CS161 Midterm Exam

Do not turn this page until you are instructed to do so!

Instructions: Solve all questions to the best of your abilities. Please pay attention to the instructions at the beginning of each section, which provide guidance about the sort of answer we are expecting. Make sure to look at all pages. You may cite any result we have seen in class or CLRS (or in any resource linked from the course website, except Piazza) without proof. You have **80 minutes** to complete this exam. You may use one two-sided sheet of notes that you have prepared yourself. You may not use any other notes, books, or online resources. There is one blank page at the end that you may tear off as scratch paper, and one blank page for extra work. Please write your name at the top of all pages.

The following is a statement of the Stanford University Honor Code:

1. *The Honor Code is an undertaking of the students, individually and collectively:*

   (1) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

   (2) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

2. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*

3. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

By signing your name below, you acknowledge that you have abided by the Stanford Honor Code while taking this exam.

Signature: _____

Name: _____

SUNetID: _____

Question	#1	#2	#3	#4	#5	Total
Score						
Maximum	20	20	10	20	30	100

1 True or False (20 points)

Please answer all questions in this section True or False. No explanation is required.

1.1. (2 pts) A function can be both $O(n)$ and $O(n^2)$.

True **False**

1.2. (2 pts) RadixSort is a comparison-based sorting algorithm.

True **False**

1.3. (2 pts) Computing the median of n elements requires $\Omega(n \log(n))$ time for any comparison-based algorithm.

True **False**

1.4. (2 pts) Consider the following task: given n bits $b_1, \ldots, b_n \in \{0, 1\}$, decide if the number of b_i so that $b_i = 1$ is even or odd. This task requires time $\Omega(n)$.

True **False**

1.5. (2 pts) It takes time $O(1)$ to sort an array of 100 elements.

True **False**

1.6. (2 pts) The Master Theorem applies to the recurrence $T(n) = T(n/2) + T(n/10) + 7$.

True **False**

1.7. (2 pts) Consider the recurrence relation $T(n) = T(n/2) + T(n/20) + O(1)$, where $T(m) = 1$ for all $m \leq 20$. Then $T(n) = \Omega(n^2)$.

True **False**

1.8. (2 pts) Consider the recurrence relation $T(n) = 2T(n/2) + \sqrt{n}$, where $T(m) = 1$ for all $m \leq 2$. Then $T(n) = \Theta(n)$.

True **False**

1.9. (2 pts) A Red-Black Tree on n nodes always has depth $O(\log(n))$.

True **False**

1.10. (2 pts) For a given input array A, there is a nonzero probability that the running time of QuickSort with randomly chosen pivots will be $\Omega(n^2)$. However, there is **no** input array A on which QuickSort with randomly chosen pivots will always (with probability 1) take time $\Omega(n^2)$.

True **False**

2 Short answers (20 points)

For all of the problems in this section, please answer with **at most a few sentences.**

2.1. (5 pts) **(QuickSort v. MergeSort)** QuickSort is very efficient in practice, and has an expected running time $O(n \log(n))$, the same as that of MergeSort. Why would anyone ever use MergeSort over QuickSort?

2.2. (5 pts) **(MergeSort v. RadixSort)** RadixSort runs in time $O(n)$, while MergeSort requires time $\Omega(n \log(n))$. Why would anyone ever use MergeSort over RadixSort?

2.3. (5 pts) **(BFS and DFS)** Give one application of Breadth-First Search and one different application of Depth-First Search.

2.4. (5 pts) **(Hash families)** Let $h : \{1, \ldots, 10\} \to \{0, 1\}$ be the function $h(x) = x \bmod 2$. Your friend claims that $\mathcal{H} = \{h\}$ is a one-element universal hash family, which hashes a 10-element universe into $n = 2$ buckets. They give the following reasoning. Suppose that x and y are drawn independently and uniformly at random from $\{1, \ldots, 10\}$. Then the probability that $h(x) = h(y)$ is $1/2$. Thus, $\Pr\{h(x) = h(y)\} \leq 1/n$, which is the definition of a universal hash family. Your friend has made a big conceptual error: what is it?

