University of Waterloo
CS240 Spring 2026
Assignment 3

Due Date: Tuesday, June 30 at 5:00pm

Read `https://student.cs.uwaterloo.ca/~cs240/s26/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 30 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.

Reminder: all logarithms are in base 2 unless stated otherwise.

Question 1 [1+1+5 =7 marks]

Consider the following algorithm.

```
Mystery(A)
A: an array of size n ≥ 2 storing distinct numbers
1.  sum ← 0
2.  sorted ← true
3.  for i = 1 to n − 1
4.      if A[i − 1] > A[i]
5.          sorted ← false
6.  if sorted
7.      for i = 1 to 2^n
8.          sum ← sum + i ∗ sum
9.      return sum
10. if A[0] > A[1]
11.     for i = 1 to n^2
12.         sum ← sum + i ∗ sum
13.     return sum
14. return sum
```

a) What is the best-case running time? Use Θ notation. Briefly justify.

b) What is the worst-case running time? Use Θ notation. Briefly justify.

c) Derive the average-case running time. Use Θ notation.

Question 2 [2+2+4=8 marks]

Consider the algorithm below, where *random(k)* returns an integer from the set of $\{0, 1, 2, \ldots, k-1\}$ uniformly and at random.

```
ArrayAlg(A)
A: an array storing numbers
1.  if A.size == 1
2.      return
3.  i ← random(A.size)
4.  for j = 0 to i do
5.      print(A[j])
6.  ArrayAlg(A[0, ..., A.size − 2])
```

a) For an array A of size n, what is the fastest possible execution of $ArrayAlg(A)$? Justify.

b) For an array A of size n, what is the slowest possible execution of $ArrayAlg(A)$? Justify.

c) Let $T^{exp}(n)$ be the expected running time of $ArrayAlg$ an input of size n. Give a tight asymptotic bound for $T^{exp}(n)$. Justify your answer.

Question 3 Merging AVL Trees [3+4=7 marks]

a) Describe an algorithm `Join`(T_A, T_B, h_A, h_B) which merges two AVL trees T_A and T_B whose initial heights h_A and h_B are given. You may assume that:

(i) each node stores the height of the subtree rooted at it.

(ii) Every key in A is less than every key in B.

(iii) $h_A = h_B - 1$

Your algorithm must run in time $O(h_A + h_B)$. Justify that your algorithm produces a valid AVL tree and that it runs in the specified runtime.

b) Describe an algorithm `Join2`(T_A, T_B, h_A, h_B) as in part a), except that assumption (iii) is replaced by the following:

(iii) $h_A < h_B$

As in part a), justify that your algorithm produces a valid AVL tree and that it runs in $O(h_A + h_B) = O(h_B)$ time.

The following exercises are provided for additional practice and are not part of the graded portion of the assignment. While these questions will not be evaluated, we strongly encourage you, and expect you, to work through them to deepen your understanding of the material. You are welcome to upload your solutions to Crowdmark for your own record-keeping and organization; however, no grades or feedback will be provided for these submissions. You may also ask questions about these exercises during office hours or on Piazza.

Question 4

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.**

Question 5

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.

Question 6

We consider biased skiplists, where Heads come up with probability $p \in (0, 1)$, and Tails with probability $1 - p$. You can assume that the number of keys n is $n = 1/p^K$ for some integer K.

(a) Give an upper bound on the expected height of a biased skiplist (give a explicit inequality, not a big-O). Your upper bound should look like $g(n, p) + h(p)$, for some rather simple functions g, h and when $p = 1/2$, your answer should be similar to the one seen in class.

To analyze the expected cost of search, we are going to consider search paths backward (so either going left or going up at each step).

(b) Justify the following observation: if we follow a search path backward, then whenever we can go up, we must go up.

(c) Fix a level $i \geq 0$, let $-\infty < k_1 < \cdots < k_m < \infty$ be the keys at that level and consider the search path going backward from some k_j. For $j = 0, \ldots, m$, call E_j the expected number of left-steps the path does before going up.

Give a recurrence relation for E_j, and use it to derive a closed form expression for it. Alternative: you may get the closed form expression directly, without using a recurrence, if you favor a more brute-force kind of approach.

(d) Give an upper bound $E_j \leq f(p)$, for some very simple function $f(p)$

Let us use $P = \sum_{i=0}^{\text{expected height}-1}(f(p)+1)$ as a proxy for the expected cost of search, where the $+1$ accounts for the steps up.

(e) At this stage, you can write $P \leq A(n,p) + B(p)$, for some simple functions A, B. For a given n, what value of p minimizes the term $A(n,p)$?

Formulas you can use

- if $S_0 = 0$ and $S_{j+1} = aS_j + b$, $a \neq 1$, then $S_j = b\frac{1-a^j}{1-a}$
- $\sum_{j=0}^{N-1} ja^j = \frac{a - Na^N + (N-1)a^{N+1}}{1-a^2}$ for $a \neq 1$
- $\sum_{i=0}^{N} i^2 = N(N+1)(2N+1)/6$.