A Program to check if strings are rotations of each other or not

Given a string s1 and a string s2, write a snippet to say whether s2 is a rotation of s1 using only one call to strstr routine?
(eg given s1 = ABCD and s2 = CDAB, return true, given s1 = ABCD, and s2 = ACBD , return false)

Algorithm: areRotations(str1, str2)

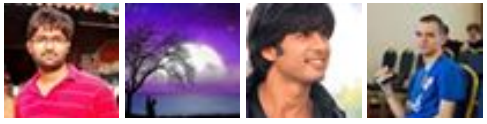
```
1. Create a temp string and store concatenation of str1 to
   str1 in temp.
                      temp = str1.str1
2. If str2 is a substring of temp then str1 and str2 are
   rotations of each other.

Example:
                 str1 = "ABACD"
                 str2 = "CDABA"

 temp = str1.str1 = "ABACDABACD"
 Since str2 is a substring of temp, str1 and str2 are
 rotations of each other.
```

Implementation:

```
# include <stdio.h>
```

```
# include <string.h>
# include <stdlib.h>

/* Function checks if passed strings (str1 and str2)
   are rotations of each other */
int areRotations(char *str1, char *str2)
{
  int size1   = strlen(str1);
  int size2   = strlen(str2);
  char *temp;
  void *ptr;

  /* Check if sizes of two strings are same */
  if (size1 != size2)
     return 0;

  /* Create a temp string with value str1.str1 */
  temp   = (char *)malloc(sizeof(char)*(size1*2 + 1));
  temp[0] = '\0';
  strcat(temp, str1);
  strcat(temp, str1);

  /* Now check if str2 is a substring of temp */
  ptr = strstr(temp, str2);

  free(temp); // Free dynamically allocated memory

  /* strstr returns NULL if the second string is NOT a
    substring of first string */
  if (ptr != NULL)
    return 1;
  else
    return 0;
}

/* Driver program to test areRotations */
int main()
{
    char *str1 = "AACD";
    char *str2 = "ACDA";

    if (areRotations(str1, str2))
       printf("Strings are rotations of each other");
    else
       printf("Strings are not rotations of each other");

    getchar();
```

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
    return 0;
}
```

Output:

```
Strings are rotations of each other
```

Library Functions Used:
strstr:
strstr finds a sub-string within a string.
Prototype: char * strstr(const char *s1, const char *s2);
See http://www.lix.polytechnique.fr/Labo/Leo.Liberti/public/computing/prog/c/C/MAN/strstr.htm for more details

strcat:
strncat concatenate two strings
Prototype: char *strcat(char *dest, const char *src);
See http://www.lix.polytechnique.fr/Labo/Leo.Liberti/public/computing/prog/c/C/MAN/strcat.htm for more details

Time Complexity: Time complexity of this problem depends on the implementation of strstr function.
If implementation of strstr is done using KMP matcher then complexity of the above program is (-)(n1 + n2) where n1 and n2 are lengths of strings. KMP matcher takes (-)(n) time to find a substrng in a string of length n where length of substring is assumed to be smaller than the string.

Related Tpoics:

- Printing Longest Common Subsequence
- Suffix Array | Set 2 (nLogn Algorithm)
- Rearrange a string so that all same characters become d distance away
- Recursively remove all adjacent duplicates
- Find the first non-repeating character from a stream of characters
- Dynamic Programming | Set 33 (Find if a string is interleaved of two other strings)
- Remove “b” and “ac” from a given string
- Dynamic Programming | Set 29 (Longest Common Substring)

Writing code in comment? Please use **ideone.com** and share the link here.

49 Comments GeeksforGeeks

Sort by Newest

Recent Comments

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 20 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 40 minutes ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 40 minutes ago

@meya Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Join the discussion…

Guest • 4 days ago

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

#include<stdbool.h>

bool Rotable(char* str1,char* str2)

{

int i=0,x=0,z=0;

int flag=0;

char temp;

temp=str2[0];
```

see more

Reply • Share ›

sourabh • 2 months ago

I dont understand the logic here we are appending two strings and then find wl concatenated string . Will it not always be contained in the temp?

