|  |
| --- |
| **Experiment No. 2** |
| **To implement Insertion Sort** |
| Date of Performance: |
| Date of Submission: |

## Experiment No. 2

**Title:** Insertion Sort

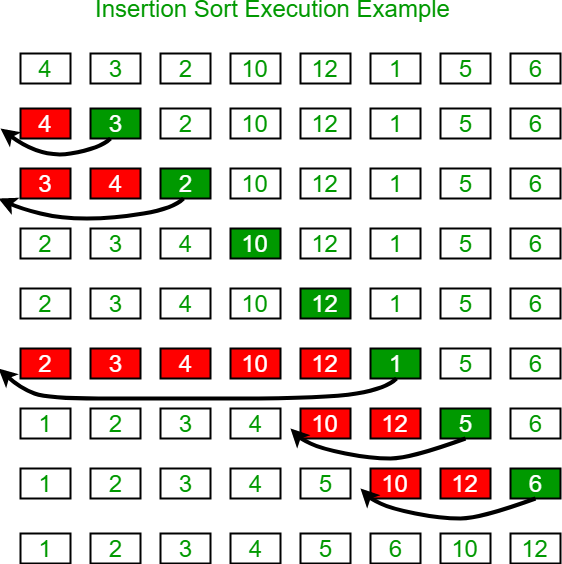
**Aim:** To study, implement and Analyze Insertion Sort Algorithm

**Objective:** To introduce the methods of designing and analyzing algorithms

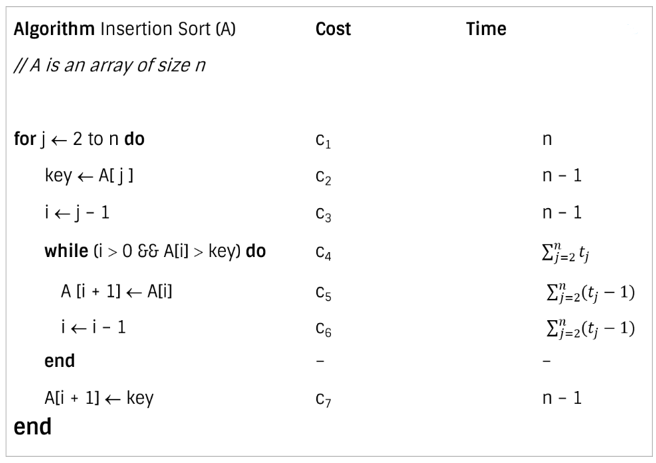
#### Theory:

Insertion sort is a simple sorting algorithm that works similar to the way you sort the 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.

#### Example:



**Algorithm and Complexity:**



**Best case analysis:**

* Let size of the input array is n. Total time taken by algorithm is the summation of time taken by each of its instruction.

A black and white math equation

Description automatically generated with medium confidence

* The best case offers the lower bound of the algorithm’s running time.
* When data is already sorted, the best scenario for insertion sort happens.
* In this case, the condition in the while loop will never be satisfied, resulting in tj = 1.

A screenshot of a math problem

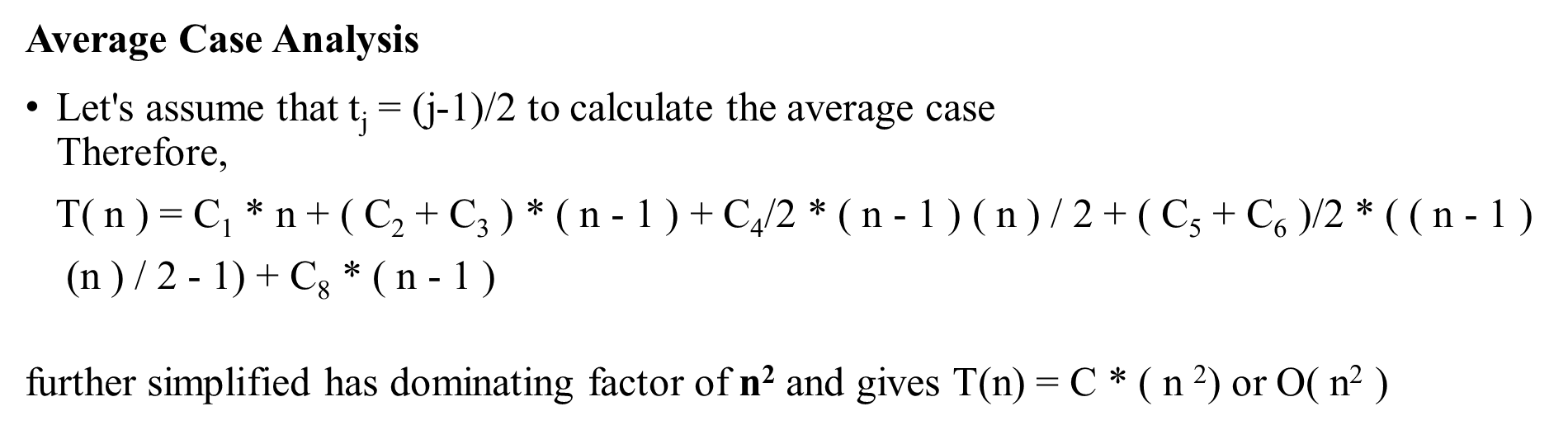
Description automatically generated

**Worst case analysis:**

* The worst-case running time gives an upper bound of running time for any input.
* The running time of algorithm cannot get worse than its worst-case running time.
* Worst case for insertion sort occurs when data is sorted in reverse order.
* So we must have to compare A[j] with each element of sorted array A[1 … j – 1]. So, tj = j

