Rotate an Array by d - Counterclockwise or Left - GeeksforGeeks

Source: https://www.geeksforgeeks.org/array-rotation/

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 Rotate an Array by d - Counterclockwise or Left Last Updated : 23 Jul, 2025 Given an array of integers arr[] of size n , the task is to rotate the array elements to the left by d positions. Examples: Input: arr[] = {1, 2, 3, 4, 5, 6}, d = 2 Output: {3, 4, 5, 6, 1, 2} Explanation : After first left rotation, arr[] becomes {2, 3, 4, 5, 6, 1} and after the second rotation, arr[] becomes {3, 4, 5, 6, 1, 2} Input: arr[] = {1, 2, 3}, d = 4 Output: {2, 3, 1} Explanation: The array is rotated as follows: After first left rotation, arr[] = {2, 3, 1} After second left rotation, arr[] = {3, 1, 2} After third left rotation, arr[] = {1, 2, 3} After fourth left rotation, arr[] = {2, 3, 1} Try it on GfG Practice Table of Content [Naive Approach] Rotate one by one - O(n * d) Time and O(1) Space [Better Approach] Using Temporary Array - O(n) Time and O(n) Space [Expected Approach 1] Using Juggling Algorithm - O(n) Time and O(1) Space [Expected Approach 2] Using Reversal Algorithm - O(n) Time and O(1) Space [Naive Approach] Rotate one by one - O(n * d) Time and O(1) Space In each iteration, shift the elements by one position to the left in a circular fashion (the first element becomes the last). Perform this operation d times to rotate the elements to the left by d positions. Illustration: Let us take arr[] = {1, 2, 3, 4, 5, 6} , d = 2 . First Step: => Rotate to left by one position. => arr[] = {2, 3, 4, 5, 6, 1} Second Step: => Rotate again to left by one position => arr[] = {3, 4, 5, 6, 1, 2} Rotation is done 2 times. So the array becomes arr[] = {3, 4, 5, 6, 1, 2} C++ // C++ Program to left rotate the array by d positions // by rotating one element at a time #include <bits/stdc++.h> using namespace std ; // Function to left rotate array by d positions void rotateArr (vector < int >& arr , int d) { int n = arr . size (); // Repeat the rotation d times for (int i = 0 ; i < d ; i ++) { // Left rotate the array by one position int first = arr [0]; for (int j = 0 ; j < n - 1 ; j ++) { arr [j] = arr [j + 1]; } arr [n - 1] = first ; } } int main () { vector < int > arr = { 1 , 2 , 3 , 4 , 5 , 6 }; int d = 2 ; rotateArr (arr , d); for (int i = 0 ; i < arr . size (); i ++) cout << arr [i] << " " ; return 0 ; } C // C Program to left rotate the array by d positions // by rotating one element at a time #include <stdio.h> // Function to left rotate array by d positions // Repeat the rotation d times void rotateArr (int arr [], int n , int d) { for (int i = 0 ; i < d ; i ++) { // Left rotate the array by one position int first = arr [0]; for (int j = 0 ; j < n - 1 ; j ++) { arr [j] = arr [j + 1]; } arr [n - 1] = first ; } } int main () { int arr [] = { 1 , 2 , 3 , 4 , 5 , 6 }; int d = 2 ; int n = sizeof (arr) / sizeof (arr [0]); rotateArr (arr , n , d); for (int i = 0 ; i < n ; i ++) printf ("%d " , arr [i]); return 0 ; } Java // Java Program to left rotate the array by d positions // by rotating one element at a time import java.util.Arrays ; class GfG { // Function to left rotate array by d positions static void rotateArr (int [] arr , int d) { int n = arr . length ; // Repeat the rotation d times for (int i = 0 ; i < d ; i ++) { // Left rotate the array by one position int first = arr [0] ; for (int j = 0 ; j < n - 1 ; j ++) { arr [j] = arr [j + 1] ; } arr [n - 1] = first ; } } public static void main (String [] args) { int [] arr = { 1 , 2 , 3 , 4 , 5 , 6 }; int d = 2 ; rotateArr (arr , d); for (int i = 0 ; i < arr . length ; i ++) System . out . print (arr [i] + " "); } } Python # Python Program to left rotate the array by d positions # by rotating one element at a time # Function to left rotate array by d positions def rotateArr (arr , d): n = len (arr) # Repeat the rotation d times for i in range (d): # Left rotate the array by one position first = arr [0] for j in range (n - 1): arr [j] = arr [j + 1] arr [n - 1] = first if __name__ == "__main__" : arr = [1 , 2 , 3 , 4 , 5 , 6] d = 2 rotateArr (arr , d) for i in range (len (arr)): print (arr [i], end = " ") C# // C# Program to left rotate the array by d positions // by rotating one element at a time using System ; class GfG { // Function to left rotate array by d positions static void rotateArr (int [] arr , int d) { int n = arr . Length ; // Repeat the rotation d times for (int i = 0 ; i < d ; i ++) { // Left rotate the