Reply • Share ›

Sourabh sourabh • 2 months ago

sorry my bad its concatenating the first string twice

Neha I think that is what it should return as, in...

Find depth of the deepest odd level leaf node · 2 hours ago

sorry my bad its concatenating the first string twice.

Reply • Share ›

jugal • 5 months ago

s1 = "ABAB" & s2 = "ABAB".
is s2 a rotation of s1??

Reply • Share ›

Swagato Mondal • 8 months ago

This code doesn't work for a input as first string-"ABACD",second string-"CDA
returns true

Reply • Share ›

Ashish ➦ Swagato Mondal • 3 months ago

not wrong.. concatenation of ABACD to itself is ABACDABACD, which
.

Reply • Share ›

pavansrinivas ➦ Swagato Mondal • 7 months ago

No the solution is correct as we first check the lengths of two strings...

4 Reply • Share ›

Vivek Venkatesh • 10 months ago

This can be done with a space complexity of O(1).

Also I believe I handled the special case of repeating characters.

```
boolean isRotationWith(String input, String input2) {


                if(input.length() != input2.length()) {
                        // Two Strings can be a rotation only if they
                        return false;
```

```
}
else {
        boolean found = false;
        int i;
        int foundAt = 0;
        for(i=0;i<input.length();i++) {

                /*AIM: Find the position of 1st charac
                 *
```

see more

Reply • Share ›

Jitendra.BITS • 10 months ago

Is it advisable to use library functions like strlen, strcat, strcmp, strstr in the int
Or should we implement these functions themselves?

Reply • Share ›

Abhishek Kumar ➔ Jitendra.BITS • 21 days ago

i also have same doubt bro...:/

Reply • Share ›

GeeksforGeeks • 10 months ago

By convention a string is called rotation of itself. If you n times rotate a string, y
number of characters in the string.

Reply • Share ›

Abirami Ramalingam • 10 months ago

@geeksforgeeks :if str1=abc and str2=abc, these two are not rotation of each
they are substring of each other.

Reply • Share ›

GeeksforGeeks • 11 months ago

Thanks for pointing this out. We have updated the code.

Reply • Share ›

Harris Siddiqui • 11 months ago

arre matlab roshan babu keher dha rahe ho aajkal to bilkul :D

Reply • Share ›

Roshan Jindani • 11 months ago

it should be temp = (char *) malloc(sizeof(char) *(size1 * 2 + 1);.
not temp = (char *) malloc(sizeof(char) * size1 * 2 + 1);.

Reply • Share ›

oqardZ • 11 months ago

Hi,

code given above has two problems with it:

1)

Line
temp = (char *) malloc(sizeof(char) * size1 * 2 + 1);
should actually be
temp = (char *) malloc(sizeof(char) * (size1 * 2 + 1));

2)

You have a memory leak. After call to strstr, next line should be:
free(temp);

Reply • Share ›

GeeksforGeeks oqardZ • 11 months ago

Thanks for pointing this out. We have updated the code. Keep it up!

oqardZ ➚ GeeksforGeeks • 11 months ago

Hi, please take a moment and re-check malloc line. You have n

GeeksforGeeks ➚ oqardZ • 11 months ago

Thanks, corrected!

