ECE360C: Algorithms — Homework Assignment #7
University of Texas at Austin — Due: March 15, 2025 (11:59pm)
STUDENT CHOICE: ☐

Homework Assignment #7

Problem 1: Remember [20 points]

Answer the following true/false questions.

True	False	
☐	☐	Kruskal's algorithm will find a minimum spanning tree, even if the graph contains edges with negative weights.
☐	☐	Prim's algorithm will find a minimum spanning tree, even if the graph contains edges with negative weights.
☐	☐	Dijkstra's algorithm can be used to find the *longest* path in a graph by (only) changing the min-heap to a max-heap and using `Extract-Max` instead of `Extract-Min`
☐	☐	Dijkstra's algorithm can be used to solve the *single-destination shortest paths* problem in a directed graph by (1) creating G_{rev} which is the same as G with all of the edges reversed and (2) running Dijkstra's algorithm from the target destination.
☐	☐	Given a cut $(S, V - S)$ of an undirected graph G, there is always exactly one *light edge* that crosses the cut.
☐	☐	In a graph G where all of the edge weights are distinct, if an edge is the heaviest edge on a cycle in G, then that edge will be in every minimum spanning tree of G.
☐	☐	In a graph G, if an edge e is not part of a any cycle in G, it will be in every minimum spanning tree of G.
☐	☐	Breadth-first search can be used instead of Dijkstra's algorithm to find the shortest cost path between two nodes when all edges in the graph have the same weight.
☐	☐	During Dijkstra's algorithm's search for a shortest path, a node's estimate of its distance can go up and down, but it will eventually settle on the correct shortest path cost.
☐	☐	Let G be a graph with strictly positive edges and let T be an MST of G. Suppose you double the weight of every edge in the graph G, then T is still an MST of G; i.e., the edges of the MST remain the same.

Problem 2: Understand

Consider an undirected graph that may have negative edge weights.

Will Kruskal's algorithm still generate the minimum spanning tree? If so, explain why. If not, give an example on which Kruskal's algorithm will fail.

Will Prim's algorithm still generate the minimum spanning tree? If so, explain why. If not, give an example on which Prim's algorithm will fail.

Problem 3: Apply [20 points]

You are given an undirected, connected graph G, and each edge is colored red or blue (there are no edge weights). G has n vertices and m edges. Give an algorithm with running time $O(m)$ to find a spanning tree with the minimum number of blue edges. Justify the running time.

Hint: You don't have enough time to run the regular Prim's or Kruskal's. Think about how to modify Prim's to take linear time in this case.

Problem 4: Analyze [20 points]

Suppose you are given an array A of n sorted numbers that has been *circularly shifted* to the right by k positions. For example, $\{35, 42, 5, 15, 27, 29\}$ is a sorted array that has been circularly shifted $k = 2$ positions, while $\{27, 29, 35, 42, 4, 15\}$ has been shifted $k = 4$ positions. Give an $O(\log n)$ algorithm to find the largest number in A. You may assume the elements of A are distinct. Write the recurrence for your algorithm and show that its recurrence solves to $O(\log n)$ (e.g., using the Master Method, a recursion tree, or an inductive proof).

Problem 5: Evaluate [20 points]

Consider a situation where you have to find available classrooms for n different lectures. Of course, you must avoid scheduling two or more overlapping lecture in the same room. Each lecture i begins at s_i and ends at t_i.

(a) Find an algorithm that assigns the smallest number of rooms possible.

(b) Show that your algorithm is optimal.

Problem 6: Create [20 points]

A group of network designers at the communication company CluNet find themselves facing the following problem. They have a connected graph $G(E, V)$ in which nodes represent sites that want to communicate, each edge e is a communication link with a given distinct bandwidth . For each pair of nodes they want to select a single path P on which to communicate. The bandwidth of this path $B(P)$ is the minimum bandwidth of any edge on the path P. The best achievable bandwidth (BAB) for the pair u, v is the maximum bandwidth over all paths between u and v in G.
The problem is to find BAB for every pair of nodes in the graph, which is very complicated due to the huge number of paths. A network designer makes a bold suggestion: Maybe one can find a spanning tree T of G so that for every pair of nodes u,v the unique path between u and v in the tree actually attains the BAB. The idea is tossed away by his colleagues since there is a natural reason for skepticism: each pair of nodes might want a very different path to get BAB; why should there be a single tree that simultaneously makes everybody happy? But after some failed attempts to rule out the idea, designers at NetSD begin to suspect it could be possible. Prove that such a spanning tree exists and give an efficient algorithm to find it. That is, give an algorithm constructing a spanning tree T in which, for each u, v in V, the bottleneck rate of the $u - v$ path in T is equal to the best achievable bottleneck rate for the pair u, v in G.

(Hint: The cycle property of MSTs says that the edge with the largest weight in a cycle will never be in a MST.)