Bubble Sort Algorithm

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

How does Bubble Sort Work?

Bubble Sort

Input: *arr[] = {6, 3, 0, 5}*

First Pass:

- *Bubble sort starts with very first two elements, comparing them to check which one is greater.*

- *(3 0 **6 5**) –> (3 0 **5 6**), Swap since 6 > 5*

Second Pass:

- *Now, during second iteration it should look like this:*
  - *(**3 0** 5 6) –> (**0 3** 5 6), Swap since 3 > 0*
  - *(0 **3 5** 6) –> (0 **3 5** 6), No change as 5 > 3*

Third Pass:

- *Now, the array is already sorted, but our algorithm does not know if it is completed.*
- *The algorithm needs one **whole** pass without **any** swap to know it is sorted.*
  - *(**0 3** 5 6) –> (**0 3** 5 6), No change as 3 > 0*

Array is now sorted and no more pass will happen.

Follow the below steps to solve the problem:

- Run a nested for loop to traverse the input array using two variables **i** and **j**, such that 0 ≤ i < n-1 and 0 ≤ j < n-i-1
- If **arr[j]** is greater than **arr[j+1]** then swap these adjacent elements, else move on
- Print the sorted array

Below is the implementation of the above approach:

C

```
// C program for implementation of Bubble sort
#include <stdio.h>

void swap(int* xp, int* yp)
{
```

```c
// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)

        // Last i elements are already in place
        for (j = 0; j < n - i - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(&arr[j], &arr[j + 1]);
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver program to test above functions
int main()
{
    int arr[] = { 5, 1, 4, 2, 8 };
    int n = sizeof(arr) / sizeof(arr[0]);
    bubbleSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

C++

```cpp
// C++ program for implementation
// of Bubble sort
#include <bits/stdc++.h>
using namespace std;

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n - 1; i++)

        // Last i elements are already
        // in place
```

```
// Function to print an array
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

// Driver code
int main()
{
    int arr[] = { 5, 1, 4, 2, 8};
    int N = sizeof(arr) / sizeof(arr[0]);
    bubbleSort(arr, N);
    cout << "Sorted array: \n";
    printArray(arr, N);
    return 0;
}
// This code is contributed by rathbhupendra
```

Java

```
// Java program for implementation of Bubble Sort
import java.util.*;

class BubbleSort {
    void bubbleSort(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++)
            for (int j = 0; j < n - i - 1; j++)
                if (arr[j] > arr[j + 1]) {
                    // swap arr[j+1] and arr[j]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
    }

    /* Prints the array */
    void printArray(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");
```

```
    public static void main(String args[])
    {
        BubbleSort ob = new BubbleSort();
        int arr[] = { 5, 1, 4, 2, 8 };
        ob.bubbleSort(arr);
        System.out.println("Sorted array");
        ob.printArray(arr);
    }
}
/* This code is contributed by Rajat Mishra */
```

Python3

```
# Python program for implementation of Bubble Sort


def bubbleSort(arr):
    n = len(arr)

    # Traverse through all array elements
    for i in range(n):

        # Last i elements are already in place
        for j in range(0, n-i-1):

            # traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]


# Driver code to test above
if __name__ == "__main__":
  arr = [5, 1, 4, 2, 8]

  bubbleSort(arr)

  print("Sorted array is:")
  for i in range(len(arr)):
      print("%d" % arr[i], end=" ")
```

C#

```
// C# program for implementation
```

```
    static void bubbleSort(int[] arr)
    {
        int n = arr.Length;
        for (int i = 0; i < n - 1; i++)
            for (int j = 0; j < n - i - 1; j++)
                if (arr[j] > arr[j + 1]) {
                    // swap temp and arr[i]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
    }

    /* Prints the array */
    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 = { 5, 1, 4, 2, 8};
        bubbleSort(arr);
        Console.WriteLine("Sorted array");
        printArray(arr);
    }
}

