CS3000: Algorithms & Data — Summer 2023 — Laney Strange

Recitation 3
Due Tuesday May 30 @ 9pm Gradescope

Name:
Collaborators:

- One recitation problem each week is graded; the rest are there for practice. That problem will be graded on completeness – full credit for making an honest effort. It is also closely linked to the quicksort problem on the upcoming Short Homework, so it's good to review it now!
- Recitations can be written by hand; submit a picture/screenshot on Gradescope.
- Put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This recitation is due Tuesday May 30 @ 9pm Gradescope. If you miss the in-person recitation, or need to submit your solution later than the end of your section, please fill out this form: https://forms.gle/CLrhrkVauXYzC7U57
- Collaboration is strongly encouraged during recitation! Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.

Problem 1. *Heaps (graded)*

Although we visualize heaps a lot like binary trees, we implement them with an underlying array. Position 1 is the root; positions 2 and 3 are the second level; positions 4, 5, 6, and 7 are the third level, etc.

(a) Draw the Heap represented by this array: $[5, 14, 23, 32, 41, 87, 90, 50, 64, 53]$. Is it a valid min-heap?

Solution:

(b) Draw the Heap represented by this array: $[5, 14, 23, 12, 26, 34, 20, 24, 35]$. Is it a valid min-heap?

Solution:

(c) What would the heap in Part A look like after we remove the minimum element and re-heapify?

Solution:

Problem 2. *Huffman (for practice; not graded)*

Recall that our greedy implementation for Huffman codes requires us to repeatedly find the two characters with the lowest frequencies and merge them together to make one node of a binary tree. Every character x has a corresponding frequency $x.freq$. In class, we did a high-level version of the pseudocode; for this problem, give pseudocode for a ”real” solution.

Your HUFFMAN algorithm should take in a min-heap of n binary trees. We can refer to an entire tree by its root node. Each binary tree starts out as a single/root node that contains a character and its frequency. You can assume you can do all of the following:

- EXTRACT-MIN(H) removes and returns the smallest binary tree in the heap, and re-heapifies the remaining elements.
- Allocate a new node for your binary tree
- Once you've allocated a new node z, you can ”merge” your two lowest-freq characters x and y together by setting $z.left = x$ and $z.right = y$
- INSERT(H, n) inserts the root of a binary tree, n, into Heap H
- You can assume that your heap has a length, $H.length$, or you can pass in its length n as another parameter.

Solution: