Reverse a String – Complete Tutorial - GeeksforGeeks

Source: https://www.geeksforgeeks.org/reverse-a-string/

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 Reverse a String – Complete Tutorial Last Updated : 3 Oct, 2025 Given a string s , the task is to reverse the string. Reversing a string means rearranging the characters such that the first character becomes the last , the second character becomes second last and so on. Examples: Input: s = "GeeksforGeeks" Output: "skeeGrofskeeG" Explanation : The first character G moves to last position, the second character e moves to second-last and so on. Input: s = "abdcfe" Output: "efcdba" Explanation: The first character a moves to last position, the second character b moves to second-last and so on. Try it on GfG Practice Table of Content Using backward traversal – O(n) Time and O(n) Space Using Two Pointers - O(n) Time and O(1) Space Using Recursion - O(n) Time and O(n) Space Using Stack - O(n) Time and O(n) Space Using Inbuilt methods - O(n) Time and O(1) Space Using backward traversal – O(n) Time and O(n) Space The idea is to start at the last character of the string and move backward, appending each character to a new string res . This new string res will contain the characters of the original string in reverse order. C++ // C++ program to reverse a string using backward traversal #include <iostream> #include <string> using namespace std ; string reverseString (string & s) { string res ; // Traverse on s in backward direction // and add each charecter to a new string for (int i = s . size () - 1 ; i >= 0 ; i --) { res += s [i]; } return res ; } int main () { string s = "abdcfe" ; string res = reverseString (s); cout << res ; return 0 ; } C // C program to reverse a string using backward traversal #include <stdio.h> #include <string.h> #include <stdlib.h> char * reverseString (char * s) { int n = strlen (s); char * res = (char *) malloc ((n + 1) * sizeof (char)); int j = 0 ; // Traverse on s in backward direction // and add each character to a new string for (int i = n - 1 ; i >= 0 ; i --) { res [j] = s [i]; j ++ ; } // Null-terminate the result string res [n] = '\0' ; return res ; } int main () { char s [] = "abdcfe" ; char * res = reverseString (s); printf ("%s" , res); return 0 ; } Java // Java program to reverse a string using backward traversal class GfG { static String reverseString (String s) { StringBuilder res = new StringBuilder (); // Traverse on s in backward direction // and add each character to a new string for (int i = s . length () - 1 ; i >= 0 ; i --) { res . append (s . charAt (i)); } return res . toString (); } public static void main (String [] args) { String s = "abdcfe" ; String res = reverseString (s); System . out . print (res); } } Python # Python program to reverse a string using backward traversal def reverseString (s): res = [] # Traverse on s in backward direction # and add each character to the list for i in range (len (s) - 1 , - 1 , - 1): res . append (s [i]) # Convert list back to string return '' . join (res) if __name__ == "__main__" : s = "abdcfe" print (reverseString (s)) C# // C# program to reverse a string using backward traversal using System ; using System.Text ; class GfG { static string reverseString (string s) { StringBuilder res = new StringBuilder (); // Traverse on s in backward direction // and add each character to a new string for (int i = s . Length - 1 ; i >= 0 ; i --) { res . Append (s [i]); } // Convert StringBuilder to string return res . ToString (); } static void Main (string [] args) { string s = "abdcfe" ; string res = reverseString (s); Console . WriteLine (res); } } JavaScript // JavaScript program to reverse a string using backward traversal function reverseString (s) { let res = []; // Traverse on s in backward direction // and add each character to the array for (let i = s . length - 1 ; i >= 0 ; i --) { res . push (s [i]); } return res . join (''); } const s = "abdcfe" ; console . log (reverseString (s)); Output efcdba Time Complexity: O(n) for backward traversal Auxiliary Space: O(n) for storing the reversed string. Using Two Pointers - O(n) Time and O(1) Space The idea is to maintain two pointers: left and

right , such that left points to the beginning of the string and right points to the end of the string. While left pointer is less than the right pointer, swap the characters at these two positions. After each swap, increment the left pointer and decrement the right pointer to move towards the center of the string. This will swap all the characters in the first half with their corresponding character in the second half. Illustration: C++ // C++ program to reverse a string using two pointers #include <bits/stdc++.h> using namespace std ; string reverseString (string & s) { int left = 0 , right = s . length () - 1 ; // Swap characters from both ends till we reach // the middle of the string while (left < right) { swap (s [left], s [right]); left ++ ; right -- ; } return s ; } int main () { string s = "abdcfe" ; cout << reverseString (s); return 0 ; } C // C program to reverse a string using two pointers #include <stdio.h> #include <string.h> char * reverseString (char * s) { int left = 0 , right = strlen (s) - 1 ; // Swap characters from both ends till we reach // the middle of the string while (left < right) { char temp = s [left]; s [left] = s [right]; s [right] = temp ; left ++ ; right -- ; } return s ; } int main () { char s [] = "abdcfe" ; printf ("%s" , reverseString (s)); return 0 ; } Java // Java program to reverse a string using two pointers class GfG { static String reverseString (String s) { int left = 0 , right = s . length () - 1 ; // Use StringBuilder for mutability StringBuilder res = new StringBuilder (s); // Swap characters from both ends till we reach // the middle of the string while (left < right) { char temp = res . charAt (left); res . setCharAt (left , res . charAt (right)); res . setCharAt (right , temp); left ++ ; right -- ; } // Convert StringBuilder back to string return res . toString (); } public static void main (String [] args) { String s = "abdcfe" ; System . out . println (reverseString (s)); } } Python # Python program to reverse a string using two pointers # Function to reverse a string using two pointers def reverseString (s): left = 0 right = len (s) - 1 # Convert string to a list for mutability s = list (s) # Swap characters from both ends till we reach # the middle of the string while left < right : s [left], s [right] = s [right], s [left] left += 1 right -= 1 # Convert list back to string return '' . join (s) if __name__ == "__main__" : s = "abdcfe" print (reverseString (s)) C# // C# program to reverse a string using two pointers using System ; using System.Text ; class GfG { static string reverseString (string s) { // Use StringBuilder for mutability StringBuilder res = new StringBuilder (s); int left = 0 , right = res . Length - 1 ; // Swap characters from both ends till we reach // the middle of the string while (left < right) { char temp = res [left]; res [left] = res [right]; res [right] = temp ; left ++ ; right -- ; } // Convert StringBuilder back to string return res . ToString (); } static void Main (string [] args) { string s = "abdcfe" ; Console . WriteLine (reverseString (s)); } } JavaScript // JavaScript program to reverse a string using two pointers function reverseString (s) { let left = 0 , right = s . length - 1 ; // Convert string to array for mutability s = s . split (''); // Swap characters from both ends till we reach // the middle of the string while (left < right) { [s [left], s [right]] = [s [right], s [left]]; left ++ ; right -- ; } return s . join (''); } const s = "abdcfe" ; console . log (reverseString (s)); Output efcdba Time Complexity: O(n) Auxiliary Space: O(1) Using Recursion - O(n) Time and O(n) Space The idea is to use recursion and define a recursive function that takes a string as input and reverses it. Inside the recursive function, Swap the first and last element. Recursively call the function with the remaining substring. C++ // C++ Program to reverse an array using Recursion #include <iostream> #include <vector> using namespace std ; // recursive function to reverse a string from l to r void reverseStringRec (string & s , int l , int r) { // If the substring is empty, return if (l >= r) return ; swap (s [l], s [r]); // Recur for the remaining string reverseStringRec (s , l + 1 , r - 1); } // function to reverse a string string reverseString (string & s) { int n = s . length (); reverseStringRec (s , 0 , n - 1); return s ; } int main () { string s = "abdcfe" ; cout << reverseString (s) << endl ; return 0 ; } C // C program to reverse a string using Recursion #include <stdio.h> #include <string.h> // Recursive function to reverse a string from l to r void reverseStringRec (char * s , int l , int r) { if (l >= r) return ; // Swap the characters at the ends char temp = s [l]; s [l] = s [r]; s [r] = temp ; // Recur for the remaining string reverseStringRec (s , l + 1 , r - 1); } char * reverseString (char * s) { int n = strlen (s); reverseStringRec (s , 0 , n - 1); return s ; } int main () { char s [] = "abdcfe" ; printf ("%s \n " , reverseString (s)); return 0 ; } Java // Java program to reverse a string using Recursion class GfG { // Recursive function to reverse a string from l to r static void reverseStringRec (char [] s , int l , int r) { if (l >= r) return ; // Swap the characters at the ends char temp = s [l] ; s [l] = s [r] ; s [r] = temp ; // Recur for the remaining string reverseStringRec (s , l + 1 , r - 1); } // Function to reverse a string static String reverseString (String s) { char [] arr = s . toCharArray (); reverseStringRec (arr , 0 , arr . length - 1); return new String (arr); } public static void main (String [] args) { String s = "abdcfe" ; System . out . println (reverseString (s)); } } Python # Python program to reverse a string using Recursion # Recursive Function to reverse a string def reverseStringRec (arr , l , r): if l >= r : return # Swap the characters at the ends arr [l], arr [r] = arr [r], arr [l] # Recur for the remaining string reverseStringRec (arr , l + 1 , r - 1) def reverseString (s): # Convert string to list of characters arr = list

(s) reverseStringRec (arr , 0 , len (arr) - 1) # Convert list back to string return '' . join (arr) if __name__ == "__main__" : s = "abdcfe" print (reverseString (s)) C# // C# program to reverse a string using Recursion using System ; using System.Text ; class GfG { // recursive function to reverse a string from l to r static void reverseStringRec (StringBuilder s , int l , int r) { if (l >= r) return ; char temp = s [l]; s [l] = s [r]; s [r] = temp ; // Recur for the remaining string reverseStringRec (s , l + 1 , r - 1); } // function to reverse a string static string reverseString (string input) { StringBuilder s = new StringBuilder (input); int n = s . Length ; reverseStringRec (s , 0 , n - 1); return s . ToString (); } static void Main () { string s = "abdcfe" ; Console . WriteLine (reverseString (s)); } } JavaScript // JavaScript program to reverse a string using Recursion // Recursive Function to reverse a string function reverseStringRec (res , l , r) { if (l >= r) return ; // Swap the characters at the ends [res [l], res [r]] = [res [r], res [l]]; // Recur for the remaining string reverseStringRec (res , l + 1 , r - 1); } function reverseString (s) { // Convert string to array of characters let res = s . split (''); reverseStringRec (res , 0 , res . length - 1); // Convert array back to string return res . join (''); } let s = "abdcfe" ; console . log (reverseString (s)); Output efcdba Time Complexity: O(n) where n is length of string Auxiliary Space: O(n) Using Stack - O(n) Time and O(n) Space The idea is to use stack for reversing a string because Stack follows Last In First Out (LIFO) principle. This means the last character you add is the first one you'll take out. So, when we push all the characters of a string into the stack, the last character becomes the first one to pop. Illustration: C++ // C++ program to reverse a string using stack #include <bits/stdc++.h> using namespace std ; string reverseString (string & s) { stack < char > st ; // Push the charcters into stack for (int i = 0 ; i < s . size (); i ++) { st . push (s [i]); } // Pop the characters of stack into the original string for (int i = 0 ; i < s . size (); i ++) { s [i] = st . top (); st . pop (); } return s ; } int main () { string s = "abdcfe" ; cout << reverseString (s); return 0 ; } Java // Java program to reverse a string using stack import java.util.