Midterm Exam, CS-250: Algorithms I, 2024

Do not turn the page before the start of the exam. This document is double-sided and has 8 pages. Do not unstaple.

- The exam consists of three parts. The first part consists of multiple-choice questions (Problem 1), the second part consists of a short open question (Problem 2), and the last part consists of three open-ended questions (Problems 3, 4, 5).
- For the open-ended questions, your explanations should be clear enough and in sufficient detail that a fellow student can understand them. In particular, do not only give pseudocode without explanations. A good guideline is that a description of an algorithm should be such that a fellow student can easily implement the algorithm following the description.
- You are allowed to refer to material covered in the lectures including algorithms and theorems (without reproving them). You are however *not* allowed to simply refer to material covered in exercises.

Good luck!

Problem 1: Multiple Choice Questions (35 points)

For each question, select the correct alternative. Note that each question has **exactly one** correct answer. Wrong answers are **not penalized** with negative points.

1a. Asymptotics *(7 points).* Let $f : \mathbb{N} \setminus \{0\} \to \mathbb{R}$ be a function from the positive integers to the reals. Which of the following implications does **not** hold?

A. If $f(n) = \log_2 n + \log_2(1 + \log_2 n)$ for all positive integers n, then $f(n) = O(\log_2^2 n)$.

B. If $f(n) = \log_2(n!)$ for all positive integers n, then $f(n) = \Theta(n \log_2 n)$.

C. If $f(n) = n$ for all positive integers n, then $f(n) = \Omega(1)$.

D. If $f(n) = n$ for all positive integers n, then $f(n) = O(1)$.

E. If $f(n) = \log_{10000} n$ for all positive integers n, then $f(n) = \Theta(\log_2 n)$.

1b. Heaps *(7 points).* Consider the array $A =$ | 7 | 10 | 6 | 9 | 4 | 1 | 2 | indexed from 1 to 7. Which of the following arrays is the result of calling BuildMaxHeap on A as shown in class? Recall that BuildMaxHeap works by sequentially calling MaxHeapify.

A. | 10 | 9 | 7 | 6 | 4 | 2 | 1 |

B. | 7 | 10 | 6 | 9 | 4 | 1 | 2 |

C. | 10 | 7 | 6 | 9 | 4 | 1 | 2 |

D. | 1 | 2 | 4 | 6 | 7 | 9 | 10 |

E. | 10 | 9 | 6 | 7 | 4 | 1 | 2 |

1c. Sorting *(7 points).* Consider the array $B =$ | 7 | 3 | 1 | 4 | indexed from 1 to 4. We sort it using insertion sort by calling InsertionSort(A, n) with $A = B$ and $n = 4$ (see below for pseudocode).

InsertionSort(A, n)

for $j = 2, \ldots, n$
    $k \leftarrow A[j]$
    $i \leftarrow j - 1$
    while $i > 0$ and $A[i] > k$
        $A[i+1] \leftarrow A[i]$
        $i \leftarrow i - 1$
    $A[i+1] \leftarrow k$
    print(A)

Which of the following sequence of arrays corresponds to the outputs printed in line 8?

A. [3 | 7 | 1 | 4], [1 | 3 | 7 | 4], [1 | 3 | 4 | 7]

B. [1 | 7 | 4 | 3], [1 | 7 | 3 | 4], [1 | 3 | 4 | 7]

C. [7 | 3 | 4 | 1], [7 | 4 | 3 | 1], [7 | 4 | 3 | 1]

D. [3 | 7 | 1 | 4], [1 | 7 | 3 | 4], [1 | 3 | 4 | 7]

E. [7 | 4 | 3 | 1], [7 | 4 | 3 | 1], [7 | 4 | 3 | 1]

1d. Time analysis *(7 points).* Consider the following algorithms.

Algorithm I: it takes as input a positive integer n and an array A of n integers, performs operations that run in $\Theta(1)$ time in total, computes $q = \lfloor n/3 \rfloor$, calls itself recursively on $A[1, \ldots, q]$, then performs other operations that run in $\Theta(1)$ time in total, and finally returns.

Algorithm II: it takes as input a positive integer n and an array A of n integers, performs operations that run in $\Theta(1)$ time in total, computes $q = \lfloor n/3 \rfloor$, calls itself recursively on $A[1, \ldots, q]$ and $A[n-q+1, \ldots, n]$, then performs other operations that run in $\Theta(1)$ time in total, and finally returns.

Algorithm III: it takes as input a positive integer n and an array A of n integers, performs operations that run in $\Theta(n)$ time in total, computes $q = \lfloor n/3 \rfloor$, calls itself recursively on $A[1, \ldots, q]$, $A[q+1, \ldots, 2q]$, $A[n-q+1, \ldots, n]$, then performs other operations that run in $\Theta(n)$ time in total, and finally returns.

Which of the following statements holds? *Hint:* $\log_3 2 \approx 0.63$.

