Dynamic Programming | Set 33 (Find if a string is interleaved of two other strings)

Given three strings A, B and C. Write a function that checks whether C is an interleaving of A and B. C is said to be interleaving A and B, if it contains all characters of A and B and order of all characters in individual strings is preserved.

We have discussed a simple solution of this problem here. The simple solution doesn't work if strings A and B have some common characters. For example A = "XXY", string B = "XXZ" and string C = "XXZXXXY". To handle all cases, two possibilities need to be considered.

a) If first character of C matches with first character of A, we move one character ahead in A and C and recursively check.

b) If first character of C matches with first character of B, we move one character ahead in B and C and recursively check.

If any of the above two cases is true, we return true, else false. Following is simple recursive implementation of this approach (Thanks to Frederic for suggesting this)

```
// A simple recursive function to check whether C is an interleaving of A and
bool isInterleaved(char *A, char *B, char *C)
{
    // Base Case: If all strings are empty
    if (!(*A || *B || *C))
        return true;

    // If C is empty and any of the two strings is not empty
    if (*C == '\0')
        return false;

    // If any of the above mentioned two possibilities is true,
    // then return true, otherwise false
    return ( (*C == *A) && isInterleaved(A+1, B, C+1))
           || ((*C == *B) && isInterleaved(A, B+1, C+1));
}
```

Dynamic Programming

The worst case time complexity of recursive solution is $O(2^n)$. The above recursive solution certainly has many overlapping subproblems. For example, if wee consider A = “XXX”, B = “XXX” and C = “XXXXXX” and draw recursion tree, there will be many overlapping subproblems.
Therefore, like other typical Dynamic Programming problems, we can solve it by creating a table and store results of subproblems in bottom up manner. Thanks to Abhinav Ramana for suggesting this method and implementation.

```
// A Dynamic Programming based program to check whether a string C is
// an interleaving of two other strings A and B.
#include <iostream>
#include <string.h>
using namespace std;

// The main function that returns true if C is
// an interleaving of A and B, otherwise false.
bool isInterleaved(char* A, char* B, char* C)
{
    // Find lengths of the two strings
    int M = strlen(A), N = strlen(B);

    // Let us create a 2D table to store solutions of
    // subproblems.  C[i][j] will be true if C[0..i+j-1]
    // is an interleaving of A[0..i-1] and B[0..j-1].
    bool IL[M+1][N+1];

    memset(IL, 0, sizeof(IL)); // Initialize all values as false.

    // C can be an interleaving of A and B only of sum
```

```
    // of lengths of A & B is equal to length of C.
    if ((M+N) != strlen(C))
       return false;

    // Process all characters of A and B
    for (int i=0; i<=M; ++i)
    {
        for (int j=0; j<=N; ++j)
        {
            // two empty strings have an empty string
            // as interleaving
            if (i==0 && j==0)
                IL[i][j] = true;

            // A is empty
            else if (i==0 && B[j-1]==C[j-1])
                IL[i][j] = IL[i][j-1];

            // B is empty
            else if (j==0 && A[i-1]==C[i-1])
                IL[i][j] = IL[i-1][j];

            // Current character of C matches with current character of A,
            // but doesn't match with current character of B
            else if(A[i-1]==C[i+j-1] && B[j-1]!=C[i+j-1])
                IL[i][j] = IL[i-1][j];

            // Current character of C matches with current character of B,
            // but doesn't match with current character of A
            else if (A[i-1]!=C[i+j-1] && B[j-1]==C[i+j-1])
                IL[i][j] = IL[i][j-1];

            // Current character of C matches with that of both A and B
            else if (A[i-1]==C[i+j-1] && B[j-1]==C[i+j-1])
                IL[i][j]=(IL[i-1][j] || IL[i][j-1]) ;
        }
    }

    return IL[M][N];
}

// A function to run test cases
void test(char *A, char *B, char *C)
{
    if (isInterleaved(A, B, C))
        cout << C <<" is interleaved of " << A <<" and " << B << endl;
    else
        cout << C <<" is not interleaved of " << A <<" and " << B << endl;
}


// Driver program to test above functions
int main()
```

```
{
    test("XXY", "XXZ", "XXZXXXY");
    test("XY" ,"WZ" ,"WZXY");
    test ("XY", "X", "XXY");
    test ("YX", "X", "XXY");
    test ("XXY", "XXZ", "XXXXZY");
    return 0;
}
```

Output:

```
XXZXXXY is not interleaved of XXY and XXZ
WZXY is interleaved of XY and WZ
XXY is interleaved of XY and X
XXY is not interleaved of YX and X
XXXXZY is interleaved of XXY and XXZ
```

See this for more test cases.

Time Complexity: O(MN)
Auxiliary Space: O(MN)

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

Related Topics:

- Recursively print all sentences that can be formed from list of word lists
- Check if a given sequence of moves for a robot is circular or not
- Find the longest substring with k unique characters in a given string
- Function to find Number of customers who could not get a computer
- Find maximum depth of nested parenthesis in a string
- Find all distinct palindromic sub-strings of a given string
- Find if a given string can be represented from a substring by iterating the substring “n” times
- Suffix Tree Application 6 – Longest Palindromic Substring

Tags: Dynamic Programming

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

It can be solved in O(M+N) time , here is the following logic:
Here is the link :
http://ideone.com/pp6Z6z

• Reply • Share ›

Guest • 3 months ago
Why use dynamic programming when we can get answer in O(m+n) Complexity?

• Reply • Share ›

Itachi Uchiha ➚ Guest • 2 months ago
Dude! O(m+n) solution doesn't handle the case when there are common characters! in A and B! . You than go for recursive version which handles common characters case but is exponential (2^n). So finally we get to the dp solution above handling all cases!

• Reply • Share ›

Simone Pistocchi • 5 months ago
http://ideone.com/10ZiY3
this solution seems to work well also if there are characters in common but it is O(N+M) and doesn't use additional memory.
Correct me if i am wrong

• Reply • Share ›

Avata This comment was deleted.

Ravi Balocha ➚ Guest • 7 days ago
Nobody is interested in your solutions. so please stop spamming geeksforgeeks with your shitty comments. Sorry for being harsh on you but thats true.

• Reply • Share ›

Alex ➚ Simone Pistocchi • 5 months ago
fails on "dabc","def" and "dadbecf"

• Reply • Share ›

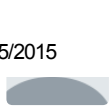
Simone Pistocchi ➚ Alex • 5 months ago
I correct it in this version http://ideone.com/GVWoco

• Reply • Share ›

ishan ➚ Simone Pistocchi • 4 months ago
fails on "aabc", "abad", "aabcabad"

• Reply • Share ›

Guest • 6 months ago
There are multiple errors in indexing the array.



• Reply • Share ›

Cathy Liu • 8 months ago

The dynamic programming version will not work if A or B is an empty string. An index out bound exception will be occur for A="", B="x", and C="y"

The loop should terminate like this for i == 0 and j == 0

```
// A is empty
else if (i==0){
if (B[j-1]==C[j-1])
IL[i][j] = IL[i][j-1];
}

// B is empty
else if (j==0){
if (A[i-1]==C[i-1])
IL[i][j] = IL[i-1][j]; }
```

• Reply • Share ›

Saurabh • 9 months ago

A O(n) approach for this problem: (Iterative Solution)
http://ideone.com/e6NH0F

Please comment if it fails for any cases.

• Reply • Share ›

majineu ➚ Saurabh • 8 months ago

```
char *a ="aabbb";

char *b= "aaabbc";

char *c= "aaabbcaabbb";
```

• Reply • Share ›

Mohit Gandhi • 10 months ago

Please somebody help me out! Cant We just check first if strlen(C) is equal to sum of lengths of A and B. And if it is, Traverse C two times , seperately for A and B and check whether both are sub"sequence" of C, If yes return TRUE., Please Let me know what is wrong in this approach, Thanks in advance.

