University of Waterloo
CS240 - Spring 2023
Assignment 3

Due Date: Wednesday June 21, 5pm

You should read and sign the Academic Integrity Declaration and submit it before the deadline of June 21; i.e.**read, sign, and submit AID02.txt now or as soon as possible** The agreement will indicate what you must do to ensure the integrity of your grade. If you are having difficulties with the assignment, course staff are there to help (provided it isn't last minute).

The Academic Integrity Declaration must be signed and submitted on time or the assessment will not be marked.

Please read `https://student.cs.uwaterloo.ca/~cs240/s23/assignments.phtml#guidelines` for guidelines on submission. **Each question must be submitted individually to MarkUs as a PDF** with the corresponding file names: a3q1.pdf, a3q2.pdf, ... It is a good idea to submit questions as you go so you aren't trying to create several PDF files at the last minute.

Late Policy: Assignments are due at 5:00pm, with the grace period until 11:59pm. Assignments submitted after 11:59pm on the due date will not be accepted but may be reviewed (by request) for feedback purposes only.

Question 1 [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.

Question 2 [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 $T(n) \in o(n \log n)$ **without proving it**.

Question 3 [5+4=9 marks]

You have found a treasure chest filled with $n \geq 3$ coins. Exactly 2 coins are counterfeit, the rest are genuine. All genuine coins weigh the same and the two counterfeit coins weigh the same, but a counterfeit coin weighs less than a genuine coin. Your task is to separate the genuine coins from the counterfeit coins. To accomplish this you will compare the weight of pairs of subsets of the coins using a balance scale. The outcome of one weighing will determine that each subset of coins weighs the same, or that one or the other subset of coins weighs more.

a) Give a precise (not big-Omega) lower bound for the number of weighings required in the worst case to determine which coins are genuine and which are counterfeit for $n \geq 3$. Briefly justify.

b) Describe an algorithm called `FindGenuine` to determine the genuine coins when $n = 4$. Use the names C_1, C_2, C_3, C_4 for the four coins, and the function

$$\text{CompareWeight}\left(\{first\_subset; second\_subset\}\right),$$

which returns 1 if $first\_subset$ weighs more, 0 if both subsets weigh the same, and -1 if $second\_subset$ weighs more. Your function should return the set of genuine coins. Give an exact worst-case analysis of the number of weightings required by your algorithm. For full marks, this should match exactly the lower bound from Part (a).

Question 4 [3+2+4+1+1+3+3=17 marks]

In this problem, we study a variant of QuickSort, for which we hope to obtain a better worst-case bound. To simplify the discussion, we measure the runtime of our algorithm by only counting comparisons between elements. In what follows, you can use the fact that to sort n elements, insertion sort uses at most n^2 element comparisons.

We have to sort an array of size m, where we assume that m is a square (so $\sqrt{m}$ is an integer), and $m \geq 9$. We do not want to have to worry about breaking ties, so we assume that all entries in our array are pairwise distinct. We proceed recursively, and we call n the size of the array we receive in a given recursive call (so $n \leq m$). Here is the algorithm:

- If $n \leq 2\sqrt{m}$, we use insertion sort.
- Else, sort the first $2\sqrt{m} + 1$ entries of the array using insertion sort and let p be their median (as defined in module 03, slide 14/45). Then, we partition our array using p as pivot, and we continue as in the usual QuickSort algorithm.

Questions:

a) Show that for $n \leq 2\sqrt{m}$, the number of comparisons is at most λm, for some constant λ.

b) Show that if $n > 2\sqrt{m}$, both parts in the partition have size at least $\sqrt{m}$.

c) Suppose that the worst case for this algorithm happens when one part in the partition has the minimum possible size (this is true, but we are not asking you to prove it). Show that in the worst case, the number of comparisons satisfies a relation of the form

$$T(n) \leq T(n - (\sqrt{m} + 1)) + \lambda' m$$

for $2\sqrt{m} + 1 \leq n \leq m$, for some constant λ'.

d) Prove that

$$T(m) \leq T(m - (\sqrt{m} + 1)) + \lambda' m.$$

e) Let $k = \sqrt{m} - 2$. Prove that

$$m - (k-1)(\sqrt{m} + 1) \geq 2\sqrt{m} + 1$$

and

$$m - k(\sqrt{m} + 1) \leq 2\sqrt{m}.$$

f) Prove that

$$T(m) \leq T(m - k(\sqrt{m} + 1)) + k\lambda' m.$$

g) Using a) and f), deduce that $T(m) \in O(m^{3/2})$.