Bharatiya Vidya Bhavan's

Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India

(Autonomous College Affiliated to University of Mumbai)

Name	Tejas Jadhav
UID No.	2022301006
Class	COMPS A (B batch)
Experiment No.	01 B

Aim: Experiment on finding the running time of an algorithm.

Algorithm:

```
InsertionSort(A)
    For i = 2 to A.length
        key = A[i]
        j = i -1
        while j > 0 and A[j] > key
            A[j + 1] = A[j]
            j  = j – 1
        A[j + 1] = key
```

```
SelectionSort(A)
    For i = 1 to A.length – 1
        i_min = i
        for j = i +1 to A.length
            if (A[j] < A[i_min])
                i_min = j
        if i != i_min
            swap(A[i], A[i_min])
```

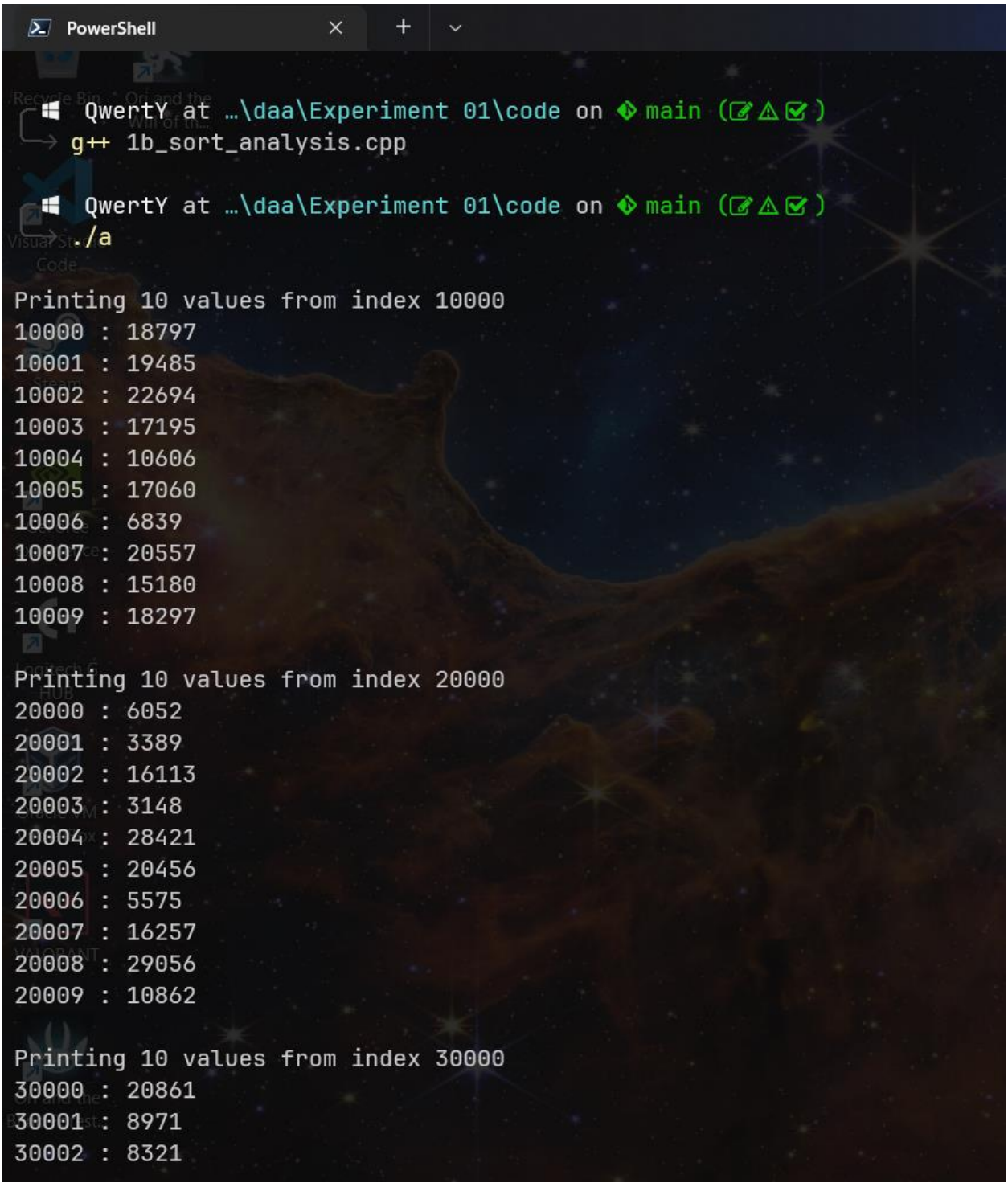
Code:

