

Check whether two strings are anagram of each other

Write a function to check whether two given strings are **anagram** of each other or not. An anagram of a string is another string that contains same characters, only the order of characters can be different. For example, "abcd" and "dabc" are anagram of each other.

Method 1 (Use Sorting)

- 1) Sort both strings
- 2) Compare the sorted strings

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

/* Fucntion prototype for srting a given string using quick sort */
void quickSort(char *arr, int si, int ei);

/* function to check whether two strings are anagram of each other */
bool areAnagram(char *str1, char *str2)
{
    // Get lenghts of both strings
    int n1 = strlen(str1);
    int n2 = strlen(str2);

    // If lenght of both strings is not same, then they cannot
    // be anagram
    if (n1 != n2)
        return false;

    // Sort both strings
    quickSort (str1, 0, n1 - 1);
    quickSort (str2, 0, n2 - 1);

    // Compare sorted strings
    for (int i = 0; i < n1; i++)
```

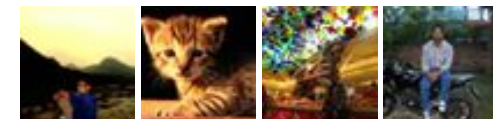
Google™ Custom Search



GeeksforGeeks



52,731 people like [GeeksforGeeks](#).



Interview Experiences

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

Pattern Searching

Divide & Conquer

Mathematical Algorithms

Recursion

Geometric Algorithms

```

        if (str1[i] != str2[i])
            return false;

    return true;
}

```

// Following functions (exchange and partition are needed for quickSort)

```

void exchange(char *a, char *b)
{
    char temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

int partition(char A[], int si, int ei)
{
    char x = A[ei];
    int i = (si - 1);
    int j;

    for (j = si; j <= ei - 1; j++)
    {
        if (A[j] <= x)
        {
            i++;
            exchange(&A[i], &A[j]);
        }
    }
    exchange (&A[i + 1], &A[ei]);
    return (i + 1);
}

/* Implementation of Quick Sort
A[] --> Array to be sorted
si --> Starting index
ei --> Ending index
*/
void quickSort(char A[], int si, int ei)
{
    int pi;    /* Partitioning index */
    if (si < ei)
    {
        pi = partition(A, si, ei);
        quickSort(A, si, pi - 1);
        quickSort(A, pi + 1, ei);
    }
}

```



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

```

}

/* Driver program to test to pront printDups*/
int main()
{
    char str1[] = "test";
    char str2[] = "ttew";
    if ( areAnagram(str1, str2) )
        printf("The two strings are anagram of each other");
    else
        printf("The two strings are not anagram of each other");

    return 0;
}

```

Output:

The two strings are not anagram of each other

Time Complexity: Time complexity of this method depends upon the sorting technique used. In the above implementation, quickSort is used which may be $O(n^2)$ in worst case. If we use a $O(n\log n)$ sorting algorithm like merge sort, then the complexity becomes $O(n\log n)$

Method 2 (Count characters)

This method assumes that the set of possible characters in both strings is small. In the following implementation, it is assumed that the characters are stored using 8 bit and there can be 256 possible characters.

- 1) Create count arrays of size 256 for both strings. Initialize all values in count arrays as 0.
- 2) Iterate through every character of both strings and increment the count of character in the corresponding count arrays.
- 3) Compare count arrays. If both count arrays are same, then return true.

```

# include <stdio.h>
# define NO_OF_CHARS 256

/* function to check whether two strings are anagram of each other */
bool areAnagram(char *str1, char *str2)
{
    // Create two count arrays and initialize all values as 0
    int count1[NO_OF_CHARS] = {0};
    int count2[NO_OF_CHARS] = {0};
    int i;

```



Recent Comments

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

[Backtracking | Set 7 \(Sudoku\)](#) · 16 minutes ago

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

[Flipkart Interview | Set 6](#) · 36 minutes ago

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

[Software Engineering Lab, Samsung Interview | Set 2](#) · 36 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

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

[Find depth of the deepest odd level leaf node](#) · 2 hours ago

AdChoices

[► Anagram Solver](#)

[► C Strings](#)

```
// For each character in input strings, increment count in
// the corresponding count array
for (i = 0; str1[i] && str2[i]; i++)
{
    count1[str1[i]]++;
    count2[str2[i]]++;
}

// If both strings are of different length. Removing this condition
// will make the program fail for strings like "aaca" and "aca"
if (str1[i] || str2[i])
    return false;

// Compare count arrays
for (i = 0; i < NO_OF_CHARS; i++)
    if (count1[i] != count2[i])
        return false;

return true;
}

/* Driver program to test to pront printDups*/
int main()
{
    char str1[] = "geeksforgeeks";
    char str2[] = "forgeeksgeeks";
    if (areAnagram(str1, str2) )
        printf("The two strings are anagram of each other");
    else
        printf("The two strings are not anagram of each other");

    return 0;
}
```

Output:

The two strings are anagram of each other

The above implementation can be further to use only one count array instead of two. We can increment the value in count array for characters in str1 and decrement for characters in str2. Finally, if all count values are 0, then the two strings are anagram of each other. Thanks to [Ace](#) for suggesting this optimization.

```
bool areAnagram(char *str1, char *str2)
{
```

```


// Create two count arrays and initialize all values as 0
int count[NO_OF_CHARS] = {0};
int i;

// For each character in input strings, increment count in
// the corresponding count array
for (i = 0; str1[i] && str2[i]; i++)
{
    count[str1[i]]++;
    count[str2[i]]--;
}

// If both strings are of different length. Removing this condition
// will make the program fail for strings like "aaca" and "aca"
if (str1[i] || str2[i])
    return false;

// See if there is any non-zero value in count array
for (i = 0; i < NO_OF_CHARS; i++)
    if (count[i])
        return false;
return true;
}


```

AdChoices 

► [C Strings](#)

► [Just for Strings](#)

► [Strings All](#)

AdChoices 

► [String Java](#)

► [Anagram Anagrams](#)

► [Replace String](#)

If the possible set of characters contains only English alphabets, then we can reduce the size of arrays to 52 and use `str[i] - 'A'` as an index for count arrays. This will further optimize this method.

Time Complexity: $O(n)$

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Better Than Hadoop.

HPCC Systems is Big Data Processing and Analytics
Open Source. Proven. Trusted.

 LexisNexis® [Learn More](#) 

Related Topics:

- 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)



17

 Tweet

2



2

Writing code in comment? Please use [ideone.com](https://www.ideone.com) and share the link here.

72 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion



with the algorithm...



kinshuk chandra • 5 days ago

I liked the counting approach as it is $O(n)$ timewise and $O(1)$ space wise, as o my blog with [same post](#).

^ | v • Reply • Share ›



Yogeswari Rengarajan • 13 days ago

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

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

```
#define ANAGRAM 1
```

```
#define NOTANAGRAM 0
```

```
void test_anagram();
```

```
void string_count(char x[], char count[])
```

```
{
```

```
int i;
```

```
for(i=0;x[i]!='\0';i++)
```

```
{
```

```
count[x[i] - 'a']++;
```

[see more](#)

^ | v • Reply • Share ›



vaibhav • 19 days ago

```
import java.util.*;
```

```
public class Anagrams {  
  
    public static void main(String args[])  
  
    {  
  
        String str1= "this is that high";  
  
        String str= "this is that high always";  
  
        String s[]=str.split("");  
  
        String s2[]=str1.split("");  
  
        ArrayList al=new ArrayList(Arrays.asList(s));  
  
        ArrayList al2=new ArrayList(Arrays.asList(s2));  
  
        Collections.sort(al);
```

[see more](#)

^ | v • Reply • Share ›



dwa • a month ago

counting method will fail ... eg., abaz and aabz .. these two are not anagrams

1 ^ | v • Reply • Share ›



Saravana Kumar • 4 months ago

How abt the following son?

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

```
int main(void) {
```

```
// your code goes here
```



```
printf("%s",(AreAnagrams("life", "fiel")?"Anagrams " : "Not Anagrams") );  
  
return 0;  
  
}  
  
int AreAnagrams(char *str1, char *str2)  
  
{  
  
int cnt=0,i=0;  
  
