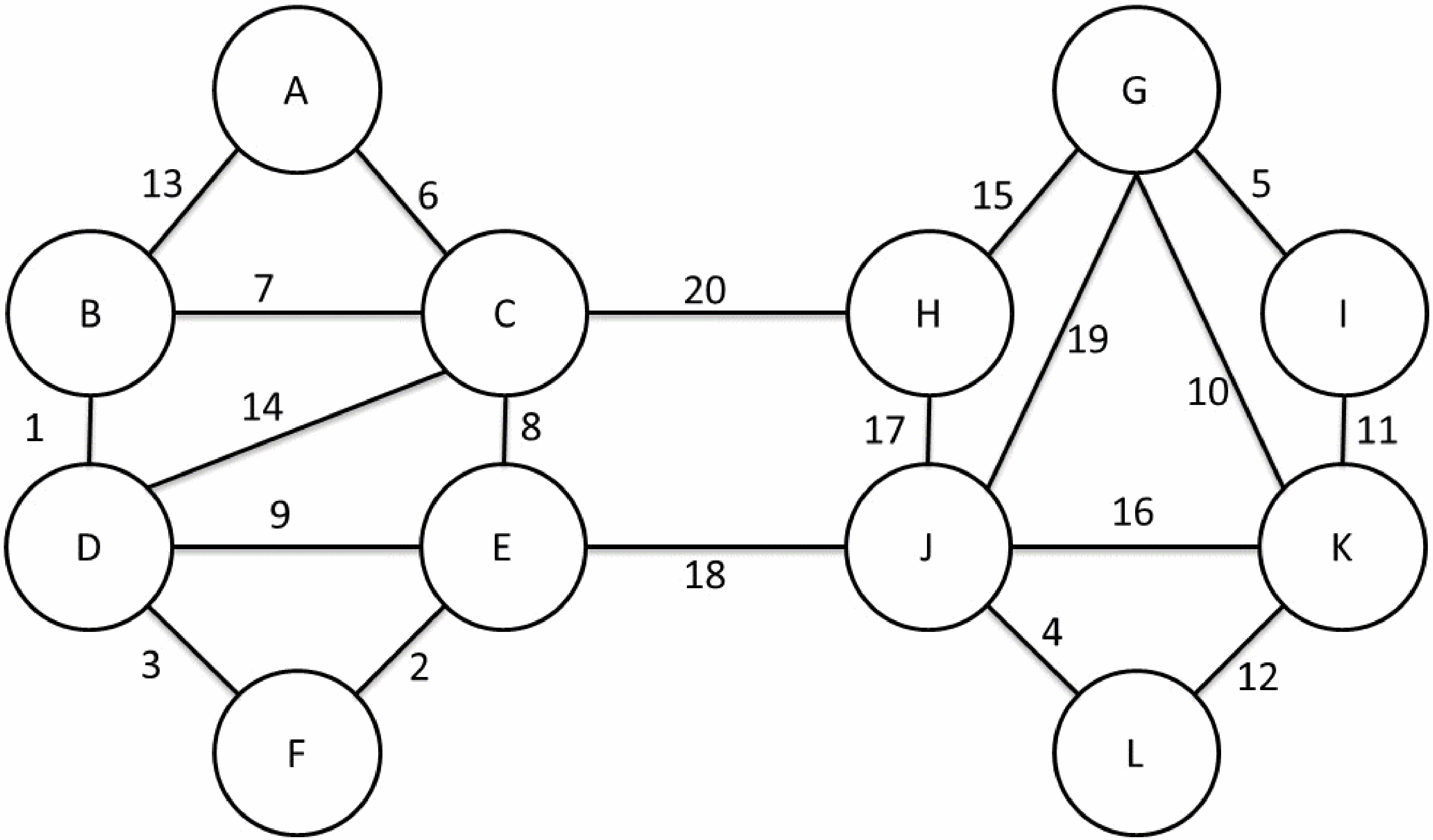
Outline

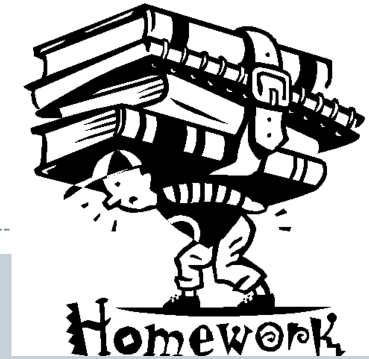
- Terminology and Representations
- Graph Traversals
- Shortest-Paths Problems
- Minimum-Cost Spanning Trees
  - Kruskal's Algorithm
  - Jarnik-Prim's Algorithm
  - Boruvka's Algorithm

Boruvka's Algorithm Pseudo-Code

1. Initialize a forest `T` to be a set of one-vertex trees, one for each vertex of the graph
2. **While** `T` has more than one component:
3.   **For each** component `C` of `T`:
4.   Begin with an empty set of edges `S`
5.   **For each** vertex `v` in `C`:
6.   Find the cheapest edge from `v` to a vertex outside of `C`, and add it to `S`
7.   Add the cheapest edge in `S` to `T`
8.   Combine trees connected by edges to form bigger components
9. Output: `T` is the minimum spanning tree of `G`.

Boruvka's Algorithm Animation





Write functions of Boruvka’s Algorithm (Minimum Spanning Tree) base on pseudo-code in the slide.

To be continued…