CS3000: Algorithms & Data — Summer 2023 — Laney Strange

Homework 1 - Long
Due Tuesday May 16 @ 9pm Gradescope

Name:
Collaborators:

- Put your name on the first page. If you are using the LaTeX template we provided, then you can make sure it appears by filling in the `yourname` command.
- This assignment is due Tuesday May 16 @ 9pm Gradescope. You may submit up to 48 hours late for no penalty, but expect a delay in grading.
- Show ALL your work, even if the problem doesn't specify it.
- You'll have an opportunity to resubmit two homeworks (one long, one short) at the end of the semester.
- Solutions must be typeset, preferably in LaTeX. If you need to draw any diagrams, you may draw them by hand as long as they are embedded in the PDF. I recommend using the source file for this assignment to get started.
- I encourage you to work with your classmates on the homework problems. *If you do collaborate, you must write all solutions by yourself, in your own words.* Do not submit anything you cannot explain. Please list all your collaborators in your solution for each problem by filling in the `yourcollaborators` command.
- If you get stuck on a homework problem, come by office hours or post on Piazza! We recommend you spend about 30 minutes trying to figure out a problem, and then ask for help. We'll be happy to clarify material from class and algorithm concepts, but we will not give out solutions or confirm your answers are correct.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class, is strictly forbidden.

Problem 1. *Mystery Algorithm One (2 + 4 + 2 = 8 points)*

You encounter the following pseudocode, and you can assume it calls the procedure LINEARSEARCH implemented as we covered in class.

```
MYSTERY(A)
1  for i = 1 to A.length
2      if LINEARSEARCH(A, i) == −1
3          return i
4  return NIL
```

(a) Assume that the algorithm accepts an array of distinct integers. In English, describe what this algorithm is trying to accomplish.

Solution:

(b) Formally show the worst-case run-time $T(n)$ of the Mystery Algorithm and give a tight bound, assuming that each single operation takes 1 step (but also account for the run-time of Linear Search!). As a first step, fill in the two rightmost columns of the table below.

line	cost	times
1		
2		
3		
4		

Solution:

(c) Give an example of A that would result in the best-case run time. Give a bound on that run-time.

Solution:

Problem 2. *Mystery Algorithm Two (4 + 4 = 8 points)*

You encounter the following pseudocode:

MYSTERY(a, n)

```
1  if n == 1
2        return (1, a)
3  elseif n == 2
4        return (a, a · a)
5  elseif n is odd
6        (u, v) = MYSTERY(a, ⌊(n+1)/2⌋)
7        return (u · u, u · v)
8  elseif n is even
9        (u, v) = MYSTERY(a, ⌊(n+1)/2⌋)
10       return (u · v, v · v)
```

(a) What would this function return in the follow examples, assuming a is any integer?

- MYSTERY$(a, 1)$?

  Solution:

- MYSTERY$(a, 2)$?

  Solution:

- MYSTERY$(a, 3)$?

  Solution:

- MYSTERY$(a, 4)$?

  Solution:

(b) What does MYSTERY do in general, when given any integer a and an integer $n > 0$? Prove your assertion by induction on n.

Solution:

Problem 3. *Recurrence (4 + 4 = 8 points)*

(a) Use the iteration method to solve the recurrence $T(n) = 2T(n/2) + n + c$, where c is a constant, with base case T(2) = 1.

Solution:

(b) Give a bound on the recurrence.

Solution:

Problem 4. *Recursive Selection Sort (4 + 2 = 6 points)*

In class this week, we studied Selection Sort as an iterative sorting algorithm. It can also be implemented in a recursive way, where we (iteratively) find the minimum element and place it in position $A[i]$, then recursively sort $A[i+1....n]$.

(a) Write the pseudocode for recursive Selection Sort

Solution:

(b) Give a recurrence for this algorithm's worst-case run-time. (Remember to include the base case!)

Solution:

Problem 5. *Binary Search* $(1 + 1 + 1 + 2 + 2 = 7$ *points)*

In class, the version of Binary Search we learned used a Binary Search Tree. It can also be implemented with the data stored in a sorted array, and it has the same run-time for the Search operation.

Binary Search on a sorted array works by finding the element in the middle and comparing that to the key. We compare the key to the midpoint element and discard either the left or right half of the array. (If the array has an odd number of elements, then there is only one choice for the midpoint; if it has an even number, there are two possible midpoints and we pick the leftmost one.)

There are tradeoffs to binary search with a BST vs. binary search with a sorted array, and for this problem we'll do a little comparison.

(a) What is the midpoint of the array $< 1, 4, 5, 6, 16, 20, 21, 23, 24] >$, assuming we care about the whole array?

Solution:

(b) What is the midpoint of the array $< 1, 4, 5, 6, 16, 20, 21, 23, 24 >$, assuming we care only about the subarray from position 6 through position 9 inclusive?

Solution:

(c) In general, given an array A where we care about the subarray from position l to position r, give a formula for the midpoint that will work no matter how many elements in the array.

Solution:

(d) The worst-case run-time on an insertion operation into a balanced Binary Search Tree is $\Theta(\lg n)$. Give a bound on the worst-case run-time on an insertion operation into a sorted array.

Solution:

(e) Binary Search requires our data to be in some kind of sorted order. If we start with an unsorted array of length n, we could sort it one of two ways: (1) sort the array using a $\Theta(n \lg n)$ sorting algorithm, or (2) turn the array into a BST by repeatedly calling its insert operation. Give a bound on the worst-case run-time to turn the unsorted array into a BST.

Solution: