GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

Longest Palindromic Substring | Set 1

Given a string, find the longest substring which is palindrome. For example, if the given string is "forgeeksskeegfor", the output should be "geeksskeeg".

Method 1 (Brute Force)
The simple approach is to check each substring whether the substring is a palindrome or not. We can run three loops, the outer two loops pick all substrings one by one by fixing the corner characters, the inner loop checks whether the picked substring is palindrome or not.

Time complexity: O (n^3)
Auxiliary complexity: O (1)

Method 2 (Dynamic Programming)

The time complexity can be reduced by storing results of subproblems. The idea is similar to this post. We maintain a boolean table[n][n] that is filled in bottom up manner. The value of table[i][j] is true, if the substring is palindrome, otherwise false. To calculate table[i][j], we first check the value of table[i+1][j-1], if the value is true and str[i] is same as str[j], then we make table[i][j] true. Otherwise, the value of table[i][j] is made false.

```
// A dynamic programming solution for longest palindr.
// This code is adopted from following link
// http://www.leetcode.com/2011/11/longest-palindromic-substring-part-i.html

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

// A utility function to print a substring str[low..high]
void printSubStr( char* str, int low, int high )
{
    for( int i = low; i <= high; ++i )
        printf("%c", str[i]);
}

// This function prints the longest palindrome substring
// of str[].
// It also returns the length of the longest palindrome
int longestPalSubstr( char *str )
{
    int n = strlen( str ); // get length of input string

    // table[i][j] will be false if substring str[i..j]
    // is not palindrome.
    // Else table[i][j] will be true
    bool table[n][n];
    memset(table, 0, sizeof(table));

    // All substrings of length 1 are palindromes
    int maxLength = 1;
    for (int i = 0; i < n; ++i)
        table[i][i] = true;

    // check for sub-string of length 2.
    int start = 0;
    for (int i = 0; i < n-1; ++i)
    {
        if (str[i] == str[i+1])
        {
            table[i][i+1] = true;
            start = i;
            maxLength = 2;
        }
    }

    // Check for lengths greater than 2. k is length
    // of substring
    for (int k = 3; k <= n; ++k)
```

```
    {
        // Fix the starting index
        for (int i = 0; i < n-k+1 ; ++i)
        {
            // Get the ending index of substring from
            // starting index i and length k
            int j = i + k - 1;

            // checking for sub-string from ith index to
            // jth index iff str[i+1] to str[j-1] is a
            // palindrome
            if (table[i+1][j-1] && str[i] == str[j])
            {
                table[i][j] = true;

                if (k > maxLength)
                {
                    start = i;
                    maxLength = k;
                }
            }
        }
    }

    printf("Longest palindrome substring is: ");
    printSubStr( str, start, start + maxLength - 1 );

    return maxLength; // return length of LPS
}

// Driver program to test above functions
int main()
{
    char str[] = "forgeeksskeegfor";
    printf("\nLength is: %d\n", longestPalSubstr( str ) );
    return 0;
}
```

Output:

```
Longest palindrome substring is: geeksskeeg
Length is: 10
```

Time complexity: O (n^2)
Auxiliary Space: O (n^2)

We will soon be adding more optimized methods as separate posts.

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.

http://ideone.com/mK9mDy

• Reply • Share ›

Aditya Goel • 3 months ago

How is this a DP problem??

• Reply • Share ›

The_Geek • 3 months ago

This can be managed with a single table, like this.
http://ideone.com/tvLqbV

• Reply • Share ›

WP • 3 months ago

For checking the palindromes of length 2, the look should run from 0 to n-2

1 • Reply • Share ›

kevin • 5 months ago

in inner loop if(k>maxlength) is not required because k is always increasing

• Reply • Share ›

rohit • 5 months ago

this solution is highly memory intensive as it uses a table of nxn which might be huge for a large array..

• Reply • Share ›

Matt Clifford • 5 months ago

Search Manacher's algorithm for an O(n) solution, I have not seen it posted yet. Other O(n) algorithms exist as well. This algorithm computes the string left to right, and relies on the symmetric property of palindrome. For a given current palindrome, this property allows us to skip over certain parts of a palindrome if it has already been determined as a palindrome in the left half, and otherwise we continue to build outwards until we reach an index which does not grow the current palindrome.

• Reply • Share ›

Ankit Bhardwaj • 5 months ago

http://wcipeg.com/wiki/index.p...

We can find out longest palindrome in a string in O(n) time, but I am not getting that solution. Can someone help me out ?

• Reply • Share ›

Ankit Bhardwaj ➚ Ankit Bhardwaj • 5 months ago

and there is one more solution which takes O(nlogn) time using suffix array.

