Homework Assignment #4
Due: October 21, 2020, by 11:59 pm

- You must submit your assignment through the Crowdmark system. You will receive by email an invitation through which you can submit your work in the form of separate PDF documents with your answers to each question of the assignment. To work with a partner, you and your partner must form a group on Crowdmark. Crowdmark does not enforce a limit on the size of groups. The course policy that limits the size of each group to at most two remains in effect: submissions by groups of more than two persons will not be graded.
- It is your responsibility to ensure that the PDF files you submit are legible. To this end, I encourage you to learn and use the LaTeX typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You are not required to produce the PDF files you submit using LaTeX; you may produce it any way you wish, as long as the resulting document is legible.
- By virtue of submitting this assignment you (and your partner, if you have one) acknowledge that you are aware of the policy on homework collaboration for this course.
- For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbooks, by referring to it.
- Unless we explicitly state otherwise, you may describe algorithms in high-level pseudocode or point-form English, whichever leads to a clearer and simpler description.
- Unless we explicitly state otherwise, you should justify your answers. Your paper will be graded based on the correctness and efficiency of your answers, and the clarity, precision, and conciseness of your presentation.

"In each homework assignment you may collaborate with at most one other student who is currently taking CSCC73. If you collaborate with another student on an assignment, you and your partner must submit only one copy of your solution, with both of your names. The solution will be graded in the usual way and both partners will receive the same mark. Collaboration involving more than two students is not allowed. For help with your homework you may consult only the instructor, TAs, your homework partner (if you have one), your textbook, and your class notes. You may not consult any other source."

Question 1. (10 marks) An interval is a pair \((\ell, r)\) of numbers such that \(\ell \leq r\), or the empty interval denoted \(\emptyset\). The intersection of two intervals \((\ell, r)\) and \((\ell', r')\) is the interval \((\max(\ell, \ell'), \min(r, r'))\) if \(\max(\ell, \ell') \leq \min(r, r')\); otherwise, it is \(\emptyset\). The length of interval \((\ell, r)\) is \(r - \ell\), and the length of \(\emptyset\) is 0.

We are given a list of \(n \geq 2\) nonempty intervals \((\ell_1, r_1), (\ell_2, r_2), \ldots, (\ell_n, r_n)\). We want to find two distinct intervals in this list whose intersection is as long as possible. It is straightforward to do this in \(\Theta(n^2)\) time. Describe a divide-and-conquer algorithm that solves this problem in \(O(n \log n)\) time. Explain why your algorithm is correct and analyze its running time. (Hint. Split the given list of intervals by their left (or right) endpoints.)

Question 2. (10 marks) Consider the following abstract data type. The state of the abstract data type consists of a multiset \(S\) of integers, initially empty. (A multiset is like a set, except that it can contain multiple copies of elements. In this respect, it is like a sequence; unlike a sequence, however, there is no order to the elements of a multiset.) There are two operations on the abstract data type:

- \(\text{INSERT}(x)\), where \(x\) is an integer, adds \(x\) to \(S\).
• **DELETEMINHALF** deletes the smallest half elements of \( S \). More precisely (since duplicates may exist in \( S \)), this operation deletes exactly \( \lceil |S|/2 \rceil \) elements of \( S \), and every element deleted is less than or equal to every element that remains in \( S \) after the operation.

Describe a data structure to represent the multiset \( S \), and algorithms to implement the INSERT and DELETEMINHALF operations, so that the **amortized** cost of an operation in any sequence of INSERT and DELETEMINHALF operations is \( O(1) \). Justify why your implementation achieves this amortized cost. What is the worst-case running time of an operation in a sequence of \( n \) INSERT and DELETEMINHALF operations in your implementation? Justify your answer. (You need not justify your implementation’s correctness; this should be completely straightforward from your algorithm descriptions.)

**Question 3.** (10 marks) Let \( S \) be a set of at least two numbers, and let

\[
\text{ads}(S) = \frac{\max(S) - \min(S)}{|S| - 1}.
\]

Intuitively, this is the average distance between pairs of successive elements of \( S \) in sorted order: there are \(|S| - 1\) such pairs, and the distances between all of them add up to \( \max(S) - \min(S) \). We say that \( x, y \in S \) are **close in** \( S \) if \( x \neq y \) and \(|x - y| \leq \text{ads}(S)\) — i.e., the distance between \( x \) and \( y \) is at most the average distance between successive elements of \( S \). Note that any set of at least two numbers must contain a pair of close numbers: this is for the same reason that it is impossible for all of you to receive an above-average grade! Also note that a pair of numbers that are close in \( S \) need not be successive in sorted order; nor is every pair of successive numbers necessarily close in \( S \).

Describe an algorithm that, given a set \( S \) of \( n \geq 2 \) numbers in arbitrary order, finds a pair of numbers in \( S \) that are close; your algorithm should run in \( O(n) \) time. Justify the correctness and running time of your algorithm.

**Hint.** Use divide-and-conquer, taking advantage of the following fact, which you should prove: Let \( p \) be any element of \( S \), \( S_0 = \{x \in S : x \leq p\} \) and \( S_1 = \{x \in S : x \geq p\} \). (Note that \( p \) belongs to both \( S_0 \) and \( S_1 \), and so \(|S_0| + |S_1| = |S| + 1\).) Let \( a_i = \text{ads}(S_i) \), for \( i \in \{0, 1\} \), and let \( m \in \{0, 1\} \) be such that \( a_m = \min(a_0, a_1) \). Then every close pair in \( S_m \) is a close pair in \( S \).