**DEPARTMENT OF MATHEMATICS, IIT - Kharagpur**

**End Semester Examination 2022**

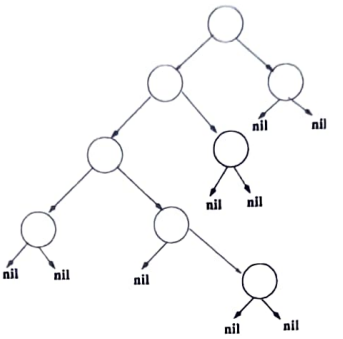
**MA30207/MA21007 Design & Analysis of Algorithms**

**No. of students: 190 Total Points: 50 DURATION: 3 Hours**

---

Answer **ALL QUESTIONS**. All the notations are standard and no query or doubts will be entertained. If any data/statement is missing, identify it in your answer script.

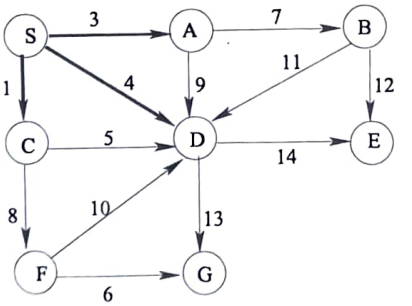
1. **(10 points)** (a) Assign the key $2, 3, 5, 7, 11, 13, 17, 19$ to the nodes of the binary search tree below so that they satisfy the binary-search-tree property.




   (b) Explain why this binary search tree cannot be colored to form a legal red-black tree.

   (c) The binary search tree can be transformed into a red-black tree by performing a single rotation. Draw the red-black tree that results, labeling each node with "red" or "black". Include the keys from part (a).

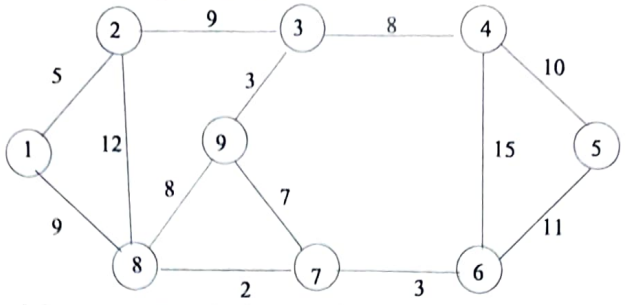
2. **(8 points)** We are running one of these three algorithms on the graph below, where the algorithm has already "processed" the **bold-face** edges. (Ignore the directions on the edges for Prim's and Kruskal's algorithms.)
   - Prim's for the minimum spanning tree, starting from $S$.
   - Dijkstra's for shortest path from $S$.




   (a) Which two edges would be added next in Prim's algorithm? Be sure to indicate the order in which they are added.

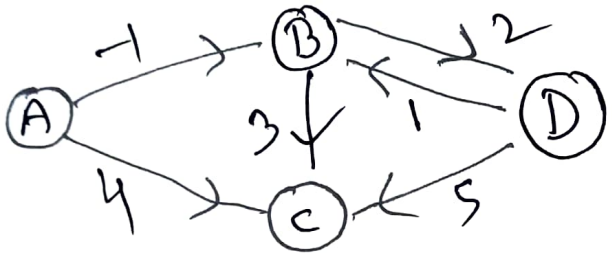
   (c) At this point in the running of Dijkstra's algorithm, $S$ has been taken off the top of the min priority queue and marked. Which four vertices would be marked next in Dijkstra's algorithm? Be sure to indicate the order in which they are marked. Which final edges would Dijkstra's algorithm choose as part of the shortest path to these vertices?

3. (7 points)Consider the following graph $G$:




Find a Minimal Spanning Tree in $G$ using Prim's algorithm.

4. (10 points)Write Bellman-Ford Algorithm for solving shortest path problem and apply on the given graph to short all pair shortest path.(weight only), & matrix.




5 (15 points) **TRUE OR FALSE?** Argue why it is or give a counterexample.

(a) Given a set of $n$ elements, one can output in sorted order of $k$ elements following the median in sorted order in time $O(n + k \log k)$.

(b) Deleting a node from a binary serach tree on $n$ nodes takes $O(\log n)$ time in the worst case.

(c) Given a graph $G = (V, E)$ with cost on edges and a set $S \subset V$, let $(u, v)$ be an edge such that $(u, v)$ is the minimum cost edge between any vertex in $S$ and any vertex in $V - S$. Then, the minimum spanning tree of $G$ must include the edge $(u, v)$. (You may assume the costs on all edges are distinct, if needed.)

(d) A red-black tree on 128 keys must have at least 1 red node.

(e) In the worst case, a red-black tree insertion requires $O(1)$ rotations.

(f) You are given a graph in which all edges have non-negative weights, except those leaving the source vertex s. Then Dijkstra's algorithm may fail to correctly determine the shortest paths from $s$.

(g) If a graph has a negative-weight cycle, then one can repeatedly relax edges in such a way that the shortest path estimates change after each relaxation.

(h) When running Dijkstra's algorithm on a sparse graph (i.e., $E = O(V)$), using a heap implementation of a priority queue will always be less efficient than using a linear-array implementation.

(i) If some of the edge weights in a graph are negative, the shortest path from $s$ to $t$ can be obtained using Dijkstra's algorithm by first adding a large constant $C$ to each weight, where $C$ is chosen large enough that every resulting edge weight will be nonnegative.

(j) Dijkstra's algorithm asymptotically beats the Bellman-Ford algorithm at solving the single-source shortest path problem.

——The End——