NAME:	Vinit Madhyan
UID:	2021300070
SUBJECT	Design and Analysis of Algorithm
EXPERIMENT NO :	2
AIM:	Experiment on finding the running time of an algorithm.
Algorithm	**1. Insertion sort-** a. procedure insertionSort(A: list of sortable items) b. n = length(A) c. for i = 1 to n - 1 do d. j = i e. while j > 0 and A[j-1] > A[j] do f. swap(A[j], A[j-1]) g. j = j - 1 h. end while i. end for j. end procedure **2. Selection sort-** a. Repeat Steps b and c for i = 0 to n-1 b. CALL SMALLEST(arr, i, n, pos) c. [INITIALIZE] SET key = i d. Repeat for j = i+1 to n e. if (arr[key] > arr[j]) f. SET key=j g. [END OF if] h. [END OF LOOP] i. SWAP arr[i] with arr[pos] j. [END OF LOOP] k. EXIT

PROGRAM:

```cpp
#include <bits/stdc++.h>
#include <fstream>
using namespace std;

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

void selectionsort(vector<int> arr, int num)
{
    int key;
    for (int i = 0; i < num - 1; i++)
    {
        key = i;
        for (int j = i + 1; j < num; j++)
        {
            if (arr[j] < arr[key])
            {
                key = j;
            }
        }
        int temp;
        temp = arr[key];
        arr[key] = arr[i];
        arr[i] = temp;
    }
}

int main()
{
    vector<int> arr;
    clock_t t1, t2, t3, t4;
    string filename("values.txt");
    ifstream fin(filename);
    if (!fin.is_open())
```

```
    {
        cerr << "Could not open the file - '"
             << filename << "'" << endl;
        return EXIT_FAILURE;
    }

    while (!fin.eof())
    {
        int tmp;
        fin >> tmp;
        arr.push_back(tmp);
    }

    int num = 100;
    for (int i = 0; i < 150; i++)
    {
        t1 = clock();
        insertionsort(arr, num);
        t2 = clock();
        t3 = clock();
        selectionsort(arr, num);
        t4 = clock();
        double insertiontime = double(t2 - t1) /
double(CLOCKS_PER_SEC);
        double selectiontime = double(t4 - t3) /
double(CLOCKS_PER_SEC);
        cout << endl;
        cout << i + 1 << "  " << fixed << insertiontime <<
setprecision(5) << '\t';
        cout << fixed << selectiontime << setprecision(5);

        num += 100;
    }
    fin.close();
    return 0;
}
```

Values.cpp

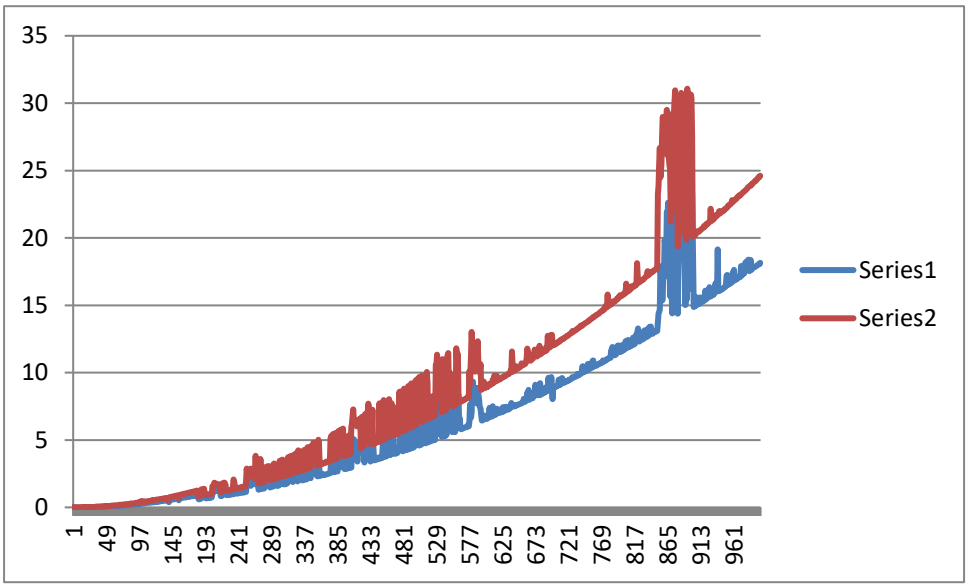
```
#include <iostream>
#include <cstdlib>
#include <cstdio>
using namespace std;

int main()
{
    for (int i = 1; i <= 100000; i++)
```

```
    {
        cout << rand() << " ";
    }

    return 0;
}
```

Observation (SNAPSHOT)



Observation

For insertion sort

the graph of the running time of insertion sort reveals that as the size of the input data increases, the running time also increases at a steady and consistent rate. The graph exhibits a characteristic upward trend, with the slope of the curve becoming steeper as the size of the input data increases. This is indicative of the quadratic nature of the running time of insertion sort.

For selection sort

the graph of the running time of selection sort shows that it also increases as the size of the input data increases, similar to insertion sort. However, the rate of increase is relatively slower compared to insertion sort, which is evident from the less steep slope of the graph. This suggests that selection sort is a more efficient algorithm for small input sizes.

Conclusion

The experiment on finding the running time of insertion sort and selection sort shows that the running time of both algorithms is dependent on the size of the input data. The running time of insertion sort is quadratic in nature, while that of selection sort is also quadratic but performs better on small inputs.