COMPSCI 311: Introduction to Algorithms **Spring 2020**

Homework 3

Released 2/25/2019 Due 3/11/2019 11:59pm in Gradescope

Instructions. You may work in groups, but you must write solutions yourself. List collaborators on your submission. Also list any sources of help (including online sources) other than the textbook and course staff.

If you are asked to design an algorithm, please provide: (a) the pseudocode or precise description in words of the algorithm, (b) an explanation of the intuition for the algorithm, (c) a proof of correctness, (d) the running time of your algorithm and (e) justification for your running time analysis.

Submissions. Please submit a PDF file. You may submit a scanned handwritten document, but a typed submission is preferred. Please assign pages to questions in Gradescope.

1. **(20 points) Three Friends and a Car** Three friends, Alice, Bob, and Cora, want to drive from s to t. They have a complete map with all rest stops (vertices, including s and t), roads (edges) between rest stops, and the length of each road segment. They want to share driving equitably, so they set the following rules:
1) A segment between any two rest stops is driven by a single person.
2) On any part of their trip consisting of more than three consecutive segments, any of them can drive at most once more than any other. This also applies to the entire trip, regardless of the number of segments.
3) Alice insists on driving the first and the last segment.

Give an algorithm that finds the shortest (not necessarily simple) path from s to t that can be driven with the rules above and an assignment of segments to drivers; otherwise report this is impossible.

2. **(20 points) Bottleneck Edge** Consider an undirected weighted graph and think of edge weights as *capacities*. The capacity of any path is defined as the minimum capacity over all edges on the path.
Given two nodes s and t, the *bottleneck capacity* between s and t is the maximum capacity of any $s-t$ path. (This is ∞ if $s = t$, and $-\infty$ if there is no $s - t$ path).

a) Show that the *maximum* spanning tree of G contains a path of maximum capacity between any two nodes.

b) Design an algorithm that answers in $O(m + n)$ time whether the bottleneck capacity between two given nodes s and t is at most a given value C.

c) Design an $O((m+n)\log n)$ algorithm that finds the bottleneck capacity between two given nodes s and t.

3. **(20 points) Paintball** Having recently watched the classic film *The Deer Hunter*, n members of the UMass Paintball Club decide to play the following variant of Russian Roulette at their next session.
The n players stand in a row in some order. In each round, each player shoots up to two other players:
(1) the closest player to their left who is taller than they are (if any), and
(2) the closest player to their right who is taller than they are (if any).
We assume that no two players have the same height and that the shooters always hit their targets. After a round, the remaining unshot players continue (in the same relative order) until there is only one player left.

(a) Assume we are given an array H such that entry $H[i]$ is the height of the i^{th} player from the left. Describe a simple $O(n^2)$ time algorithm to determine the two targets of each player in the first round.

(b) Give an $O(n \log n)$ divide-and-conquer algorithm to find the two targets of each player in the next round.

(c) Prove that at least $\lfloor n/2 \rfloor$ of the players will be shot in the first round.

(d) It is clear that the game will end with only the shortest player remaining. Give an algorithm to find how many rounds this will take, given the initial array H. For full credit your algorithm should run in $O(n)$ time.

4. **(20 points) Recurrences** Give asymptotic upper and lower bounds in the following recurrences. Assume that $T(n)$ is bounded by a constant for $n \leq 3$. Make your bounds as tight as possible and justify your answers.

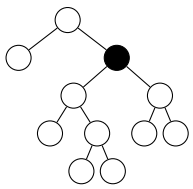
a) $T(n) = 2T(n/3) + 2T(n/9) + n$

b) $T(n) = \frac{1}{3}(T(n-1) + T(n-2) + T(n-3)) + cn$, with $c > 0$

c) $T(n) = T(n/2) + T(n/3) + T(n/6) + n$

d) $T(n) = \sqrt{2n}T(\sqrt{2n}) + \sqrt{n}$ (assume $T(n) \geq \sqrt{n}$)

5. **(20 points) Largest Perfect Tree**. Consider a *proper* binary tree (every node has 0 or 2 children). You are allowed to delete any nodes and edges.
Give an algorithm that determines the largest *perfect* binary tree you can obtain. A perfect binary tree is a proper binary tree whose leaves are all on the same level. Your algorithm should return the root node and the depth (number of edges between root and leaves). In the given figure, this is the black node, with depth 2.



6. **(0 points).** How long did it take you to complete this assignment?