* ; class GfG { static String reverseString (String s) { Stack < Character > st = new Stack <> (); // Push the characters into stack for (int i = 0 ; i < s . length (); i ++) st . push (s . charAt (i)); StringBuilder res = new StringBuilder (); // Pop the characters of stack into the original string for (int i = 0 ; i < s . length (); i ++) res . append (st . pop ()); return res . toString (); } public static void main (String [] args) { String s = "abdcfe" ; System . out . println (reverseString (s)); } } Python # Python program to reverse a string using stack def reverseString (s): stack = [] # Push the characters into stack for char in s : stack . append (char) # Prepare a list to hold the reversed characters rev = [''] * len (s) # Pop the characters from stack into the reversed list for i in range (len (s)): rev [i] = stack . pop () # Join the list to form the reversed string return '' . join (rev) if __name__ == "__main__" : s = "abdcfe" print (reverseString (s)) C# // C# program to reverse a string using stack using System ; using System.Collections.Generic ; using System.Text ; class GfG { static string reverseString (string s) { Stack < char > st = new Stack < char > (); // Push the characters into stack for (int i = 0 ; i < s . Length ; i ++) st . Push (s [i]); // Create a StringBuilder to hold the reversed string StringBuilder res = new StringBuilder (); // Pop the characters from stack into the StringBuilder while (st . Count > 0) res . Append (st . Pop ()); // Convert the StringBuilder back to a string return res . ToString (); } static void Main () { string s = "abdcfe" ; Console . WriteLine (reverseString (s)); } } JavaScript // JavaScript program to reverse a string using stack function reverseString (s) { // To store the characters of the original string. const stack = []; // Push the characters into the stack. for (let i = 0 ; i < s . length ; i ++) { stack . push (s [i]); } // Create an array to hold the reversed characters const res = new Array (s . length); // Pop the characters of the stack and store in the array for (let i = 0 ; i < s . length ; i ++) { res [i] = stack . pop (); } // Join the array to form the reversed string return res . join (''); } let str = "abdcfe" ; console . log (reverseString (str)); Output efcdba Time Complexity: O(n) Auxiliary Space: O(n) Using Inbuilt methods - O(n) Time and O(1) Space The idea is to use built-in reverse method to reverse the string. If built-in method for string reversal does not exist, then convert string to array or list and use their built-in method for reverse. Then convert it back to string. C++ #include <bits/stdc++.h> using namespace std ; string reverseString (string & s) { reverse (s . begin (), s . end ()); return s ; } int main () { string s = "abdcfe" ; cout << reverseString (s) ; return 0 ; } Java // Java program to reverse a string using StringBuffer class import java.io.* ; import java.util.* ; class GFG { static String stringReverse (String s) { StringBuilder res = new StringBuilder (s); res . reverse (); return res . toString (); } public static void main (String [] args) { String s = "abdcfe" ; System . out . println (stringReverse (s)); } } Python # Function to reverse a string def reverseString (s): # Reverse the string using slicing return s [:: - 1] if __name__ == "__main__" : str = "abdcfe" print (reverseString (str)) C# // C# program to reverse a string using System ; class GfG { static string reverseString (string s) { // Reverse the string using the built-in method char [] arr = s . ToCharArray (); Array . Reverse (arr); return new string (arr); } static void Main () { string s = "abdcfe" ; Console . WriteLine (reverseString

(s)); } } JavaScript // Function to reverse a string function reverseString (s) { // convert to array to make it mutable s = s . split (''); // Reverse the string using the built-in method s = s . reverse (); return s . join (''); } let str = "abdcfe" ; console . log (reverseString (str)); Output efcdba Time Complexity: O(n) Auxiliary Space: O(1) in C++ and python and O(n) in Java, C# and JavaScript (extra space is used to store in array or list or StringBuilder for reversal). Comment Article Tags: Article Tags: Misc Strings DSA Reverse