Name: Trevor Buck

ID: 109081318

CSCI 3104, Algorithms
Problem Set 1

Profs. Grochow & Layer
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) $n^2 + 2n - 4 = \Omega(n^2)$

   $2n-4 \geq 0$ for $n \geq 2$

   $\Rightarrow n^2+2n-4 \geq n^2$ for all $n \geq 2$

   $\boxed{\text{Thus } n^2+2n-4 = \Omega(n^2) \text{ for all } n \geq 2}$ → True

   *Note could also read: $n^2+2n-4 = \Theta(n^2)$

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

   This is a constant

   By definition, any constant is equal to '$\Theta(1)$'

   $\boxed{\text{Therefor} \rightarrow \text{True} \rightarrow 10^{100} = \Theta(1)}$

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

False $\rightarrow$ $\ln^2(n) \geq \log^2(n)$ for all $n \geq 2$

Thus the correct statement should read:

$\boxed{\ln^2 n = \Omega(\log^2 n)}$

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

False $\rightarrow$ $2^n \leq 2^{n+7}$ for all $n \geq 0$

Thus the correct statement should read:

$2^n = O(2^{n+7})$

OH, not theta

Name: Trevor Buck
ID: 109081318

CSCI 3104, Algorithms
Problem Set 1

Profs. Grochow & Layer
Spring 2019, CU-Boulder

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

$n \leq n^4$ for all $n \geq 0$

$1 \leq n^4$ for all $n \geq 1$

There for, $\boxed{\text{true}}$ $\rightarrow$ $n + 1 = O(n^4)$

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

$1 \geq 1/n$ for all $n \geq 1$

$\boxed{\text{False}}$ $\rightarrow$ statement should read

$$\boxed{1 = \Omega(1/n)}$$

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

False

$3^{3n} \geq 9^n$ for all $n \geq 0$

True statement: $\boxed{3^{3n} = \Omega(9^n)}$

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

False

$2^{2n} \geq 2^n$ for all $n \geq 0$ (exponentially)

True statement: $\boxed{2^{2n} = \Omega(2^n)}$

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

$\boxed{\text{False:}}$ $2^{n+1} \leq 2^{n \lg n}$ for all $n \geq 100$

True statement: $\boxed{2^{n+1} = O\left(2^{n \log n}\right)}$

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

$\boxed{\text{False:}}$ $\sqrt{n} \geq \log n$ for all $n \geq 1$

True statement: $\boxed{\sqrt{n} = \Omega(\log n)}$

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 {   → n times (n)
        for j = i+1 to length(A) {   → n times (n)
            coins = A[j] - A[i]
            if (coins > maxCoinsSoFar) { maxCoinsSoFar = coins }   ] constant (1)
    }}
    return maxCoinsSoFar
```

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

$\boxed{\text{running time} = \Theta(n^2)}$

$n \rightarrow$ first 'for' loop

$n \rightarrow$ second 'for' loop

$1 \rightarrow$ constant subtraction and comparison

total $= (n \times n \times 1) = n^2$

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

- If 'A' has length zero or length one, the 'for loops' won't run a single comparison, and 'zero' will be returned.
- Or if all items in the array have the same value, a zero will be returned

(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*

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

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.

```
minnie = 0
m[0] = A[0]
for j=1 to length(A) {
    if (A(j) < A(minnie) {
        m[j] = A[j]
        minnie = j }
    else {m[j] = m[minnie] }
```

run time = $\Theta(n)$

↑
only one for loop

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

```
maxcoins = 0
for k=0 to length(A)-1 {
    if (maxcoins < (A(k) - m(k))) {
        maxcoins = (A(k) - m(k))
    }
}
```

(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.*

```
minnie = A(0)
maxcoins = 0

for i=1 to length(A)-1 {
    if (A(i) < minnie) {minnie = A(i)}
    if (A(i) - minnie > maxcoins) {
        maxcoins = (A(i)-minnie)
}
return maxcoins
```

Name: Trevor Buck

ID: 109081318

CSCI 3104, Algorithms
Problem Set 1

Profs. Grochow & Layer
Spring 2019, CU-Boulder

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.*

```
index = NIL
for i=0 to length(A)-1 {
    if (A[i]==v) {
        index = i
    }
return index
```

Name: Trevor Buck
ID: 109081318

CSCI 3104, Algorithms
Problem Set 1

Profs. Grochow & Layer
Spring 2019, CU-Boulder

(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.*

Loop invarient: index equals 'NIL' or the value 'i' such that A[i] == V

Initialization: i = 0 ⟹ Because we set index equal to NIL, it is true before the iteration begins

Maitenance: index will have the value of 'NIL' until 'v' is located because the value of index only changes if the conditions are met

Termination: index will contain 'NIL' or the value 'i' such that A[i] == V

Name: Trevor Buck

ID: 109081318

CSCI 3104, Algorithms
Problem Set 1

Profs. Grochow & Layer
Spring 2019, CU-Boulder

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)
}
```

11 14 10 12 9

A 1 4 12 9 10 12 12 14

(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.*

bugs

- $(l \le r) \rightarrow$ always true if more than two elements
  - $\rightarrow$ should read: 'if $l < 0$ or $r > n$'
- $(A[p] < v) \rightarrow$ Assumes that the values are sorted
  - $\rightarrow$ fix: add a sort(A)
- (begin with calling binarysearch(A, 0, n-1, v)

```
bsearch (A,v) {
    Sort (A)
    return binary (A, 0, n-1, v)
}

binary (A, l, r, v) {
    if (l < 0 or r > n) return -1
    p = floor ( (l+r)/2 )
    if (A[p] == v) then return p
    if (A[p] < v) then return binary (A, p+1, r, v)
        else return binary (A, l, p-1, v)
}
```

Maintenance: we adjust either 'r' or 'l' depending on where the value A[p] is

Termination: we only end once we have checked the first or the last number or found the value

Loop invarient: if the value of 'v' exists, it rests in betwen A[l] and A[r]

Initialization: 'l' is set to the first indexed number, and 'r' is set to the last. This means it covers the entire array

Name: Trevor Buck
ID: 109081318

(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.*

Ron is correct. Because our only task is to find the right value, it would be faster to check 4 different numbers for every call to binary Search.

* Note: The sorting algorithm is the same in both cases and takes $\boxed{O(n \log n)}$. This is the dominant factor in the entire process and would therefor make the actual times similair no matter how many ways you divide the array later on in the code.