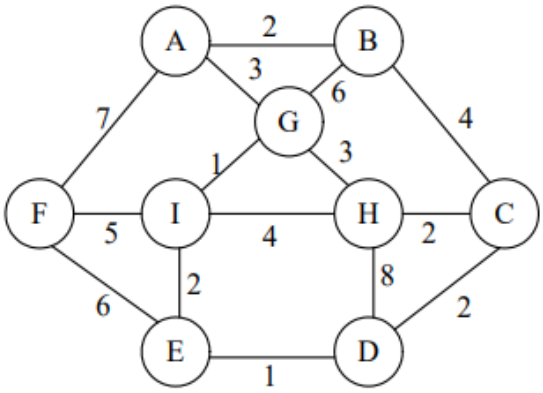
Ashcon Abae

Design & Analysis of Algorithms

1 December 2019

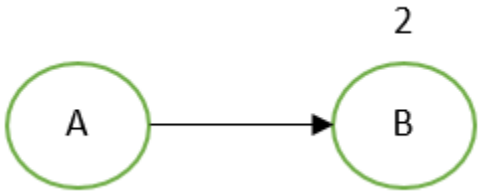
Homework 5

1. Given:

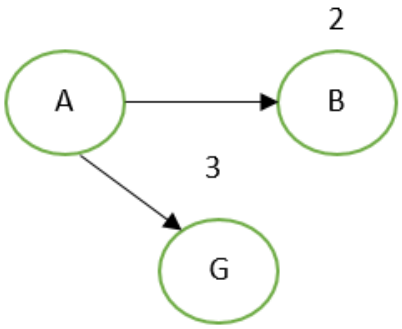


The execution of Prim's minimum spanning tree is shown below:

a. **Step 1**
Starting vertex is A.
There are 3 paths from A: to B with distance of 2, to G with distance of 3, and to F with distance of 7.
For minimum spanning tree, we choose path with minimum distance, thus we go from A to B.

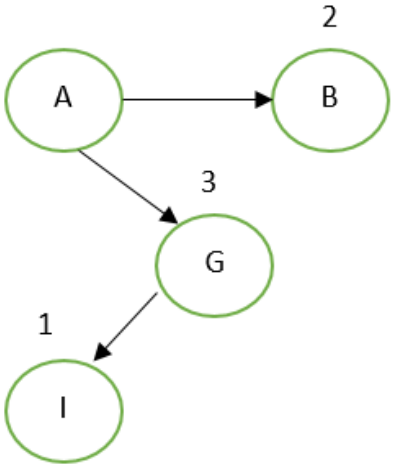


b. **Step 2**
For further paths, we remember which vertices are already being traversed.
From vertex A, the paths are now: to G with distance of 3, to F with distance of 7, and to B to C with distance of 2 + 4 = 6.
For minimum spanning tree, we choose path with minimum distance, thus we go from A to G.



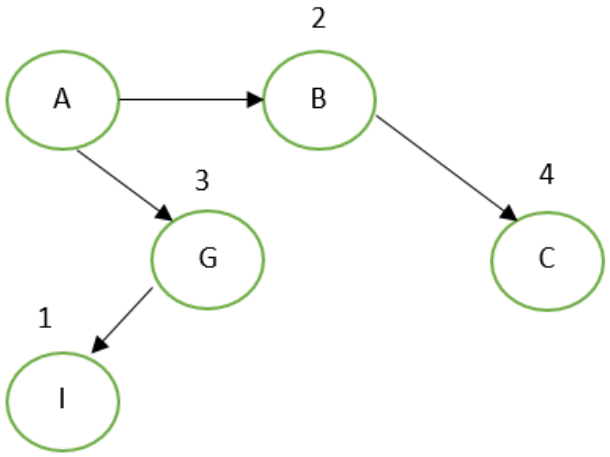
c. **Step 3**

From vertex A, the paths are now: to F with distance of 7, to G to H with distance of 3 + 3 = 6, to G to I with distance of 3 + 1 = 4, and to B to C with distance of 6.
Since vertex B is already being traversed, the path from A to G to B is not being considered for minimum spanning tree.
For minimum spanning tree, we choose path with minimum distance, thus we go from A to G to I.