// This code is contributed by Sam007
```

PHP

```
<?php
// PHP program for implementation
// of Bubble Sort

function bubbleSort(&$arr)
{
    $n = sizeof($arr);

    // Traverse through all array elements
    for($i = 0; $i < $n; $i++)
    {
```

```
            // Swap if the element found is greater
            // than the next element
            if ($arr[$j] > $arr[$j+1])
            {
                $t = $arr[$j];
                $arr[$j] = $arr[$j+1];
                $arr[$j+1] = $t;
            }
        }
    }
}

// Driver code to test above
$arr = array(5, 1, 4, 2, 8);

$len = sizeof($arr);
bubbleSort($arr);

echo "Sorted array : \n";

for ($i = 0; $i < $len; $i++)
    echo $arr[$i]." ";

// This code is contributed by ChitraNayal.
?>
```

Javascript

```
function swap(arr, xp, yp)
{
    var temp = arr[xp];
    arr[xp] = arr[yp];
    arr[yp] = temp;
}

// An optimized version of Bubble Sort
function bubbleSort( arr, n)
{
var i, j;
for (i = 0; i < n-1; i++)
{
    for (j = 0; j < n-i-1; j++)
    {
        if (arr[j] > arr[j+1])
        {
        swap(arr,j,j+1);
```

```
}
}

/* Function to print an array */
function printArray(arr, size)
{
    var i;
    for (i=0; i < size; i++)
        document.write(arr[i]+ " ");
    document.write("\n");
}

// Driver program to test above functions
  var arr = [5, 1, 4, 2, 8];
    var n = 5;
    document.write("UnSorted array: \n");
    printArray(arr, n);

    bubbleSort(arr, n);
    document.write("Sorted array: \n");
    printArray(arr, n);
```

Output

```
Sorted array:
1 2 4 5 8
```

Time Complexity: O(N^2)

Auxiliary Space: O(1)

Optimized Implementation of Bubble Sort:

The above function always runs **O(N^2)** time even if the array is sorted. It can be optimized by stopping the algorithm if the inner loop didn't cause any swap.

Below is the implementation for the above approach:

C

```
// Optimized implementation of Bubble sort
#include <stdio.h>
#include <stdbool.h>
```

```
    *xp = *yp;
    *yp = temp;
}

// An optimized version of Bubble Sort
void bubbleSort(int arr[], int n)
{
   int i, j;
   bool swapped;
   for (i = 0; i < n-1; i++)
   {
     swapped = false;
     for (j = 0; j < n-i-1; j++)
     {
        if (arr[j] > arr[j+1])
        {
           swap(&arr[j], &arr[j+1]);
           swapped = true;
        }
     }

     // IF no two elements were swapped by inner loop, then break
     if (swapped == false)
        break;
   }
}

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
}

// Driver program to test above functions
int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

C++

```
// An optimized version of Bubble Sort
void bubbleSort(int arr[], int n)
{
   int i, j;
   bool swapped;
   for (i = 0; i < n-1; i++)
   {
     swapped = false;
     for (j = 0; j < n-i-1; j++)
     {
        if (arr[j] > arr[j+1])
        {
           swap(arr[j], arr[j+1]);
           swapped = true;
        }
     }

     // IF no two elements were swapped
     // by inner loop, then break
     if (swapped == false)
        break;
   }
}

// Function to print an array
void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        cout <<" "<< arr[i];
}

// Driver program to test above functions
int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int N = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, N);
    cout <<"Sorted array: \n";
    printArray(arr, N);
    return 0;
}
// This code is contributed by shivanisinghss2110
```