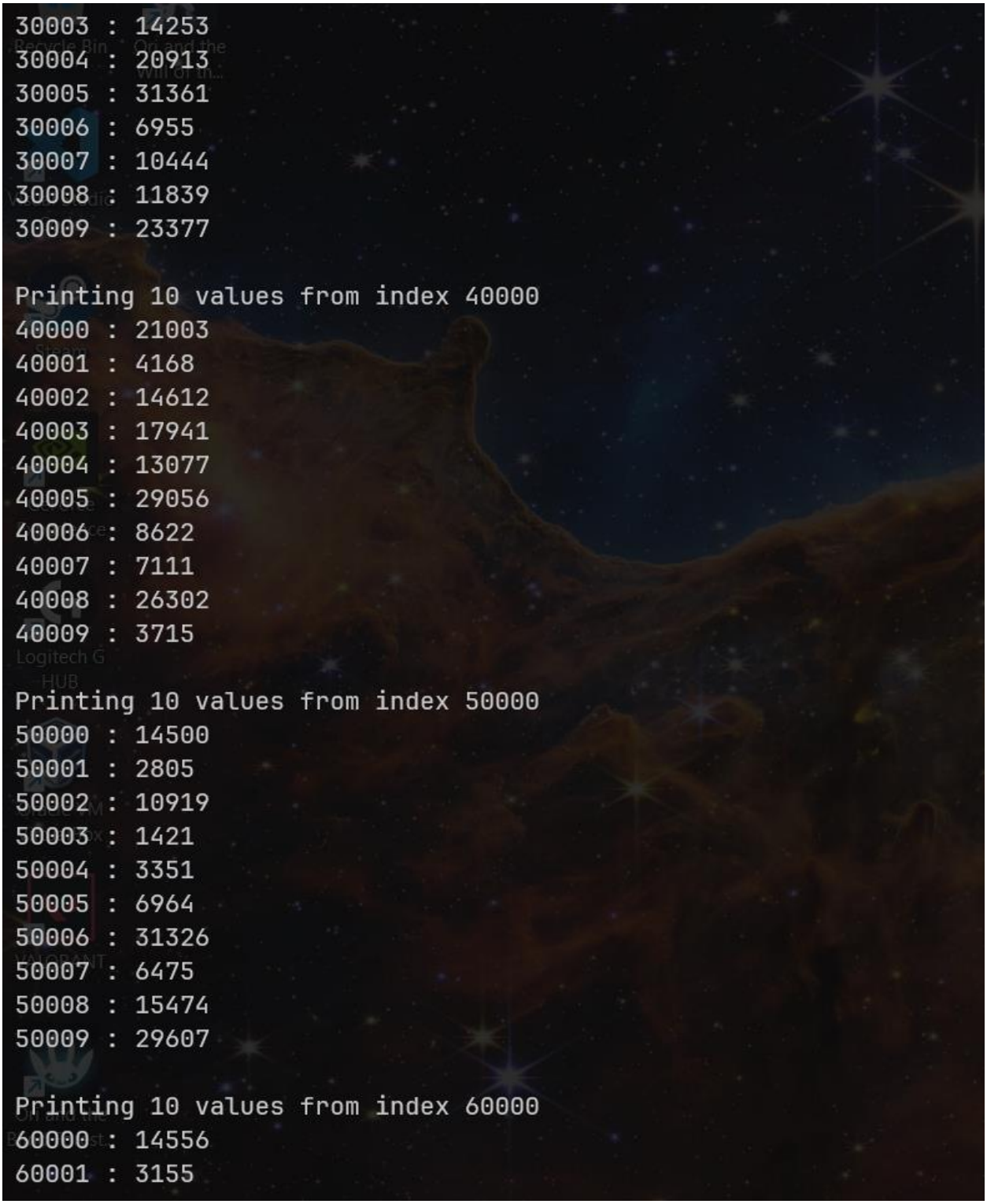
```
#include <stdio.h>
#include <chrono>
#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>
using namespace std;

double insertion_comparision = 0.0;
double selection_comparision = 0.0;

void insertion_sort(int* arr, int size) {
    for (int i = 1; i < size; i++) {
        int current = arr[i];
        int j = i - 1;
        while (j >= 0 && current < arr[j]) {
            arr[j + 1] = arr[j];
            j--;
            insertion_comparision++;
        }
        arr[j + 1] = current;
    }
}

void swap(int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
}

void selection_sort(int* arr, int size) {
    for (int i = 0; i < size - 1; i++) {
        int i_min = i;
        for (int j = i + 1; j < size; j++) {
            selection_comparision++;
            if (arr[j] < arr[i_min]) {
                i_min = j;
            }
        }
        if (i != i_min) {
```

```
            swap(arr[i], arr[i_min]);
        }
    }
}

int digits(int num) {
    return num == 0 ? 1 : floor(log10(abs(num))) + 1;
}

int main() {
    int arr_ins[100000];
    int arr_sel[100000];
    ifstream nums("random_numbers.txt");
    ofstream output("../csv/sort_analysis.csv");
    output << "block_size,insertion,selection\n";

    for (int i = 1; i <= 100000; i++) {
        int val = 0;
        nums >> val;
        arr_ins[i] = val;
        arr_sel[i] = val;
    }

    // 1000 blocks of 100 numbers
    for (int i = 1; i <= 1000; i++) {
        // print 10 values at index 10000, 20000, ...
        int index = i * 100;
        if (index % 10000 == 0 && index != 100000) {
            cout << "\nPrinting 10 values from index " << index << endl;
            for (int t = 0; t < 10; t++) {
                cout << index + t << " : " << arr_ins[index + t] << "\n";
            }
        }

        // insertion
        auto ins_start = chrono::high_resolution_clock::now();
        insertion_sort(arr_ins, i * 100);
        auto ins_end = chrono::high_resolution_clock::now();
        chrono::duration<double> ins_time = (ins_end - ins_start);
```

```
        // selection
        auto sel_start = chrono::high_resolution_clock::now();
        selection_sort(arr_sel, i * 100);
        auto sel_end = chrono::high_resolution_clock::now();
        chrono::duration<double> sel_time = (sel_end - sel_start);

        output << i * 100 << "," << ins_time.count() << "," <<
sel_time.count() << "\n";
    }

    cout << "Sorting completed !" << endl;

    cout << "\nSmallest Number = " << arr_ins[0] << "\tDigits = " <<
digits(arr_ins[0]) << endl;
    cout << "Largest Number = " << arr_ins[99999] << "\tDigits = " <<
digits(arr_ins[99999]) << endl;

    printf("Insertion sort comparision count: %.0lf\n",
insertion_comparision);
    printf("Selection sort comparision count: %.0lf\n",
selection_comparision);

    return 0;
}
```

