| |
|---|
| Experiment No.1 |
| Insertion Sort |
| Date of Performance: |
| Date of Submission: |

## Experiment No.1

**Title**: Insertion Sort

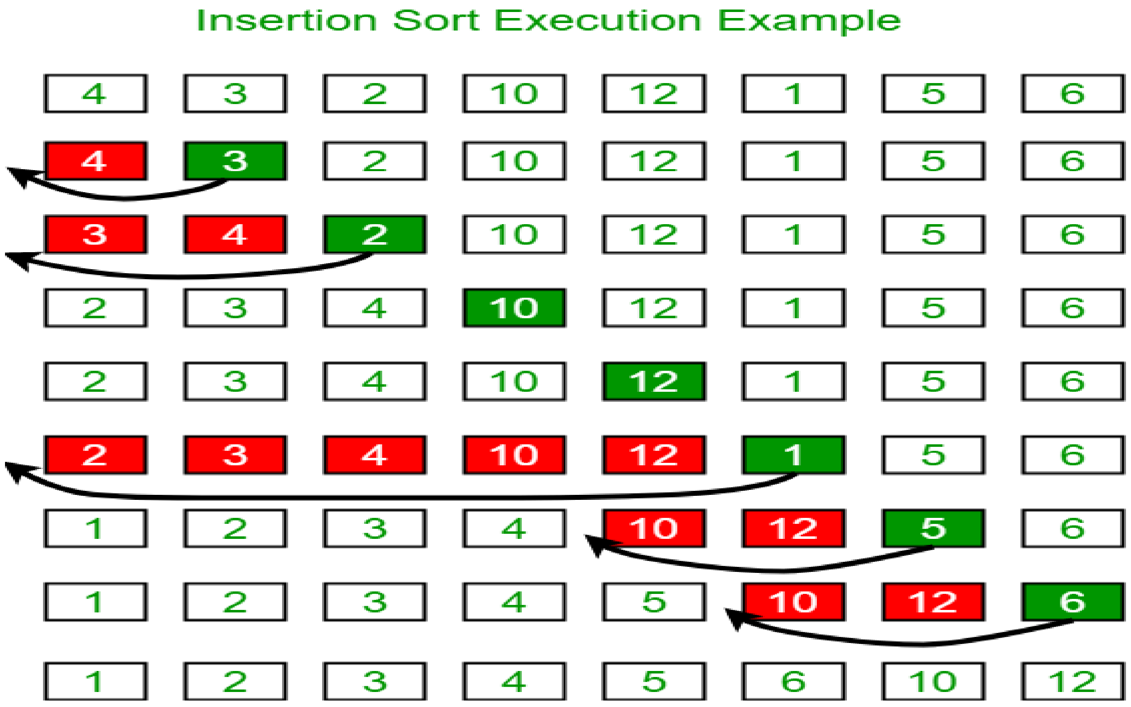
**Aim**: To implement Selection Comparative analysis for large values of 'n'

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

**Theory**:

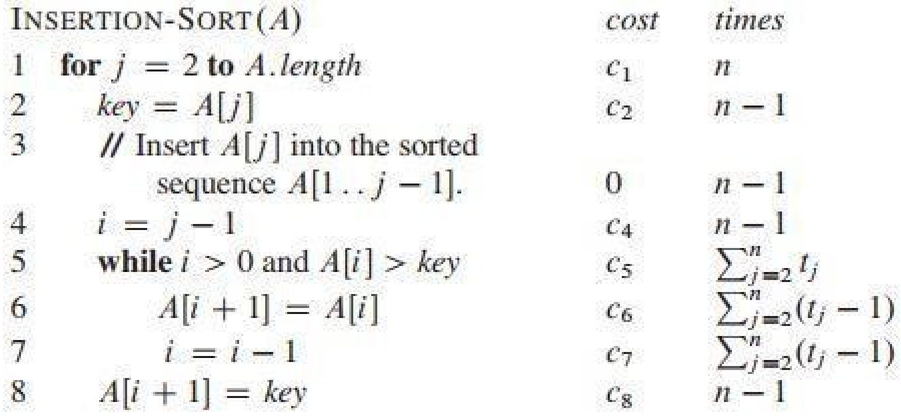
Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

**Example:**



```
4  3  2  10 12 1  5  6
4  3  2  10 12 1  5  6
3  4  2  10 12 1  5  6
2  3  4  10 12 1  5  6
2  3  4  10 12 1  5  6
2  3  4  10 12 1  5  6
1  2  3  4  10 12 5  6
1  2  3  4  5  10 12 6
1  2  3  4  5  6  10 12
```

**Algorithm and Complexity:**



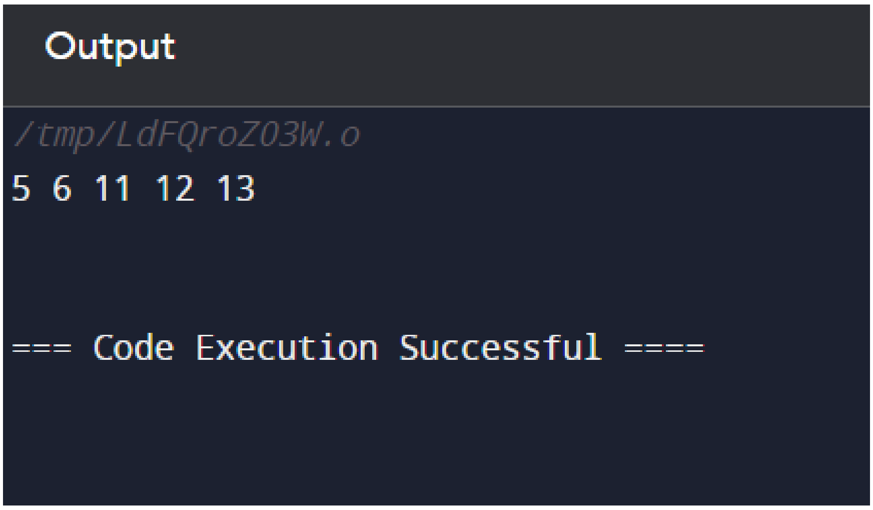
```
INSERTION-SORT(A)                                 cost    times
for $j = 2$ to $A.length$                         $c_1$   $n$
    $key = A[j]$                                  $c_2$   $n - 1$
    // Insert $A[j]$ into the sorted
          sequence $A[1 \,..\, j - 1]$.           0       $n - 1$
    $i = j - 1$                                   $c_4$   $n - 1$
    while $i > 0$ and $A[i] > key$                $c_5$   $\sum_{j=2}^{n} t_j$
        $A[i + 1] = A[i]$                         $c_6$   $\sum_{j=2}^{n} (t_j - 1)$
        $i = i - 1$                               $c_7$   $\sum_{j=2}^{n} (t_j - 1)$
    $A[i + 1] = key$                              $c_8$   $n - 1$
```

**Implementation:**

```
#include <math.h>
#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;
    }}
void printArray(int arr[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
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:**



```
/tmp/LdFQroZO3W.o
5 6 11 12 13


=== Code Execution Successful ====
```

**Conclusion:** Insertion Sort is a straightforward sorting algorithm with a manageable implementation and good performance characteristics for small datasets or nearly sorted arrays. However, its quadratic time complexity limits its efficiency for larger datasets compared to more advanced sorting algorithms.