Sample Final Exam

CSE 680: Introduction to Algorithms and Data Structures

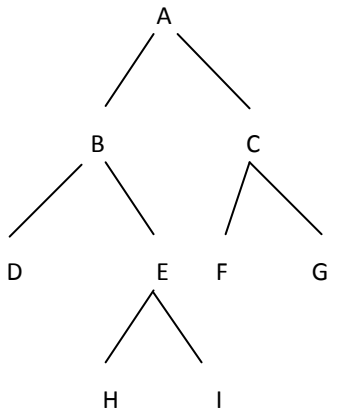
1. Solve the following recurrences by giving tight Θ-notation bounds using the Master Method. If I thought you would need the actual value of a logarithm, I have given it to you.

   (a) T (n)=9T (n/3)+n
   see book

   (b) T (n)=T (2n/3)+1
   see book

   (c) T (n)=3T (n/4)+nlgn
   see book

2. Give the Inorder, Postorder and Preorder traversals of the following binary tree.



Note: Traversal orders only make sense with Depth-first traversal.

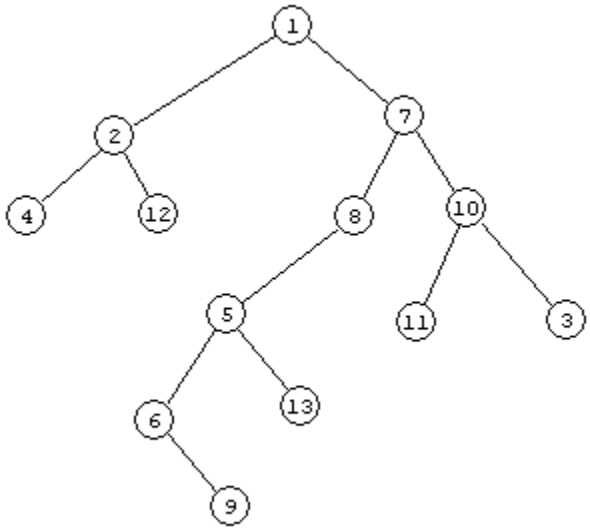
Inorder: DBHEIAFCG

Preorder: ABDEHICFG

Postorder: DHIEBFGCA

3. The picture below represents a binary search tree. The numbers shown are arbitrary node labels, not numbers representing the contents of the nodes. **The contents are not shown**. If node 1 is deleted, using binary search tree deletion, what will be the new root node?

The successor to node 1, which is node 6. Node 9 moves to where node 6 is.



4. Answer True or False. Justify your answer. Each answer is for 3 points.

a. The topological sort of an arbitrary directed graph $G(V;E)$ can be computed in linear time.

True. We can use Depth First Traversal to compute the finish times and then return the nodes in order of decreasing finishing times. We can also easily check for cycles as we do this and report no sort is possible if a cycle exists.

b. Kruskal's algorithm for minimum weight spanning trees is an example of a divide and conquer programming algorithm.

False. Kruskal's algorithm is a greedy algorithm. Divide and conquer would imply splitting the problem into fractional parts. Kruskal's reduces its problem size by one each time.

c. The shortest path between two vertices is unique if all edge weights are distinct.

False. The sum of several weights may be the same. For example (3+2+10 = 7+8 = 15).

d. An arbitrary graph with $G(V;E)$, with $|E| = |V|$ edges is a tree.

False. A tree has only $|V|-1$ edges.

e. A directed graph is strongly connected if and only if a DFS started from any vertex will visit every vertex in the graph without needing to be restarted.

True. A DFS will provide the nodes reachable from a vertex *u*. Since the statement indicates it can be any vertex, then this will hold for all vertices, which means every vertex is reachable from every other vertex.

5. Given an adjacency **matrix** representation of a graph how long does it take to compute the out-degree of all vertices? How long does it take to compute in-degree of all vertices? How long does it take to compute in-degree and out-degree of a single vertex?

outdegree: Each vertex needs to scan a row, so O(n) for each vertex and $O(n^2)$ overall.

indegree: Same, but need to scan a column, $O(n^2)$.

In and out or a single vertex: Look at the row and the column, so 2n or O(n).

