CSCI 3104, Algorithms **Profs. Clauset & Grochow**
Problem Set 1 **Spring 2018, 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.

---

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)	$n+1$	$=$	$O(n^4)$
(b)	2^{2n}	$=$	$O(2^n)$
(c)	2^n	$=$	$\Theta(2^{n+7})$
(d)	1	$=$	$O(1/n)$
(e)	$\ln^2 n$	$=$	$\Theta(\lg^2 n)$
(f)	n^2+2n-4	$=$	$\Omega(n^2)$
(g)	3^{3n}	$=$	$\Theta(9^n)$
(h)	2^{n+1}	$=$	$\Theta(2^{n \lg n})$
(i)	$\sqrt{n}$	$=$	$O(\lg n)$
(j)	10^{100}	$=$	$\Theta(1)$

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 at 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.

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

(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 $A[0..i]$.

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

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

(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.

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.

   (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.

4. (15 pts) Crabbe and Goyle are arguing about binary search. Goyle writes the following pseudocode on the board, which he 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 Crabbe determine whether this code performs a correct binary search. If it does, prove to Goyle 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 Goyle that your fixed algorithm is correct.

(b) Goyle tells Crabbe that binary search is efficient because, at worst, it divides the remaining problem size in half at each step. In response Crabbe 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.

5. (10 pts extra credit) You are given two arrays of integers A and B, both of which are sorted in ascending order. Consider the following algorithm for checking whether or not A and B have an element in common.

```
findCommonElement(A, B) :
    # assume A,B are both sorted in ascending order
    for i = 0 to length(A) {                    # iterate through A
        for j = 0 to length(B) {                # iterate through B
            if (A[i] == B[j]) { return TRUE }   # check pairwise
        }
    }
    return FALSE
```

(a) If arrays A and B have size n, what is the worst case running time of the procedure `findCommonElement`? Provide a Θ bound.

(b) For $n = 5$, describe input arrays A_1, B_1 that will be the best case, and arrays A_2, B_2 that will be the worst case for `findCommonElement`.

(c) Write pseudocode for an algorithm that runs in $\Theta(n)$ time for solving the problem. Your algorithm should use the fact that A and B are sorted arrays.
(Hint: repurpose the `merge` procedure from MergeSort.)