d. **Step 4**

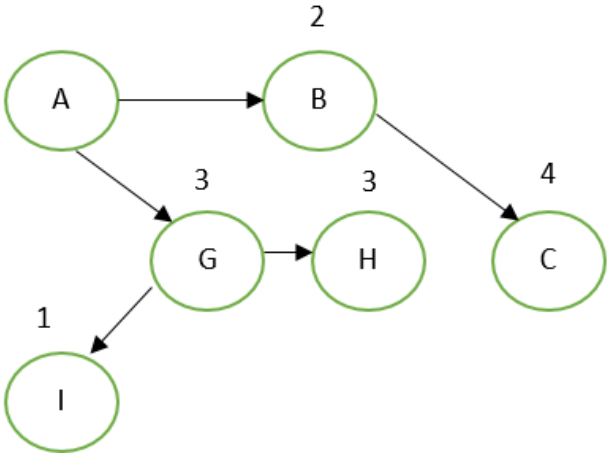
From vertex A, the paths are now: to F with distance of 7, to G to H with distance of 6, to G to I to F with distance of 3 + 1 + 5 = 9, to G to I to E with distance of 3 + 1 + 2 = 6, to G to I to H with distance of 3 + 1 + 4 = 8, and to B to C with distance of 2 + 4 = 6.
For minimum spanning tree, we choose path with minimum distance. Since the minimum distance is 6 and there are multiple paths with that distance, we choose any one of them. We choose to use path A to B to C.



e. **Step 5**

From vertex A, the paths are now: to F with distance of 7, to G to H with distance of 6, to G to I to F with distance of 9, to G to I to E with distance of 6, and to G to I to H with distance of 6.

Choosing any of the 2 minimum distance paths, we will chose A to G to H.

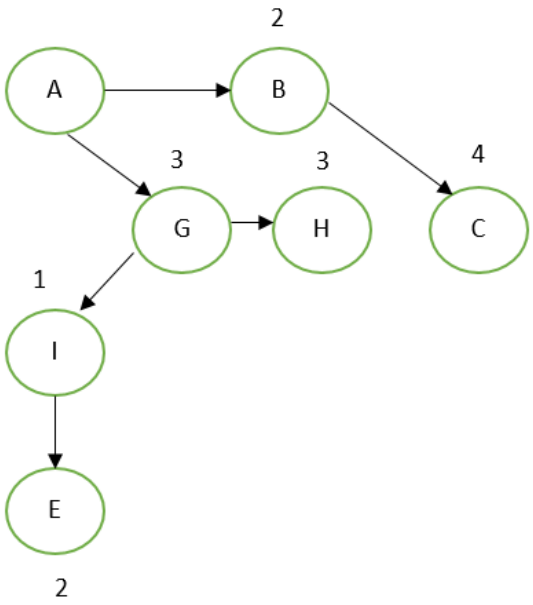


f. **Step 6**

From vertex A, the paths are now: to F with distance of 7, to G to H to D with distance of 14, to G to I to F with distance of 9, and to G to I to E with distance of 6.

Since H is already in our minimum spanning tree, the paths from A to G to I to H and from A to B to C to H are not being considered.

Our minimum distance path is A to G to I to E.

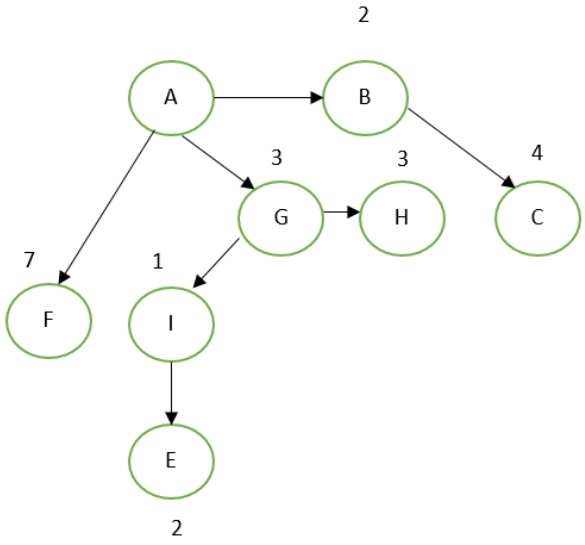


g. **Step 7**

From vertex A, the paths are now: to F with distance of 7, to G to H to D with distance of 14, to G to I to F with distance of 9, to G to I to E to F with distance of 12, to G to I to E to D with distance of 7, and to B to C to D with distance of 8.

