**ANALYSIS OF ALGORITHM**

**EXPERIMENT – 1**

**Name of Student: HERAMBA LIMAYE**

**Roll no. : 19102A0033**

**PROBLEM STATEMENT :**

**Selection and Insertion Sort**

Implement selection and insertion sort and execute them for different values of n as follows. (Values of n should be taken from 10000 to 50000 with interval of 10000). Make use of files to store the integers generated randomly and count and mention the time required for each one.

**THEORY:**

**Selection sort:**

The steps involved in selection sort

1. Starting from the first element, we search the smallest element in the array, and replace it with the element in the first position.
2. We then move on to the second position, and look for smallest element present in the subarray, starting from index 1, till the last index.
3. We replace the element at the **second** position in the original array, or we can say at the first position in the subarray, with the second smallest element.
4. This is repeated, until the array is completely sorted.

**Insertion sort:**

The steps involved in insertion sort:

1. We start by making the second element of the given array, i.e. element at index 1, the key. The key element here is the new card that we need to add to our existing sorted set of cards(remember the example with cards above).
2. We compare the key element with the element(s) before it, in this case, element at index 0:
   * If the key element is less than the first element, we insert the key element before the first element.
   * If the key element is greater than the first element, then we insert it after the first element.
3. Then, we make the third element of the array as key and will compare it with elements to it's left and insert it at the right position.
4. And we go on repeating this, until the array is sorted.

**IMPLEMENTATION :**

**CODE:**

#include <stdio.h>

#include <conio.h>

#include <time.h>

#include <stdlib.h>

void swap(long \*xp, long \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void selectionSort(long arr[], int n)

{

int i, j, min\_idx;

/\*One by one move boundary of unsorted subarray\*/

for (i = 0; i < n-1; i++)

{

/\*Find the minimum element in unsorted array\*/

min\_idx = i;

for (j = i+1; j < n; j++)

if (arr[j] < arr[min\_idx])

min\_idx = j;

/\*Swap the found minimum element with the first element\*/

swap(&arr[min\_idx], &arr[i]);

}

}

void selectionSortReverse(long arr[], int n)

{

int i, j, min\_idx;

/\*One by one move boundary of unsorted subarray\*/

for (i = 0; i < n-1; i++)

{

/\*Find the minimum element in unsorted array\*/

min\_idx = i;

for (j = i+1; j < n; j++)

if (arr[j] > arr[min\_idx])

min\_idx = j;

/\*Swap the found minimum element with the first element\*/

swap(&arr[min\_idx], &arr[i]);

}

}

void insertionSort(long arr[], int n)

{

int i, key, j;

for (i = 1; i < n; i++)

{

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

void insertionSortReverse(long arr[], int n)

{

int i, key, j;

for (i = 1; i < n; i++) {

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] < key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

int main()

{

FILE \*fp;

int i, j, k, l, m, a, min\_idx;

long arr[50001];

char file1[50];

clock\_t clock\_time\_array[10][2];

float clockt[5];

clock\_t time\_1, time\_2, time\_3, time\_interval;

for(k=10000; k <= 50000; k += 10000)

{

printf("Enter name of a file : ");

scanf("%s", file1);

/\*Getting k elements\*/

fp=fopen(file1,"w");

for(i = 1; i <= k; i++)

{

fprintf(fp,"%d ",rand()%1000);

if(i % 10 == 0)

fprintf(fp, "\n");

}

fclose(fp);

/\*Store in array\*/

fp=fopen(file1,"r");

for(i = 1; i <= k; i++)

{

fscanf(fp,"%d",&a);

arr[i]=a;

}

fclose(fp);

selectionSortReverse(arr, k);

time\_1 = clock();

selectionSort(arr,k);

time\_2 = clock();

selectionSort(arr,k);

time\_3 = clock();

/\*store in file again\*/

fp = fopen(file1,"w");

for(i = 1; i <= 10000;i++)

{

fprintf(fp,"%d ",arr[i]);

if(i % 10 == 0)

fprintf(fp, "\n");

}

fclose(fp);

clock\_time\_array[k/10000 - 1][0] = time\_2 - time\_1;

clock\_time\_array[k/10000 - 1][1] = time\_3 - time\_2;