A math equations on a white background

Description automatically generated



**Code:**

**#include <stdio.h>**

**void insertionSort(int arr[], int n) {**

**int i, key, j;**

**for (i = 1; i < 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;**

**}**

**}**

**int main() {**

**int n;**

**printf("Enter number of elements: ");**

**scanf("%d", &n);**

**int arr[n];**

**printf("Enter %d integers: ", n);**

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

**scanf("%d", &arr[i]);**

**}**

**insertionSort(arr, n);**

**printf("Sorted array: ");**

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

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

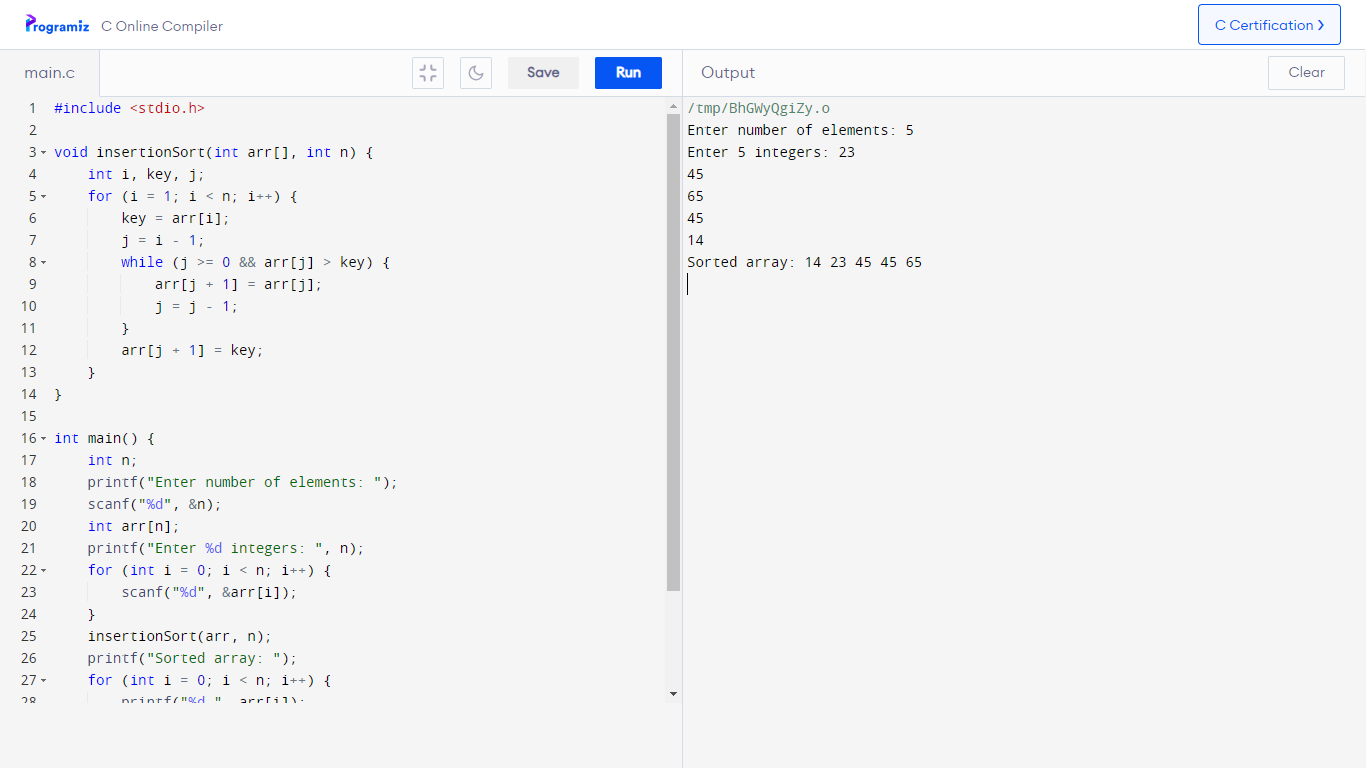
**}**

**printf("\n");**

**return 0;**

**}**

**Output:**



**Conclusion:**

**Insertion sort is a simple sorting algorithm that builds the final sorted array one element at a time. It repeatedly takes an element from the unsorted part and inserts it into its correct position in the sorted part of the array. Despite its simplicity, insertion sort is efficient for small datasets and has a relatively straightforward implementation. However, its time complexity is O(n^2), which makes it less suitable for large datasets compared to more efficient algorithms like merge sort or quicksort.**