We choose any of the 2 minimum paths.

Our path is A to F with distance of 7.

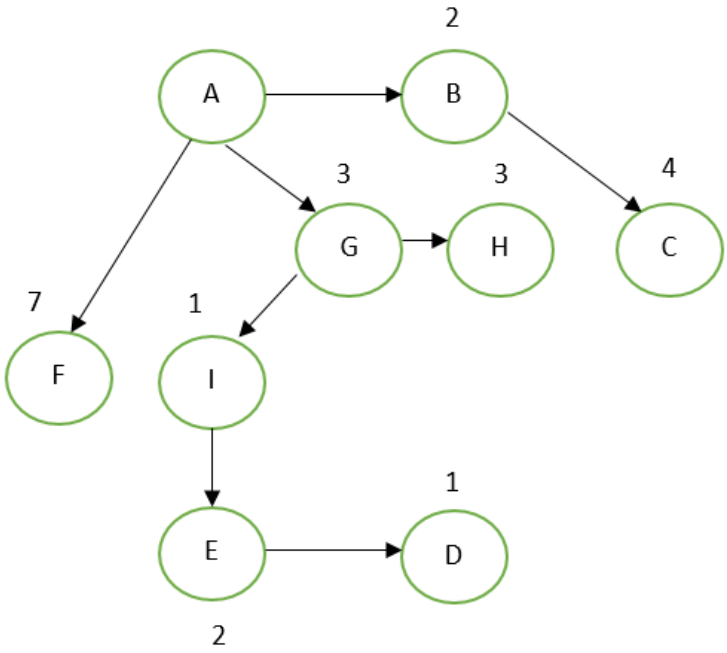


h. **Step 8**

From vertex A, the paths are now: to G to H to D with distance of 14, to G to I to E to D with distance of 7, and to B to C to D with distance of 8.

We choose minimum distance path.

Our path is A to G to I to E to D with distance of 7.



i. **Step 9**
Since all vertices are traversed, we are done with Prim's minimum spanning tree algorithm.

2. To execute Kruskal's algorithm, first we sort the edges in a priority queue in ascending order. Then, of this priority queue we retain those edges that don't form a cycle and those that have a minimum weight. We will mark all edges that don't meet these requirements (edges are listed on left, weights in middle, and if edge doesn't meet requirements on right):

Priority Queue:

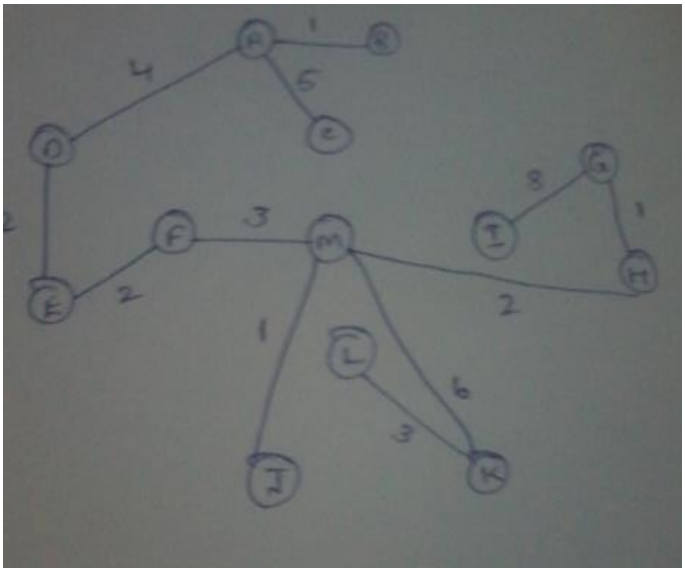
AB	1	
GH	1	
JM	1	
DE	2	
EF	2	
HM	2	
DF	3	(CYCLE FORMED)
FM	3	
KL	3	
AD	4	
EM	4	(CYCLE FORMED)
AC	5	
GM	5	(CYCLE FORMED)
BC	6	(CYCLE FORMED)
KM	6	
BG	7	(CYCLE FORMED)