}

printf("\nThe clock time array for Selection sort is as follows : \n\n");

printf("Worst\tBest\n\n");

for(l = 0; l < 5; l++)

{

for(m = 0; m < 2; m++)

{

printf("%d\t",clock\_time\_array[l][m]);

}

printf("\n");

}

for(k=10000; k <= 50000; k += 10000)

{

printf("Enter name of a file : ");

scanf("%s", file1);

/\*Getting k elements\*/

fp=fopen(file1,"w");

for(i = 1; i <= k; i++)

{

fprintf(fp,"%d ",rand()%1000);

if(i % 10 == 0)

fprintf(fp, "\n");

}

fclose(fp);

/\*Store in array\*/

fp=fopen(file1,"r");

for(i = 1; i <= k; i++)

{

fscanf(fp,"%d",&a);

arr[i]=a;

}

fclose(fp);

insertionSortReverse(arr, k);

time\_1 = clock();

insertionSort(arr, k);

time\_2 = clock();

insertionSort(arr, k);

time\_3 = clock();

/\*store in file again\*/

fp = fopen(file1,"w");

for(i=0;i<=10000;i++)

{

fprintf(fp,"%d ",arr[i]);

if(i % 10 == 0)

fprintf(fp, "\n");

}

fclose(fp);

clock\_time\_array[k/10000 + 4][0] = time\_2 - time\_1;

clock\_time\_array[k/10000 + 4][1] = time\_3 - time\_2;

}

printf("\nThe clock time array for insertion sort is as follows : \n\n");

printf("Worst\tBest\n\n");

for(l = 5; l < 10; l++)

{

for(m = 0; m < 2; m++)

{

printf("%d\t",clock\_time\_array[l][m]);

}

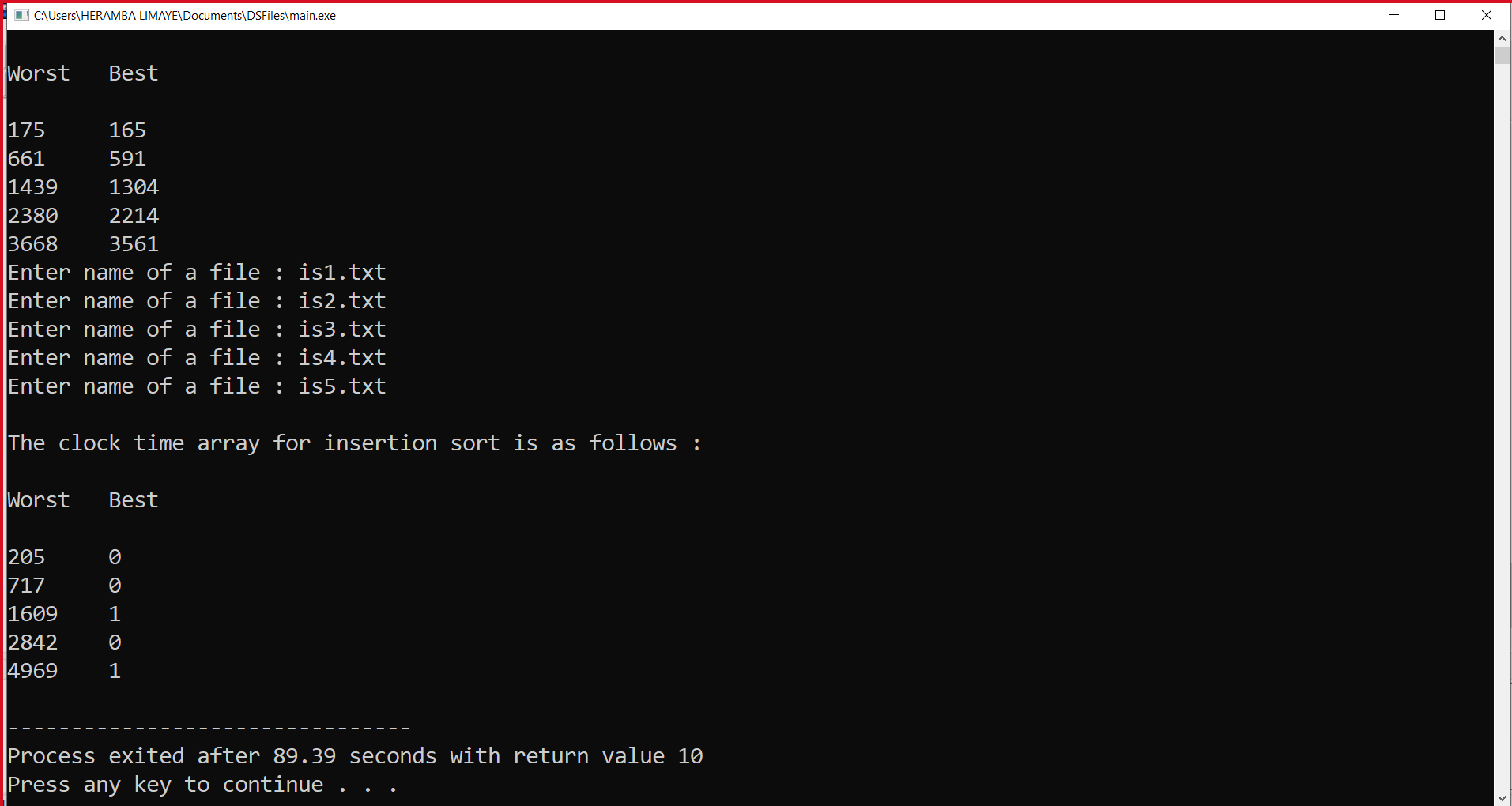
printf("\n");