• Reply • Share ›

anand kiran • 5 months ago

http://ideone.com/RAOiKI

• Reply • Share ›

Saksham Gupta • 6 months ago

Greedy approach solution:

http://ideone.com/ZvMrPB

• Reply • Share ›

Shrikant Polawar • 6 months ago

Hi, for finding the largest palindromic sub-string we have to locate the center of the largest palindrome and then expand it. So if abcba is the string c is center, if abddba is the string first d is center. If we scan the string and check for every such center and expand the string as much as possible we can do it in place(auxilary space : O(1)) and in O(n^2) worst case time. Worst case will occur when all characters in the string are identical. Here is the link to solution http://ideone.com/Qloolu .

9 • Reply • Share ›

The_Geek ➦ Shrikant Polawar • 3 months ago

Logic is excellent, but when palindrome length is odd, then it does not print exact string, but can be easily corrected. No worries !

• Reply • Share ›

aa1992 • 7 months ago

here is the perfect java code for the longest palindromic string,

http://ideone.com/FeMOmC

• Reply • Share ›

jayandranath jaya • 7 months ago

hi the above code wont for all cases ex:hoforaofob
for the above string it prints oforaofo which is not palindrome correct ans should ofo
so this code is not correct..

• Reply • Share ›

Dominic ➦ jayandranath jaya • 5 months ago

It does return "ofo" as the answer

• Reply • Share ›

Abhi • 8 months ago

str1 = “abacdfgdcaba”, rev(str1) = “abacdgfdcaba”.
The longest common substring between str1 and rev(str1) is “abacd”. this is not a valid

palindrome.
the longest common substring method fails when there exists a reversed copy of a non-palindromic substring in some other part of S. To rectify this, each time we find a longest common substring candidate, we check if the substring's indices are the same as the reversed substring's original indices. If it is, then we attempt to update the longest palindrome found so far; if not, we skip this and find the next candidate.

• Reply • Share ›

xinwei he ➚ Abhi • 2 months ago

abacddcaba

• Reply • Share ›

Kim Jong-il • 8 months ago

I think we can solve this problem in o(n^2) time, without using any space.

What we have to do that for each and every character we have to check the possibility of even length palindrome and odd length palindrome. And we know that if a middle element is given then how we can check that if it is palindrome or not.

2 • Reply • Share ›

typing.. • 8 months ago

we can have space complexity O(n), by considering only two arrays of length n.

1 • Reply • Share ›

ALEX • 8 months ago

One of the best way to do....I don't know the complexity..... :P

can anybody tell me the complexity on this approach....

http://ideone.com/GhDvI2

• Reply • Share ›

Kim Jong-il ➚ ALEX • 8 months ago

O(n^2).

• Reply • Share ›

Amandeep Kamboj • 10 months ago

I think this is the better solution

public static String findPalindrome(String str) {

StringBuilder finalStr1 = new StringBuilder();

StringBuilder finalStr = new StringBuilder();

```
char[] a = str.toCharArray();

int start = 0;

int end = a.length - 1;

for (int i = start; i < end; i++) {

int _end = end;

boolean append = false;

int _start = i;
```

see more

• Reply • Share ›

vaibhav • 10 months ago

The normal approach would be of creating suffix array for the actual string and suffix array for reverse of string. Apply radix sort on both the arrays. the problem is reduced to find the longest prefix length. The worst case time complexity is O(n^2) whereas auxiliary space is of O(n). But the average case time complexity is O(nlgn). I

• Reply • Share ›

Karshit Jaiswal • 10 months ago

One more DP approach to Longest Common Substring.
Simple and easy to grab.
http://ideone.com/FbSbR5

2 • Reply • Share ›

Amitesh Sinha ➚ Karshit Jaiswal • 9 months ago

Hi, The approach mentioned in ideone.com has to first reverse the string. We can also do this without reversing :

```
public class LargestPalindrome {

/**

* @param args

*/

static String astring = "abcdadcbaaaaa";

static int ROW = astring.length();

static int Answer, index;

public static void main(String[] args) {
```

public static void main(String[] args) {

int[][] matrix = new int[ROW][ROW];

see more

• Reply • Share ›

Unique Karshit Jaiswal • 9 months ago

I think this soln won't work for "abcdba"
answer should b 0 bt ur method gives 2!!

• Reply • Share ›

Karshit Jaiswal Unique • 9 months ago

Every character is a palindrome in itself buddy.
The answer can never be zero until an unless the string is empty.

1 • Reply • Share ›

Sanchita Tiwari Karshit Jaiswal • 9 months ago

u r method is Wrong...see dis link:
http://leetcode.com/2011/11/lo...

