Name: $\boxed{\text{Michael Rogers}}$
ID: $\boxed{105667404}$

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Problem Set 1 **Spring 2019, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Hyperlinks for convenience:	1a 1b 1c 1d 1e 1f 1g 1h 1i 1j	2a 2b 2c 2d 2e
	3a 3b	4a 4b

1. *(10 pts total) For each of the following claims, determine whether they are true or false. Justify your determination (show your work). If the claim is false, state the correct asymptotic relationship as O, Θ, or Ω.*

   (a) $n^2 + 2n - 4 = \Omega(n^2)$

   By L'Hospital's Rule:

   $$\lim_{n\to\infty} \frac{n^2+2n-4}{n^2} = \lim_{n\to\infty} \frac{2n+2}{2n} = \lim_{n\to\infty} \frac{2}{2} = 1$$

   This means $n^2 + 2n - 4 = O(n^2)$ by the limit test. If we reverse this we get:

   $$\lim_{n\to\infty} \frac{n^2}{n^2+2n-4} = \lim_{n\to\infty} \frac{2n}{2n+2} = \lim_{n\to\infty} \frac{2}{2} = 1$$

   Which means $n^2 = O(n^2 + 2n - 4)$.

   This means that $n^2 + 2n - 4 \geq n^2$ which implies $n^2 + 2n - 4 = \Omega(n^2)$ Thus proving the statement true.

(b) $10^{100} = \Theta(1)$

Big-O:

$$\lim_{n\to\infty} \frac{10^{100}}{1} = 10^{100} \Rightarrow 10^{100} = O(1)$$

Big-Ω:

$$\lim_{n\to\infty} \frac{1}{10^{100}} = \frac{1}{10^{100}} \Rightarrow 10^{100} = O(10^{100}) \Rightarrow 10^{100} = \Omega(1)$$

Since Big-O and Big-Ω are equal, then the means that $10^{100} = \Theta(1)$

(c) $\ln^2 n = \Theta(\lg^2 n)$

Using the change of base formula, $\lg^2 n$ becomes $\frac{\ln^2 n}{\ln^2(2)}$
Now using the limit test to find Big-O:

$$\lim_{n\to\infty} \frac{\ln^2 n}{\frac{\ln^2 n}{\ln^2(2)}} = \lim_{n\to\infty} \frac{(\ln^2 n)(\ln^2(2))}{\ln^2 n} = \lim_{n\to\infty} \ln^2(2) = \ln^2(2) \Rightarrow \ln^2 n = O(\lg^2 n)$$

Big-Ω:

$$\lim_{n\to\infty} \frac{\frac{\ln^2 n}{\ln^2(2)}}{\ln^2 n} = \lim_{n\to\infty} \frac{\ln^2 n}{(\ln^2 n)(\ln^2(2))} = \lim_{n\to\infty} \frac{1}{\ln^2(2)} = \frac{1}{\ln^2(2)} \Rightarrow \ln^2 n = \Omega(\lg^2 n)$$

Since $\ln^2 n = O(\lg^2 n) = \Omega(\lg^2 n)$

Then $\ln^2 n = \Theta(\lg^2 n)$

(d) $2^n = \Theta(2^{n+7})$

I don't know

(e) $n + 1 = O(n^4)$

I don't know

(f) $1 = O(1/n)$

I don't know

(g) $3^{3n} = \Theta(9^n)$

I don't know

(h) $2^{2n} = O(2^n)$

I don't know

(i) $2^{n+1} = \Theta(2^{n \lg n})$

I don't know

(j) $\sqrt{n} = O(\lg n)$

I don't know

2. *(15 pts) Professor Dumbledore needs your help optimizing the Hogwarts budget. You'll be given an array A of exchange rates for muggle money and wizard coins, expressed as integers. Your task is help Dumbledore maximize the payoff by buying at some time i and selling at a future time $j > i$, such that both $A[j] > A[i]$ and the corresponding difference of $A[j] - A[i]$ is as large as possible.*

