NAME:	VEDANT TUSHAR DAPOLIKAR
UID:	2021700016
BRANCH:	CS-DS

EXPERIMENT-2

- **AIM:** Experiment on finding the running time of an insertion sort and selection sort algorithm.

- **ALGORITHM:**

  - ❖ FOR SELECTION SORT:

1)Initially ,set the minimum value to array[0]
2)Then,search the array for the minimum element
3)If the traversed element is smaller than the min value,swap both the elements
4)After,traversing the array set min to array[1]
5)Keep repeating till the array is sorted

  - ❖ FOR INSERTION SORT:

1)Initially ,assume that the starting element is already sorted.
2)Then,store the next element in a new variable
3)Compare the new with all elements of the array
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.
5)Now insert the value
6)Keep repeating till the array is sorted

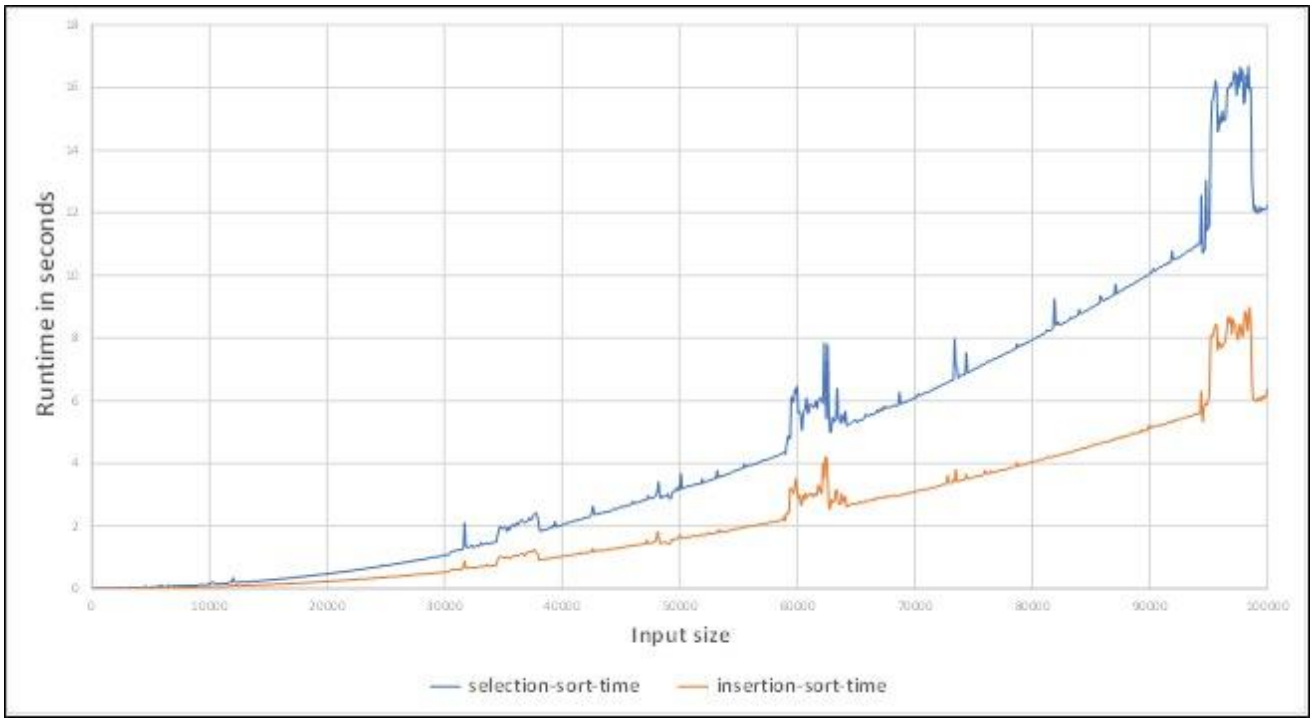
- **CODE:**

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

void selectionsort(int array[], int end){
  for (int i = 0; i < end; i++)
  {
    int mini = i;
    for (int j = i + 1; j < end; j++)
    {
      if (array[j] < array[mini])
      {
        mini = j;
      }
    }
    int temp = array[i];
    array[i] = array[mini];
    array[mini] = temp;
  }
}

void insertionsort(int a[], int n){
  for (int i = 1; i < n; i++){
    int key = a[i];
    int j = i - 1;
    while (j >= 0 && a[j] > key){
      a[j + 1] = a[j];
      j = j - 1;
    }
    a[j + 1] = key;
  }
}

int main(){

  FILE *fptr;
  fptr = fopen("randomm.txt", "w");
  if (fptr == NULL){
```

```
    printf("ERROR Creating File!");
    exit(1);
  }

  int n = 100000;
  srand(time(0));
  for (int i = 1; i <= n; i++){
    int r = rand() % 100;
    fprintf(fptr, "%d\n", r);
  }
  fclose(fptr);

  int block = 1;
  printf("Block\tSelection\tInsertion\n");
  for (int i = 100; i <= n; i += 100){
    fptr = fopen("randomm.txt", "r");
    int arr[i];

    for (int j = 0; j < i; j++){
      fscanf(fptr, "%d", &arr[j]);
    }
    clock_t t;
    t = clock();
    selectionsort(arr, i);
    t = clock() - t;
    double time_takenss = ((double)t) / CLOCKS_PER_SEC;
    fclose(fptr);

    fptr = fopen("randomm.txt", "r");
    int arr2[i];
    for (int j = 0; j < i; j++){
      fscanf(fptr, "%d", &arr2[j]);
    }
    clock_t t2;
    t2 = clock();
    insertionsort(arr2, i);
    t2 = clock() - t2;
    double time_takenis = ((double)t2) / CLOCKS_PER_SEC;

    printf("%d\t%f\t%f\n", block, time_takenss, time_takenis);
```

```
        fclose(fptr);
        block++;

    }
    return 0;
}
```

- **OUTPUT:**



The graph plots the execution time in seconds for 1000 blocks of size 100 each i.e 100000 inputs for insertion and selection sort algorithms.

- **RESULT ANALYSIS**:

1)Here we have plotted graphs of time complexities of insertion sort and selection Sort.
2)Through this graph it is quite evident that insertion sort is better than selection sort,as there is a vast difference between the execution time of both the Algorithms.
3)In insertion sort it inserts the value in the presorted array to sort the set of values in the array,whereas,in selection sort it finds the minimum number from the list and sort it in ascending / descending order