University of Waterloo
CS240 Winter 2023
Assignment 2

Due Date: Wednesday, February 15 at 5:00pm

If you are having difficulties with the assignment, course staff are there to help (provided it isn't last minute).

Please read `https://student.cs.uwaterloo.ca/~cs240/w23/assignments.phtml#guidelines` for guidelines on submission. **Each question must be submitted individually to MarkUs as a PDF** with the corresponding file names: a2q1.pdf, a2q2.pdf, ... , a2q6.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 11:59 on the due date will not be accepted but may be reviewed (by request) for feedback purposes only.

Notes:

- Logarithms are in base 2, if not mentioned otherwise.
- If you use MSD or LSD radix sort in this assignment, you are free to chose any value for radix R you wish. If you chose radix R other than 10 (standard decimal representation), you must account for the time it takes to transform a number in decimal representation to base R.
- Array indices start from 0.

Question 1 [2+2+3+3+5 = 15 marks]

a) Starting with an empty max-heap, construct the heap resulting from insertion of 21, 50, 18, 4, 25, 61, 100, in that order. Show the final heap. No explanation is necessary.

b) Let array $A = [6, 2, 10, 9, 4, 11, 27]$. Show the resulting array after applying heapify to A. No explanation is necessary.

c) Let A be an array containing numbers in increasing order. Give an in-place algorithm that modifies A so that it is a valid max-heap. Your algorithm should not perform any comparisons of elements in A, i.e. you cannot simply run heapify on A. Briefly justify your algorithm.

d) Let array H of size $2^k + 1$ store a permutation of integers in $\{1, .., 2^k + 1\}$, where k is an integer larger than 2. Array H is also a valid max-heap. What is the smallest key that can be stored in $H[3]$? Explain.

e) Write an algorithm that reports the k largest elements in heap H in $O(k \log k)$ time. You can use standard binary tree functions such as $left(v)$, $parent(v)$, etc. for a tree node v. You must provide both pseudocode and a brief description of your algorithm. Briefly justify the running time and correctness of your algorithm.

Question 2 [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. Your derivation must be based on the definition of the average-case running time (not on an equivalence of running time to some randomized version of $Mystery(A)$).

Question 3 [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) What is the best-case (i.e. the best luck) running time of *ArrayAlg* on an array A of size n? Justify.

b) What is the worst-case (i.e. the worst luck) running time of *ArrayAlg* on an array A of size n? Justify.

c) Let $T^{exp}(n)$ be the expected running time of *ArrayAlg* of size n. Show how to derive a recurrence relation for $T^{exp}(n)$ and then solve it. Express $T^{exp}(n)$ using big-O asymptotic notation. Your bound must be asymptotically tight, but you need not prove that it is tight.

Question 4 [4 marks]

You are given an array A of size n storing integers. Assume A does not store the integer 0. Write an algorithm $reorder(A)$ that reorders the elements of the array so that it first contains negative odd integers, then negative even integers, then positive odd integers, then positive even integers. For example, with an initial array $A = [47, 50, -78, -76, 7, -9]$, one possible correct output is $A = [-9, -78, -76, 47, 7, 50]$. There are other valid outputs; your algorithm needs to return any one of them. Your algorithm must be in-place and must run in $O(n)$ time. You must provide a detailed explanation of your algorithm in words. You can also provide pseudocode, if you wish, but it is not required. Briefly justify correctness and the running time.

Question 5 [4 marks]

Let $sortSemiSorted(A, k)$ be an algorithm with parameters A, an array of size n storing arbitrary numbers, and k, a positive integer. Furthermore, array A is such that the first k numbers are sorted, the next k numbers are sorted, and so on. For example, if $k = 4$, then $A = [6, 9, 10, 12, 1, 2, 5, 7]$ is a valid input. You can assume n is divisible by k. Algorithm *sortSemiSorted* sorts A. Prove $sortSemiSorted(A, k)$ must have running time $\Omega(n \log \frac{n}{k})$. You can use the fact that for a positive integer m, $(\frac{m}{e})^m \leq m! \leq m^m$.

Question 6 [2+2+3+3 = 10 marks]

a) Show the contents of array $A = [45, 112, 83, 8]$ after one round of LSD sort, where one round means one application of single digit bucket sort. No explanation is required.

b) Perform MSD sort on array $A = [736, 212, 213, 376, 354, 850]$. For each number, underline the digits (if any) which are not examined by MSD sort. You only need to underline the unexamined digits, nothing else, and no explanation is required for your answer.

c) Explain how to sort n numbers in the range $[0, n^5)$ in $O(n)$ time.

d) Let A be an array storing non-negative integers in radix R. The size of A is $n = R^b$ for some positive integer b, and the smallest number in A is larger than R^b. MSD sort on array A examines at most k digits for each number. What is the smallest value k can take? In other words, what is the smallest possible height of the recursion tree of MSD sort?