For example, let $A = [8, 9, 3, 4, 14, 12, 15, 19, 7, 8, 12, 11]$. If we buy stock at time $i = 2$

with $A[i] = 3$ *and sell at time* $j = 7$ *with* $A[j] = 19$*, Hogwarts gets in income of* $19 - 3 = 16$ *coins.*

(a) *Consider the pseudocode below that takes as input an array* A *of size* n*:*

```
makeWizardMoney(A) :
    maxCoinsSoFar = 0
    for i = 0 to length(A)-1 {
        for j = i+1 to length(A) {
            coins = A[j] - A[i]
            if (coins > maxCoinsSoFar) { maxCoinsSoFar = coins }
    }}
    return maxCoinsSoFar
```

What is the running time complexity of the procedure above? Write your answer as a Θ *bound in terms of* n*.*

I don't know

(b) *Explain (1–2 sentences) under what conditions on the contents of A the* `makeWizardMoney` *algorithm will return 0 coins.*

I don't know

(c) *Dumbledore knows you know that* `makeWizardMoney` *is wildly inefficient. With a wink, he suggests writing a function to make a new array M of size n such that*

$$M[i] = \min_{0 \leq j \leq i} A[j] \ .$$

That is, $M[i]$ gives the minimum value in the subarray of $A[0..i]$.

Write pseudocode to compute the array M. What is the running time complexity of your pseudocode? Write your answer as a Θ bound in terms of n.

I don't know

(d) *Use the array M computed from (2c) to compute the maximum coin return in time $\Theta(n)$.*

I don't know

(e) *Give Dumbledore what he wants: rewrite the original algorithm in a way that combine parts (2b)–(2d) to avoid creating a new array M.*

I don't know

Name: Michael Rogers
ID: 105667404

3. *(15 pts) Consider the problem of linear search. The input is a sequence of n numbers $A = \langle a_1, a_2, \ldots, a_n \rangle$ and a target value v. The output is an index i such that $v = A[i]$ or the special value NIL if v does not appear in A.*

   (a) *Write pseudocode for a simple linear search algorithm, which will scan through the input sequence A, looking for v.*

```
linSearch(A,v){
for i in A{
   if A[i] == v{
      return A[i]
   }
}
return NIL
```

(b) *Using a loop invariant, prove that your algorithm is correct. Be sure that your loop invariant and proof covers the initialization, maintenance, and termination conditions.*

I don't know

4. *(20 pts) Ron and Hermione are arguing about binary search. Hermione writes the following pseudocode on the board, which she claims implements a binary search for a target value* `v` *within input array* `A` *containing n elements.*

```
bSearch(A, v) {
   return binarySearch(A, 1, n-1, v)
}

binarySearch(A, l, r, v) {
   if l <= r then return -1
   p = floor( (l + r)/2 )
   if A[p] == v then return p
   if A[p] < v then
     return binarySearch(A, p+1, r, v)
     else return binarySearch(A, l, p-1, v)
}
```

(a) *Help Ron determine whether this code performs a correct binary search. If it does, prove to Hermione that the algorithm is correct. If it is not, state the bug(s), give line(s) of code that are correct, and then prove to Hermione that your fixed algorithm is correct.*

I don't know

Name: Michael Rogers
ID: 105667404

(b) *Hermione tells Ron that binary search is efficient because, at worst, it divides the remaining problem size in half at each step. In response Ron claims that four-nary search, which would divide the remaining array A into fourths at each step, would be way more efficient. Explain who is correct and why.*

I don't know

I just wanted to add that I did not leave enough time to do this problem set. I had never used LaTeX before and I underestimated how long it would take me to learn some syntax. It does not reflect on me as a student and I promise to manage my time more efficiently in the future. Also in terms of citing, I used this URL to get the equation for the change of base formula:
https://www.purplemath.com/modules/logrules5.htm