CM	7	(CYCLE FORMED)
JL	7	(CYCLE FORMED)
JK	7	(CYCLE FORMED)
AM	8	(CYCLE FORMED)
GI	8	
HI	8	(CYCLE FORMED)
HK	8	(CYCLE FORMED)
BM	9	(CYCLE FORMED)
DM	9	(CYCLE FORMED)
EJ	9	(CYCLE FORMED)
IM	9	(CYCLE FORMED)
LM	9	(CYCLE FORMED)

From our priority queue above, we remove all edges that we marked with a (CYCLE FORMED), and we will get our minimum spanning tree:

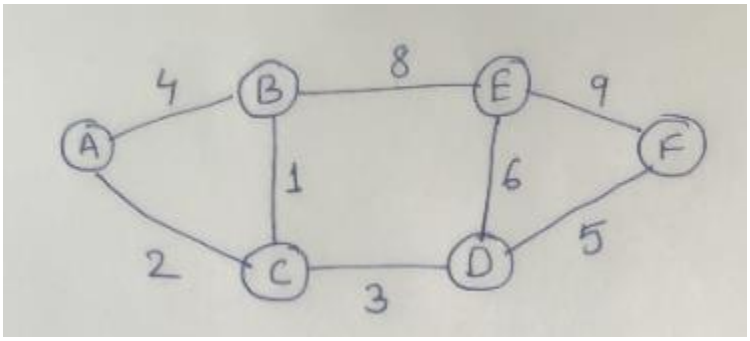
Priority Queue:

AB	1
GH	1
JM	1
DE	2
EF	2
HM	2
FM	3
KL	3
AD	4
AC	5
KM	6
GI	8

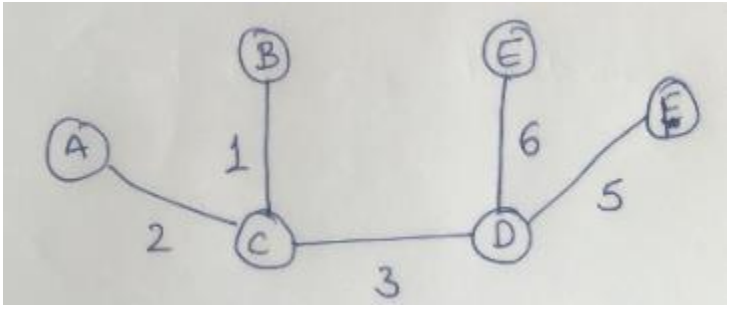


3.

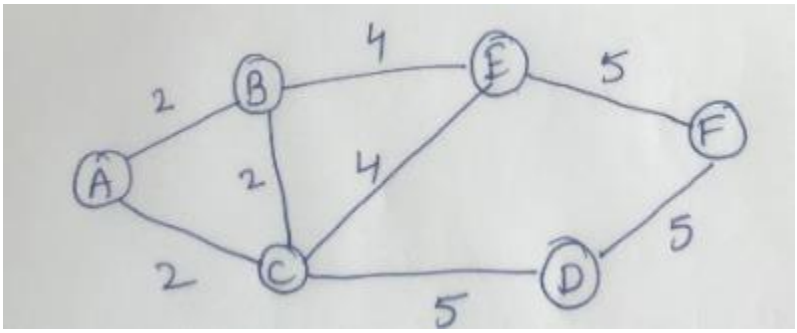
a. To get a unique minimum spanning tree from a weighted graph, there must exist unique edge weights in the weighted graph. Consider the weighted graph below:



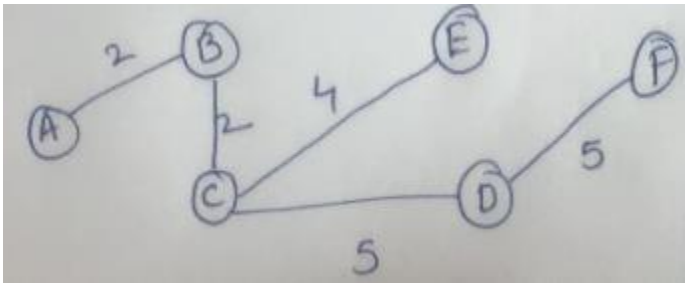
The unique minimum spanning tree of the above graph is:



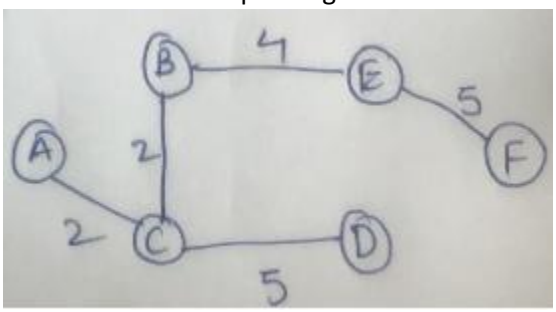
b. Below is an example of a weighted graph that has multiple minimum spanning trees:



One minimum spanning tree of the above weighted graph is:

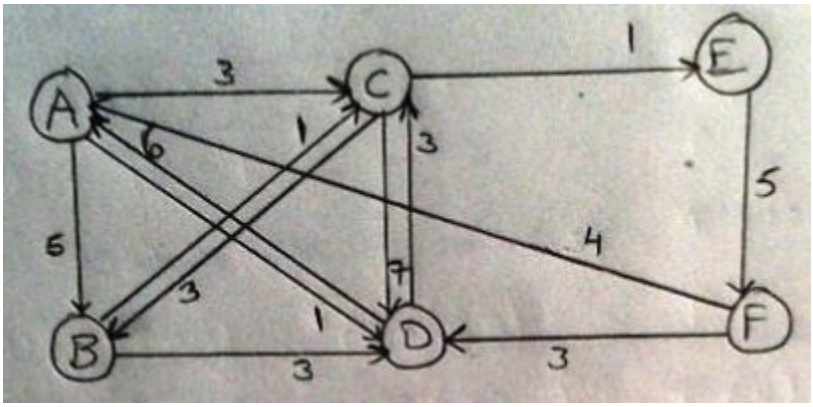


Another minimum spanning tree of the above weighted graph is:



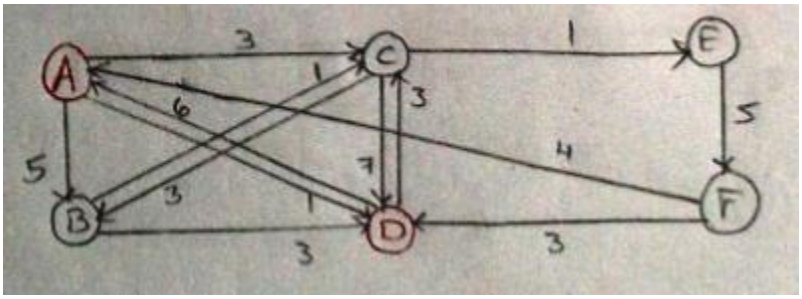
A weighted graph can have several minimum spanning trees. For example, if all the edge weights of a graph are the same, then every spanning tree of said weighted graph is minimum. The only way for a weighted graph to have a unique minimum spanning tree is for all the edge weights of the graph to be unique. If a node can be reached from various nodes with equal weight, then we get different minimum spanning trees.