6. Consider the graph in Figure 2. Unless otherwise indicated, always visit adjacent nodes in alphabetical order.

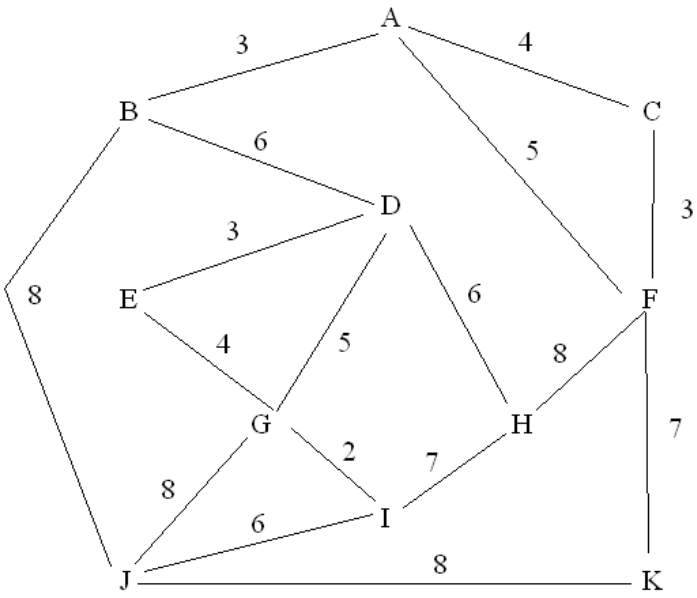
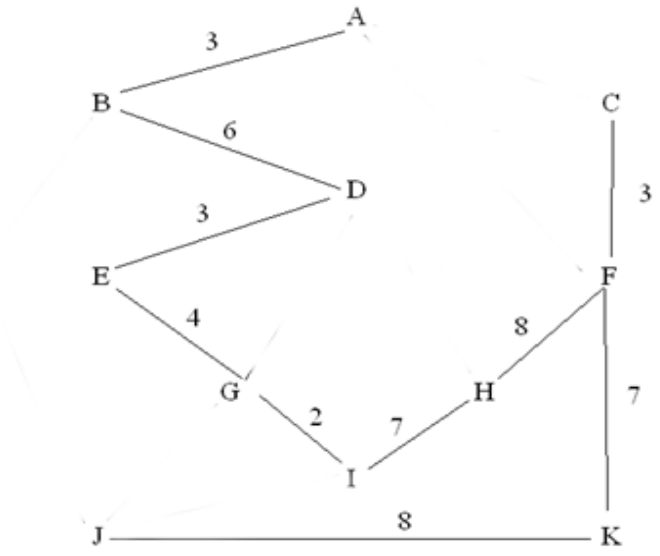
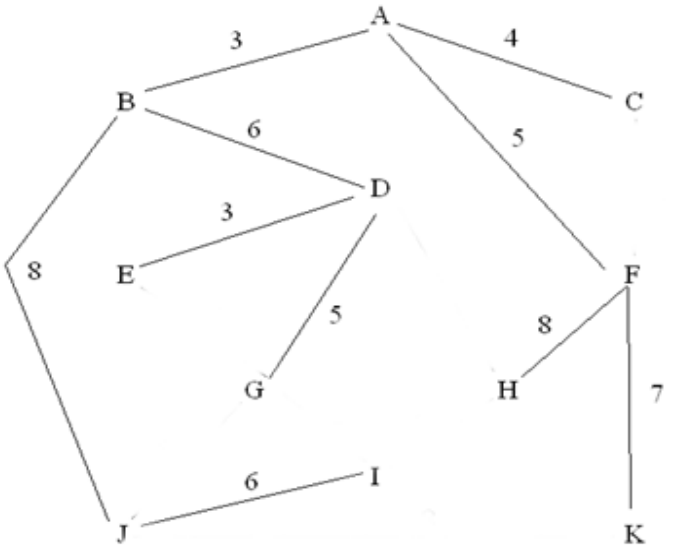