Tanusri • 11 months ago

I have tried to cover different cases-

```
#include
#include
#include
#include
#include

bool rotate(char *s1,char *s2)
{
if(strlen(s1)!=strlen(s2))
return 0;
int j=0,i=0;
int bitmap[26]={0};
while(i< strlen(s2))
{
bitmap[s2[i]-65]=bitmap[s2[i]-65]+1;

i++;
```

rohit • a year ago

@geeksforgeek:you also need to check strlen(s1)==strlen(s2)...eg-:ABCD,CD

 • Reply • Share ›

gags • a year ago

[sourcecode language="java"]
public static boolean isRotated(String original, String rotated) {
if (original == null || rotated == null)
return false;
if (original.length() != rotated.length())
return false;
int len = original.length();
for (int i = 0; i < len; i++) {
if (rotated.charAt(i) == original.charAt(0)) {
if (check(original, rotated, i))
return true;
}
}
return false;
}

private static boolean check(String original, String rotated, int start) {
int end = start - 1;

see more

 • Reply • Share ›

abhishek08aug • a year ago

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define NO_OF_CHARS 256
```

```
#define NO_OF_CHARS 256

int are_rotations(char * str1, char * str2) {
  int size1=strlen(str1);
  int size2=strlen(str2);

  if(size1!=size2) {
    return 0;
  }
  char * temp=(char *)malloc(sizeof(char)*size1*2+1);
  strcat(temp, str1);
  strcat(temp, str1);

  return strstr(temp, str2);
}
```

see more

• Reply • Share ›

Ramasubramani • a year ago

Awesome Logic.

• Reply • Share ›

arun • a year ago

awesome logic

```
/* Paste your code here (You may delete these lines if not writing c
```

• Reply • Share ›

radhakrishna • 3 years ago

traverse array from 0 to n-1 for str1 and from n-1 to 0 for str2 (both at once)
keep xoring str1 and str2 elements till u get 0

if index resulted reaches boundary not rotated string
else {
compare rest of elements and if all same print rotated
}

Reply • Share ›

joker ➚ radhakrishna • 2 years ago

@radhakrishna
your algo won't work.try it on some case's which have even number re

eg. AA , BB
now xoring will give 0. but they are not rotation of each other.

Reply • Share ›

Don • 3 years ago

```
if(length(string1) length(string2))
return false;

for i = 0:length(string1)
if(string1[i] == string2[0])
break;

for j=0:length(string2)
i++;
if(string1[j] string2[i%length(string2)]
return false;

return true;
```

Reply • Share ›

Don ➚ Don • 3 years ago

Darn html! here it is again:

```
if(length(string1) != length(string2))
return false;

for i = 0:length(string1)
if(string1[i] == string2[0])
break;

for j=0:length(string2)
i++;
if(string1[j] != string2[i%length(string2)]
return false;

return true;
```

▲ | ▼ • Reply • Share ›

atul ➦ Don • 2 years ago

it will fails for following test case :-

```
char str1[]="ABCDA";
char str2[]="AABCD";
```

> /* Paste your code here (You may **delete** these lines **if**

▲ | ▼ • Reply • Share ›

Swapna • 3 years ago

```
#include
#include

using namespace std;

int recurse(char *s1 , char *s2, char *s2_orig) {
if(*s1 == '')
return 1;
```

```
return 1;
if(*s1 == *s2)
{
s1 += 1;
if(*(s2 + 1) == '')
s2 = s2_orig;
else
s2 += 1;
return recurse(s1, s2, s2_orig);
}
return 0;
}
```

see more

• Reply • Share ›

zoso • 3 years ago

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
int rot(char *s1, char *s2)
{
    int n1=strlen(s1),i=0,j=0;
    if(strlen(s2)!=n1)
      return 0;
    else
    {
        while(s1[i])
        {

            int k=i;
            while(s1[k]==s2[j]&&(k!=(i+n1-1)%n1))
```

```
            {
                k=(k+1)%n1;
```

see more

• Reply • Share ›

zoso ➚ zoso • 3 years ago

one thing missed is that in rot() the inner while loop will have a j=0 state

• Reply • Share ›

Sunil • 3 years ago

Awesome logic :) i m loving geeksforgeeks portal

• Reply • Share ›

rams • 3 years ago

```
 void Prefix(char P[],int Pre[])
{
    int k=0,q;
    int m=strlen(P);
    Pre[0]=0;

    for(q=1; q0 & & P[k]!=P[q])
        k=Pre[k];
    if(P[k]==P[q])
        k++;
    Pre[q]=k;
}
}
```

```
int KMP(char T[],char P[])
```

see more

Reply • Share ›

priyadarshi • 3 years ago

```
 int are_rotations(char *s1,char *s2)
{
        int len,i,j,k;
   if(strlen(s1)!=strlen(s2))
        return 0;
   len=strlen(s1);
   for(i=0;i<len;i++)
   {
        if(s2[i]==s1[0])
        {
        for(j=1,k=i+1;j<len;j++,k=(k+1)%len)
         {
                if(s1[j]!=s2[k])
                break;
         }
         if(j==len)
                return 1;
      }
   }
   return 0;
}
/*this program runs in O(n) in best case, when all elements are distir
& the strings are rotations of each other, but the time reduces to O(r
otherwise.*/
```

Reply • Share ›

Algoseekar • 3 years ago

@GeeksForGeeks,Sandeep,....Its O(n^2) Solution can you let me know how w

Reply • Share ›

aimless ➚ Algoseekar • 3 years ago

if you want O(n) then don't go for strstr
->have an array of chars say arr[256] - i am not considering DOUBLE
->initialize with zero.
->when you read one string for every occurrence increment arr[str1[i]]-
->for second char decrement arr[str[i]]--
->if string length is same you shall meet a situation when you will have already zero. report it. else at the end every element will be zero.

Reply • Share ›

donbosio • 3 years ago

@geeks:in the comments u wrote that strstr() returns null if second is a subst

Reply • Share ›

Sandeep ➚ donbosio • 3 years ago

@donbosio: Thanks for pointing this out. I have corrected the typo.

Reply • Share ›

Ritesh • 3 years ago

if str1 = "abc"
and str2 = "cba"
will it work?

Reply • Share ›

Srish ➚ Ritesh • 3 years ago

@Ritesh:

Yes it should work perfectly fine in that case, "abc" and "cba" are not ro
"cba" will not be found in "abcabc" so it will return 0.
"abc" and "cab" infact would return 1.

Reply • Share ›

geeksforgeeks • 4 years ago

@manoj: Your solution may not work for strings with duplicate characters. For following strings but they are rotations of each other.

s = "ACAB"
t = "ABAC"

Reply • Share ›

manoj • 4 years ago

Forgot to mention that when first string finishes, start it again from 0th index.

Reply • Share ›

manoj • 4 years ago

following would work without temp string and strstr function.

s = "ABACD"
t = "CDABA"

1. start from s and t
2. compare first counter at 'A' of s and second at 'C' of t
3. if equal, increment both, else increment first only.
4. if at any point they do not match after first match, then return false, else true

This is O(m+n) time and no space algorithm.

Reply • Share ›

mohitk ➚ manoj • a year ago

^Really nice optimization by manoj.

Changes the algorithm to O(1) space, while maintaining O(n) time.

code for the same, hope I have covered everything:

```
public boolean isRotation(String s1, String s2)
        {
                // If 2nd string is null
                if (s2 == null)
                        return true;

                // If 1st string is null or the strings have ur
                if(s1 == null || (s1.length() != s2.length()))
                        return false;

                int i = 0, j = 0;
                boolean first_flg = false;
```

see more

Reply • Share ›

hsg ➚ manoj • 4 years ago

this will not work for string1: ABACD
and String 2 : ACDAB
even though its a rotation.

Reply • Share ›

Sergey ➚ hsg • 3 years ago

No, it works perfectly. And why it should not?

Reply • Share ›

abc ➚ Sergey • 2 years ago

in case to work for
String1:ABACD
String2:ACDAB

we must work out such that if there is any mismatch of
be set as 0.

• Reply • Share ›

Subscribe Add Disqus to your site