4. First, let's generate our weighted graph based on the given adjacency lists with edge weights:

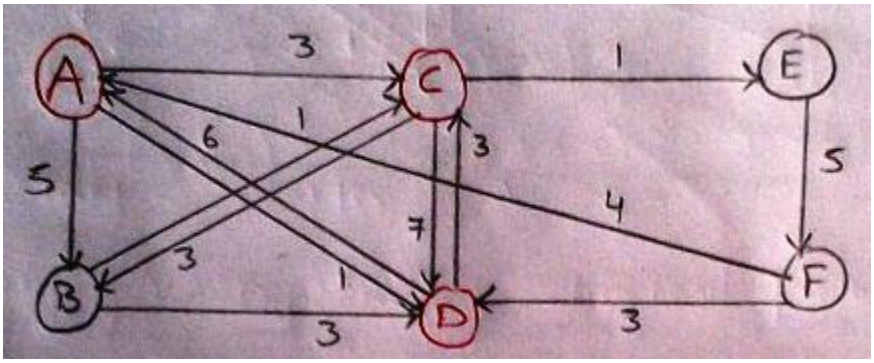


Now, we can start applying Dijkstra's Shortest Path Algorithm. Visited Vertices will be marked in the color red:

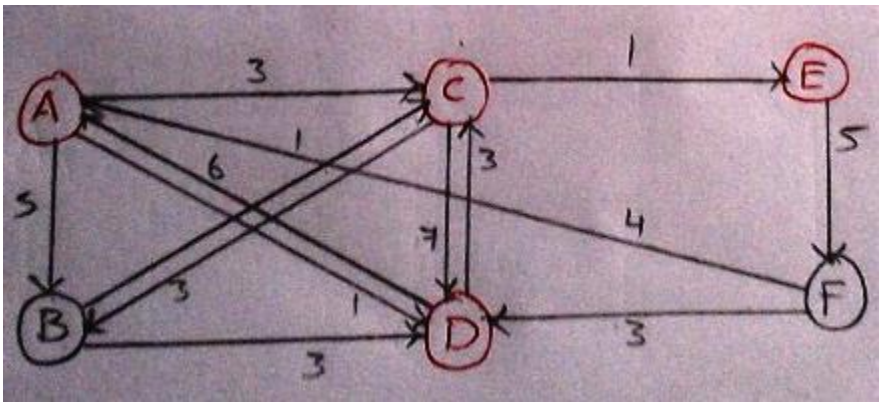
Step 1:
Unvisited Vertex List = {A, B, C, D, E, F}
We start with and visit vertex A. Find adjacent vertices of A: B(5), C(3), D(1)
The adjacent vertex with the shortest distance from A is D. Visit vertex D.



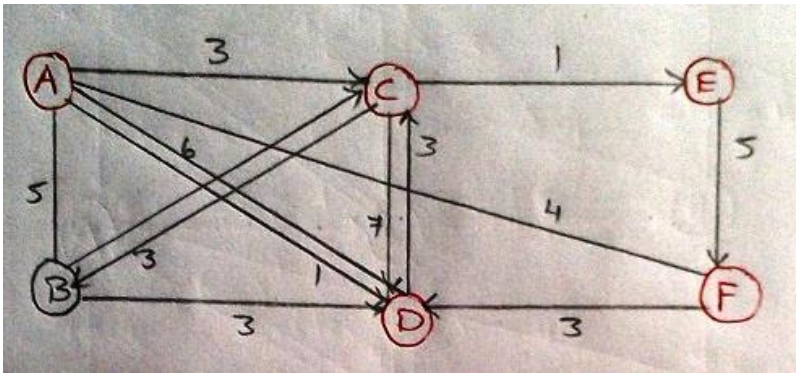
Step 2:
Unvisited Vertex List = {B, C, E, F}
Find adjacent vertices of D: A(6), C(3).
We choose C because A is already visited. Visit vertex C.



Step 3:
Unvisited Vertex List = {B, E, F}
Find adjacent vertices of C: B(3), D(7), E(1).
We choose vertex E because it is shortest distance. Visit vertex E.



Step 4:
Unvisited Vertex List = {B, F}
Find adjacent vertices of E: F(5).
We choose vertex F and visit.



Step 5:

Unvisited Vertex List = {B}

Although we have not visited vertex B, we have gone from vertex A to vertex F in the shortest possible path (A -> D -> C -> E -> F). Thus, our shortest path is given as: