Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Insertion Sort/5a12e8a046765b2b63e3474e

TUTORIAL

Insertion Sort

Topics

One of the simplest methods to sort an array is an insertion sort. An example of an insertion sort occurs in everyday life while playing cards. To sort the cards in your hand you extract a card, shift the remaining cards, and then insert the extracted card in the correct place. This process is repeated until all the cards are in the correct sequence. Following algorithm will describe the insertion sort procedure: -

```
for i = 1 to n-1
   Pick element at position i
   insert it into sorted sequence from index 0 to i-1.
end
```

Let's take the below example: -

```
15   11   14   12   18
```

To sort this array in ascending order, Insertion sort will perform following steps: -

```
When i=1
```

15 is the element, first element is always sorted so nothing to do.

```
When i=2
```

11 is the element. Now to insert 11 in array from 0 to 1 (i-1=1) we have to shift 15 and then insert 11. So array is

```
11   15   14   12   18
When i=3
```

14 is the element. Now to insert 14 in array from 0 to 2 (i-1=2) we have to shift 15 and then insert 14. So array is

```
11   14   15   12   18
When i=4
```

12 is the element. Now to insert 12 in array from 0 to 3 (i-1=3) we have to shift 15, 14 and then insert 12. So array is

```
11   12   14   15   18
When i=5
```

18 is the element. Now to insert 18 in array from 0 to 4 (i-1=4) we do not have to shift any number as 18 is the largest number. So array is

```
11   12   14   15   18
```

Following is the implementation of insertion sort: -

Insertion Sort Code

Javascript

```
function insertionSort(array,n){
  let i, key, j;
  for (i = 0; i < n; i++){
    key = array[i];
    j = i-1;
    while (j >= 0 && array[j] > key)
    {                    // find the correct position of the element
      array[j+1] = array[j];        // shift all lesser elements
      j = j-1;
    }
    array[j+1] = key;           // place the element at position
```

```
  }
}

function main(){
  let A = [15,11,14,12,18]
  console.log('Unsorted Array:')
  console.log(A.join(' '));
  console.log()

  insertionSort(A,A.length);

  console.log('Sorted Array');
  console.log(A.join(' '));
}

main()
```

C

```
#include<stdio.h>

void printArray(int array[], int size){
  int i;
  for (i=0; i < size; i++)
    printf("%d ", array[i]);
  printf("\n");
}

void insertionSort(int array[], int n){
  int i, key, j;
  for (i = 0; i < n; i++){
    key = array[i];
    j = i-1;
    while (j >= 0 && array[j] > key){
      // find the correct position of the element
      array[j+1] = array[j];        // shift all lesser elements
      j = j-1;
    }
    array[j+1] = key;           // place the element at position

  }
```

```
}

int main()
{
  int array[] = {15, 11, 14, 12, 18};
  int n = 5;
  /* we can calculate the number of elements in an array by using sizeof(array)/sizeof(array[0]).
       */
  printf("Un-Sorted array: \n");
  printArray(array, n);     // Unsorted array
  insertionSort(array, n);     // Call the sorting routine
  printf("\nSorted array: \n");
  printArray(array, n);   // Sorted array
  return 0;
}

```

Java

```
import java.util.Scanner;
// Other imports go here
// Do NOT change the class name
class Main{
    static void printArray(int array[], int size){
        int i;
        for (i=0; i < size; i++)
            System.out.printf("%d ", array[i]);
        System.out.printf("\n");
    }

    static void insertionSort(int array[], int n)
    {
        int i, key, j;
        for (i = 0; i < n; i++){
            key = array[i];
            j = i-1;

            while (j >= 0 && array[j] > key)
            {                    // find the correct position of the element
                array[j+1] = array[j];        // shift all lesser elements
                j = j-1;
```

```
            }
            array[j+1] = key;           // place the element at position
        }
    }

    public static void main(String[] args){
        int array[] = {15, 11, 14, 12, 18};
        int n = 5;
        /* we can calculate the number of elements in an array by using sizeof(array)/sizeof(array[0]).*/
        System.out.printf("Un-Sorted array: \n");
        printArray(array, n);     // Unsorted array
        insertionSort(array, n);     // Call the sorting routine
        System.out.printf("\nSorted array: \n");
        printArray(array, n);   // Sorted array
    }
}

```

Python 3

```
def printArray(A,size):
    for i in range(size):
        print(A[i],end=' ');
    print()

def insertionSort(array,size):
    for i in range(1, size):
        key = array[i]
        j = i-1
        while j >=0 and key < array[j] :
            array[j+1] = array[j]
            j -= 1
        array[j+1] = key

if __name__=="__main__":
    A = [15,11,14,12,18]
    print('Unsorted Array:')
    printArray(A,len(A));
    print()

```

```
    insertionSort(A,len(A));

    print('Sorted Array');
    printArray(A,len(A));
```

C++

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

void printArray(int array[], int size){
    int i;
    for (i=0; i < size; i++)
        cout<<array[i]<<" ";
    cout<<endl;
}

void insertionSort(int array[], int n){
    int i, key, j;
    for (i = 0; i < n; i++){
        key = array[i];
        j = i-1;
        while (j >= 0 && array[j] > key)
        {                       // find the correct position of the element
            array[j+1] = array[j];        // shift all lesser elements

            j = j-1;
        }
        array[j+1] = key;           // place the element at position
    }
}

int main(){
    int array[] = {15, 11, 14, 12, 18};
    int n = 5;
    /* we can calculate the number of elements in an array by using sizeof(array)/sizeof(array[0]).*/
    cout<<"Un-Sorted array:"<<endl;
    printArray(array, n);     // Unsorted array
    insertionSort(array, n);     // Call the sorting routine
    cout<<endl<<"Sorted array:"<<endl;
```

```
    printArray(array, n);   // Sorted array
    return 0;
}

```

Output of above program: -

```
Un-Sorted array:
15 11 14 12 18

Sorted array:
11 12 14 15 18
```

The output above shows the elements shifted in each pass and final array after each pass. The algorithm goes to inner loop only if required. So, the time complexity of this algorithm is O(n^2) if both loops in insertionSort() function will run n times. Otherwise the complexity will be O(n) if array is already sorted.

Properties of Insertion sort

Worst and Average Case Time Complexity: O(n^2). Worst case occurs when array is sorted in opposite direction.

Best Case Time Complexity: O(n). Best case occurs when array is already sorted.

Auxiliary Space: O(1)

Sorting In Place: Yes

Stable: Yes

Insertion sort is used when number of elements is small. It can also be useful when input array is almost sorted, only few elements are misplaced in complete big array.

Video Solution

```
<iframe width="560" height="315"
src="https://www.youtube.com/embed/_fXogocanDQ"
title="YouTube video player" frameborder="0" allow="accelerometer;
autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-
picture" allowfullscreen></iframe>
```