- DSA Introduction
- Data Structures (I)
- Data Structures (II)
- Tree based DSA (I)
- Tree based DSA (II)
- Graph based DSA
- Sorting and Searching Algorithms
  - Bubble Sort
  - Selection Sort
  - **Insertion Sort**
  - Merge Sort
  - Quicksort
  - Counting Sort
  - Radix Sort
  - Bucket Sort
  - Heap Sort
  - Shell Sort
  - Linear Search
  - Binary Search
- Greedy Algorithms
- Dynamic Programming
- Other Algorithms

Insertion Sort Algorithm

In this tutorial, you will learn about insertion sort algorithm and its implementation in C, C++, Java and Python.

Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration.

Insertion sort works similarly as we sort cards in our hand in a card game.

We assume that the first card is already sorted then, we select an unsorted card. If the unsorted card is greater than the card in hand, it is placed on the right otherwise, to the left. In the same way, other unsorted cards are taken and put in their right place.

A similar approach is used by insertion sort.

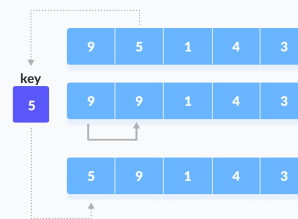## Working of Insertion Sort

Suppose we need to sort the following array.

| 9 | 5 | 1 | 4 | 3 |

Initial array

1. The first element in the array is assumed to be sorted. Take the second element and store it separately in `key` .

   Compare `key` with the first element. If the first element is greater than `key` , then `key` is placed in front of the first element.
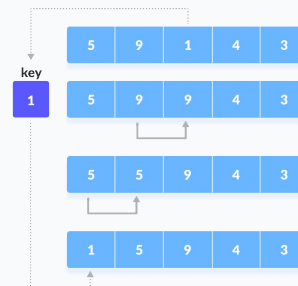

   step = 1

   If the first element is greater than key, then key is placed in front of the first element.

2. Now, the first two elements are sorted.

   Take the third element and compare it with the elements on the left of it. Placed it just behind the element smaller than it. If there is no element smaller than it, then place it at the beginning of the array.

   step = 2

   Place 1 at the beginning
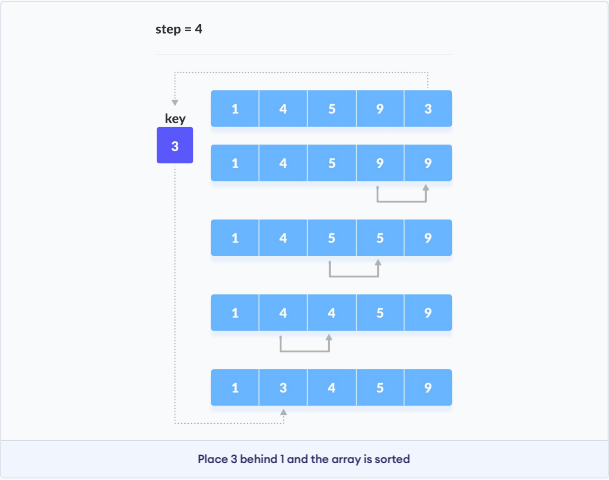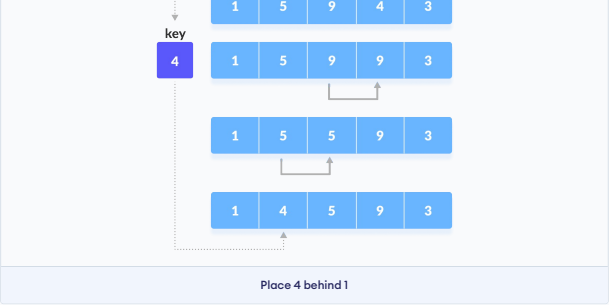
3. Similarly, place every unsorted element at its correct position.

   step = 3

Related Topics

Selection Sort Algorithm

Bubble Sort

Shell Sort Algorithm

Counting Sort Algorithm

Radix Sort Algorithm

Quicksort Algorithm

| | 1 | 5 | 9 | 4 | 3 |

key
4

| | 1 | 5 | 9 | 9 | 3 |

| | 1 | 5 | 5 | 9 | 3 |

| | 1 | 4 | 5 | 9 | 3 |

Place 4 behind 1

step = 4

| | 1 | 4 | 5 | 9 | 3 |

key
3

| | 1 | 4 | 5 | 9 | 9 |

| | 1 | 4 | 5 | 5 | 9 |

| | 1 | 4 | 4 | 5 | 9 |

| | 1 | 3 | 4 | 5 | 9 |

Place 3 behind 1 and the array is sorted

Insertion Sort Algorithm

```
insertionSort(array)
  mark first element as sorted
  for each unsorted element X
    'extract' the element X
    for j <- lastSortedIndex down to 0
      if current element j > X
        move sorted element to the right by 1
    break loop and insert X here
end insertionSort
```

Insertion Sort in Python, Java, and C/C++

| Python | Java | C | C++ |

```java
// Insertion sort in Java

import java.util.Arrays;

class InsertionSort {

  void insertionSort(int array[]) {
    int size = array.length;

    for (int step = 1; step < size; step++) {
      int key = array[step];
      int j = step - 1;

      // Compare key with each element on the left of it until an element smaller than
      // it is found.
      // For descending order, change key<array[j] to key>array[j].
      while (j >= 0 && key < array[j]) {
        array[j + 1] = array[j];
        --j;
      }

      // Place key at after the element just smaller than it.
      array[j + 1] = key;
    }
  }

  // Driver code
  public static void main(String args[]) {
    int[] data = { 9, 5, 1, 4, 3 };
```

Insertion Sort Complexity

Time Complexity	
Best	O(n)
Worst	$O(n^2)$
Average	$O(n^2)$
Space Complexity	O(1)
Stability	Yes

Time Complexities

- **Worst Case Complexity:** $O(n^2)$

  Suppose, an array is in ascending order, and you want to sort it in descending order. In this case, worst case complexity occurs.

  Each element has to be compared with each of the other elements so, for every nth element, $(n-1)$ number of comparisons are made.

  Thus, the total number of comparisons = $n*(n-1) \sim n^2$

- **Best Case Complexity:** $O(n)$

  When the array is already sorted, the outer loop runs for n number of times whereas the inner loop does not run at all. So, there are only n number of comparisons. Thus, complexity is linear.

- **Average Case Complexity:** $O(n^2)$

  It occurs when the elements of an array are in jumbled order (neither ascending nor descending).

Space Complexity

Space complexity is $O(1)$ because an extra variable key is used.

Insertion Sort Applications

The insertion sort is used when:

- the array is has a small number of elements
- there are only a few elements left to be sorted

Similar Sorting Algorithms

1. Bubble Sort
2. Quicksort
3. Merge Sort
4. Selection Sort

Share on:

Did you find this article helpful?

😀 🙁

Related Tutorials

DS & Algorithms

Selection Sort Algorithm

DS & Algorithms

Bubble Sort

DS & Algorithms

Shell Sort Algorithm

DS & Algorithms

Counting Sort Algorithm

Tutorials

Python 3 Tutorial
JavaScript Tutorial
SQL Tutorial
C Tutorial
Java Tutorial
Kotlin Tutorial
C++ Tutorial
Swift Tutorial
C# Tutorial
Go Tutorial
DSA Tutorial

Examples

Python Examples
JavaScript Examples
C Examples
Java Examples
Kotlin Examples
C++ Examples

Company

About
Advertising
Privacy Policy
Terms & Conditions
Contact
Blog
Youtube

Apps

Learn Python
Learn C Programming
Learn Java