Insertion Sort Algorithm - GeeksforGeeks

Source: https://www.geeksforgeeks.org/insertion-sort/

Courses Tutorials Practice Jobs DSA Tutorial Interview Questions Quizzes Must Do Advanced DSA System Design Aptitude Puzzles Interview Corner DSA Python Technical Scripter 2026 Explore DSA Fundamentals Logic Building Problems Analysis of Algorithms Data Structures Array Data Structure String in Data Structure Hashing in Data Structure Linked List Data Structure Stack Data Structure Queue Data Structure Tree Data Structure Graph Data Structure Trie Data Structure Algorithms Searching Algorithms Sorting Algorithms Introduction to Recursion Greedy Algorithms Tutorial Graph Algorithms Dynamic Programming or DP Bitwise Algorithms Advanced Segment Tree Binary Indexed Tree or Fenwick Tree Square Root (Sqrt) Decomposition Algorithm Binary Lifting Geometry Interview Preparation Interview Corner GfG160 Practice Problem GeeksforGeeks Practice - Leading Online Coding Platform Problem of The Day - Develop the Habit of Coding DSA Course 90% Refund Insertion Sort Algorithm Last Updated : 6 Dec, 2025 Insertion sort is a simple sorting algorithm that works by iteratively inserting each element of an unsorted list into its correct position in a sorted portion of the list. It is like sorting playing cards in your hands. You split the cards into two groups: the sorted cards and the unsorted cards. Then, you pick a card from the unsorted group and put it in the right place in the sorted group. Start with the second element as the first element is assumed to be sorted. Compare the second element with the first if the second is smaller then swap them. Move to the third element, compare it with the first two, and put it in its correct position Repeat until the entire array is sorted. Try it on GfG Practice C++ // C++ program for implementation of Insertion Sort #include <iostream> using namespace std ; /* Function to sort array using insertion sort */ void insertionSort (int arr [], int n) { for (int i = 1 ; i < n ; ++ i) { int key = arr [i]; int 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 to print array of size n */ void printArray (int arr [], int n) { for (int i = 0 ; i < n ; ++ i) cout << arr [i] << " " ; cout << endl ; } // Driver method int main () { int arr [] = { 12 , 11 , 13 , 5 , 6 }; int n = sizeof (arr) / sizeof (arr [0]); insertionSort (arr , n); printArray (arr , n); return 0 ; } /* This code is contributed by Hritik Shah. */ C // C program for implementation of Insertion Sort #include <stdio.h> /* Function to sort array using insertion sort */ void insertionSort (int arr [], int n) { for (int i = 1 ; i < n ; ++ i) { int key = arr [i]; int 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 to print array of size n */ void printArray (int arr [], int n) { for (int i = 0 ; i < n ; ++ i) printf ("%d " , arr [i]); printf (" \n "); } // Driver method int main () { int arr [] = { 12 , 11 , 13 , 5 , 6 }; int n = sizeof (arr) / sizeof (arr [0]); insertionSort (arr , n); printArray (arr , n); return 0 ; } /* This code is contributed by Hritik Shah. */ Java // Java program for implementation of Insertion Sort public class InsertionSort { /* Function to sort array using insertion sort */ void sort (int arr []) { int n = arr . length ; for (int i = 1 ; i < n ; ++ i) { int key = arr [i] ; int 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 to print array of size n */ static void printArray (int arr []) { int n = arr . length ; for (int i = 0 ; i < n ; ++ i) System . out . print (arr [i] + " "); System . out . println (); } // Driver method public static void main (String args []) { int arr [] = { 12 , 11 , 13 , 5 , 6 }; InsertionSort ob = new InsertionSort (); ob . sort (arr); printArray (arr); } } /* This code is contributed by Hritik Shah. */ Python # Python program for implementation of Insertion Sort # Function to sort array using insertion sort def insertionSort (arr): for i in range (1 , len (arr)): 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 and key < arr [j]: arr [j + 1] = arr [j] j -= 1 arr [j + 1] = key # A utility function to print array of size n def printArray (arr): for i in range (len (arr)): print (arr [i], end = " ") print () # Driver method if __name__ == "__main__" : arr = [12 , 11 , 13 , 5 , 6] insertionSort (arr) printArray (arr) # This code is contributed by Hritik Shah. C# // C# program for implementation of Insertion Sort using System ; class InsertionSort { /* Function to sort array using insertion sort */ void sort (int [] arr) { int n = arr . Length ; for (int i = 1 ; i < n ; ++ i) { int key = arr [i]; int 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 to print array of size n */ static void printArray (int [] arr) { int n = arr . Length ; for (int i = 0 ; i <

n ; ++ i) Console . Write (arr [i] + " "); Console . WriteLine (); } // Driver method public static void Main () { int [] arr = { 12 , 11 , 13 , 5 , 6 }; InsertionSort ob = new InsertionSort (); ob . sort (arr); printArray (arr); } } /* This code is contributed by Hritik Shah. */ JavaScript // Javascript program for insertion sort // Function to sort array using insertion sort function insertionSort (arr) { for (let i = 1 ; i < arr . length ; i ++) { let key = arr [i]; let 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 to print array of size n function printArray (arr) { console . log (arr . join (" ")); } // Driver method let arr = [12 , 11 , 13 , 5 , 6]; insertionSort (arr); printArray (arr); // This code is contributed by Hritik Shah. PHP <?php // PHP program for insertion sort // Function to sort an array using insertion sort function insertionSort (& $arr , $n) { for ($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 to print an array of size n function printArray (& $arr , $n) { for ($i = 0 ; $i < $n ; $i ++) echo $arr [$i] . " " ; echo " \n " ; } // Driver Code $arr = array (12 , 11 , 13 , 5 , 6); $n = sizeof ($arr); insertionSort ($arr , $n); printArray ($arr , $n); // This code is contributed by Hritik Shah. ?> Output 5 6 11 12 13 Illustration arr = {23, 1, 10, 5, 2} Initial: Current element is 23 The first element in the array is assumed to be sorted. The sorted part until 0th index is : [23] First Pass: Compare 1 with 23 (current element with the sorted part). Since 1 is smaller, insert 1 before 23 . The sorted part until 1st index is: [1, 23] Second Pass: Compare 10 with 1 and 23 (current element with the sorted part). Since 10 is greater than 1 and smaller than 23 , insert 10 between 1 and 23 . The sorted part until 2nd index is: [1, 10, 23] Third Pass: Compare 5 with 1 , 10 , and 23 (current element with the sorted part). Since 5 is greater than 1 and smaller than 10 , insert 5 between 1 and 10 The sorted part until 3rd index is : [1, 5, 10, 23] Fourth Pass: Compare 2 with 1, 5, 10 , and 23 (current element with the sorted part). Since 2 is greater than 1 and smaller than 5 insert 2 between 1 and 5 . The sorted part until 4th index is: [1, 2, 5, 10, 23] Final Array: The sorted array is: [1, 2, 5, 10, 23] Complexity Analysis of Insertion Sort Time Complexity Best case: O(n) , If the list is already sorted, where n is the number of elements in the list. Average case: O(n 2) , If the list is randomly ordered Worst case: O(n 2) , If the list is in reverse order Space Complexity Auxiliary Space: O(1), Insertion sort requires O(1) additional space, making it a space-efficient sorting algorithm. Please refer Complexity Analysis of Insertion Sort for details. Advantages and Disadvantages of Insertion Sort Advantages Simple and easy to implement. Stable sorting algorithm. Efficient for small lists and nearly sorted lists. Space-efficient as it is an in-place algorithm. Adoptive. the number of inversions is directly proportional to number of swaps. For example, no swapping happens for a sorted array and it takes O(n) time only. Disadvantages Inefficient for large lists. Not as efficient as other sorting algorithms (e.g., merge sort, quick sort) for most cases. Applications of Insertion Sort Insertion sort is commonly used in situations where: The list is small or nearly sorted. Simplicity and stability are important. Used as a subroutine in Bucket Sort Can be useful when array is already almost sorted (very few inversions) Since Insertion sort is suitable for small sized arrays, it is used in Hybrid Sorting algorithms along with other efficient algorithms like Quick Sort and Merge Sort. When the subarray size becomes small, we switch to insertion sort in these recursive algorithms. For example IntroSort and TimSort use insertions sort. What are the Boundary Cases of the Insertion Sort algorithm? Insertion sort takes the maximum time to sort if elements are sorted in reverse order. And it takes minimum time (Order of n) when elements are already sorted. What is the Algorithmic Paradigm of the Insertion Sort algorithm? The Insertion Sort algorithm follows an incremental approach. Is Insertion Sort an in-place sorting algorithm? Yes, insertion sort is an in-place sorting algorithm. Is Insertion Sort a stable algorithm? Yes, insertion sort is a stable sorting algorithm. When is the Insertion Sort algorithm used? Insertion sort is used when number of elements is small. It can also be useful when the input array is almost sorted, and only a few elements are misplaced in a complete big array. Comment Article Tags: Article Tags: Sorting DSA Cisco Dell MAQ Software Juniper Networks Grofers Veritas Accenture + 5 More