• Reply • Share ›

Karshit Jaiswal Sanchita Tiwari • 9 months ago

My mistake... Yes the method wont work if you have the reversed string as a part of the original string..!!
Thanks.. :)

• Reply • Share ›

saiyam agarwal Karshit Jaiswal • 10 months ago

Plz elaborate the logic.

• Reply • Share ›

Karshit Jaiswal saiyam agarwal • 10 months ago

Logic : Reverse the given string and apply Longest Common Substring Algorithm.
P.S : Longest Common Substring and not Longest Common Subsequence.

2 • Reply • Share ›

Red Karshit Jaiswal • 10 months ago

Awesome post, buddy. Much, much better than the explanation above.

• Reply • Share ›

Karshit Jaiswal Red • 9 months ago

Thanks.. :)

• Reply • Share ›

Sanchita Tiwari ➦ Karshit Jaiswal • 9 months ago

we just do reverse and apply LCSubstring..for more details see:

http://leetcode.com/2011/11/lo...

• Reply • Share ›

Lohith Ravi • a year ago

This is also DP is it not ? please correct me if im wrong

main()

{

for(int i=0;i<n;i++) {="" for(int="" j="i+1;j<n;j++)" {="" if(j-i+1="">> max){

if(isPalindrome(i,j,array))

{

max = j-i+1;

resultLowIndex=i;

resultHighIndex=j

}

}

see more

• Reply • Share ›

sukisukimo • a year ago

/*

* Program to find the longest palindrome
* The algo time complexity is O(n^2) and space complexity is O(1)

*/

static String getBigPalindrome(String str)

{

int i = 0, j = 0, tempi = 0;

int maxLen = 0, currentLen = 0;

```
String maxpalin = null;

String currentpalin = null;

char[] carr = str.toCharArray();
```

see more

• Reply • Share ›

neelabhsingh • 2 years ago

Very Important CHECK this condition

```
for( int k = 3; k <= n; ++k )
{
// Fix the starting index
for( int i = 0; i < n - k + 1 ; ++i )
{
// Get the ending index of substring from starting index i and length k
int j = i + k - 1;
// checking for sub-string from ith index to jth index iff str[i+1]
// to str[j-1] is a palindrome
if( table[i+1][j-1] && str[i] == str[j] )
{
table[i][j] = true;
if( k > maxLength )
{
start = i;
maxLength = k;
```

see more

• Reply • Share ›

baba neelabhsingh • a year ago

table[i][i] have all been set to true.

• Reply • Share ›

Sumit Monga • 2 years ago

To find whether a substring starting at index 'i' and ending at index 'j' is a palindrome or not, we need information regarding 'i+1' and 'j-1'. So the solution to the problem is building the table starting from the end of the string and not from the first index in the string .The below code finds the length as well as prints the longest palindromic substring in the given string:

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

```
#include<string.h>

int lps(char * str)

{

int n = strlen(str);

int i,j,start_ind,end_ind;

int max = 0;

bool max_pal[n][n];
```

see more

• Reply • Share ›

Divya • 2 years ago

Is the following code correct? Can somebody please authenticate?

```
int main()

{

char *input = "abforgeeksskeegforba";

char outputarr[100] = {NULL};

//printf("strlen is %d\n", strlen(input));

int i = 0;

int j = strlen(input) - 1;

int k = 0;

bool isPal = false;

while (i <= j)
```

see more

1 • Reply • Share ›

jayasurya j ➚ Divya • a year ago

i guess its wrong

• Reply • Share ›

Sanjith Sakthivel • 2 years ago

Follow the manacher&#039s Algorithm that&#039s simple and takes less space and time

complexity.

• Reply • Share ›

prity • 2 years ago

It seems O(n) solution is available. Refer this http://codeinterview.blogspot....

• Reply • Share ›

Shweta prity • 2 years ago

The solution given at the link is not O(n).

• Reply • Share ›

Purushotham • 2 years ago

I agree the above solution is using DP. However this problem has a better solution with O(n) time & O(n) space. Below is the code using Greedy approach.

```
public static int Lps(String s){
char[] c = s.toCharArray();
int[] lps = new int[s.length()];
int tmp, max_len = 1;
lps[0] = 1;
for(int i = 1; i 0 && c[i] == c[tmp])
lps[i] = lps[i-1] + 2;
else if(c[i] == c[i-1])
lps[i] = 2;
else
lps[i] = 1;

if(lps[i] > max_len) max_len = lps[i];
}

return max_len;
}
```

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

• Reply • Share ›

Load more comments

- 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

Recent Comments

- lt_k

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

  Java Programming Language · 1 hour ago

- Piyush

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

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

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

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour 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