Java

```java
class GFG
{
    // An optimized version of Bubble Sort
    static void bubbleSort(int arr[], int n)
    {
        int i, j, temp;
        boolean swapped;
        for (i = 0; i < n - 1; i++)
        {
            swapped = false;
            for (j = 0; j < n - i - 1; j++)
            {
                if (arr[j] > arr[j + 1])
                {
                    // swap arr[j] and arr[j+1]
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }

            // IF no two elements were
            // swapped by inner loop, then break
            if (swapped == false)
                break;
        }
    }

    // Function to print an array
    static void printArray(int arr[], int size)
    {
        int i;
        for (i = 0; i < size; i++)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    // Driver program
    public static void main(String args[])
    {
        int arr[] = { 64, 34, 25, 12, 22, 11, 90 };
        int n = arr.length;
        bubbleSort(arr, n);
        System.out.println("Sorted array: ");
        printArray(arr, n);
    }
}
```

Python3

```
# Optimized Python program for implementation of Bubble Sort

def bubbleSort(arr):
    n = len(arr)
    # Traverse through all array elements
    for i in range(n):
        swapped = False

        # Last i elements are already in place
        for j in range(0, n-i-1):

            # traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True
        if (swapped == False):
            break


# Driver code to test above
if __name__ == "__main__":
    arr = [64, 34, 25, 12, 22, 11, 90]

    bubbleSort(arr)

    print("Sorted array is:")
    for i in range(len(arr)):
        print("%d" % arr[i], end=" ")

# This code is modified by Suraj krushna Yadav
```

C#

```
// Optimized C# implementation
// of Bubble sort
using System;

class GFG
{
    // An optimized version of Bubble Sort
    static void bubbleSort(int []arr, int n)
```

```
        {
            swapped = false;
            for (j = 0; j < n - i - 1; j++)
            {
                if (arr[j] > arr[j + 1])
                {
                    // swap arr[j] and arr[j+1]
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }

            // IF no two elements were
            // swapped by inner loop, then break
            if (swapped == false)
                break;
        }
    }

    // Function to print an array
    static void printArray(int []arr, int size)
    {
        int i;
        for (i = 0; i < size; i++)
            Console.Write(arr[i] + " ");
        Console.WriteLine();
    }

    // Driver method
    public static void Main()
    {
        int []arr = {64, 34, 25, 12, 22, 11, 90};
        int n = arr.Length;
        bubbleSort(arr,n);
        Console.WriteLine("Sorted array");
        printArray(arr,n);
    }

}
// This code is contributed by Sam007
```

PHP

```
<?php
// PHP Optimized implementation
```

```
{
    $n = sizeof($arr);

    // Traverse through all array elements
    for($i = 0; $i < $n; $i++)
    {
        $swapped = False;

        // Last i elements are already
        // in place
        for ($j = 0; $j < $n - $i - 1; $j++)
        {

            // traverse the array from 0 to
            // n-i-1. Swap if the element
            // found is greater than the
            // next element
            if ($arr[$j] > $arr[$j+1])
            {
                $t = $arr[$j];
                $arr[$j] = $arr[$j+1];
                $arr[$j+1] = $t;
                $swapped = True;
            }
        }

        // IF no two elements were swapped
        // by inner loop, then break
        if ($swapped == False)
            break;
    }
}

// Driver code to test above
$arr = array(64, 34, 25, 12, 22, 11, 90);
$len = sizeof($arr);
bubbleSort($arr);

echo "Sorted array : \n";

for($i = 0; $i < $len; $i++)
    echo $arr[$i]." ";

// This code is contributed by ChitraNayal.
?>
```

Javascript

```
// An optimized version of Bubble Sort
function bubbleSort(arr, n)
    {
        var i, j, temp;
        var swapped;
        for (i = 0; i < n - 1; i++)
        {
            swapped = false;
            for (j = 0; j < n - i - 1; j++)
            {
                if (arr[j] > arr[j + 1])
                {
                    // swap arr[j] and arr[j+1]
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }

            // IF no two elements were
            // swapped by inner loop, then break
            if (swapped == false)
                break;
        }
    }

    // Function to print an array
    function printArray(arr, size)
    {
        var i;
        for (i = 0; i < size; i++)
            document.write(arr[i] + " ");
        document.writeln();
    }

    // Driver program
        var arr = [ 64, 34, 25, 12, 22, 11, 90 ];
        var n = arr.length;
        bubbleSort(arr, n);
        document.write("Sorted array: ");
        printArray(arr, n);

// This code is contributed shivanisinghss2110
```

Output

Time Complexity: $O(N^2)$

Auxiliary Space: O(1)

Worst Case Analysis for Bubble Sort:

The **worst-case** condition for bubble sort occurs when elements of the array are arranged in decreasing order.

In the worst case, the total number of iterations or passes required to sort a given array is **(n-1).** where 'n' is a number of elements present in the array.

At pass 1 : *Number of comparisons = (n-1)*
Number of swaps = (n-1)

At pass 2 : *Number of comparisons = (n-2)*
Number of swaps = (n-2)

At pass 3 : *Number of comparisons = (n-3)*
Number of swaps = (n-3)

.
.
.

At pass n-1 : *Number of comparisons = 1*
Number of swaps = 1

Now , calculating total number of comparison required to sort the array
= (n-1) + (n-2) + (n-3) + . . . 2 + 1
= (n-1)(n-1+1)/2 { by using sum of N natural Number formula }*
*= **n (n-1)/2***

For the Worst case:

Total number of swaps = Total number of comparison
*Total number of comparison (Worst case) = **n(n-1)/2***
*Total number of swaps (Worst case) = **n(n-1)/2***

Worst and Average Case Time Complexity: *$O(N^2)$. The worst case occurs when an array is reverse sorted.*
Best Case Time Complexity: *O(N). The best case occurs when an array is already sorted.*
Auxiliary Space: *O(1)*

Recursive Implementation Of Bubble Sort:

The idea is to place the largest element in its position and keep doing the same for every other element.

Algorithm:

1. Start with an array of unsorted numbers
2. Define a function called "**bubbleSort**" that takes in the array and the length of the array as parameters
3. In the function, create a variable called "**sorted**" that is set to true
4. Create a for loop that iterates through the array starting at index **0** and ending at the length of the array **-1**
5. Within the for loop, compare the current element with the next element in the array
6. If the current element is greater than the next element, swap their positions and set "**sorted**" to false
7. After the for loop, check if "**sorted**" is false
8. If "**sorted**" is false, call the "**bubbleSort**" function again with the same array and length as parameters
9. If "**sorted**" is true, the array is now sorted and the function will return the sorted array
10. Call the "**bubbleSort**" function with the initial unsorted array and its length as parameters to begin the sorting process.

C

```c
#include "stdbool.h"
#include <stdio.h>

void bubbleSort(int a[], int n)
{
    bool sorted = true;
    // we are assuming that array is sorted

    for (int i = 0; i < n - 1; i++) {
        if (a[i] > a[i + 1]) {
            int t = a[i];
            a[i] = a[i + 1];
            a[i + 1] = t;

            sorted = false;
            // now array is not sorted
        }
        // if there are no swaps then we can
        // say that array is sorted.
    }
    if (sorted == false) {
        // recursively calling until it was sorted.
        bubbleSort(a, n);
    }
}

int main()
{

    int ar[] = { 5, 4, 8, 2, 9, 7, 3 };
    int n = sizeof(ar) / sizeof(int);
    bubbleSort(ar, n);

    printf("Sorted array : ");
    for (int i = 0; i < n; i++) {
        printf("%d ", ar[i]);
    }
    printf("\n");

    return 0;
}

// This code is contributed by aeroabrar_31
```

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

void bubbleSort(int a[], int n)
{
    bool sorted = true;
    //we are assuming that array is sorted

    for (int i = 0; i < n - 1; i++) {
        if (a[i] > a[i + 1]) {
            int t = a[i];
            a[i] = a[i + 1];
            a[i + 1] = t;

            sorted = false;
          //now array is not sorted
        }
        //if there are no swaps then we can
        //say that array is sorted.
    }
    if (sorted == false)
    {
          //recursively calling until it was sorted.
        bubbleSort(a, n);
    }
}

int main()
{

    int ar[] = { 5, 4, 8, 2, 9, 7, 3 };
    int n = sizeof(ar) / sizeof(int);
    bubbleSort(ar, n);

    cout << "Sorted array : ";
    for (int i = 0; i < n; i++) {
        cout << ar[i] << " ";
    }
    cout << endl;

    return 0;
}
// This code is contributed by aeroabrar_31
```

Java

```
/*package whatever //do not write package name here */
```

```
    public static void main(String[] args) {

        int[] ar={5,4,8,2,9,7,3};
        bubbleSort(ar,ar.length);

