INT102 Algorithmic Foundations
Problem Session 2, Week 4

Location: SC176

Question 1

Given the Bubble sort algorithm as below:

```
ALGORITHM BubbleSort(A[0..n − 1])
//Sorts a given array by bubble sort
//Input: An array A[0..n − 1] of orderable elements
//Output: Array A[0..n − 1] sorted in ascending order
for i=0 to n − 2 do
    for j = n-1 downto i+1 do
        if A[j ]<A[j-1 ] swap A[j ] and A[j - 1]
```

1. What is the number of swapping operations needed to sort the numbers A[0..5]=[6，1，2，3，4，5] in ascending order using the Bubble sort algorithm?
2. What is the number of key comparisons needed to sort the numbers A[0..5]= [6，1，2，3，4，5] in ascending order using the Bubble sort algorithm?

Question 2

Given the Merge sort algorithm as below:

```
Algorithm Mergesort(A[0..n-1])
  if n > 1 then begin
    copy A[0..⌊n/2⌋-1] to B[0..⌊n/2⌋-1]
    copy A[⌊n/2⌋..n-1] to C[0..⌈n/2⌉-1]
    Mergesort(B[0..⌊n/2⌋-1])
    Mergesort(C[0..⌈n/2⌉-1])
    Merge(B, C, A)
  End

Algorithm Merge(B[0..p-1], C[0..q-1], A[0..p+q-1])
        Set i=0, j=0, k=0
        while i<p and j<q do
        begin
                if B[i]≤C[j] then set A[k]=B[i] and increase i
                else set A[k] = C[j] and increase j
                k = k+1
        end
        if i==p then copy C[j..q-1] to A[k..p+q-1]
        else copy B[i..p-1] to A[k..p+q-1]
```

What is the number of key comparisons needed to sort the numbers A[0..5]= [6， 1， 2， 3， 4， 5] in ascending order using the Mergesort algorithm?

Question 3:

The time complexity of the merge sort algorithm can be described by the following recurrence for T(n).

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/2) + n & \text{if } n > 1 \end{cases}$$

In the lecture we have proved that *T(n) = O(n log n)* using the substitution method (i.e., using mathematical induction). Now prove that *T(n) = O(n log n)* using the iterative method (unfolding the recurrence). Assume that $n = 2^k$

Question 4

1. Write a pseudocode for a divide-and-conquer algorithm for finding a **position** of the largest element in an array of n numbers.

2. Set up and solve (for $n = 2^k$) a recurrence relation for the number of key comparisons made by your algorithm.

Question 5

1. Design a divide-and-conquer algorithm for finding values of both the largest and smallest elements in an array of *n* numbers.

2. Set up and solve (for $n = 2^k$) a recurrence relation for the number of key comparisons made by your algorithm.