Data Structure and Algorithms

Lab # 10

Hassaan Akbar Cheema

174351

BSCS 6C

**Task1:**

**Implement Insertion sort algorithms in C++.**

// C program for insertion sort

#include <stdio.h>

#include <math.h>

/\* Function to sort an array using insertion sort\*/

void insertionSort(int arr[], int n)

{

int i, key, j;

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

{

key = arr[i];

j = i - 1;

/\* Move elements of arr[0..i-1], that are

greater than key, to one position ahead

of their current position \*/

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

{

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

j = j - 1;

}

arr[j + 1] = key;

}

}

// A utility function ot print an array of size n

void printArray(int arr[], int n)

{

int i;

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

printf("%d ", arr[i]);

printf("\n");

}

/\* Driver program to test insertion sort \*/

int main()

{

int arr[] = { 12, 11, 13, 5, 6 };

int n = sizeof(arr) / sizeof(arr[0]);

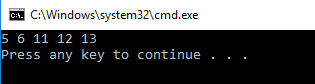
insertionSort(arr, n);

printArray(arr, n);

return 0;

}

**OUTPUT:**



**Task 2 Empirical Study:**

**The next step is to compare the algorithm with two different inputs.**

**• Find O(n) for the average case**

**100,000 random numbers.**

**Code:**

int main()

{

const int n = 100000;

int arr[n];

int arrdes[n];

srand(time(NULL));

//int n = sizeof(arr) / sizeof(arr[0]);

for(int i = 0; i < n; i++){

arr[i] = 0 + rand() % n + 1;

}

clock\_t start = clock(); // start timer

insertionSort(arr, n);

clock\_t stop = clock();

int cpu\_time\_used = float(stop - start) / CLOCKS\_PER\_SEC; //time in seconds

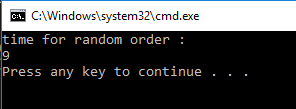
cout << "time for ascending order :";

cout << endl << cpu\_time\_used << endl;

return 0;

}

**OUTPUT:**



**• Find O(n) for the best case**

**100,000 numbers sorted in ascending order.**

Time = 0.08s

**• Find O(n) for the worst case**

**100,000 numbers sorted in the descending order.**

**Code:**

int main()

{

const int n = 100000;

int arr[n];

int arrdes[n];

srand(time(NULL));

//int n = sizeof(arr) / sizeof(arr[0]);

for(int i = 0; i < n; i++){

arr[i] = 0 + rand() % n + 1;

}

insertionSort(arr, n);

for (int i = n,j = 0; i > 0; i--,j++){

arrdes[i] = arr[j];

}

clock\_t start = clock(); // start timer

insertionSort(arrdes, n);

clock\_t stop = clock();

int cpu\_time\_used = float(stop - start) / CLOCKS\_PER\_SEC; //time in seconds

cout << "time for descending order :";

cout << endl << cpu\_time\_used << endl;

return 0;

}

**OUTPUT:**