}

}

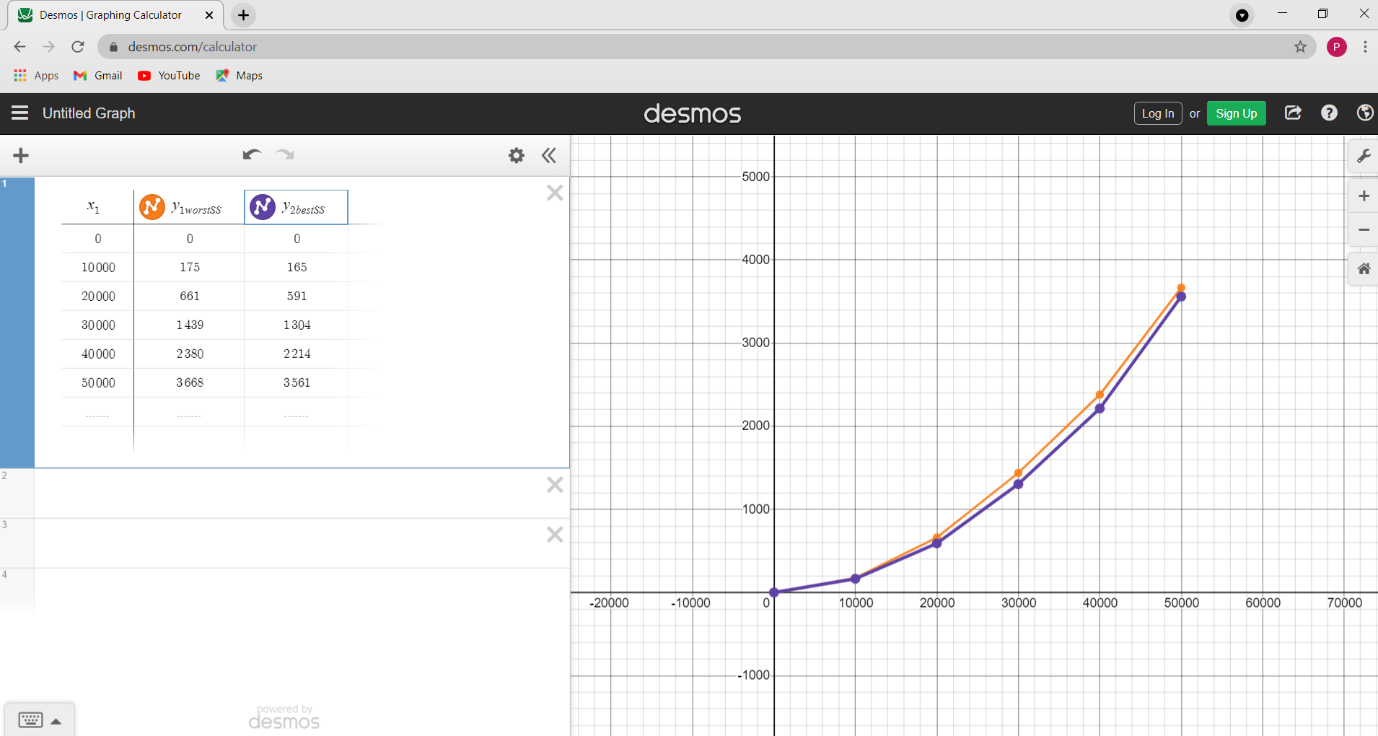
**OUTPUT :**

(the first worst and best time are of Selection sort)