A. Algorithms I, II, III run in time $\Omega(n^{1/3})$, $O(n^{2/3})$, $\Theta(n^{3/4})$ respectively.

B. Algorithms I, II, III run in time $\Theta(\log n)$, $\Omega(\sqrt{n})$, $O(n \log n)$ respectively.

C. Algorithms I, II, III run in time $\Theta(\log n)$, $\Theta(\log n)$, $\Theta(n \log n)$ respectively.

D. Algorithms I, II, III all run in time $\Theta(n \log n)$.

E. Algorithms I, II, III run in time $\Theta(\log n)$, $\Theta(n \log n)$, $\Omega(n^{3/2})$ respectively.

1e. Data Structures *(7 points).* Let S be a stack and Q be a queue, initially empty. Consider the following sequence of operations on S and Q:

Push(S, 1),
Enqueue(Q, 4),
Push(S, 1),
Push(S, 4),
Enqueue(Q, Pop(S)),
Enqueue(Q, Pop(S)),
Push(S, Dequeue(Q)).

We recall that $\textsc{Pop}$ and $\textsc{Dequeue}$ also return the item that they removed from the stack and queue respectively. Which of the following statements about S and Q holds true after having run the sequence of operations above?

A. The output of $\textsc{Dequeue}(Q), \textsc{Pop}(S)$ is $4, 1$ (in order).

B. The output of $\textsc{Dequeue}(Q), \textsc{Dequeue}(Q)$ is the same as $\textsc{Pop}(S), \textsc{Pop}(S)$ (in order).

C. The output of $\textsc{Pop}(S)$ is 1.

D. The stack S contains only 1's and the queue Q contains 1's and 4's.

E. The stack S contains 1's and 4's and the queue Q contains only 4's.

Problem 2: Magical Computation (10 points)

Consider the following procedure UNKNOWN that takes as input an array $A[\ell \ldots r]$ of $n = r - \ell + 1$ numbers with the left-index ℓ and the right-index r:

```
UNKNOWN(A, ℓ, r)

if ℓ > r
        return 0
elseif ℓ = r
        return A[ℓ]
else
        p ← ℓ + ⌊(r−ℓ)/4⌋
        q ← r − ⌈(r−ℓ)/4⌉
        Term1 ← UNKNOWN(A, ℓ, p)
        Term2 ← UNKNOWN(A, p + 1, q)
        Term3 ← UNKNOWN(A, q + 1, r)
        return Term1 + Term2 + Term3
```

2a. *(5 points)* Let $A[1 \ldots 8] =$

3	7	5	5	2	3	1	9

. What does a call to UNKNOWN(A, 1, 8) return?

2b. *(5 points)* Let $T(n)$ be the time it takes to execute UNKNOWN(A, ℓ, r) where $n = r - \ell + 1$ is the number of elements in the array. Give the recurrence relation of $T(n)$.

Problem 3: Searching in a mountain (20 points)

Consider an array A containing n distinct integers. It is indexed starting at 1, i.e. its first element is $A[1]$ and last element is $A[n]$.

The array has the following structure: there is an unknown index k $(1 < k < n)$ such that the array is increasing until its k^{th} element and decreasing afterwards. More precisely $A[1] < A[2] \ldots < A[k]$ and $A[k] > A[k+1] > \ldots > A[n]$. For example, the array $[8, 9, 13, 10, 4]$ satisfies the property for $k = 3$.

You need to design and analyze an algorithm that searches for an integer x in the array A. If x is in A then you should return the index of x in A, otherwise return -1.

Your algorithm must run in $O(\log n)$ time.

(You will receive partial points (10 out of 20) if you design an algorithm with $O(\log n)$ running time assuming the index k is given to you.)

Problem 4: Finding the highest peaks (15 points)

Alice, who recently relocated to Switzerland, thrives on adventure and hiking challenges. Eager to take on the hardest challenges, she sets out to identify highest mountains of Switzerland. Armed with an array containing the heights of all N Swiss mountains, she faces a daunting task due to Switzerland's abundance of peaks. Alice seeks an efficient algorithm to determine the heights of the $\sqrt{N}$ highest mountains. As an experienced algorithm designer, she looks for a solution that runs in $O(N)$ time complexity. Can you assist her in this endeavor? Your task is to design and analyze an algorithm which, given an array of N positive integers, outputs the $\sqrt{N}$ highest numbers and runs in $O(N)$ time.

For your convenience you can assume that $\sqrt{N}$ is an integer.

Problem 5: Mirror mirror on the wall (20 points)

For a binary tree T, the mirror tree $M(T)$ is another binary tree where the children of all non-leaf nodes are interchanged. In other words, the left child of every node becomes its right child and the right child becomes its left child. An example of a tree and its mirror tree can be seen in Figure 1.

Figure 1.

Input Tree: T **Mirror: M(T)**

Your task is to design and analyze an algorithm that, given a binary tree of n nodes, outputs its mirror image in time $O(n)$.