CSC373 Fall'19
Assignment 1

Due Date: October 14, 2019, by 11:59pm

Instructions

1. Be sure to include your name and student number with your assignment. Typed assignments are preferred (e.g., PDFs created using LaTeX or Word), especially if your handwriting is possibly illegible or if you do not have access to a good quality scanner. Please submit a single PDF on MarkUS at `https://markus.teach.cs.toronto.edu/csc373-2019-09`

2. You will receive 20% of the points for any (sub)problem for which you write "I do not know how to approach this problem." (you will receive 10% if you leave the question blank and do not write this or a similar statement).

3. You may receive partial credit for the work that is clearly on the right track. But if your answer is largely irrelevant, you will receive 0 points.

4. This assignment has 4 questions. The total number of marks is 80.

Note: The theme of this assignment is the circus!

Q1 [25 Points] Raising the Bar

There are n clowns in the circus numbered $1, 2, \ldots, n$. The height of clown i is h_i. All clowns have equal-sized heads of height f. The clowns form a queue whose ordering is defined by an array of integers L, where $L[j]$ is the index of the j^{th} clown in the queue.

The first clown $L[1]$ walks under a metal bar cranked-up from the ground by the strong-man of the circus, just high enough so the clown can pass under it. Then $L[2]$ stands on $L[1]$'s shoulders and they both walk under the bar, again raised from the ground just high enough so they both can pass. This continues until a tower of clowns standing on each other's shoulders, with $L[1]$ at the bottom and $L[n]$ at the top, walk under the bar. Specifically, when the tower of clowns $L[1], \ldots, L[k]$ pass under the bar, the strong-man raises the bar to the height $H_k = \left(\sum_{j=1}^{k} h_{L[j]}\right) - (k-1) * f$. The effort made by the strong-man is proportional to the cumulative height $\sum_{k=1}^{n} H_k$ to which he must crank the bar.

(a) [3 Points] Design a greedy algorithm to compute the optimal order of clowns in the queue L that will minimize the strong-man's effort.

(b) [5 Points] Prove that your greedy algorithm from (a) will always yield a minimum-effort solution.

(c) [2 Points] What is the worst-case time complexity of this algorithm?

(d) [4 Points] Suppose now that the strong-man can only crank the bar up to a maximum height of H. We again want successive towers of clowns to pass under the bar. Unlike part (a), note that a tower consisting of all clowns may no longer be feasible given the maximum height of H. Our goal is now to construct a sequence of towers of clowns such that (a) the final tower in our solution is as tall as possible (we want the finale to be impressive!), and (b) among all such sequences, our solution minimizes the strong-man's effort. Design a greedy algorithm for this problem. Bizarrely, you are told that the shoulder heights $h_i - f$ of the clowns are all unique powers of 2 (that is, each $h_i - f$ is a power of 2, and $h_i - f \neq h_j - f$ for $i \neq j$).

(e) [5 Points] Prove that the your greedy algorithm from (d) will always yield an optimal solution.

(f) [3 Points] What is the worst-case time complexity of your algorithm from (d)?

(g) [3 Points] Let us go back to part (a) without any maximum height restriction. Suppose now that the clowns also have girths, where the girth of clown i is g_i. To allow a tower of clowns $L[1], \ldots, L[k]$ to pass, the strong-man now needs to both raise the bar to the height H_k *and* spread two vertical posts by width $W_k = \max(g_{L[1]}, \ldots, g_{L[k]})$, which is just wide enough for the clowns to pass. The effort made by the strong-man is now proportional to the cumulative area $\sum_{k=1}^{n} W_k \cdot H_k$. Is your greedy algorithm of (a) still guaranteed to provide a solution that minimizes the strong-man's effort? Either prove that it does, or provide a counter-example.

Q2 [25 Points] Circus Arena

The circus handyman tasked with creating a roped off arena in the circus arbitrarily lays n stakes into the ground at points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ while sleepwalking. In the morning, a rope is placed loosely surrounding all the stakes and then tightened to mark the arena.

(a) [7 Points] Design a divide and conquer algorithm to determine the clockwise circular sequence of stakes that anchor the rope. Formally, your algorithm should output an array $P[1, \ldots, k]$ such that $P[1]$ is some anchor touching the rope, $P[2]$ is the next anchor touching the rope in the clockwise order, $P[3]$ is the next anchor touching the rope in the clockwise order, and so on. The divide step should reduce the problem into approximately equal-sized smaller instances of the problem. The combine or merge step should then use the circular sequences of stakes that anchor sub-arenas to compute the solution for the single larger arena.

(b) [7 Points] Prove the correctness of your algorithm in (a).

(c) [3 Points] What is the worst-case time complexity of the algorithm?

(d) [3 Points] Suppose now that a number of nested arenas are created by recursively tightening a rope around stakes left inside the outer arena (i.e. stakes not touching the outer rope). What is the worst-case time complexity of recursively running your algorithm from part (a) until all nested arenas are created so that each stake is touching a rope?

(e) [5 Points] Suppose lions start parachuting into this arena. Given the output from (a), design an recursive $O(\log k)$ time algorithm to check if a lion touching down at point (u, v) will land inside the arena (here, k is the length of the array found in (a)).

Q3 [10 Points] Lions and Tamers

There are m lions $L_1, \ldots, L_m$ and $n - m$ tamers $T_{m+1}, \ldots, T_n$ on pedestals connected by planks. The plank connecting lion/tamer i with lion/tamer j has length $l(i, j)$, for each $i, j \in \{1, \ldots, n\}$. You want to select a subset of planks that satisfy the following conditions:

- No lions are directly connected to each other by a plank.
- Every lion is directly connected by a plank to at most one tamer.
- There is a path of planks connecting any two tamers.

(a) [5 Points] Describe an algorithm that, subject to the above conditions, selects the subset of planks minimizing the total length. (You can directly refer to any algorithm mentioned in the lecture slides; you do not need to describe such an algorithm in detail.)

(b) [5 Points] Prove that correctness of your algorithm and analyze worst-case time complexity.

Q4 [20 Points] Trapeeze

There are $m + n$ acrobats on a long horizontal circus rope. Among the acrobats, m are "holders" suspended upside-down and pulling the rope up, while the rest n are "hangers" hanging from the rope and pulling the rope down. The force applied by acrobat k from the left end of the rope is f_k, and index $i_k \in \{\text{holder,hanger}\}$ indicates whether acrobat k is a holder or a hanger. Acrobats at the two ends of the rope, i.e. acrobats 1 and $m + n$, are holders.

For the safety of the rope, it is important that the net force pulling down in any segment of the rope (i.e. the total force applied by hangers in that segment minus the total force applied by holders in that segment) does not exceed F.

(a) [5 Points] Design a divide and conquer algorithm to determine if the rope is safe given $f_1, \dots, f_{m+n}$ and $i_1, \dots, i_{m+n}$.

(b) [5 Points] Prove that your algorithm is correct, and analyze its worst-case time complexity.

(c) [5 Points] Among the $m + n$ acrobats, you are now given which acrobats know which other acrobats. You need to assign them costumes of different colors such that no two acrobats who know each other are assigned costumes of the same color. Design a greedy algorithm to assign costumes to acrobats which uses at most one more color than the maximum number of acrobats that any acrobat knows. Prove that your algorithm produces a valid coloring and uses no more than the desired number of colors.

(d) [5 Points] Provide a counterexample to show that the algorithm from (c) does not always produce costume assignments with the smallest possible number of colors.