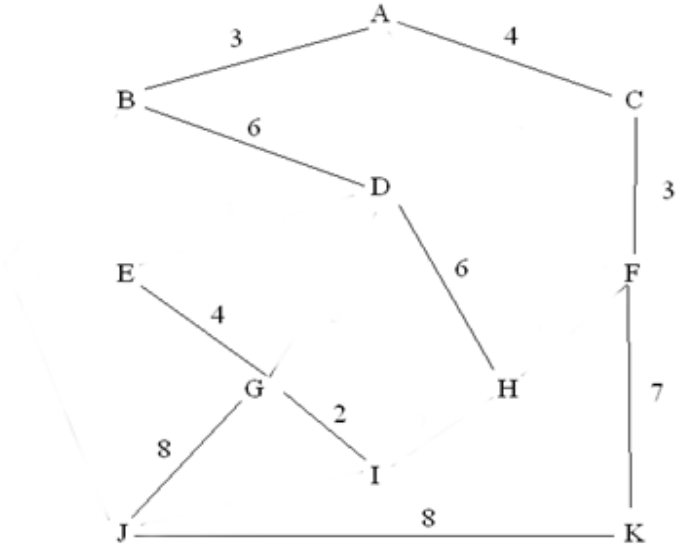
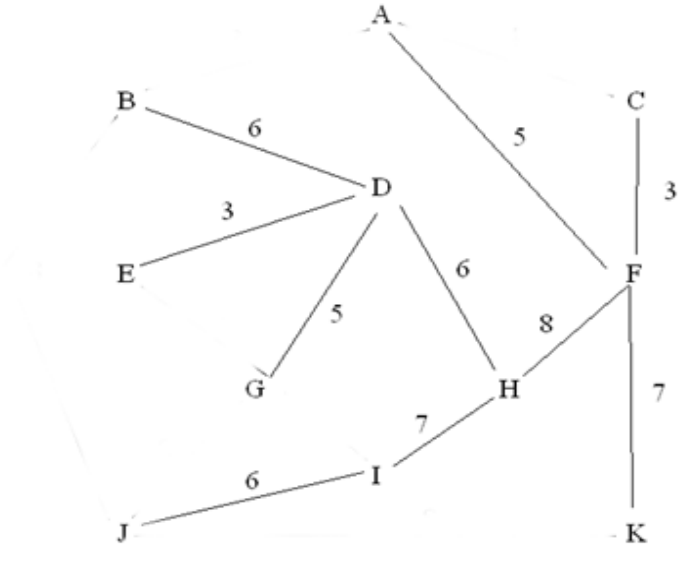
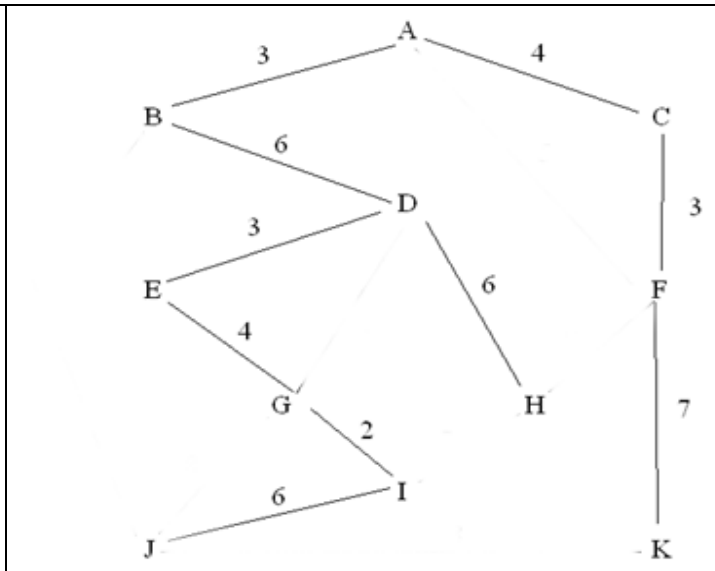
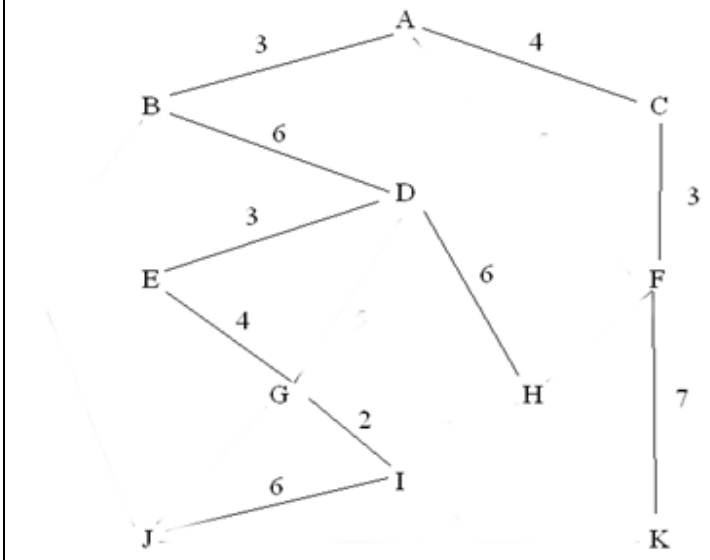
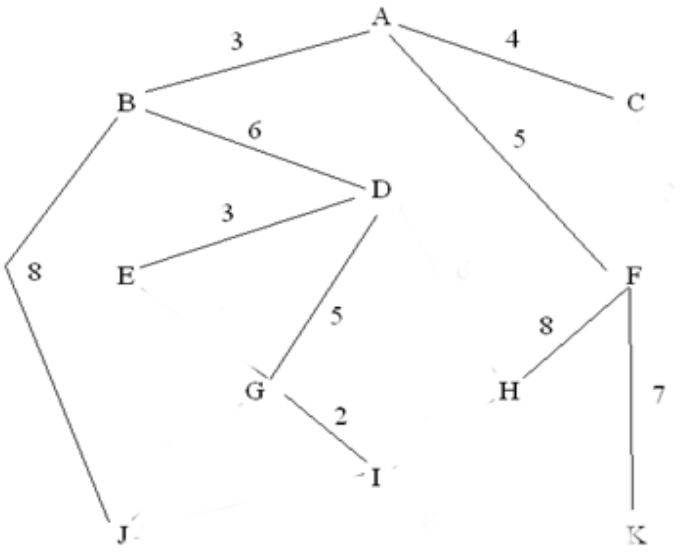