Output:

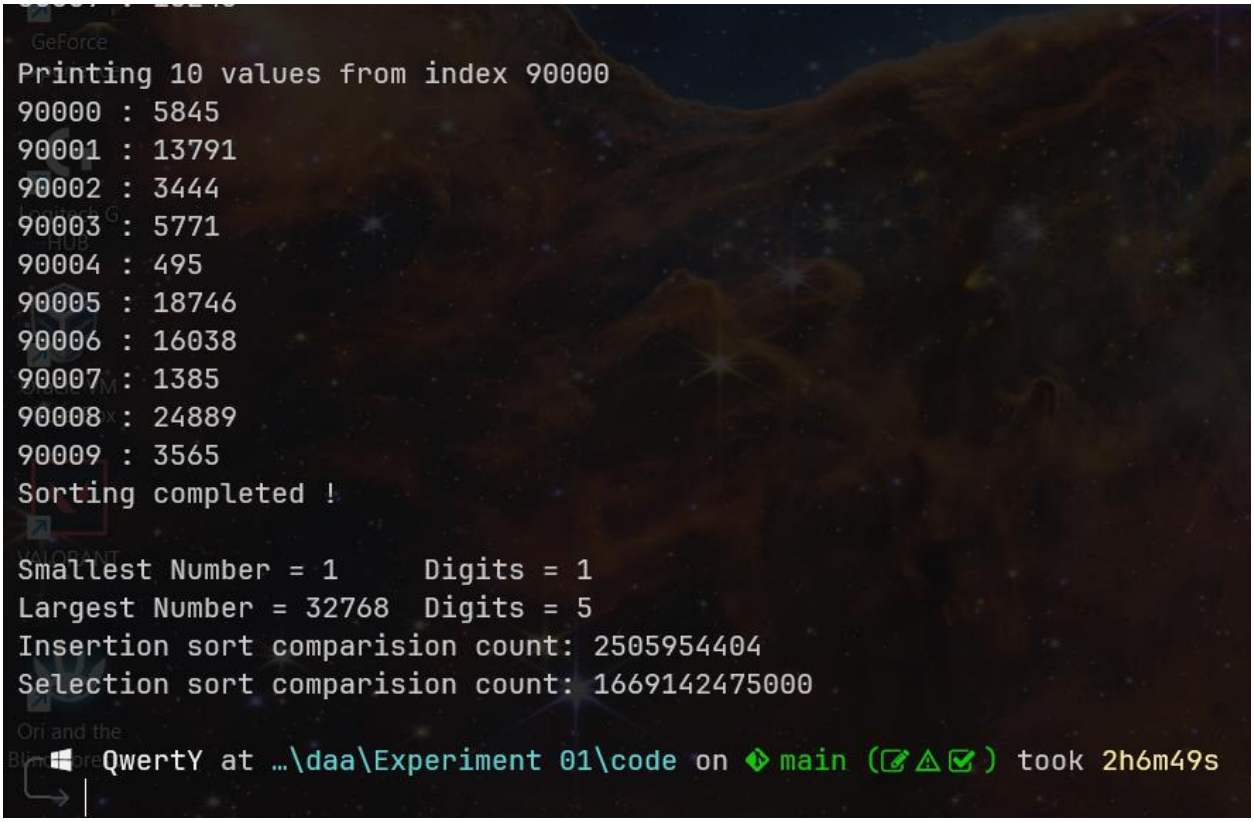

PowerShell
QwertY at …\daa\Experiment 01\code on main
g++ 1b_sort_analysis.cpp
QwertY at …\daa\Experiment 01\code on main
./a
Printing 10 values from index 10000
10000 : 18797
10001 : 19485
10002 : 22694
10003 : 17195
10004 : 10606
10005 : 17060
10006 : 6839
10007 : 20557
10008 : 15180
10009 : 18297
Printing 10 values from index 20000
20000 : 6052
20001 : 3389
20002 : 16113
20003 : 3148
20004 : 28421
20005 : 20456
20006 : 5575
20007 : 16257
20008 : 29056
20009 : 10862
Printing 10 values from index 30000
30000 : 20861
30001 : 8971
30002 : 8321



```
30003 : 14253
30004 : 20913
30005 : 31361
30006 : 6955
30007 : 10444
30008 : 11839
30009 : 23377

Printing 10 values from index 40000
40000 : 21003
40001 : 4168
40002 : 14612
40003 : 17941
40004 : 13077
40005 : 29056
40006 : 8622
