NAME :	APULKI DUBEY
UID	2021300032
CLASS :	SE COMPS A
THEORY	**INSERTION SORT** : It is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part. **SELECTION SORT** The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array. ● The subarray is already sorted. ● The remaining subarray is unsorted. In every iteration of the selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.
ALGORITHM	**INSERTION SORT** Step 1 - If the element is the first element, assume that it is already sorted. Return 1. Step2 - Pick the next element, and store it separately in a key. Step3 - Now, compare the key with all elements in the sorted array. Step 4 - If the element in the sorted array is smaller than **the**

current element, then move to the next element. Else, shift greater elements in the array towards the right.

Step 5 - Insert the value.

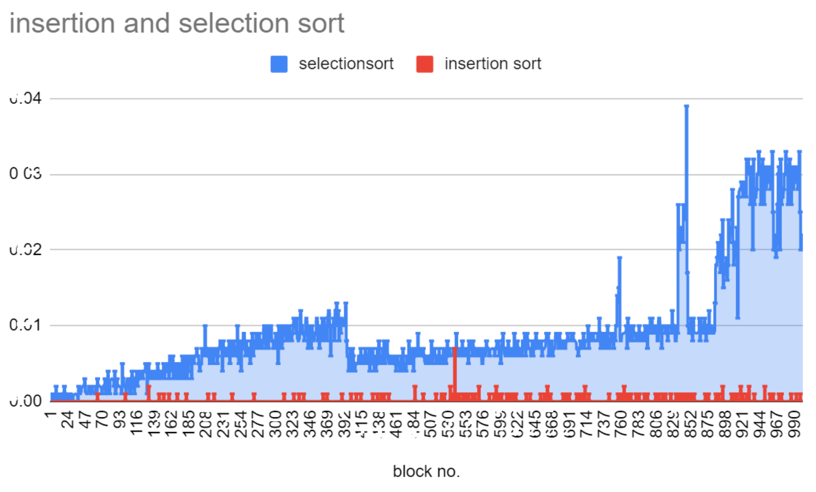
Step 6 - Repeat until the array is sorted.

SELECTION SORT

```
Step 1: Repeat Steps 2 and 3 for i = 0 to n-1

Step 2: CALL SMALLEST(arr, i, n, pos)

Step 3: SWAP arr[i] with arr[pos]

[END OF LOOP]

Step 4: EXIT


SMALLEST (arr, i, n, pos)

Step 1: [INITIALIZE] SET SMALL = arr[i]

Step 2: [INITIALIZE] SET pos = i

Step 3: Repeat for j = i+1 to n

if (SMALL > arr[j])

    SET SMALL = arr[j]

SET pos = j

[END OF if]

[END OF LOOP]

Step 4: RETURN pos
```

RESULT:	CODE:

```
#include<iostream>
#include<fstream>
#include<time.h>
#include<cstdlib>
#include<fstream>
#include<string.h>
#include<chrono>
#include<numeric>
#include<iomanip>
using namespace std;

void getInput()
{
  ofstream fp;
  fp.open("input.text");
  for(int i=0;i<100000;i++)
  fp<< rand()%100000 << endl;
  fp.close();
}

void readfile(int arr[])
{
  int i=0;
  ifstream fp;
  fp.open("input.text");
  while(fp.good())
  {
    fp >> arr[i];
    i++;
  }
  fp.close();
}


void insertionsort(int arr[], int n)
{
    int i, key, j;
    if(n==1001)
    return;
    for (i=n+1;  i<=100* 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 swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void selectionsort(int arr[], int n)
{
    int i, j, min_idx;
    // One by one move boundary of
    // unsorted subarray
    for (i = n; i <=100*n; 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
        if (min_idx!=i)
            swap(&arr[min_idx], &arr[i]);
    }
}

int main()
{

  int arr[100000];
  clock_t start,end;
  double time_taken;
  getInput();
  readfile(arr);
  printf("The randomly generated numbers:\n");

     int k=2;
   while(k--){
  printf("\nPlease choose any one of them:\n1. Insertion
Sort\n2.Selection Sort\n");
  printf("\n Please enter your choice:");
   int ch;
   scanf("%d", &ch);


   if(ch==1)
   {
       printf("hi\n");
        for(int i=1;i<=1000;i++){


      start=clock();



      insertionsort(arr,i);
      end=clock();
      time_taken=double(end-start)/double(CLOCKS_PER_SEC);
      cout<<"The time taken by program to execute insertion sort
is with block number :"<<i<<"
"<<fixed<<time_taken<<setprecision(5);
      cout<<"seconds "<<endl;

   }
   }
   else if(ch==2)
   {
       for(int i=1;i<=1000;i++){
       start=clock();
       selectionsort(arr,i);
       end=clock();
       time_taken=double(end-start)/double(CLOCKS_PER_SEC);
       cout<<" The time taken by program to execute selection sort
is with block number :"<<i<<"
"<<fixed<<time_taken<<setprecision(5);
       cout<<"seconds "<<endl;

   }
   }
   else
   {
       cout<<"Please choose from the above choices\n"<<endl;
   }
   }
  return 0;

}
```

OBSERVATION	
	The speed of insertion sort and selection sort can vary based on several factors, such as the size of the input array, the order of the elements in the array, and the specific implementation of the algorithm. However, in general, insertion sort tends to be faster than selection sort. This is because insertion sort has the advantage of sorting elements in place and it is more efficient when the array is partially sorted. On the other hand, selection sort has to repeatedly find the minimum element in the unsorted portion of the array, which can be time-consuming for larger arrays. Both insertion sort and selection sort have a space complexity of O(1), meaning that the algorithms use a constant amount of extra memory regardless of the size of the input array. This is because both algorithms are considered to be in-place sorting algorithms, meaning that they sort the elements within the original array and do not require any additional memory to store the sorted result.
CONCLUSION:	I have successfully performed and understood the insertion sort and selection sort algorithm.I have realized that insertion sort is faster than selection sort.