Algorithms, Data Structures, and Problem Solving
(DA4002)

Group Activity
Complexity of Algorithms
October 3, 2016

---

1. Insertion Sort

Insertion sort is a simple sorting algorithm that builds a sorted list of n elements given in a random order. The algorithm somehow works similarly to the way people manually sort items. Basically, in each iteration, an item is inserted in the proper place into an (initially empty) list by comparing it with each item in the list until it finds the new element's successor or the end of the list. Here is the pseudo code of the insertion sort.

```
function INSERTION-SORT(A)
    for j ← 2 to length[A] do
        key ← A[j]
        Insert A[j] into the sorted sequence A[1, j − 1].
        i ← j − 1
        while i > 0 and A[i] > key do
            A[i + 1] ← A[i]
            i ← i − 1
        end while
        A[i + 1] ← key
    end for
end function
```

Considering the given pseudo code of the algorithm, discuss and try to find

1. the best, worst, and average cases of the algorithm, and
2. the average runtime of the algorithm (Big-O notation).

2. Merge Sort

Merge sort is a *Divide and Conquer* algorithm. It sorts a list of n elements, given in a random order, by dividing the input array in two halves, calling itself for the two halves, and then merging the two sorted halves. The **merge()** function, used for merging two halves, is the key process. It assumes that the inputs (say A1 and A2) are sorted arrays and merges them into one sorted array.

function MERGE-SORT(A)
  $n \leftarrow lenght[A]$
  if then$(n == 1)$
    return A
  end if
  $A1 = A[0]...A[n/2]$
  $A2 = A[n/2 + 1]...A[n]$
  $A1 = MERGE - SORT(A1)$
  $A2 = MERGE - SORT(A2)$
  return $merge(A1, A2)$
end function

Considering the given pseudo code of the algorithm, discuss and try to find

1. the best, worst, and average cases of the algorithm, and
2. the complexity of the merge sort (Big-O notation).