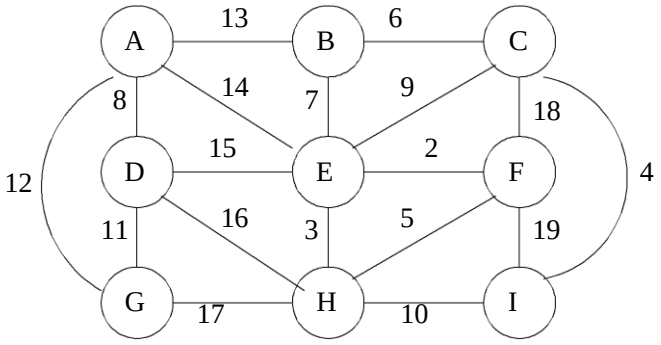
Problem Set 7

Assigned: July 8
Due: July 15
Name: Subhankari Mishra

Problem 1

A. Trace the execution of Prim's algorithm in finding the minimum spanning tree for the graph below. Show the sequence in which edges are added to the tree, and the successive values of the array D[v].

B. Trace the execution of Kruskal's algorithm in finding the minimum spanning tree for the graph below. Show the sequence in which edges are added to the tree.



Ans:

A. T = Ø;
S = {A}
D =

A	B	C	D	E	F	G	H	I
0	13	∞	8	14	∞	12	∞	∞

P =

A	B	C	D	E	F	G	H	I
-	A	-	A	A	-	A	-	-

W = D
S = {A, D}
T = {A-D}
D =

A	B	C	D	E	F	G	H	I
0	13	∞	8	14	∞	11	16	∞

P =

A	B	C	D	E	F	G	H	I
-	A	-	A	A	-	D	D	-

W = G
S = {A, D, G}
T = {A-D, D-G}
D =

A	B	C	D	E	F	G	H	I
0	13	∞	8	14	∞	11	16	∞

P =

A	B	C	D	E	F	G	H	I
-	A	-	A	A	-	D	D	-

W = B
S = {A, D, G, B}
T = {A-D, D-G, A-B}
D =

A	B	C	D	E	F	G	H	I
0	13	6	8	7	∞	11	16	∞

P =

A	B	C	D	E	F	G	H	I
-	A	B	A	B	-	D	D	-

W = C
S = {A, D, G, B, C}
T = {A-D, D-G, A-B, B-C}
D =

A	B	C	D	E	F	G	H	I
0	13	6	8	7	18	11	16	4

P =

A	B	C	D	E	F	G	H	I
-	A	B	A	B	C	D	D	C

W = I
S = {A, D, G, B, C, I}
T = {A-D, D-G, A-B, B-C, C-I}
D =

A	B	C	D	E	F	G	H	I
0	13	6	8	7	18	11	10	4

P =

A	B	C	D	E	F	G	H	I
-	A	B	A	B	C	D	I	C

W = E
S = {A, D, G, B, C, I, E}
T = {A-D, D-G, A-B, B-C, C-I, B-E}
D =

A	B	C	D	E	F	G	H	I
0	13	6	8	7	2	11	3	4

P =

A	B	C	D	E	F	G	H	I
-	A	B	A	B	E	D	E	C

W = F
S = {A, D, G, B, C, I, E, F}
T = {A-D, D-G, A-B, B-C, C-I, B-E, E-F}
D =

A	B	C	D	E	F	G	H	I

0	13	6	8	7	2	11	3	4

P =

A	B	C	D	E	F	G	H	I
-	A	B	A	B	E	D	E	C

W = H
S = {A, D, G, B, C, I, E, F, H}
T = {A-D, D-G, A-B, B-C, C-I, B-E, E-F, E-H}
D =

A	B	C	D	E	F	G	H	I
0	13	6	8	7	2	11	3	4

P =

A	B	C	D	E	F	G	H	I
-	A	B	A	B	E	D	E	C

B. Kruskal's algorithm:

E = {E-F, E-H, C-I, F-H, B-C, B-E, A-D, C-E, H-I, D-G, A-G, A-B, A-E, D-E, D-H, G-H, C-F, F-I}
T = Ø
The FOR loop on E:
E-F <= not connected, add to T
T= {E-F}
E-H <= not connected, add to T
T = {E-F, E-H}
C-I <= not connected, add to T
T = {E-F, E-H, C-I}
F-H <= already connected
B-C <= not connected, add to T
T = {E-F, E-H, C-I, B-C}
B-E <= not connected, add to T
T = {E-F, E-H, C-I, B-C, B-E}
A-D <= not connected, add to T
T = {E-F, E-H, C-I, B-C, B-E, A-D}
C-E <= already connected
H-I <= already connected
D-G <= not connected, add to T
T = {E-F, E-H, C-I, B-C, B-E, A-D, D-G}
A-G <= already connected
A-B <= not connected, add to T
T = {E-F, E-H, C-I, B-C, B-E, A-D, D-G, A-B}
|T| = 8 = (N-1)
Return T;

Problem 2

(Siegel, Ex. 8.24) Let G be an undirected weighted graph and let F be a subgraph of G that is a forest (a collection of separate trees). Design an efficient algorithm to find a spanning tree of G that contains F and has the minimal total cost over all spanning trees containing F.

Ans: Let E be the set of edges in G, assuming all edges in G which are not part of F have distinct weight/cost below would be algorithm to find the minimal total cost spanning tree.

```
MST (G) {
        For (each U-V in E)
          If ((U not in F) || (V not in F)) {
                Add U-V to E';
          Else
                Add U-V to E'';
          End if.
        End for.
        Let G' be the graph formed by the nodes and edges in E'.
        T = Kruskal(G');

        Append E'' to T;
        Return T;
}
```

Problem 3

Let G be an undirected graph and let X be a subset of the vertices of G. A connecting tree on X is a tree composed out of the edges of G that contains all the vertices in X. One way to compute a connecting tree consists of two steps: (1) Compute a minimum spanning tree T over G. (2) Delete all the edges out of T not needed to connect vertices in X.

A. Give an algorithm to carry out step 2 above in time θ (N) where N is the number of vertices in G.

Ans:

```
Let all nodes have an initial color white.
DeleteEdge (T) {
        If ((no leaf node with white color) or (T == Ø))
                Return.
        End if.
        For (each leaf node (U) of T with (U.color == white))
          If (U ε X)
                U.color = grey.
          Else
                Delete the edge from U to parent V.
          End if.
        Endfor.
        DeleteEdge(T);   //where this T is the Tree after the deleted edges

}
```

B. The Steiner tree for X is the minimum cost connecting tree. Give an example to show that the above algorithm does not always return the Steiner tree.

Ans: Considering the graph in Problem 1, as per the algorithm mentioned above, below would be the calculated minimum spanning tree:

T = {E-F, E-H, C-I, B-C, B-E, A-D, D-G, A-B}

Let X = {A, E}

As per the above algorithm upon deleting the edges out of T not needed to connect vertices we have,

T = {A-B, B-E}

TotalCost of T = 13 + 7 = 20.

Whereas we had a direct edge from A-E with cost 14 which is lesser than the current total cost i.e. 20.