for(i=0;(str2[i]&&str2[i]):i++)
```

[see more](#)

6 ^ | v • Reply • Share ›



Satyam Dhar → Saravana Kumar • 24 days ago

This will not work because the sum of different strings can be same. e

ABC and AAD

both sum up to 198 but are not anagrams of each other.

^ | v • Reply • Share ›



Gourav Mahindra J • 6 months ago

In java store the char of the first string as a key in a map. Value for it will be the number of times it appears in the first string. Then start decrementing this count from the map. When the count becomes 0, remove it from the map.

At the end if the map is empty, it means the strings are anagrams.

4 ^ | v • Reply • Share ›



aam admi • 6 months ago



Why to take 2 count arrays? Only one can meet the purpose. For string 1 increment and for string 2 decrement. Finally just check if every value in count array is zero!

2 ^ | v • Reply • Share ›



Gourav Mahindra J → aam admi • 6 months ago

Yes one is sufficient with an additional check - when the count becomes zero, break the loop and return false. This condition can be used only when we deal with strings.

^ | v • Reply • Share ›



Dhaval Dave • 8 months ago

Can't we do it with Sum of Ascii values? example "abc" will sum into 97+98+99 (If both string's length are same).

->I have tested for some random test cases,

->Please Provide some failed test cases in above method if I am wrong

-> PS : as taking strings I am not considering \b ,\n ,\t as character in string

^ | v • Reply • Share ›



RahulMahale → Dhaval Dave • 3 months ago

there can be two different combination of characters that gives same sum

For EX ACE , BDC

2 ^ | v • Reply • Share ›



shashi kumar → Dhaval Dave • 8 months ago

abf, ace. both sum to 296. etc...

9 ^ | v • Reply • Share ›



Ankita • 8 months ago

We can first find the lengths of the two strings. If their lengths are equal then we can XOR them..

Code in C

```
#include<stdio.h>

#include<string.h>

enum {NO, YES};

unsigned isAnagram(char *s1, char *s2)

{

int res = 0,i;

if(strlen(s1) != strlen(s2))

return NO;
```

[see more](#)

^ | v • [Reply](#) • [Share](#) ›



Jayant Gupta → [Ankita](#) • 6 months ago

XOR doesn't work.

Consider the example : "ABC", "ECF".

Both the strings are different, yet when XOR is applied they return 0.

The reason being XOR only take into account a single bit and it can ge that all the bits occur even number of times. Furthermore, XOR(s1, s2 but not the only condition.

1 ^ | v • [Reply](#) • [Share](#) ›



Guest • 8 months ago

What you guys say about this solution,
add and subtract each character code and check if the result is zero
Code in java script

```
function isAnagram(s1,s2){  
    result = 0;  
    if(s1.length != s2.length ){  
        return false;  
    }  
    for(var i=0;i<s1.length;i++){ result="" += "(s1.charCodeAt(i))" -= ""
```

1 ^ | v • Reply • Share ›



upman16 ➔ Guest • 8 months ago

So if you pass "hell\n" and "leh\nl"

ASCII values

h- 72, e - 69, l - 76, \n - 10

So add and subtract all the characters,

$72-76 + 69 - 69 + 76-72 + 76-10 + 10-76$

(h-l + e-e + ...)

Everything cancels out and you've got 0.

Cool, but do the same thing with

"heml\n" and "lfh\nl"

ASCII values

h- 72, e - 69, l - 76, \n - 10, m - 77, f - 70

$72-76 + 69 - 70 + 77-72 + 76-10 + 10-76$

You'll still get zero!

FAILS

2 ^ | v • Reply • Share ›



Guest • 8 months ago

What you guys say about this solution,

add and subtract each character code and check if the result is zero

add and subtract each character code and check if the result is zero

Code in java script

```
function isAnagram(s1,s2){  
    result = 0;  
    if(s1.length != s2.length ){  
        return false;  
    }  
    for(var i=0;i<s1.length;i++){ result=result+(s1.charCodeAt(i)) -(s2.charCodeAt(i))
```

1 ^ | v • Reply • Share ›



devesh101batra • 10 months ago

for those who are suggesting xor approach .. its wrong approach ..
consider the case "aaaab" and "aab" the xor resultant =0 but these are not anagrams

^ | v • Reply • Share ›



koolkeshaw • 10 months ago

We know XOR of same nos is 0 i.e

a XOR a=0 ; a --> is any int value

using the above logic to check for anagrams won't require any auxiliary space

For code

refer to :- <http://ideone.com/AjSvDx>

Plz do reply if I am wrong

^ | v • Reply • Share ›



Vibhu Tiwari • 10 months ago

I think that without sorting we can check for the anagram property by just storing the characters in a binary tree and then do the traversals to find the anagram. It will take just a little more time but it's just another way:

suppose the string is:

Suppose the string is:

----a-----

--b---c--

then all anagrams can be generated as:

abc- Preorder leftwise.

acb- Preorder rightwise

bac- Inorder leftwise

cab- Inorder rightwise

bca- Postorder leftwise

cba- Postorder rightwise

Thus for abc we can make 6 arrangements for all the possible anagrams. Thus during any time we obtain the value equal to the second string then we can say each other.

^ | v • Reply • Share ›



Avinash Kumar → Vibhu Tiwari • 8 months ago

It is the same as finding permutations of the string and check for each

```
/* Paste your code here (You may delete these lines if not write your code here) */
```

1 ^ | v • Reply • Share ›



zyzz • 10 months ago

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

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

```
void sort(char *s){
```

```
char *i,*j;
```

```
char temp;
```

```
for(i=s; *i!='&#092;&#048':i++){
```

```

for(j=s++; *j!='&#092&#048';j++){

    if(*i>*j){
        temp=*i;
        *i=*j ;
        *j=temp;
    }

```

see more

^ | v • Reply • Share ›



Harsh Bansal • 11 months ago

```

int charindex(char c).
{
    int result;
    result = (int)c - (int)&#039a&#039;.
    return result;
}

int check(char *a, char *b).
{
    int alen = strlen(a);.
    int blen = strlen(b);.
    int count[26] = {0};.
    int i, index;
    if(alen!=blen)
    {
        return 0;
    }
    for(i=0;i<alen;i++)
    {

```

[see more](#)

^ | v • Reply • Share ›



Gautam Singh • 11 months ago

your implementation not correct for the all value ,, only for the specific input ...

^ | v • Reply • Share ›



Khemais Djelidi • 11 months ago

boolean anagram(String s, String t) {return sort(s) == sort(t); }.

Thanks to Gayle Laakmann for her book "CRACKING THE CODING INTERVIEW
Enjoy it ;).

1 ^ | v • Reply • Share ›



friendmaddy • a year ago

```
public bool TwoStringAnagram(string source, string target)
{
    if (string.IsNullOrEmpty(source) | string.IsNullOrEmpty(target) | source.Length
    return false;
    int[] tmp = new int[256];
    for (int i = 0; i < source.Length; i++)
    {
        tmp[source language="[i"][/source]]++;
        tmp[target[i]]--;
    }

    for(int i=0;i<source.Length;i++)
    {
        if (tmp[source language="[i"][/source]] != 0 | tmp[target[i]] != 0)
        return false;
    }
    return true;
}
```


^ | v • Reply • Share ›



friendmaddy → friendmaddy • a year ago

```
public bool TwoStringAnagram(string source, string target)
{
    if (string.IsNullOrEmpty(source) | string.IsNullOrEmpty(target) | source
    return false;
    int[] tmp = new int[256];
    for (int i = 0; i < source.Length; i++)
    {
        tmp[source language="[i"][/source]]++;
        tmp[target[i]]--;
    }

    for(int i=0;i<source.length;i++) {="" if="" (tmp[source="" language="[i" ]|
    tmp[target[i]]="" !="0)" return="" false;="" }="" return="" true;="" }="">
```

^ | v • Reply • Share ›



MS GUY → friendmaddy • a year ago

cool solution...

^ | v • Reply • Share ›



timus → MS GUY • 10 months ago

nice work ...

^ | v • Reply • Share ›



harish • a year ago

you can try this one which is rather simple .. The code is in java....

```
package com.string;
```

```
import java.util.Scanner;
```

```

public class StringAnagram {

    private static int[] countChars = new int[256];

    public boolean isAnagram(String s1,String s2)
    {
        if(s1.length()!=s2.length())
            return false;

        char[] charArray1 = s1.toCharArray();
        char[] charArray2 = s2.toCharArray();

        for(int i=0;i<charArray1.length;i++)
        {

```

[see more](#)

2 ^ | v • Reply • Share ›



Lokesh Gopu • a year ago

Its not correct!.

for ex. consider CD and BE

$(C^D) ^ (B ^ E) = 0$.. But they are not anagrams

^ | v • Reply • Share ›



cyberWolf • a year ago

@GeeksForGeeks: In last approach, Should there be any length check before

Also, the length check that has been used after the hash could be faulty.

```

    if (str1[i] || str2[i])
        return false;

```

It could be possible that the strings have been memset with 0 or string has null end. So when we check for this condition (e.g. str1 = "ABC\0\0\0" , str2 = "AAE") return false as expected.

Kindly check it.

^ | v • Reply • Share ›



Jeff Ngo • a year ago

nice

^ | v • Reply • Share ›



Arpit Gupta • a year ago

```
public static boolean checkAnagramV2(String str1, String str2){
```

```
    if(str1.length()!= str2.length())
```

```
        return false;
```

```
    else{
```

```
        Integer val1 =0, val2=0;.
```

```
        for (int i=0;i<str1.length();i++).
```

```
        {
```

```
            val1 = (str1.charAt(i)-0) ^ val1;.
```

```
            val2 = (str2.charAt(i) -0) ^ val2;.
```

```
        }
```

```
        if ((val1^val2) == 0).
```

```
            return true;
```

```
        else
```

```
            return false;
```

```
    }
```

```
}
```

This is another way with complexity on $O(n)$.

^ | v • Reply • Share ›

^ | v • Reply • Share ›



alien • a year ago

Algo:

- 1.) Calculate hash values for both the strings
- 2.) Compare hash value. If same then anagram

time complexity depends on hash function. Space complexity: $O(1)$.

^ | v • Reply • Share ›



Star_Trek → alien • a year ago

@alien-gud algo

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

^ | v • Reply • Share ›



Vinay Gangoli • a year ago

Pseudo Code:

1. If $S1 \neq S2$, return false;
2. Insert every char of $S1$ into a HashMap
3. For every char in $S2$, remove it from HashMap. If no entry is found, return fa
4. return true;

This should run in $O(n+m)$, where n and m are lengths of $S1$ and $S2$.

^ | v • Reply • Share ›



Vinay Gangoli → Vinay Gangoli • a year ago

#correction

First line should read as

If $S1.length \neq S2.length$.

1 ^ | v • Reply • Share ›



Raghav · a year ago

Associate each character with a prime number and compute the product for each string. If the products are the same then they are anagrams.

$O(n)$ time (assuming character set is small and so computing prime for each character is $O(1)$)
 $O(1)$ space.

^ | v · Reply · Share ›



Swati · a year ago

Isn't it that simple.. just sort both strings in alphabetical order and compare the sorted strings.

^ | v · Reply · Share ›



Blog · a year ago

@Robin : I became attempting to send you some text nonetheless can't believe I missed your website. Would you need to get in touch?

^ | v · Reply · Share ›



sid · 2 years ago

i have a very simple approach for this... please correct me if i am wrong...

to check anagram :

1.concatenate the 1st string with itself

2.find 2nd string as a substring in new string. if substring found, then they are anagrams.

eg:

s1= abcd

s2=dabc

s3=abcdabcd // concatenate 1st string with itself

s2=dabc is a substring in s3.

hence s1 and s2 are anagrams .

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

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

^ | v • Reply • Share ›



Hassan → sid • 2 years ago

Hi. Unfortunately this does not work. This approach ONLY works in you permutation of s1. Consider the anagram s3 = badc. Your method will

^ | v • Reply • Share ›



Star_Trek → Hassan • a year ago

@sid...ur method is gud...but dnt jup into any conclusions befor possible...

:)

^ | v • Reply • Share ›



Apurva • 2 years ago

XOR the two strings and check if the result is 0.

^ | v • Reply • Share ›



vindhya → Apurva • 2 years ago

i think this method wont work for..

```
str1="dd"
```

```
str1="ss"
```

it will give ans as true but actually it is not correct.

..could you please explain how to implement what you mean by xoring



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

^ | v • Reply • Share ›



vindhya → vindhya • 2 years ago

i think this method wont work for..

```
str1="dd"
```

```
str2="ss"
```

it will give ans as true but actually it is not correct.

..could you please explain how to implement what you mean by

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

^ | v • Reply • Share ›



pr6989 • 2 years ago

//ANAGRAM CHECKING

```
#include
```

```
#include
```

```
#include
```

```
#define max 10
```

```
int compare (const void * a, const void * b)
```

```
{
```

```
return ( *(char*)b - *(char*)a );
```

```
}
```

```
int main()
```

```
{
```

```
char s[max];
```

```
char t[max];
```

```
printf("Enter a string : ");
```

```
scanf("%s",s);
```

```
getchar();
```

```
printf("Enter another string : ");
```

```
scanf("%s",t);
```

[see more](#)

^ | v • Reply • Share ›



Aman • 2 years ago



Aman • 2 years ago

What about this one?

```
#include<stdio.h>
#include<string.h>
#define MAX 10000
int main()
{
long mul1=1,mul2=1,add1=0,add2=0;
char a[MAX],b[MAX];
gets(a);
gets(b);
int k=strlen(a);
int flag=k-1;
if(strlen(a)==strlen(b))
{
while(flag>=0)
{
add1+=*(a+flag);
```

[see more](#)

^ | v • Reply • Share ›



Aman • 2 years ago

Better would be to use ASCII values. Sum of ascii values&&multiplication of as should be equal

^ | v • Reply • Share ›

Load more comments

