15-351/15-650/02-613 - Midterm 1 - Fall 2024

Instructor: Prof. Yun William Yu

Name: ______________________ Andrew ID: ______________________

Problem 1 [16pts]. For each part below, indicate True or False and provide a short explanation for your answer (either in words or formulas).

a) (True/False): All graphs can be topologically sorted using depth-first search (DFS).

b) (True/False): The minimum edge weight in a cycle of a graph will always be in the minimum spanning tree.

c) (True/False): If $f(x) = 2 + sin(x)$ and $g(x) = 2 + cos(x)$, f(x) = $\Theta(g(x))$.

d) (True/False): Using an admissible heuristic h(x) for A* search guarantees that the algorithm will choose a solution asymptotically faster than Dijkstra

e) (True/False): A BFS tree of a graph has a layer of nodes in each level from the root node. If there are no edges connecting nodes within the same layer, then there is no cycle in the graph.

f) (True/False): The edge with the highest weight in the graph will never be in the MST.

g) (True/False): The Bellman-Ford algorithm can detect the presence of negative weight cycles in a graph.

h) (True/False): Breadth-first search (BFS) can be used to find the shortest path in a graph with weighted edges.

Problem 2 [12pts]. You have to take n courses in order to finish your degree within the next k semesters. These courses are numbered from 0 to $n-1$. Each course may or may not have prerequisite courses (i.e. course 4 requires course 2, and course 2 requires course 1). These prerequisite courses are listed in a list form, where the list has m pairs. For the example above it is represented as [[4,2], [2, 1]]. You may take as many courses as you want each semester, but you can only take a course if you completed its prereqs before the start of the current semester.

Design an $O(n+m+k)$-time algorithm that determines if it is feasible for you to finish your degree.

Hint: Can you interpret the description above as a graph of some kind?

Problem 3 [14pts]. Consider an array A of n (not necessarily distinct) numbers. Write an $O(n)$ divide-and-conquer algorithm to find the maximum difference between any two elements of A, and justify your runtime analysis.

Note: For full marks, your answer *must* use divide-and-conquer – there is a very simple non-divide-and-conquer $O(n)$ algorithm as well, but that answer will not get you full marks. *Hint:* What would you need to know to determine the maximum difference between two items in a list of numbers?

Note 2: Recall that the Simplified Master Theorem states that a recurrence of form $T(n) = aT(n/b) + \Theta(n^i)$ has solution $\Theta(n^{\log_b a})$ if $a > b^i$, $\Theta(n^i \log n)$ if $a = b^i$, and $\Theta(n^i)$ if $a < b^i$.

Problem 4 [18pts]. You are given an undirected graph $G = (V, E)$ with distinct, positive costs $c(u, v)$ on edges in E.

You want to compute the path P_{st} in G from a given node s to a given node t that has the *smallest maximum* cost on any edge in P_{st} over all possible $s \to t$ paths. In other words, you want to find a path P_{st} that minimizes the amount you have to spend on the most expensive edge. Consider the following:

Algorithm sketch. Run a Dijkstra-like greedy tree-growing algorithm, but choose the frontier edge u, v that has the *smallest* value of $\max\{d(u), c(u, v)\}$ where $d(u)$ is the maximum cost on the path found to u.

Using a technique similar to the proof of correctness for Dijkstra's algorithm, prove that the algorithm above finds the desired $s \to t$ path, or give an example where it fails to.

Problem 5 [20pts]. You are given a string s of length n, and a list L of pairs of indices in the string pairs where pairs[i] = [a, b] indicates 2 indices of the string; L has length m.

You can swap the characters at any pair of indices in L any number of times. Design an $O(m + n \log n)$ algorithm that returns the lexicographically smallest string (i.e., the smallest string in alphabetical order) that s can be changed to after using the swaps.

For example:

```
Input: s = "dcab", pairs = [[0,3],[1,2]]

Output: "bacd"

Explanation:
Swap s[0] and s[3], s = "bcad"
Swap s[1] and s[2], s = "bacd"
```

Note that you **cannot** reach “abcd”, the overall lexicographically smallest string given those letters, because we only had two allowed swaps. If the list L includes all pairs of indices, then this problem is just sorting the letters, but the problem is nontrivial precisely because L may be limited.

Problem 6 [20pts]. Imagine you have an undirected, connected, graph G=(V, E), where edge weights are not necessarily unique. There could be multiple valid MSTs of G. Describe an $O(|E|^2)$ algorithm that will return all edges that can be found in at least one MST of G.

Problem 7 [Bonus, 5pts; will not be taken into account by any curve].
Recall that the Simplified Master Theorem states that a recurrence of form $T(n) = aT(n/b) + \Theta(n^i)$ has solution $\Theta(n^{\log_b a})$ if $a > b^i$, $\Theta(n^i \log n)$ if $a = b^i$, and $\Theta(n^i)$ if $a < b^i$.

Prove the Simplified Master Theorem.