        System.out.print("Sorted array : ");
        for(int ele:ar)
        {
            System.out.print(ele+" ");
        }
        System.out.println();

    }
    public static void bubbleSort(int[] a,int n)
    {
        boolean sorted=true;
      //we are assuming that array is sorted

        for(int i=0;i<n-1;i++)
        {
            if(a[i]>a[i+1])
            {
                int t=a[i];
                a[i]=a[i+1];
                a[i+1]=t;

                sorted=false;
              //now array is not sorted
            }
          //if there are no swaps then we can
          //say that array is sorted.

        }
        if(sorted==false)
        {
            //recursively calling until it was sorted.
            bubbleSort(a,n);
        }
    }
}
// This code is contributed by aeroabrar_31
```

Python3

```
def bubbleSort(a, n):

    sorted = True
```

```
                a[i], a[i+1] = a[i+1], a[i]
                sorted = False
                # now array is not sorted

            # if there are no swaps then we can
            # say that array is sorted.

        if sorted == False:
            #recursively calling until it was sorted.
            bubbleSort(arr, n)


# Driver code to test above
if __name__ == "__main__":
    arr = [5, 4, 8, 2, 9, 7, 3]
    n = len(arr)
    bubbleSort(arr, n)

    print("Sorted array : ", end=" ")
    for i in range(len(arr)):
        print("%d" % arr[i], end=" ")

# This code is contributed by aeroabrar_31
```

C#

```
using System;

public class GFG{

  public static void bubbleSort(int[] a,int n)
    {
        bool sorted=true;
      //we are assuming that array is sorted

        for(int i=0;i<n-1;i++)
        {
            if(a[i]>a[i+1])
            {
                int t=a[i];
                a[i]=a[i+1];
                a[i+1]=t;

                sorted=false;
              //now array is not sorted
            }
          //if there are no swaps then we can
```

```
        {
            //recursively calling until it was sorted.
            bubbleSort(a,n);
        }
    }

    public static void Main()
    {
        int []arr = {5,4,8,2,9,7,3};
        int n = arr.Length;
        bubbleSort(arr,n);
        Console.WriteLine("Sorted array : ");
        int i;
      for( i=0;i<n;i++)
      {
        Console.Write(arr[i]+" ");
      }
      Console.WriteLine();
    }

}
//This code is contributed by aeroabrar_31
```

Javascript

```
function bubbleSort(a, n) {
    let sorted = true; // we are assuming that array is sorted
    for (let i = 0; i < n - 1; i++) {
        if (a[i] > a[i + 1]) {
            let t = a[i];
            a[i] = a[i + 1];
            a[i + 1] = t;
            sorted = false; // now array is not sorted
        }
        // if there are no swaps then we can say that array is sorted.
    }
    if (sorted == false) {
        // recursively calling until it was sorted.
        bubbleSort(a, n);
    }
}

let ar = [5, 4, 8, 2, 9, 7, 3];
let n = ar.length;
bubbleSort(ar, n);

