Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058, India
(Autonomous College Affiliated to University of Mumbai)

Experiment No.	1.b
Aim	Insertion sort and selection sort
Name	Vishaka Hole
UID No.	2021300043
Class & Division	Comps A(A3)
Date of experiment	06/02/23
Date of submission	13/02/23

Theory/Experiment:

Insertion sort works similar to the sorting of playing cards in hands. It is assumed that the first card is already sorted in the card game, and then we select an unsorted card. If the selected unsorted card is greater than the first card, it will be placed at the right side; otherwise, it will be placed at the left side. Similarly, all unsorted cards are taken and put in their exact place.

The same approach is applied in insertion sort. The idea behind the insertion sort is that first take one element, iterate it through the sorted array. Although it is simple to use, it is not appropriate for large data sets as the time complexity of insertion sort in the average case and worst case is O(n2), where n is the number of items. Insertion sort is less efficient than the other sorting algorithms like heap sort, quick sort, merge sort, etc.

In selection sort, the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. It is also the simplest algorithm. It is an in-place comparison sorting algorithm. In this algorithm, the array is divided into two parts, first is sorted part, and another one is the unsorted part. Initially, the sorted part

of the array is empty, and unsorted part is the given array. Sorted part is placed at the left, while the unsorted part is placed at the right.

In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted.

ALGORITHM:

Insertion sort:

- Iterate from arr[1] to arr[N] over the array.
- Compare the current element (key) to its predecessor.
- If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

Selection sort:

- Initialize minimum value(min_idx) to location 0.
- Traverse the array to find the minimum element in the array.
- While traversing if any element smaller than min_idx is found then swap both the values.
- Then, increment min_idx to point to the next element.
- Repeat until the array is sorted.

CODE:

1. **FOR GENERATING RANDOM NUMBERS**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

```
5.
6. #define N 100000
7.
8. int main() {
9.   int a[N];
10.
11.  // Generate the 100000 integer numbers
12.  srand(time(0));
13.  for (int i = 0; i < N; i++) {
14.    a[i] = rand();
15.  }
16.
```

2.FOR CREATING .TXT FILE AND OPENING IT

```
// Write the numbers to a text file
  FILE *f = fopen("numbers.txt", "w");
  if (f == NULL) {
    printf("Error opening file!\n");
    return 1;
  }
  for (int i = 0; i < N; i++) {
    fprintf(f, "%d\n", a[i]);
  }
  fclose(f);

  // Read the numbers from the text file
  f = fopen("numbers.txt", "r");
  if (f == NULL) {
    printf("Error opening file!\n");
    return 1;
  }
  for (int i = 0; i < N; i++) {
    fscanf(f, "%d", &a[i]);
  }
  fclose(f);

  return 0;
}
```

3. FOR INSERTION SORT

```
4. #include <stdio.h>
5. #include <math.h>
6. #include <conio.h>
7. #include <stdlib.h>
```

```c
#include <time.h>
void insertionSort(int array[], int size) {
    for (int step = 1; step < size; step++) {
        int key = array[step];
        int j = step - 1;
        while (key < array[j] && j >= 0) {
            array[j + 1] = array[j];
            --j;
        }
        array[j + 1] = key;
    }
}
int main(){

    FILE *fptr, *sPtr;
    int index=99;
    int arrNums[100000];
    clock_t t;
    fptr = fopen("numbers.txt", "r");
    sPtr = fopen("iTimes.txt", "w");
    for(int i=0; i<=999; i++){
        for(int j=0; j<=index; j++){
            fscanf(fptr, "%d", &arrNums[j]);
        }
        t = clock();
        insertionSort(arrNums, index+1);
        t = clock() - t;
        double time_taken = ((double)t)/CLOCKS_PER_SEC;
        fprintf(sPtr, "%lf\n", time_taken);
        printf("%d\t%lf\n", (i+1), time_taken);
        index = index + 100;
        fseek(fptr, 0, SEEK_SET);
    }
    fclose(sPtr);
    fclose(fptr);

    return 0;
}
```

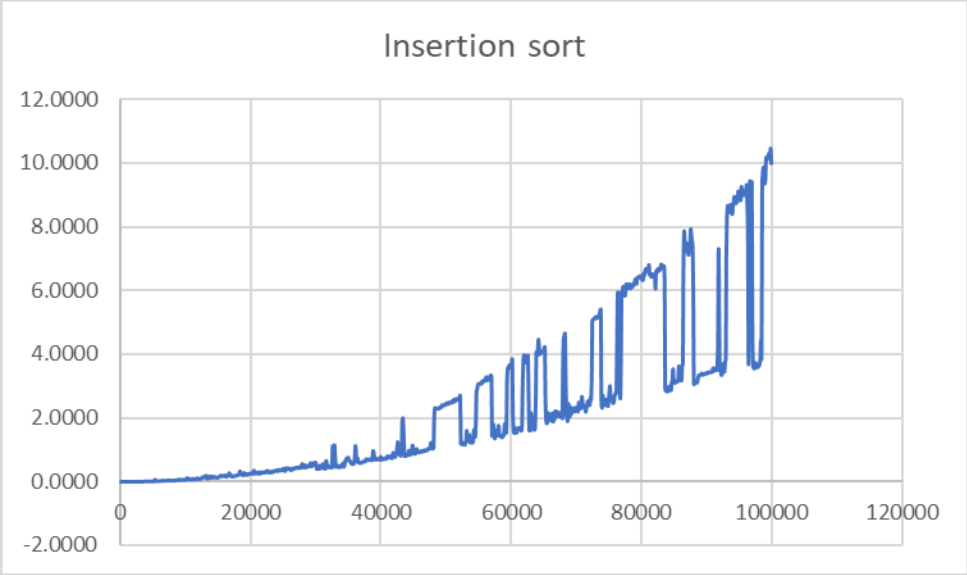
4. FOR SELECTION SORT

```c
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
```

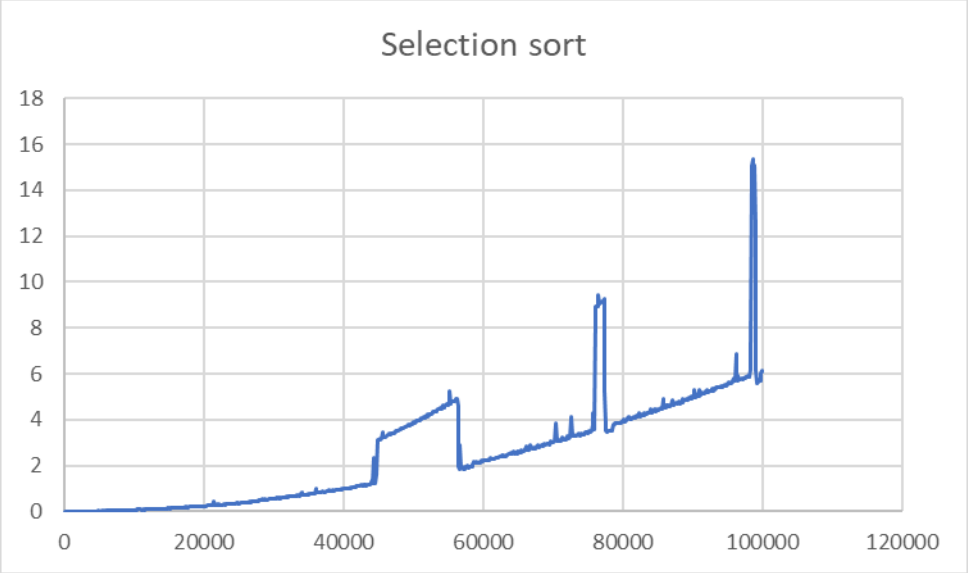
```c
#include <time.h>
void selectionSort(int arr[], int len){
    int minIndex, temp;
    for(int i=0; i<len; i++){
        minIndex = i;
        for(int j=i+1; j<len; j++){
            if(arr[j] < arr[minIndex]){
                minIndex = j;
            }
        }
        temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}
int main(){

    FILE *fptr, *sPtr;
    int index=99;
    int arrNums[100000];
    clock_t t;
    fptr = fopen("numbers.txt", "r");
    sPtr = fopen("sTimes.txt", "w");
    for(int i=0; i<=999; i++){
        for(int j=0; j<=index; j++){
            fscanf(fptr, "%d", &arrNums[j]);
        }
        t = clock();
        selectionSort(arrNums, index+1);
        t = clock() - t;
        double time_taken = ((double)t)/CLOCKS_PER_SEC;
        fprintf(sPtr, "%lf\n", time_taken);
        printf("%d\t%lf\n", (i+1), time_taken);
        index = index + 100;
        fseek(fptr, 0, SEEK_SET);
    }
    fclose(sPtr);
    fclose(fptr);

    return 0;
}
```

Result analysis:

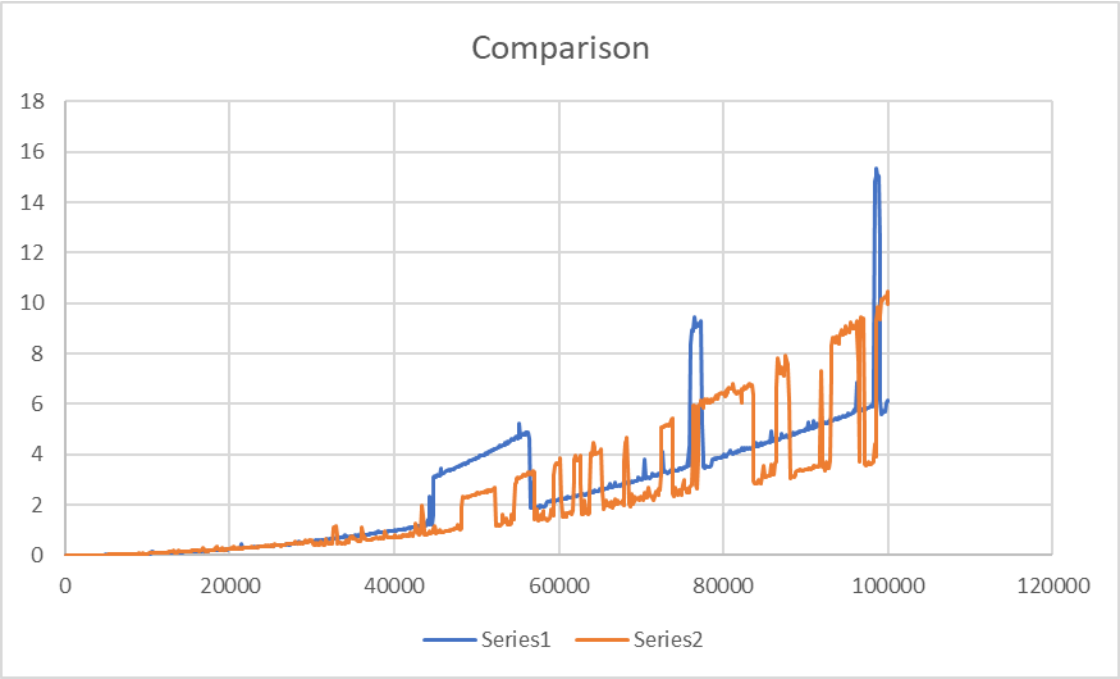
1: Insertion sort



2: Selection sort:



3: Comparison:



Conclusion: we sorted 100000 random numbers in this experiment using insertion sort and selection sort. The average and worst-case complexity of selection sort is O(n2), where n is the number of items. Due to this, it is not suitable for large data sets. So sorting using selection sort had taken 1.5 hr near about where as sorting using insertion sort had taken 15mins.