Homework 2 (problem set)

Requirements:

- Submit a .pdf file, all the solutions must be typed. Hand-written solutions will not be graded.
- Students can either work alone or in pairs. If you have a teammate, then only one of you need to submit.

Student name(s):

1. Consider the searching algorithm:

- Input: A sequence of *n* numbers A = a1 , a2 , . . . , an and a target value *v*.
- Output: the *first* index *i* such that v = A[i] or the special value *None* if v does not appear in A.

Write pseudocode for ***linear search***, which scans through the sequence, looking for v.

You can assume that the sequence A is not empty.

2. If the sequence A is pre-sorted, we can check the midpoint of the sequence against v and eliminate half of the sequence from further consideration. This is the idea of ***binary search***.

Write pseudocode for an iterative version or a recursive version of binary search algorithm.

Explain the exit condition(s) of your solution (when does your search stop?).

Explain why or why not this binary search algorithm can always find the *first* occurrence of the value *v* from the left to the right side.

3. Given the pseudocode of bubble sort below:

```
1 for i ← 1 to length[A]
2     do for j ← length[A] downto i + 1
3           do if A[j] < A[j - 1]
4                 then exchange A[j] ⇔ A[j - 1]
```

What is the worst case of this bubble sort?

What is the best case of this bubble sort?

In the worst case, what is the Big Θ?

In the best case, what is the Big Θ?

To prove Θ(f(n)), you need to find the T(n), the big-O and the big-Ω.

4. Consider n as a large enough value, which one is higher, n! or n^3? Prove it with induction. (you can start from the setup of the solution provided below)

When n = 6: $n! > n^3$

Assume that for k: $k! > k^3$

Prove: $(k+1)! > (k+1)^3$

5. Show that the solution of T (n) = T ([n/2]) + 4 is O(lg n). You can use either the substitution method or the iteration method. (assuming the base case is when n=1, the cost is constant)

6. Given the pseudocode below of a recursive function which finds the max of a sequence A.

This algorithm uses 1-based index (same to our textbook).

MAX1(A)

```
1 if len(A) ==1
2     return A[1]
3 else
4     max_remaining = MAX1(A[index 2 to end])
5     if max_remaining > A[1]
6           return max_remaining
7     else
8           return A[1]
```

What is the T(n) of this algorithm? What is the Θ of this algorithm? Prove it with the iterative method.

7. Given the pseudocode below of a recursive function which finds the max of a sequence A.

This algorithm uses 1-based index (same to our textbook).

MAX2(A)

```
1 if len(A) ==1
2     return A[1]
3 else
4     mid = ⌊len(A)/2⌋
```

```
5     max_left = MAX2(A[index 1 to mid])
6     max_right = MAX2(A[mid+1 to end])
7     if max_left > max_right
8           return max_left
9     else
10          return max_right
```

What is the T(n) of this algorithm? What is the Θ of this algorithm? Prove it with the Master Theorem.