40007 : 7111
40008 : 26302
40009 : 3715

Printing 10 values from index 50000
50000 : 14500
50001 : 2805
50002 : 10919
50003 : 1421
50004 : 3351
50005 : 6964
50006 : 31326
50007 : 6475
50008 : 15474
50009 : 29607

Printing 10 values from index 60000
60000 : 14556
60001 : 3155
```

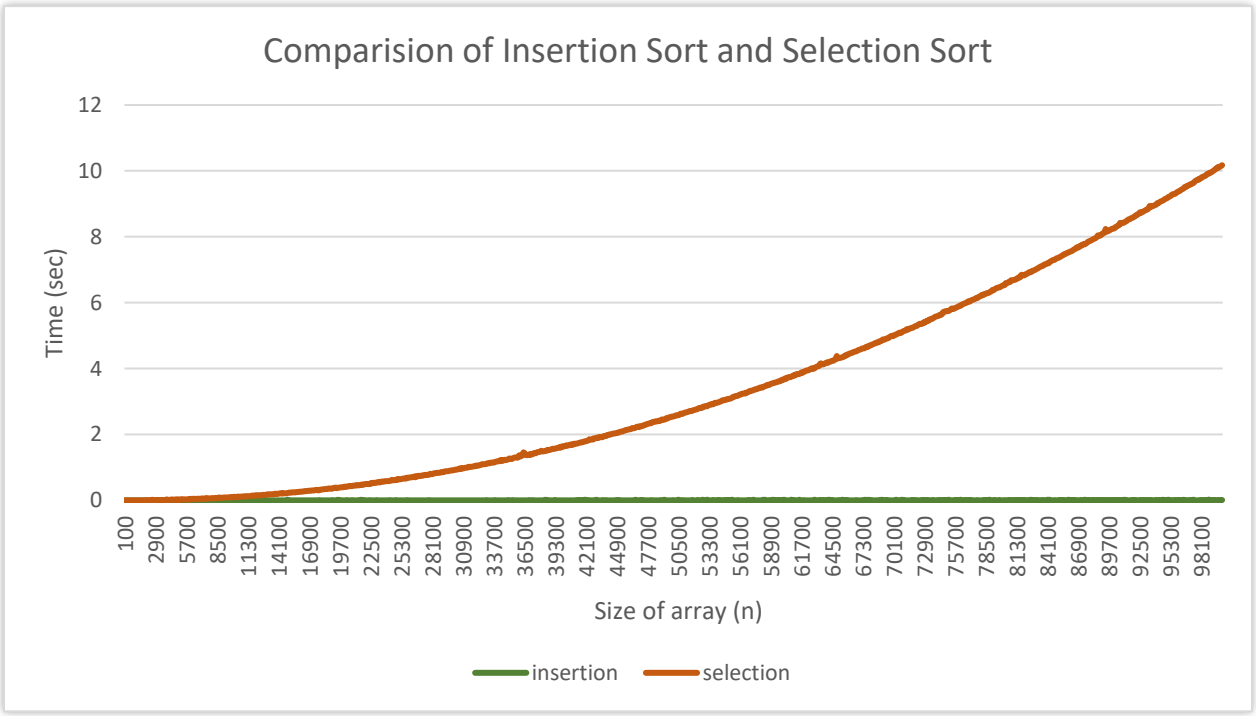
```
60002 : 19831
60003 : 8544
60004 : 16860
60005 : 9520
60006 : 8329
60007 : 9073
60008 : 19728
60009 : 17819

Printing 10 values from index 70000
70000 : 27273
70001 : 24804
70002 : 32768
70003 : 12821
70004 : 27593
70005 : 5188
70006 : 2490
70007 : 19858
70008 : 653
70009 : 23171

Printing 10 values from index 80000
80000 : 6592
80001 : 14439
80002 : 30464
80003 : 19830
80004 : 11651
80005 : 21634
80006 : 30144
80007 : 19527
80008 : 25375
80009 : 13246

Printing 10 values from index 90000
90000 : 5845
```