array by one position int first = arr [0]; for (int j = 0 ; j < n - 1 ; j ++) { arr [j] = arr [j + 1]; } arr [n - 1] = first ; } } static void Main () { int [] arr = { 1 , 2 , 3 , 4 , 5 , 6 }; int d = 2 ; rotateArr (arr , d); for (int i = 0 ; i < arr . Length ; i ++) Console . Write (arr [i] + " "); } } Javascript // JavaScript Program to left rotate the array by d positions // by rotating one element at a time // Function to left rotate array by d positions function rotateArr (arr , d) { let n = arr . length ; // Repeat the rotation d times for (let i = 0 ; i < d ; i ++) { // Left rotate the array by one position let first = arr [0]; for (let j = 0 ; j < n - 1 ; j ++) { arr [j] = arr [j + 1]; } arr [n - 1] = first ; } } let arr = [1 , 2 , 3 , 4 , 5 , 6]; let d = 2 ; rotateArr (arr , d); console . log (arr . join (" ")); Output 3 4 5 6 1 2 Time Complexity: O(n*d), the outer loop runs d times, and within each iteration, the inner loop shifts all n elements of the array by one position, resulting in a total of n*d operations. Auxiliary Space: O(1) [Better Approach] Using Temporary Array - O(n) Time and O(n) Space This problem can be solved using the below idea: The idea is to use a temporary array of size n , where n is the length of the original array. If we left rotate the array by d positions, the last n - d elements will be at the front and the first d elements will be at the end. Copy the last (n - d) elements of original array into the first n - d positions of temporary array. Then copy the first d elements of the original array to the end of temporary array. Finally, copy all the elements of temporary array back into the original array. Working: Below is the implementation of the algorithm: C++ // C++ Program to left rotate the array by d positions // using temporary array #include <bits/stdc++.h> using namespace std ; // Function to rotate vector void rotateArr (vector < int >& arr , int d) { int n = arr . size (); // Handle case when d > n d %= n ; // Storing rotated version of array vector < int > temp (n); // Copy last n - d elements to the front of temp for (int i = 0 ; i < n - d ; i ++) temp [i] = arr [d + i]; // Copy the first d elements to the back of temp for (int i = 0 ; i < d ; i ++) temp [n - d + i] = arr [i]; // Copying the elements of temp in arr // to get the final rotated vector for (int i = 0 ; i < n ; i ++) arr [i] = temp [i]; } int main () { vector < int > arr = { 1 , 2 , 3 , 4 , 5 , 6 }; int d = 2 ; rotateArr (arr , d); // Print the rotated vector for (int i = 0 ; i < arr . size (); i ++) cout << arr [i] << " " ; return 0 ; } C // C Program to left rotate the array by d positions // using temporary array #include <stdio.h> #include <stdlib.h> // Function to rotate array void rotateArr (int * arr , int d , int n) { // Handle case when d > n d %= n ; // Storing rotated version of array int temp [n]; // Copy last n - d elements to the front of temp for (int i = 0 ; i < n - d ; i ++) temp [i] = arr [d + i]; // Copy the first d elements to the back of temp for (int i = 0 ; i < d ; i ++) temp [n - d + i] = arr [i]; // Copying the elements of temp in arr // to get the final rotated array for (int i = 0 ; i < n ; i ++) arr [i] = temp [i]; } int main () { int arr [] = { 1 , 2 , 3 , 4 , 5 , 6 }; int n = sizeof (arr) / sizeof (arr [0]); int d = 2 ; rotateArr (arr , d , n); // Print the rotated array for (int i = 0 ; i < n ; i ++) printf ("%d " , arr [i]); return 0 ; } Java // Java Program to left rotate the array by d positions // using temporary array import java.util.Arrays ; class GfG { // Function to rotate array static void rotateArr (int [] arr , int d) { int n = arr . length ; // Handle case when d > n d %= n ; // Storing rotated version of array int [] temp = new int [n] ; // Copy last n - d elements to the front of temp for (int i = 0 ; i < n - d ; i ++) temp [i] = arr [d + i] ; // Copy the first d elements to the back of temp for (int i = 0 ; i < d ; i ++) temp [n - d + i] = arr [i] ; // Copying the elements of temp in arr // to get the final rotated array for (int i = 0 ; i < n ; i ++) arr [i] = temp [i] ; } public static void main (String [] args) { int [] arr = { 1 , 2 , 3 , 4 , 5 , 6 }; int d = 2 ; rotateArr (arr , d); // Print the rotated array for (int i = 0 ; i < arr . length ; i ++) System . out . print (arr [i] + " "); } } Python # Python Program to left rotate the array by d positions # using temporary array # Function to rotate array def rotateArr (arr , d): n = len (arr) # Handle case when d > n d %= n # Storing rotated version of array temp = [0] * n # Copy last n - d elements to the front of temp for i in range (n - d): temp [i] = arr [d + i] # Copy the first d elements to the back of temp for i in range (d): temp [n - d + i] = arr [i] # Copying the elements of temp in arr # to get the final rotated array for i in range (n): arr [i] = temp [i] if __name__ == "__main__" : arr = [1 , 2 , 3 , 4 , 5 , 6] d = 2 rotateArr (arr , d) # Print the rotated array for i in range (len (arr)): print (arr [i], end = " ") C# // C# Program to left rotate the array by d positions // using temporary array using System ; class GfG { // Function to rotate array static void rotateArr (int [] arr , int d) { int n = arr . Length ; // Handle case when d > n d %= n ; // Storing rotated version of array int [] temp = new int [n]; // Copy last n - d elements to the front of temp for (int i = 0 ; i < n - d ; i ++) temp [i] = arr [d + i]; // Copy the first d elements to the back of temp for (int i = 0 ; i < d ; i ++) temp [n - d + i] = arr [i]; // Copying the elements of temp in arr // to get the final rotated array for (int i = 0 ; i < n ; i ++) arr [i] = temp [i]; } static void Main (string [] args) { int [] arr = { 1 , 2 , 3 , 4 , 5 , 6 }; int d = 2 ; rotateArr (arr , d); // Print the rotated array for (int i = 0 ; i < arr . Length ; i ++) Console . Write (arr [i] + " "); } } Javascript // JavaScript Program to left rotate the array by d positions // using temporary array // Function to rotate array function rotateArr (arr , d) { let n = arr . length ; // Handle case when d > n d %= n ; // Storing rotated version of array let temp = new Array (n); // Copy last n - d elements to the front of temp for (let i = 0 ; i < n - d ; i ++) temp [i] = arr [d + i]; // Copy the first d

elements to the back of temp for (let i = 0 ; i < d ; i ++) temp [n - d + i] = arr [i]; // Copying the elements of temp in arr // to get the final rotated array for (let i = 0 ; i < n ; i ++) arr [i] = temp [i]; } const arr = [1 , 2 , 3 , 4 , 5 , 6]; const d = 2 ; rotateArr (arr , d); // Print the rotated array console . log (arr . join (" ")); Output 3 4 5 6 1 2 Time Complexity: O(n), as we are visiting each element only twice. Auxiliary Space: O(n), as we are using an additional temporary array. [Expected Approach 1] Using Juggling Algorithm - O(n) Time and O(1) Space The idea is to use Juggling Algorithm which is based on rotating elements in cycles . Each cycle is independent and represents a group of elements that will shift among themselves during the rotation. If the starting index of a cycle is i, then next elements of the cycle can be found at indices (i + d) % n , (i + 2d) % n , (i + 3d) % n ... and so on till we return to the original index i. So for any index i , we know that after rotation, the element that will occupy this position is arr[(i + d) % n] . Consequently, for every index in the cycle, we will place the element that should be in that position after the rotation is completed. Please refer Juggling Algorithm for Array Rotation to know more about the implementation. Time Complexity: O(n) Auxiliary Space: O(1) [Expected Approach 2] Using Reversal Algorithm - O(n) Time and O(1) Space The idea is based on the observation that if we left rotate the array by d positions, the last (n - d) elements will be at the front and the first d elements will be at the end. Reverse the subarray containing the first d elements of the array. Reverse the subarray containing the last (n - d) elements of the array. Finally, reverse all the elements of the array. Working: Below is the implementation of the algorithm: C++ // C++ Code to left rotate an array using Reversal Algorithm #include <bits/stdc++.h> using namespace std ; // Function to rotate an array by d elements to the left void rotateArr (vector < int >& arr , int d) { int n = arr . size (); // Handle the case where d > size of array d %= n ; // Reverse the first d elements reverse (arr . begin (), arr . begin () + d); // Reverse the remaining n-d elements reverse (arr . begin () + d , arr . end ()); // Reverse the entire array reverse (arr . begin (), arr . end ()); } int main () { vector < int > arr = { 1 , 2 , 3 , 4 , 5 , 6 }; int d = 2 ; rotateArr (arr , d); for (int i = 0 ; i < arr . size (); i ++) cout << arr [i] << " " ; return 0 ; } C // C Code to left rotate an array using Reversal Algorithm #include <stdio.h> // Function to reverse a portion of the array void reverse (int * arr , int start , int end) { while (start < end) { int temp = arr [start]; arr [start] = arr [end]; arr [end] = temp ; start ++ ; end -- ; } } // Function to rotate an array by d elements to the left void rotateArr (int * arr , int n , int d) { // Handle the case where d > size of array d %= n ; // Reverse the first d elements reverse (arr , 0 , d - 1); // Reverse the remaining n-d elements reverse (arr , d , n - 1); // Reverse the entire array reverse (arr , 0 , n - 1); } int main () { int arr [] = { 1 , 2 , 3 , 4 , 5 , 6 }; int n = sizeof (arr) / sizeof (arr [0]); int d = 2 ; rotateArr (arr , n , d); for (int i = 0 ; i < n ; i ++) printf ("%d " , arr [i]); return 0 ; } Java // Java Code to left rotate an array using Reversal Algorithm import java.util.Arrays ; class GfG { // Function to rotate an array by d elements to the left static void rotateArr (int [] arr , int d) { int n = arr . length ; // Handle the case where d > size of array d %= n ; // Reverse the first d elements reverse (arr , 0 , d - 1); // Reverse the remaining n-d elements reverse (arr , d , n - 1); // Reverse the entire array reverse (arr , 0 , n - 1); } // Function to reverse a portion of the array static void reverse (int [] arr , int start , int end) { while (start < end) { int temp = arr [start] ; arr [start] = arr [end] ; arr [end] = temp ; start ++ ; end -- ; } } public static void main (String [] args) { int [] arr = { 1 , 2 , 3 , 4 , 5 , 6 }; int d = 2 ; rotateArr (arr , d); for (int i = 0 ; i < arr . length ; i ++) System . out . print (arr [i] + " "); } } Python # Python Code to left rotate an array using Reversal Algorithm # Function to rotate an array by d elements to the left def rotateArr (arr , d): n = len (arr) # Handle the case where d > size of array d %= n # Reverse the first d elements reverse (arr , 0 , d - 1) # Reverse the remaining n-d elements reverse (arr , d , n - 1) # Reverse the entire array reverse (arr , 0 , n - 1) # Function to reverse a portion of the array def reverse (arr , start , end): while start < end : arr [start], arr [end] = arr [end], arr [start] start += 1 end -= 1 if __name__ == "__main__" : arr = [1 , 2 , 3 , 4 , 5 , 6] d = 2 rotateArr (arr , d) for i in range (len (arr)): print (arr [i], end = " ") C# // C# Code to left rotate an array using Reversal Algorithm using System ; class GfG { // Function to rotate an array by d elements to the left static void rotateArr (int [] arr , int d) { int n = arr . Length ; // Handle the case where d > size of array d %= n ; // Reverse the first d elements reverse (arr , 0 , d - 1); // Reverse the remaining n-d elements reverse (arr , d , n - 1); // Reverse the entire array reverse (arr , 0 , n - 1); } // Function to reverse a portion of the array static void reverse (int [] arr , int start , int end) { while (start < end) { int temp = arr [start]; arr [start] = arr [end]; arr [end] = temp ; start ++ ; end -- ; } } static void Main () { int [] arr = { 1 , 2 , 3 , 4 , 5 , 6 }; int d = 2 ; rotateArr (arr , d); for (int i = 0 ; i < arr . Length ; i ++) Console . Write (arr [i] + " "); } } JavaScript // JavaScript Code to left rotate an array using Reversal Algorithm // Function to rotate an array by d elements to the left function rotateArr (arr , d) { let n = arr . length ; // Handle the case where d > size of array d %= n ; // Reverse the first d elements reverse (arr , 0 , d - 1); // Reverse the remaining n-d elements reverse (arr , d , n - 1); //

Reverse the entire array reverse (arr , 0 , n - 1); } // Function to reverse a portion of the array function reverse (arr , start , end) { while (start < end) { let temp = arr [start]; arr [start] = arr [end]; arr [end] = temp ; start ++ ; end -- ; } } const arr = [1 , 2 , 3 , 4 , 5 , 6]; const d = 2 ; rotateArr (arr , d); console . log (arr . join (" ")); Output 3 4 5 6 1 2 Time complexity: O(n), as we are visiting each element exactly twice. Auxiliary Space: O(1) Comment Article Tags: Article Tags: Greedy DSA Arrays Amazon MAQ Software SAP Labs MakeMyTrip rotation Wipro + 5 More