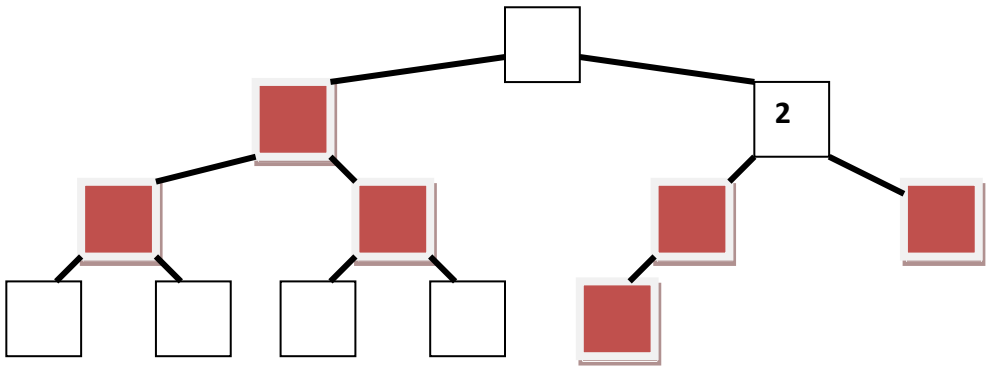
Figure 1 Weighted Graph

(a) Provide the DFS tree starting at node A.	

(b) Provide the BFS tree starting at node A.	
(c) Provide the DFS tree starting at node H.	
(d) Provide the BFS tree starting at node H.	

(e) Use Kruskal's algorithm to derive the MST. Hint: Sort the edges and then work through them.	
(f) Use Prim's algorithm to derive the MST starting at node A. Same as Kruskal's in this case.	
(g) Using Dijkstra's algorithm, determine the shortest path from node A to I. Show the steps, your tables and the resulting path. The path is A->B->D->G->I for a cost of 16. This is the last vertex finalized.	

7. Given the structure of a heap as sketched below, where the second-smallest value in the set is marked with a **2**. Mark a **4** for each node that can possibly contain the fourth-smallest value in the set. Assume that there are no duplicate node values.



The shaded nodes are the answer. Any of the nodes shown below the 2 could be the 4th smallest element. If the left child of the root is the 3rd smallest element, then either of its children, but not its grandchildren, could be the 4th smallest. If the 3rd smallest is under the 2, then the left child of the root could also be the 4th smallest.