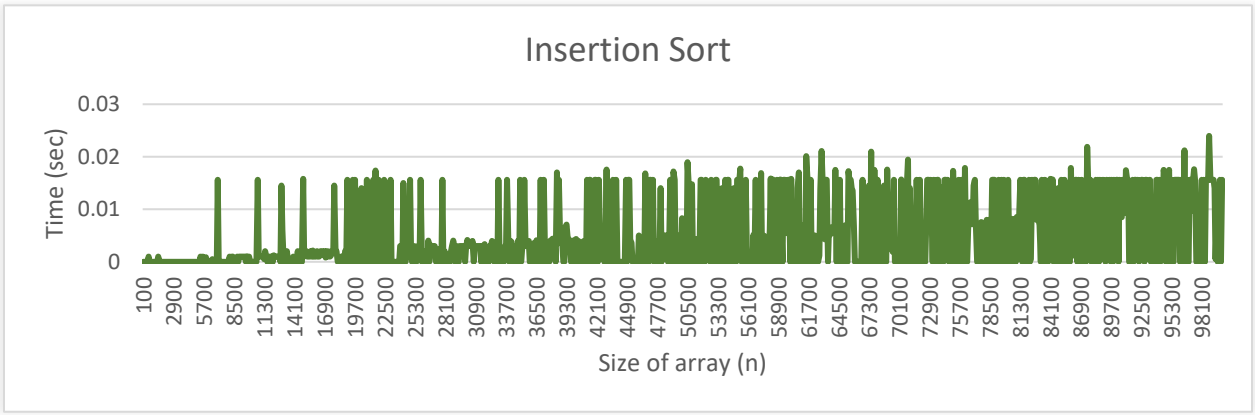
Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)



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

Chart:





Observations:

1) Initially, both sorting techniques take the same time.
2) As the size of blocks increase, I observe that selection sort begins to take more time. However, insertion sort takes almost the same amount of time at every block
3) The graph observed for selection sort resembles a scaled down quadratic curve.
4) The graph observed for insertion sort resembles a line.
5) Insertion sort works by maintaining two subarrays, one is always sorted while other is not. This problem statement mimics the behavior of insertion sort as in the only the new 100 elements are completely random while the all the previous elements are partially sorted.
6) Selection sort works by finding the smallest element in the unsorted subarray and replacing it with current element. In order to determine if an element is truly the smallest element, it is necessary to check all elements.

Conclusion:

- Insertion sort works by maintaining two subarrays, one of which always contain sorted elements. If an element from unsorted array finds its appropriate position in sorted array, we go on to the next iteration.
- The current problem statement, that goes on adding 100 unsorted elements on already sorted array, greatly favors in how insertion sort works.
- Selection sort works by again maintaining two subarrays. One is always sorted while other isn't. However, in selection sort we find the smallest element in unsorted array and swap it with current element.
- In order to find the smallest element, it is necessary to traverse the entire unsorted subarray. Hence each addition of 100 elements as per problem statement, just increases the number of comparisons for selection sort.

After conducting this experiment and analyzing the time for both sorting techniques, I conclude that the nature of problem statement and initial state of data affects greatly on the runtime on algorithm even if they have the same worst case time complexity.