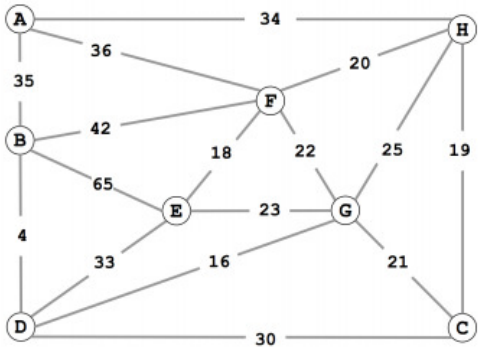
COS 226 – Data Structures and Algorithms
Fall 2014 – Flipped Lecture Section
Individual/small group worksheet
Week 8 – 11.04.14
Topics covered: digraphs, MST
Solution

Instructions: This worksheet covers directed graphs (digraphs) and Minimum spanning trees (MST). Read the worksheet first (before viewing the videos) and understand what type of questions needs to be answered. As you watch videos, if you find the answer to a problem, write the answer here and if possible in salon, so you can share it with others. Also be sure to make some comments/questions on salon.

1. MST Problem

Consider the graph given below.



(a) Complete the list of edges in the MST in the order that Kruskal's algorithm includes them. For reference, the edge weights in ascending order are:

4 16 18 19 20 21 22 23 25 30 33 34 35 36 42 65

B-D ____ ____ ____ ____ ____ ____

Solution: The idea of the kruskal's algorithm is that all edges are maintained in a minPQ (cost to build – E). Then starting with the minimum edge, add that to the MST if the new edge does not create a cycle (that is, both end points of the edge is already in the MST). We can test the connectivity of two vertices in log time using a weighted UF. The process is as follow. The red indicates that no change to MST or cut since that edge would create a cycle (if added)

Edge/cost	MST	vertices in the cut (shown are connected components)
(B,D) /4	{(B,D)}	{B,D}
(D,G) /16	{(B,D),(D,G)}	{B,D,G}
(E,F) /18	{(B,D),(D,G),(E,F)}	{B,D,G} {E,F}
(H,C) /19	{(B,D),(D,G),(E,F),(H,C)}	{B,D,G} {E,F},{H,C}
(F,H) /20	{(B,D),(D,G),(E,F),(H,C),(F,H)}	{B,D,G} {E,F,H,C}
(G,C) /21	{(B,D),(D,G),(E,F),(H,C),(F,H),(G,C)}	{B,D,G,E,F,H,C}
(F,G) /22	{(B,D),(D,G),(E,F),(H,C),(F,H),(G,C)}	{B,D,G,E,F,H,C}
(E,G) /23	{(B,D),(D,G),(E,F),(H,C),(F,H),(G,C)}	{B,D,G,E,F,H,C}
(G,H) /25	{(B,D),(D,G),(E,F),(H,C),(F,H),(G,C)}	{B,D,G,E,F,H,C}
(D,C) /30	{(B,D),(D,G),(E,F),(H,C),(F,H),(G,C)}	{B,D,G,E,F,H,C}
(D,E) /33	{(B,D),(D,G),(E,F),(H,C),(F,H),(G,C)}	{B,D,G,E,F,H,C}
(A,H) /34	{(B,D),(D,G),(E,F),(H,C),(F,H),(G,C),(A,H)}	{B,D,G,E,F,H,C,A}

MST is now complete as the MST contains |V|-1 = 8-1 = 7 edges

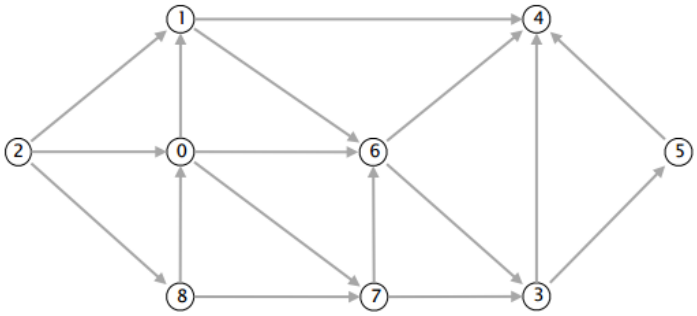
(b) Complete the list of edges in the MST in the order that Prim's algorithm includes them. Start Prim's algorithm from vertex A.

A-H _____ _____ _____ _____ _____ _____

The idea of the Prim's algorithm is that, we start with a vertex and start expanding MST, by greedily selecting the next min edge. It uses a minPQ, but unlike in Kruskals (where all edges are built into a PQ), Prim's algorithm only keeps the edges that are relevant up to that point. Given below is the process. As n Kruskals, it checks to see adding a new edge will create a cycle. Shown below is the vertex that is under process and the current minPQ (not shown as sorted. Always pick the cheapest node). The minPQ is marked with min edge (in red) and placed in MST

