CSC263 Summer 2022 Problem Set 2

Due June 24, 2022 23:59 ET on MarkUs

- Your problem sets are graded on both correctness and clarity of communication. Solutions that are correct but poorly written will not receive full marks.
- Each problem set may be completed individually or in a group of two students.
- Solutions must be typeset electronically and submitted to MarkUs as a pdf with the filename `ps2.pdf`. Handwritten submissions will receive a grade of zero.
- The work you submit must be that of your group. You may not use or copy from the work of other groups or from external sources like websites or textbooks. All the information you need to answer these questions has been covered in lecture and/or in the textbook.
- If you consult any external sources (i.e. other than the lecture notes, CLRS, or documents posted to the course website), then you must clearly indicate these additional sources on the first page of your submission.

Question 1 [8 marks]

A nonempty max-heap H is called an *Extract-Insert-Stable* (EIS) heap if and only if applying *ExtractMax(H)* followed by *Insert*(H, x) results in the original heap H, where x is the item that was returned by *ExtractMax*(H). A nonempty max-heap H is called *right-dominant* if and only if the keys of all nodes in H are distinct and, for every node x in H that has two children, the key of $x.right$ is greater than the key of $x.left$.

(a) Draw an example of an EIS max-heap with at least 6 nodes.

(b) Draw an example of a right-dominant max-heap with at least 6 nodes that is *not* an EIS heap. To show that it is not an EIS heap, draw the resulting heap after extracting and then reinserting the item in the root.

(c) Either prove or disprove the following statement: Any right-dominant heap H is an EIS heap if and only if it has $2^i - 1$ or $2^i - 2$ nodes, for some natural number i.

Question 2 [6 marks]

There is an unwritten rule that says you cannot wait within d meters of another commuter at the bus stop. Doing so would get you labelled a d-near-stander, which nobody wants. We wish to design an algorithm to determine whether there is a pair of commuters standing too close at a particular bus stop.

More formally, a *commuter* is defined as a nonnegative real number, which represents a position. A *bus stop* is defined as a list of distinct commuters $[c_1, \ldots, c_n]$, which are the positions of the people waiting at the bus stop. Notice that the commuters at a bus stop are in no particular order (i.e. they are not necessarily sorted in any way). A pair of commuters c_i, c_j are called *d-near-standers* if $|c_i - c_j| \leq d$, for some real number $d > 1$.

(a) Write an algorithm that takes a bus stop $[c_1, \ldots, c_n]$ and a real number $d > 1$ as input and determines whether there is a pair of d-near-standers waiting at the given bus stop. If there exists a pair of d-near-standers, then your algorithm should return the indices of such a pair. Otherwise, your algorithm should return `None`. You may use any of the algorithms or data structures we have learned in class so far. The worst-case runtime of your algorithm should be $O(n)$. Algorithms that are less (asymptotically) efficient than this can receive partial marks.

(b) Prove a tight asymptotic bound on the worst-case runtime of your algorithm.

Question 3 [6 marks]

The ZFL is a professional football league containing n teams. Each team in the ZFL has an associated *star power*, which determines that team's level of success during a season. After each season, the worst team (i.e. the team with the lowest star power) is kicked out of the league and replaced with a new team.

We wish to model the ZFL with the following ADT.

- Data: a set of n teams, each with a star power (an integer) and a unique name (a string). You may assume that team names are at most 30 characters long.
- Operations:
  - `Trade(`S_1`, `S_2`, `x`)`: transfer x star power from the team with name S_1 to the team with name S_2. That is, reduce the star power of the team with name S_1 by x and increase the star power of the team with name S_2 by x. The parameter x is a positive integer. You may assume that your data structure contains teams with the names S_1 and S_2.
  - `Eliminate_and_replace(`T`)`: remove a team with the lowest star power from the league. If there are multiple teams with the lowest star power, then remove any one of them. Then add the team T to the league. Finally, return the name of the team that was eliminated.

Design a data structure to implement this ADT. Both operations should have worst-case runtime $O(\log n)$.

When asked to provide an implementation of an operation, you *must* give a high-level description of your algorithm in English! You may reference algorithms seen in class without describing them, and simply describe the modifications made, if any. You do not need to prove bounds on the runtime of algorithms we have discussed in lecture. You may supplement your description with pseudo-code, but it is not necessary.

(a) Describe the data structure you will use to implement this ADT. Explain any augmentations that are needed to implement the operations. Assume that your data structure initially contains n teams and all of your augmentations are properly initialized for these teams.

(b) Provide an implementation of `Trade(`S_1`, `S_2`, `x`)`. Briefly justify why its worst case runtime is $O(\log n)$.

(c) Provide an implementation of `Eliminate_and_replace(`T`)`. Briefly justify why its worst case runtime is $O(\log n)$.