• Reply • Share ›

Krishna Kumar ➚ Mohit Gandhi • 10 months ago

Strings A and B can be overlapping, for A = XXY, B = XXZ, C = XXYZ, if parsed separately will return true, but, it should be false.

• Reply • Share ›

Harman Patel ➚ Krishna Kumar • 7 months ago

He said first check length of C with sum of length of A and B. U gave an example having unequal lengths.

• Reply • Share ›

vickychijwani ➚ Harman Patel • 6 months ago

Take A = XXY, B = XXZ, C = XXYYZZ. A and B are subsequences of C, and |A| + |B| = |C|.

• Reply • Share ›

Harman Patel ➚ vickychijwani • 6 months ago

got it
thanks :D

• Reply • Share ›

Krishna Kumar • 10 months ago

Can this be a solution with O(n) complexity?

```
private static boolean isInterleaved(char[] firstString,
char[] secondString, char[] interleavedString)
{
int aLength = firstString.length;
int bLength = secondString.length;
int cLength = interleavedString.length;

if (aLength + bLength != cLength)
{
System.out.println(String.valueOf(firstString) + " + "
+ String.valueOf(secondString) + " !IL "
+ String.valueOf(interleavedString));
return false;
}

int ai = 0;
int bi = 0;
```

see more

• Reply • Share ›

krishna • 10 months ago

authors why don't you explain concept of interleaved string with some examples

"XXZXXXY (c)is not interleaved of XXY(a) and XXZ(b)"

why it is not interleaved c contains all chars of a and b.and order is also preserved??

• Reply • Share ›

Rodrigo Bernardino ➚ krishna • 10 months ago

Please, see this:
http://algorithmsandme.blogspo...
The problem is that it has XXZ + X + XXY
It must contain precisely the same amount of chars

• Reply • Share ›

Arnab • a year ago

complexity: O(n)

public static String isInterleaved(String a,String b,String c){

if((a.length()+b.length())!=c.length())

return "No";

char[] a1=a.toCharArray();

char[] b1=b.toCharArray();

char[] c1=c.toCharArray();

int i=0,j=0,k=0;

String ans="Yes";

while(k<c.length()) {="" if(="" i<a.length()="" &&="" a1[i]="c1[k]")="" {="" system.out.print("\n="" match="" a");="" i++;k++;="" }="" else="" if(="" j<b.length()="" &&="" b1[j]="c1[k]")="" {="" system.out.print("\n="" match="" b");="" j++;k++;="" }="" else="" {="" system.out.print("\n="" match="" none");="" ans="No" ;="" break;="" }="" }="" return="" ans;="" }="">

• Reply • Share ›

minhaz palasara • a year ago

Hey Guys, tried O(n) solution.. Till now all test passed please reply in case it fails

public class DP5 {

public static void main(String args[]){
String A= "aabcc";
String B = "dbbca";
String C = "aadbcbbcac";
DP5 d = new DP5();
System.out.println(d.check(A.toCharArray(), B.toCharArray(), C.toCharArray()));

```
}

boolean check(char[] A,char[] B,char C[]){
if(A.length+B.length <= C.length){
//pointer of A
int j = 0;
//poiner of B
int k = 0;
int repeat = 0;
```

see more

• Reply • Share ›

mailoalerts • a year ago

A = AAB
B= AAC
C= AACAAB
C is interleave of A and B . If i undrstnd it right.
Bt given prgrm give Output as

"AACAAB is not interleaved of AAB and AAC".
Cn anyone plz xplain it .
whether I am getting the concept of interleaved string wrong ???

3 • Reply • Share ›

guest ➚ mailoalerts • a year ago

It is AACAAAB u missed out on a 'A'

• Reply • Share ›

Guest • a year ago

Code with Time Complexity O(m+n)

```
#include<stdio.h>
#include<string.h>
int Interleave(char str1[],char str2[],char a[])
{
int i=0,j=0,k=0;
while(a[k])
{
if(str1[i]&&(a[k]==str1[i]))
i++;
else if(str2[j]&&(a[k]==str2[j]))
j++;
else
return 0;
```

```
k++;
}
return 1;
```

see more

• Reply • Share ›

prashant saxena • a year ago

Well, the solution assumes c has to contain only characters from a,b. Say a="pqrs" and b="tuvw". If c="ptqurvswx" it wil fail. How ever it is still interleaving.. correct?

• Reply • Share ›

Gallon ➚ prashant saxena • a year ago

The example you gave is covered on sanity check.

```
if ((M+N) != strlen(C))
return false;
```

1 • Reply • Share ›

prashant saxena ➚ Gallon • a year ago

Yeah exactly so it will return false although it is still interleaving.

• Reply • Share ›

dawich77 ➚ prashant saxena • a year ago

no it isn't interleaving, only if the leftover parts could be the beginning of one of the sub strings
ex

a= 10
b= 111
c= 111101
Could be interleaving because the last 1 can be the start of a or b.
but if c=111100 then it wouldn't be.

The problem you have is x isn't accounted for in either a or b.

further clarification: http://en.wikipedia.org/wiki/I...

• Reply • Share ›

DexterLtd • a year ago

@geeksforgeeks , there is one case where some illegal operation is happening.

Consider test case in which str A is empty and str B is "XXXX" C is "XYXX".
now if (i==0 && B[j-1]==C[j-1]) condition fail for C[i+j-1] is Y so it goes to next conditions and where it will perform a[i-1] when i is actually 0.which is not good.

Also if we find that when one string is empty, then we need not to iterate till the end of other string even if we found some mismatch. its simply overhead in case of very long string.

4 • Reply • Share ›

bhopu • a year ago

i think this case may handle in O(m+n) time also please check it...

```
#include<stdio.h>

#include<string.h>

/* function to check interleaved or not */

interleaving(char *str1,char *str2,char *str)

{

int flag=0;

int i=0,j=0,k=0,s1,s2,s3;

s1=strlen(str1);

s2=strlen(str2);

s3=strlen(str);
```

see more

• Reply • Share ›

Ankit Jain ➚ bhopu • a year ago

A = "aabcc",
B = "dbbca",
C = "aadbcbbcac".

Please check yourself.

• Reply • Share ›

vinod95300 • a year ago

Its not a DP Problem at all...!!!
Step 1: Check weather the frequency of characters in C is same as of (A+B)
If not then print"NO";

Else go to Step 2
Step 2: Copy C in other string, let C1...
Step 3: Check weather A is subsequence of C1 or not by character by character and simultaneously change those characters in C1 as '1' which are found same as that of A while checking.
Step 4: Same as Step 3 BUT here check weather B is subsequence of C1(modified in Step 3).

Step 5: If we found that A and B both are subsequences of modified C1 then print "YES"

Else if Any one of A or B is found as subsequence of C1 then

Repeat the process from Step 2 only for once again BUT THIS TIME check for B first than check for A first.... If then also it was found that any one of A or B is not the subsequence of C1 then print "NO" else print "Yes"

Else if both are not the subsequence then print NO

see more

• Reply • Share ›

Rajesh vinod95300 • a year ago
This is not working for the below case. Answer should be true, but from your method, it is giving false.

A = "aabcc",
B = "dbbca",
C = "aadbcbbcac".

Please check yourself.

• Reply • Share ›

KMP • a year ago
Why not just use KMP substring search

• Reply • Share ›

Mr.Nobody • a year ago
Why is this a DP problem to begin with?
This can be done in linear time. O(strlen(C)) to be precise. Code: http://ideone.com/lnEJQG
Anything wrong with this solution?