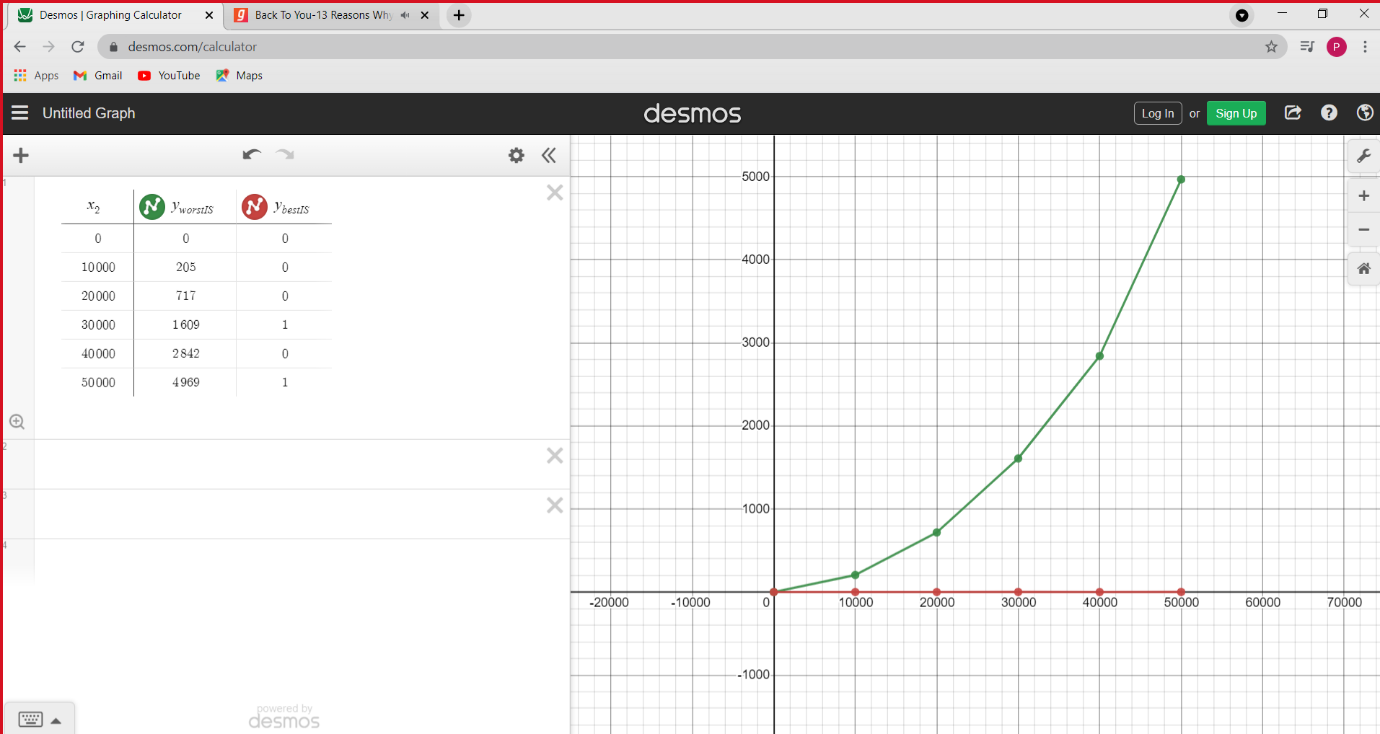


**Graphs :**

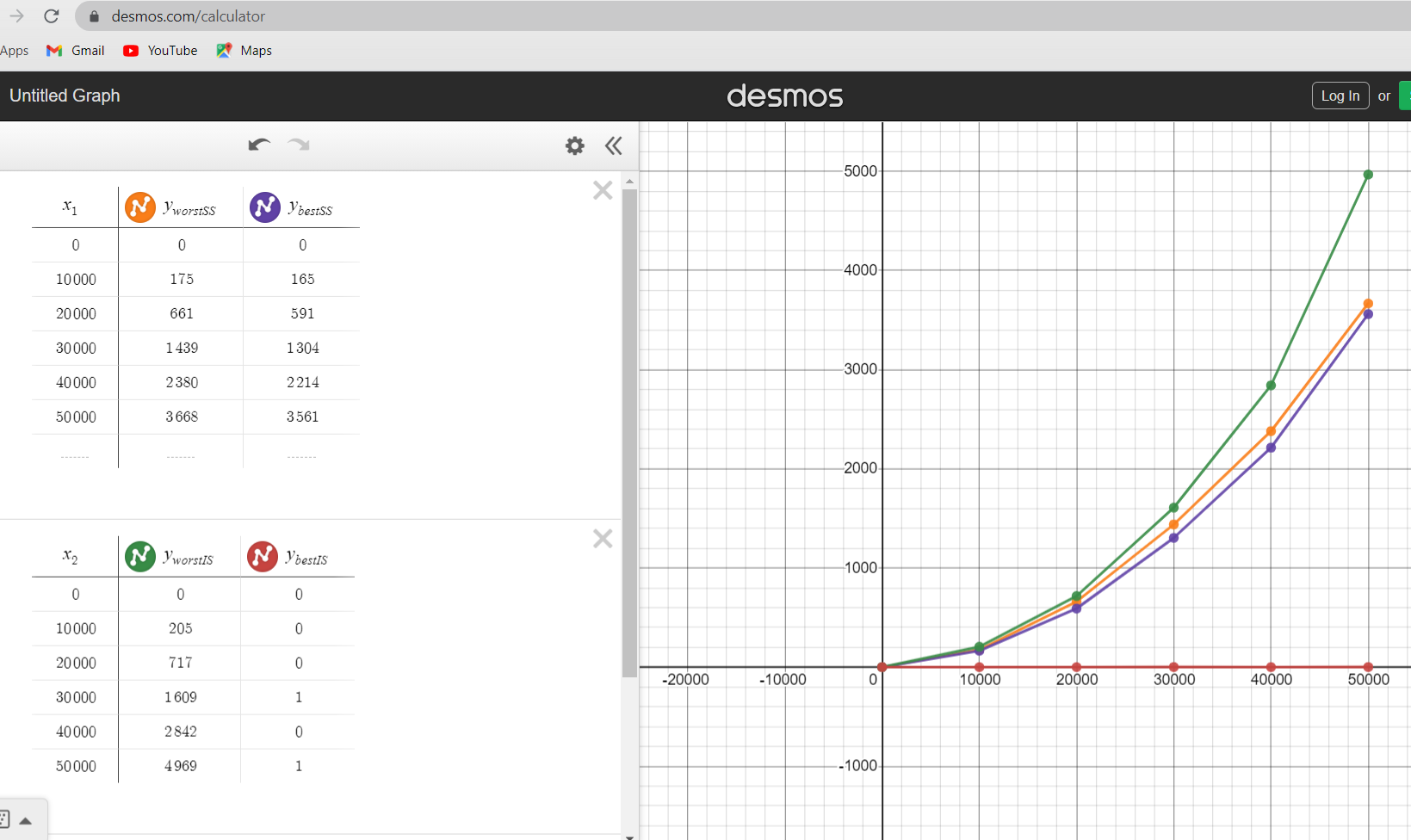
* Selection sort best and worst case comparison



* Insertion sort best and worst case comparison



* Insertion sort and Selection sort comparison together



**Conclusion :**

SELECTION SORT:

* Best case = Ω(n^2)
* Worst case =O(n^2)

INSERTION SORT:

* Best case : Ω(n)
* Worst case : O(n^2)