vertex	**minPQ (edge/cost)**	**MST**
A	{(A,H)/34, (A,F)/36, (A,B)/35}	{ }
H	{ (A,F)/36, (A,B)/35, (H,F)/20,(H,G)/25,(H,C)/19}	{ (A,H),(H,C)}
C	{ (A,F)/36, (A,B)/35,(H,G)/25,(C,G)/21, (C,D)/30,(H,F)/20}	{ (A,H),(H,C),(H,F)}
F	{ (A,F)/36, (A,B)/35,(H,G)/25,(C,G)/21, (C,D)/30,(F,G)/22,(F,E)/18,(F,B)/42}	{ (A,H),(H,C),(H,F),(F,E)}
F	{ (A,F)/36, (A,B)/35,(H,G)/25,(C,G)/21, (C,D)/30,(F,G)/22,(F,B)/42,(E,B)/65,(E,D)/33,(E,G)/23}	
	MST = {(A,H),(H,C),(H,F),(F,E),(C,G)}	
G	{ (A,F)/36, (A,B)/35,(H,G)/25,(C,D)/30,(F,G)/22,(F,B)/42,(E,B)/65,(E,D)/33,(E,G)/23,(G,C)/21,(G,D)/16}	
	MST = {(A,H),(H,C),(H,F),(F,E),(C,G),(G,D)}	
D	{ (A,F)/36, (A,B)/35,(H,G)/25,(C,D)/30,(F,G)/22,(F,B)/42,(E,B)/65,(E,D)/33,(E,G)/23,(G,C)/21,(D,C)/30,(D,E)/33,(D,B)/4}	
	MST = {(A,H),(H,C),(H,F),(F,E),(C,G),(G,D),(D,B)}	

2. Topological Sort [fin-f11]



(a) Compute the topological order by running the DFS-based algorithm and listing the vertices in reverse postorder.

Run DFS on the graph and print nodes in post order. Shown is the stack and output.

Stack	Output
2	{}
2 0	{}
2 0 6	{}
2 0 6 3	{}
2 0 6 3 5	{}
2 0 6 3 5 4	{}
2 0 6 3 5	{4}
2 0 6 3	{4,5}
2 0 6	{4,5,3}
2 0	{4,5,3,6}
2 0 7	{4,5,3,6}
2 0	{4,5,3,6,7}
2 0 1	{4,5,3,6,7}
2 0	{4,5,3,6,7,1}
2	{4,5,3,6,7,1,0}
2 8	{4,5,3,6,7,1,0}
2	{4,5,3,6,7,1,0,8}
{}	{4,5,3,6,7,1,0,8,2}

post order: 4 5 3 6 7 1 0 8 2 Reverse postorder: 2 8 0 1 7 6 3 5 4 ➔ a topological sort

(b) Run breadth-first search on the digraph, starting from vertex 2. List the vertices in the order in which they are dequeued from the FIFO queue.

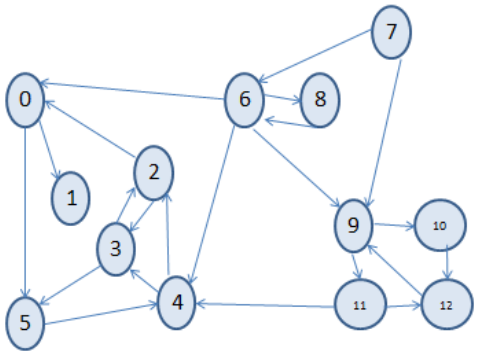
2 ___ ___ ___ ___ ___ ___ ___ ___ ___

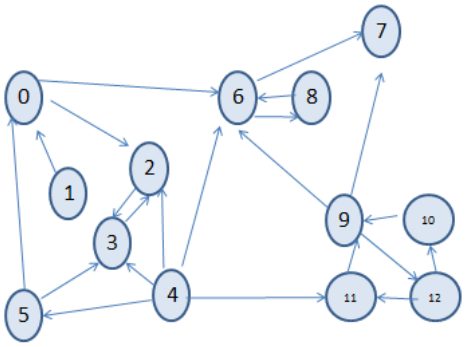
in BFS, we use a queue to run the algorithm. Shown below is queue and BFS order. Start by enque(2)

enque	Queue	deque	output
2	2	_	{}
0 1 8	2 0 1 8	2	{2}
6 7	0 1 8 6 7	0	{2,0}
4	1 8 6 7 4	1	{2,0,1}
_	8 6 7 4	8	{2,0,1,8}
_	6 7 4	6	{2,0,1,8,6}
3	7 4 3	7	{2,0,1,8,6,7}
_	4 3	4	{2,0,1,8,6,7,4}
5	3 5	3	{2,0,1,8,6,7,4,3}
_	5	5	{2,0,1,8,6,7,4,3,5} ➔ this is the BFS order

3. Strong Components(SCs)

Find all strong components (SCC's) in this digraph





Apply Kosaraju-Sharir algorithm to find a strong components.

Step 1 – Find the reverse post order of G^r (the reverse graph of C). The reverse graph of G is shown in right
start with 0 and find the post order

3 2 7 6 0 1 5 10 12 9 11 4

Find the reverse post order of G^r (the reverse graph of C)➔ 4 11 9 12 10 5 1 0 6 7 2 3

Step 2: Do DFS on G in the order found above and mark SC's with integers starting with 0

0 ➔ {4, 2, 0, 5, 3}
1 ➔ {11,12,9,10}
2 ➔ {1}
3 ➔ {6,8}
4 ➔ {7}