University of Waterloo
CS240 Spring 2025
Assignment 3

Due Date: Tuesday, June 17 at 5:00pm

Read `https://student.cs.uwaterloo.ca/~cs240/s25/assignments.phtml#guidelines` for guidelines on submission. **Each question must be submitted individually to Crowdmark.** Submit early and often.

Grace period: submissions made before 19:59PM on June 17 will be accepted without penalty. Your last submission will be graded. Please note that submissions made after 19:59PM **will not be graded** and may only be reviewed for feedback.

1. [7 marks] We want to prove the following: there is no comparison-based algorithm that can merge m sorted arrays of length m into a unique sorted array of length m^2 doing $O(m^2)$ comparisons. We argue by contradiction, and we assume that it is possible, so that we have such an algorithm (which we call FastMerge).

   Modify MergeSort in order to use FastMerge, and derive a contradiction. The following recurrence relation may show up: $T(n) = \sqrt{n}T(\sqrt{n}) + O(n)$; here you can disregard issues related to the fact that $\sqrt{n}$ is not necessarily an integer. You can use the fact that this gives $T(n) \in o(n \log n)$ **without proving it.**

2. [7 marks] Let A be an unsorted array of n integers in the range $[0, n^{42}]$. Design an algorithm that finds the minimum (non-negative) difference between any two numbers in this array. For instance, if the input was $[82, 32, 55, 78, 148]$, then the answer would be 4, witnessed by the pair 78 and 82. Your algorithm must take $O(n)$ time. It is important that your solution is explicit about how you represent the data. You may assume that the numbers are given in base n, and that computing $x \mod n$ and computing floor are constant time operations. each number is given as a word on memory. So, you don't have direct access to digits of a given number.

3. [2+5+5=12 marks] It is possible to implement AVL trees such that the nodes store only the balance factor $\{-1, 0, 1\}$ at each node instead of the height of the subtree rooted at the node.

   (a) Show that the tree T in Figure 1 is an AVL tree by writing in the balance factor in the lower half of each node.

   (b) Show the process of inserting a KVP with key 29 into the tree T in Figure 1. Specifically, draw the tree, with balance factors, after each call to restructure.

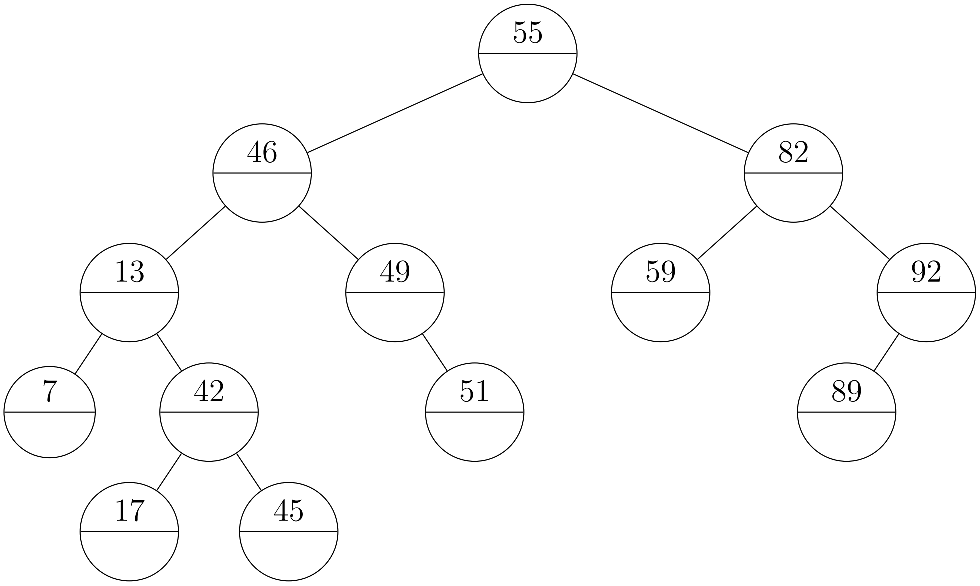

Figure 1: Binary tree T of problem 1

(c) Show the process of deleting key 49 from the *original* tree T in Figure 1. Specifically, draw the tree, with balance factors, after each call to restructure.

4. [2+6+5=13 marks]

   Consider an AVL tree T with n nodes, and let v be a leaf in T. We want to give a lower bound on the *depth* ℓ of v, that is, the length of the path from the root of T to v.

   (a) Let $v_0, \ldots, v_\ell$ be the path from the root of T to v (where v_0 is the root of T and $v_\ell = v$), and let $T_0, \ldots, T_\ell$ be the subtrees of T rooted at respectively $v_0, \ldots, v_\ell$. What is T_0 and what is T_ℓ?

   (b) Prove by induction that for $i = 0, \ldots, \ell$, $T_{\ell-i}$ has height at most $2i$.

   (c) Using the previous question, deduce a lower bound of the form $\ell \in \Omega(g(n))$, for a certain function $g(n)$.

5. [6 marks] Describe an algorithm for computing the height of a given AVL tree (where nodes store the balance factor $\{-1, 0, 1\}$ instead of height) in $O(\log n)$ time on an AVL tree of size n. In the pseudocode, use the following terminology: T.left, T.right, and T.parent indicate the left child, right child, and parent of a node T and T.balance indicates its *balance factor* (-1, 0, or 1). For example if T is the root we have T.parent=nil and if T is a leaf we have T.left and T.right equal to nil. The input is the root of the AVL tree. Justify correctness of the algorithm and provide a brief justification of the runtime.