Ch11.5 Minimum Spanning Trees (Week 15)

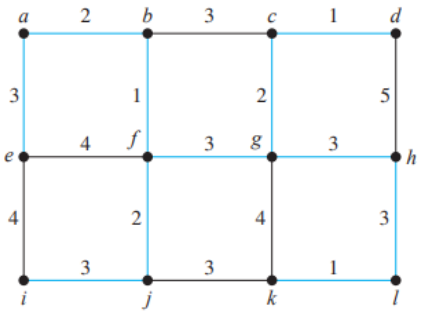
Algorithms for Minimum Spanning Trees (MST)

1 A *minimum spanning tree* in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

Both algorithms below are greedy algorithms (a procedure that makes an optimal choice at each of its steps).

ALGORITHM 1 Prim's Algorithm.

procedure *Prim*(G: weighted connected undirected graph with n vertices)
T := a minimum-weight edge
for $i := 1$ **to** $n - 2$
  e := an edge of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T
  $T := T$ with e added
return T {T is a minimum spanning tree of G}



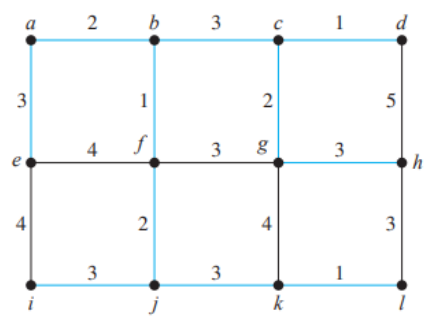
(a)

Choice	Edge	Weight
1	{b, f}	1
2	{a, b}	2
3	{f, j}	2
4	{a, e}	3
5	{i, j}	3
6	{f, g}	3
7	{c, g}	2
8	{c, d}	1
9	{g, h}	3
10	{h, l}	3
11	{k, l}	1
	Total:	24

(b)

ALGORITHM 2 Kruskal's Algorithm.

procedure *Kruskal*(G: weighted connected undirected graph with n vertices)
T := empty graph
for $i := 1$ **to** $n - 1$
  e := any edge in G with smallest weight that does not form a simple circuit when added to T
  $T := T$ with e added
return T {T is a minimum spanning tree of G}



(a)

Choice	Edge	Weight
1	{c, d}	1
2	{k, l}	1
3	{b, f}	1
4	{c, g}	2
5	{a, b}	2
6	{f, j}	2
7	{b, c}	3
8	{j, k}	3
9	{g, h}	3
10	{i, j}	3
11	{a, e}	3
	Total:	24

(b)

Difference Between Prim's and Kruskal's algorithms

Prim's → edges of min weight, incident to vertex in the tree + not formign circuit → chosen

  Edges need to be ordered.

Kruskal's → edges of min weight, *not necessarily* incident to vertex in the tree + not forming circuit → chosen

---

https://www.geeksforgeeks.org/difference-between-prims-and-kruskals-algorithm-for-mst/

Prim's Algorithm	Kruskal's Algorithm
It starts to build the Minimum Spanning Tree from any vertex in the graph.	It starts to build the Minimum Spanning Tree from the vertex carrying minimum weight in the graph.
It traverses one node more than one time to get the minimum distance.	It traverses one node only once.
Prim's algorithm gives connected component as well as it works only on connected graph.	Kruskal's algorithm can generate forest(disconnected components) at any instant as well as it can work on disconnected components
Prim's algorithm runs faster in dense graphs.	Kruskal's algorithm runs faster in sparse graphs.
Prim's algorithm has a time complexity of O(V2), V being the number of vertices and can be improved up to O(E + log V) using Fibonacci heaps.	Kruskal's algorithm's time complexity is O(E log V), V being the number of vertices.

finding MST using Prim's algorithm	finding MST using Kruskal's algorithm
1. Create a set mstSet that keeps track of vertices already included in MST.	1. Sort all the edges in non-decreasing order of their weight.
2. Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.	2. Pick the smallest edge. Check if it forms a cycle with the spanning-tree formed so far. If the cycle is not formed, include this edge. Else, discard it.
3. While mstSet doesn't include all vertices - Pick a vertex u which is not there in mstSet and has minimum key value. - Include u to mstSet. - Update the key value of all adjacent vertices of u. To update the key values, iterate through all adjacent vertices. For every adjacent vertex v, if the weight of edge u-v is less than the previous key value of v, update the key value as the weight of u-v	3. Repeat step#2 until there are (V-1) edges in the spanning tree.