4 • Reply • Share ›

Ankit Vani Mr.Nobody • a year ago
because it does not detect things like:
XXY
XXZ
XXZXXY

XXZXXY

• Reply • Share ›

KMP Mr.Nobody • a year ago

Use this seems right.

• Reply • Share ›

Satyanarayana Bolenedi • a year ago

/*Check whether a given string is an interleaving of two other given strings*/

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

// Returns true if C is an interleaving of A and B, otherwise
// returns false
bool isInterleaved (char *A, char *B, char *C)
{
char *a=A;
char *b=B;
int found=0;

// Iterate through all characters of C.
while (*C != 0)
{
found=0;
//first character of C with first character of A,
// If matches them move A to next
```

see more

• Reply • Share ›

NeedHelp • a year ago

Okay, I need help understanding:

```
// A is empty

else if (i==0 && B[j-1]==C[j-1])

IL[i][j] = IL[i][j-1];

// B is empty

else if (j==0 && A[i-1]==C[i-1])

IL[i][j] = IL[i-1][j];
```

I know that it runs the i == 0 and j ==0 if statement first, then j increments to 1 and i is still 0. It

goes to "A is empty" statement. But now does j become 0 again for the "B is empty statement"? I'm confused here...

• Reply • Share ›

gopsguru • a year ago

Is the solution of first deleting A from C and let after removing all characters of A from C we get D, Comparing D with B and if B==D then C is interleaved.

Is this solution correct. Plz give me cases when It fails.

3 • Reply • Share ›

atang gopsguru • a year ago

You are right,
The run time for you algorithm is o(m+n)

• Reply • Share ›

gopsguru atang • a year ago

The algo doesnt work if 2 strings have common char eg
A= xxx and B= xxy and C= xxxxyx

• Reply • Share ›

temp • 2 years ago

The program with dynamic solution will crash..... the for loop starts from i=0 and inside the body of the second loop we are checking A[i-1] and IL[i-1] ..... array dont take negative index... when i=0, the index would be A[0-1] ----> CRASH

2 • Reply • Share ›

Niranjan Viladkar temp • 2 years ago

Hi @temp ,
It will not crash. Please take a closer look. A[i-1] or IL[i-1] gets accessed only when i is not zero. If i is zero, then there are conditions to handle.

• Reply • Share ›

temp Niranjan Viladkar • a year ago

lets say j>0 and B[j-1]!=C[j-1] ..... then ?

4 • Reply • Share ›

Sanjay Agarwal • 2 years ago

We can solve this problem using a flag array which keeps track of the matched characters between string c and string a as well as between string c and string b.
Worst Time Complexity: O(6k).
Space Complexity: O(k)
Where:
n = no. of characters in string a.

m = no. of characters in string b.
k = no. of characters in string c.
Please feel free to comment if you find any bug in my logic or you need more explaination to this approach.

```
#include<stdio.h>
#include<stdbool.h>
#include<string.h>
int STATE_ONE = 1;
int STATE_TWO = 2;
bool is_interleaving_util(char *a, char *b, char *c, int state)
{
```

see more

• Reply • Share ›

SAGAR JHOBALIA • 2 years ago

Exactly: O(n) solution:C- (A+B) or C- (B+A). If any equal 0 thus proved.
I think the challenge would be to find if C 'contains' the A & B.
I did it here: https://github.com/Midnight-Co...
Haven't been able to code to check if A & B overlap in C

3 • Reply • Share ›

daghan dinc • 2 years ago

// O(n) solution

#include<stdio.h>

// first erases a within c, then erases b within c

// if all 3 strings are consumed returns true (1),

// otherwise false (0)

// Notes:

// - modifies c: but you can create a copy of it if you wish

// - assumes strings do not contain \n (erase char):

// you can use \0 instead but then you need to use lengths

int isInterleaving(char*a,char*b,char*c){

char *ca = c,*cb = c; // create copies of c pointer

see more

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- 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

- Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 2 hours ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 2 hours ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 2 hours ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago