Computer Science C73
Scarborough Campus

September 28, 2022
University of Toronto

Homework Assignment #4
Due: October 5, 2022, 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 docucments 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.*[a]

• *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****.*

[a] "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) Suppose we have a two-dimensional array $A[1..n, 1..n]$ of numbers, every row and column of which is sorted. That is, $A[i, j] \leq A[i, j']$, for all $1 \leq i \leq n$ and all $1 \leq j \leq j' \leq n$; and $A[i, j] \leq A[i', j]$, for all $1 \leq j \leq n$ and all $1 \leq i \leq i' \leq n$. We want to determine whether a given number x is in A.

One way to do this is to perform binary search in each row, until we either find x is some row, or determine that x is not in any row. (Of course, we could equally well do binary search in each column instead.) This clearly takes $\Theta(n \log n)$ time, but exploits only the fact that rows are sorted, ignoring that the columns are also sorted.

In an attempt to devise an algorithm that exploits the order in both dimensions, we are led to the following divide-and-conquer idea: Divide up the square array into four roughly equal two-dimensional arrays; upper-left, upper-right, lower-right, and lower-left. Look at the "middle" element of the array (roughly the element $A[n/2, n/2]$ with appropriate rounding). Based on comparing this element with x, eliminate a part of the array where we can be certain that x cannot be, and search the rest of the array.

Describe in pseudocode a divide-and-conquer algorithm to search A based on the idea sketched in the preceding paragraph, justify its correctness, analyze its running time, and compare the running time of your algorithm to the running time of the algorithm that uses binary search for each row. You may assume

that n is a power of two. As in the analysis of the first algorithm, express the running time as a function of n, the number of rows and columns of the input A. (Note that the size of the input, assuming $O(1)$-space for each entry of A, is n^2, not n.)

To think about, but not to submit with your answer: There is a better algorithm than either of the above two. Can you find one?

Question 2. (10 marks) Let S be a set of at least two numbers, and let

$$\mathbf{ads}(S) = \frac{\max(S) - \min(S)}{|S| - 1}.$$

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 $|x - y| \leq \mathbf{ads}(S)$ — 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 a divide-and-conquer 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 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 $a_i = \mathbf{ads}(S_i)$, 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.