console.log("Sorted array : " + ar.join(" "));
```

```
Sorted array : 2 3 4 5 7 8 9
```

Time Complexity : **O (N^2)**

Auxiliary space : **O(1)**

What is the Boundary Case for Bubble sort?

Bubble sort takes minimum time (Order of n) when elements are already sorted. Hence it is best to check if the array is already sorted or not beforehand, to avoid O(N^2) time complexity.

Does sorting happen in place in Bubble sort?

Yes, Bubble sort performs the swapping of adjacent pairs without the use of any major data structure. Hence Bubble sort algorithm is an in-place algorithm.

Is the Bubble sort algorithm stable?

Yes, the bubble sort algorithm is stable.

Where is the Bubble sort algorithm used?

Due to its simplicity, bubble sort is often used to introduce the concept of a sorting algorithm.
In computer graphics, it is popular for its capability to detect a tiny error (like a swap of just two elements) in almost-sorted arrays and fix it with just linear complexity (2n).

Example: It is used in a polygon filling algorithm, where bounding lines are sorted by their x coordinate at a specific scan line (a line parallel to the x-axis), and with incrementing y their order changes (two elements are swapped) only at intersections of two lines (Source: Wikipedia)

Advantages:

- Bubble sort is easy to understand and implement.
- It does not require any additional memory space.
- It's adaptability to different types of data.
- It is a stable sorting algorithm, meaning that elements with the same key

- Bubble sort has a time complexity of O(n^2) which makes it very slow for large data sets.
- It is not efficient for large data sets, because it requires multiple passes through the data.
- Bubble sort is a comparison-based sorting algorithm, which means that it requires a comparison operator to determine the relative order of elements in the input data set. While this is not necessarily a disadvantage, it can limit the efficiency of the algorithm in certain cases.

Snapshots: Quiz on Bubble Sort

Other Sorting Algorithms on GeeksforGeeks/GeeksQuiz:
Recursive Bubble Sort
Coding practice for sorting.

Last Updated : 21 Apr, 2023 686

Similar Reads

1. Comparison among Bubble Sort, Selection Sort and Insertion Sort
2. Bubble Sort algorithm using JavaScript
3. Sort an array using Bubble Sort without using loops
4. Selection Sort VS Bubble Sort
5. C Program for Bubble Sort
6. Java Program for Bubble Sort
7. Python Program for Bubble Sort
8. C++ Program for Recursive Bubble Sort
9. Java Program for Recursive Bubble Sort

Related Tutorials

1. Learn Data Structures with Javascript | DSA Tutorial
2. Introduction to Max-Heap – Data Structure and Algorithm Tutorials
3. Introduction to Set – Data Structure and Algorithm Tutorials
4. Introduction to Map – Data Structure and Algorithm Tutorials
5. What is Dijkstra's Algorithm? | Introduction to Dijkstra's Shortest Path Algorithm

Previous Next

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy | Normal | Medium | Hard | Expert

Improved By : ukasp, rathbhupendra, SumitBM, TanishJain1, akshitsaxenaa09, shivanisinghss2110, adityakangs, ajaymakvana, vibhukarnwal077, sumitgumber28, anurag_pathak, amartyaghoshgfg, animeshdey, shreyasnaphad, pragatikohli12, kashishkumar2, harendrakumar123, kushalpareek, vaishalishrivastava21, janardansthox, arashmodatma7, suraj_the_coder, aeroabrar_31, akshitaguprzj3, rohitianaditya

Article Tags : redBus, DSA, Sorting

Practice Tags : redBus, Sorting

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

NodeJS

Computer Science

GATE CS Notes

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Data Science & ML

Data Science With Python

Data Science For Beginner

Machine Learning Tutorial

Maths For Machine Learning

Pandas Tutorial

NumPy Tutorial

NLP Tutorial

Interview Corner

Company Preparation

Preparation for SDE

Company Interview Corner

Experienced Interview

Internship Interview

Competitive Programming

Aptitude

Python

Python Tutorial

Python Programming Examples

Django Tutorial

Python Projects

Python Tkinter

OpenCV Python Tutorial

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

CBSE Notes for Class 10

CBSE Notes for Class 11

CBSE Notes for Class 12

English Grammar

UPSC/SSC/BANKING

SSC CGL Syllabus

SBI PO Syllabus

IBPS PO Syllabus

UPSC Ethics Notes

UPSC Economics Notes

UPSC History Notes