CS4231: Analysis of Algorithms I, Fall 2020

Midterm Exam, September 29-30

This exam contains 5 problems, some of them composed of several parts. There are 100 points in all, and you have 75 minutes plus 15 minutes to upload your answers (total 90 minutes). Do not spend too much time on any problem. Read them all through first and attack them in the order that allows you to make the most progress.

You can use any algorithm that we covered in class by simply referring to it and specifying the input. You do not need to repeat the algorithms.

Be *clear, precise and succinct* in your answers. You will be graded not only on the correctness of your answers, but also on the clarity with which you express them.
Be neat.

Please follow the posted guidelines for the exam. You can only use the allowed resources. No collaboration is allowed.
Write your name and UNI and upload your answers within the allotted time.

Good luck!

Problem 1 [12 points, 6 points per part]

Give asymptotic solutions $T(n) = \Theta(f(n))$ for the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. All logarithms are with base 2.
Justify briefly your answers.

1. $T(n) = 4T(n/2) + n^{1.5}$

2. $T(n) = T(n-1) + \sqrt{n}$

Problem 2 [16 points, 4 points per part]
For each pair of functions f, g below, determine whether $f = o(g)$, $f = \Theta(g)$, or $f = \omega(g)$. No justification is needed. All logarithms are with base 2.

1. $f(n) = 8n^2 + 4n + 12$
   $g(n) = (n+3)(n - \log n)$

2. $f(n) = 6^{\log n}$
   $g(n) = n^3$

3. $f(n) = 2^n$
   $g(n) = \sum_{i=1}^{n} 2^i$

4. $f(n) = n^{\sqrt{n}}$
   $g(n) = 2^n$

Problem 3 [30 points, 6 points per part]

For each of the claims below state whether the claim is True or False. In each case justify carefully your answer, by providing a convincing argument why the claim is true or false. Your justifications are more important than the True/False designations, and must be ***succinct, precise and convincing***.
All the parts can be adequately answered and justified in a few lines.

All the times below refer to worst-case time.

1. Suppose that we modify the Partition routine in Quicksort so that it picks as the pivot element x around which it partitions a subarray $A[p\ldots r]$ the middle element of the subarray, i.e. $x = A[\lceil (p+r)/2 \rceil]$.

Claim: The modified Quicksort algorithm runs in $O(n \log n)$ time in the worst case if the input elements are distinct.

2. *Claim*: If the array H[1…n] is a max-heap, then the minimum element of H appears in one of the positions $\lceil n/2 \rceil, \ldots, n$ (i.e. there is an index i with $\lceil n/2 \rceil \le i \le n$ such that H[i] is the minimum element of the array H).

3. *Claim*: There is a binary search tree T which contains a path starting at the root of T such that the keys of the nodes along the path are 10, 6, 8, 4, 7 in this order (i.e. 10 is the key at the root).

4. Given a set S of n pairs of integers (i,j), where $i,j \in \{1,\ldots,n\}$, we wish to find all pairs (i,j) in the set S such that the reverse pair (j,i) is also in S.

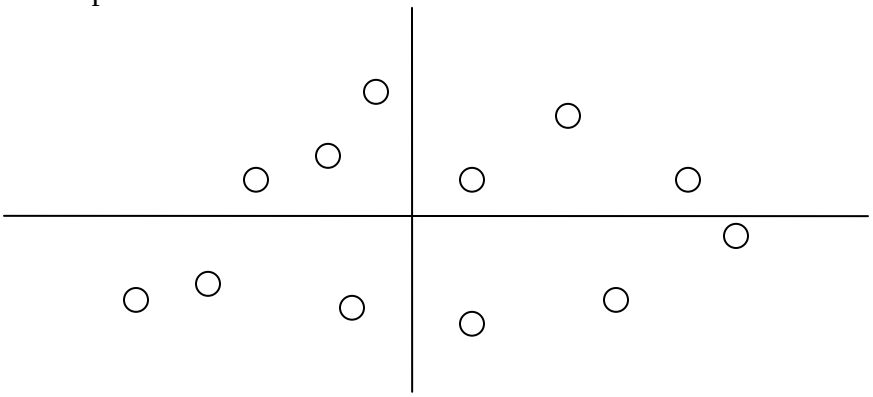
Claim: This task can be done in O(n) worst-case time and space.

5. Given a sorted array A with n elements, and another element x, we want to determine whether x appears in A, and if so, find an index i such that $A[i]=x$.

Claim: This task requires at least log$n$ comparisons in the comparison-based model (i.e., no algorithm can do this in fewer comparisons).

Problem 4. [16 points]

We are given a set *S* of $n=4m$ points on the plane with (real-valued) coordinates $p_1 = (x_1, y_1), \cdots, p_n = (x_n, y_n)$. No two of the points lie on the same horizontal or vertical line, i.e. $x_i \neq x_j$ and $y_i \neq y_j$ for all $i \neq j$. Let us say that the set *S* is *partitionable* if there is a vertical and a horizontal line that partition the plane into four parts, each of which contains *m* points of *S*.



1. [4 points] Give an example of a set of 4 points that is not partitionable.

2. [12 points] Give an algorithm that determines whether a given set *S* is partitionable. Make your algorithm as efficient as you can; the grading will depend on the efficiency of your algorithm. Justify the correctness of your algorithm.

Problem 5 [26 points]

Consider the following algorithm.

Input: (Unsorted) array $A[1\ldots n]$ of n distinct numbers
Output: a number s

```
s = 0;
for i=1 to n-1 do
    for j= i+1 to n do
        if ( A[i] > A[j] )  then s = s+1;
return s
```

1. [5 points] Give the (worst-case) time complexity of this algorithm in $\Theta(.)$ form. Justify your answer.

2. [8 points]
Suppose that the input array A has n distinct numbers in random order, where all permutations have the same probability. What is the expected value of the output s? Justify your answer.

3. [13 points]
Give a more efficient algorithm that computes the same output as the above algorithm for every input array A. You do not have to give pseudocode; clear, precise description in English suffices. State explicitly the (worst-case) running time of your algorithm, and justify its correctness and the running time.
(*Note*: Part 3 does not depend on part 2. There are several ways to design a more efficient algorithm. One way uses divide and conquer as in Merge Sort. Another way uses a suitable data structure.)