3 Fun with big-O (10 points)

3.1. (5 points) Prove formally, using the definition of asymptotic notation that we saw in class, that if $f(n) = n$ and $g(n) = n^2$, then $f(n) = O(g(n))$.

3.2. (5 points) Consider the following claim:

If $f(n) = \Omega(g(n))$, then $2^{f(n)} = \Omega(2^{g(n)})$.

This claim is **false**. Give a counter-example to show that it is false. You do not need to formally prove that your counter-example is a counter-example.

4 Algorithm Design (20 points)

Suppose that A and B are two **sorted** arrays of comparable items of length n. A has distinct elements, and B has distinct elements, but there may be elements that are in both A and B. Give a deterministic algorithm with worst-case runtime $O(n)$ that finds the *intersection* of A and B — that is, your algorithm should return all of the elements that are in both A and B.

You should write pseudocode and a (short, high-level) English description of what your algorithm does. **You do not need to prove that it is correct, or analyze the running time.**

5 Algorithm Analysis (30 points)

Suppose that A is a **sorted** array of **distinct integers.** A *fixed point* of an array is an index $i \in \{1, \ldots, n\}$ so that $A[i] = i$. For example, in the array $A =$`[-2,-1,3,5,7]`, $A[3] = 3$, so 3 is a fixed point. In the array $B =$`[2,3,4,5,6]`, there are no fixed points.

The goal of the following algorithm is to return `True` if there is a fixed point, and to return `False` otherwise.

```
isThereAFixedPoint(A):
    return isThereAFixedPoint_helper(A, 1, n)

isThereAFixedPoint_helper(A, lower, upper):
    mid = (lower + upper)/2
    if A[mid] == mid:
        return True
    if lower == upper:
        return False
    if A[mid] > mid:
        return isThereAFixedPoint_helper(A, lower, mid)
    if A[mid] < mid:
        return isThereAFixedPoint_helper(A, mid+1, upper)
```

Above, all arithmetic is integer arithmetic: that is, `3/2` rounds down to `1`.

5.1. (3 pts) Suppose the pseudocode above runs on the input array `[-2,-1,2,4,7]`. What are all of the calls to `isThereAFixedPoint_helper`?

[More parts on next page]

Algorithm analysis continued.

5.2. (10 pts) Analyze the worst-case runtime of `isThereAFixedPoint`. We are looking for a statement of the form "the worst-case running time of `isThereAFixedPoint` on an input of size n is $O(\_\_\_\_)$," along with justification for why this is correct.

Your answer should be the strongest statement you can make (that is, a bound of $O(2^{2^n})$ may be true but will not receive credit), although you do not have to prove this.

5.3. (10 pts) Suppose you were to argue by induction that `isThereAFixedPoint` is correct: that is, it returns `True` if and only if there is some $i \in \{1, \ldots, n\}$ so that $A[i] = i$. Lay out the high level overview of the argument: what inductive hypothesis would you use, what is the base case, what needs to be shown for the inductive step, and what is the conclusion? **You should prove the base case and the "conclusion/termination" step. You do not need to prove the inductive step in this part.**

[More space on next page]

Algorithm analysis continued.

(Extra space for Question 5.3 if needed).

[Another part on next page]

Algorithm analysis continued.

5.4. (7 pts) Prove the inductive step in the outline you laid out above. If your proof breaks into two cases where the proof is basically the same, you may prove the result in only one case and write "the other case is basically the same."

This is the end!

This page intentionally blank for extra space for any question.
Please indicate in the relevant problem if you have work here that you want graded, and label your work clearly.

This page is for **scratch space**. You may tear off this page. Nothing on this page will be graded.

This page is for **scratch space**. You may tear off this page. Nothing on this page will be graded.