**week 4 exam:**

The two elementary sorting algorithms are insertion sort , selection sort.

**Insertion sort:**

In insertion sort for every iteration ,suppose if i=3 the first two elements are in sorted, and the newly added element (third element) finds its correct position and place it there.it is like playing cards in our hands, in there for every new card we place it in correct position by moving cards.time complexity is O(n^2) in worst case.if all are sorted it takes n only.

**Selection Sort:**

In selection sort first we thought array two parts sorted and unsorted and from beginning, we find first min and next min element and so on.time complexity is O(n^2).

**For test case 1:Half the data is 0s, half 1s.**

Input1: insertion :0.016

Selection:0.018

Input2: insertion:0.03

Selection:0.029  
Input3: insertion:0.044

Selection:0.1

**Observation:**

For only 0’s and 1’s input

In insertion sort for every new element it needs to check already sorted array for it’s position.

In selection sort for every new element it does not need to check in already sorted array. It finds the minimum in remaining elements and add it because already sorted element are must be less then new element. So here number comparisons are more compare to insertion.

Example:

00100011101010101101010111000010001111100101001101

After sorting :

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

**For testcase 2:Half the data is 0s, half the remainder is 1s, half the remainder is 2s, and so forth.**

Input1: insertion :0.015

Selection:0.016

Input2: insertion:0.031

Selection:0.035  
Input3: insertion:0.062

Selection:0.069

**Observation:**

In the above test case only 0’s and 1’s are there but here different numbers are more so number of comparisons are more so time complexity increases.

In quick sort even less or more repeated numbers it does not care because every time it finds the minimum element in unsorted array.

**Example:**

Before **:**

31000010000000100010220031212100201010031011340212

After sort:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 4

**For test case 3:Half the data is 0s, half random int values.**

Input1: insertion :0.014

Selection:0.015

Input2: insertion:0.017

Selection:0.018  
Input3: insertion:0.037

Selection:0.039

**Observation:**

Half the elements are 0’s and half are random due to this for second half to sort,number of comparisons takes more so time complexity is increases.

Insertion is a stable sort where selection is not stable because it finds minimum in this process it becomes unstable.

Example:

Before sort:

05001500231050015002318041000058041000500150023005

After sort:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 2 2 2 3 3 3 4 4 5 5 5 5 5 5 5 5 8 8