Dynamic Programming | Set 12 (Longest Palindromic Subsequence)

Given a sequence, find the length of the longest palindromic subsequence in it. For example, if the given sequence is “BBABCBCAB”, then the output should be 7 as “BABCBAB” is the longest palindromic subseuqnce in it. “BBBBB” and “BBCBB” are also palindromic subsequences of the given sequence, but not the longest ones.

The naive solution for this problem is to generate all subsequences of the given sequence and find the longest palindromic subsequence. This solution is exponential in term of time complexity. Let us see how this problem possesses both important properties of a Dynamic Programming (DP) Problem and can efficiently solved using Dynamic Programming.

1) Optimal Substructure:
Let X[0..n-1] be the input sequence of length n and L(0, n-1) be the length of the longest palindromic subsequence of X[0..n-1].

If last and first characters of X are same, then L(0, n-1) = L(1, n-2) + 2.
Else L(0, n-1) = MAX (L(1, n-1), L(0, n-2)).

Following is a general recursive solution with all cases handled.

```
// Everay single character is a palindrom of length 1
L(i, i) = 1 for all indexes i in given sequence

// IF first and last characters are not same
If (X[i] != X[j])  L(i, j) =  max{L(i + 1, j),L(i, j - 1)}

// If there are only 2 characters and both are same
Else if (j == i + 1) L(i, j) = 2

// If there are more than two characters, and first and last
// characters are same
Else L(i, j) =  L(i + 1, j - 1) + 2
```

2) Overlapping Subproblems
Following is simple recursive implementation of the LPS problem. The implementation simply follows the recursive structure mentioned above.

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

// A utility function to get max of two integers
int max (int x, int y) { return (x > y)? x : y; }

// Returns the length of the longest palindromic subsequence in seq
int lps(char *seq, int i, int j)
{
   // Base Case 1: If there is only 1 character
   if (i == j)
     return 1;

   // Base Case 2: If there are only 2 characters and both are same
   if (seq[i] == seq[j] && i + 1 == j)
     return 2;

   // If the first and last characters match
   if (seq[i] == seq[j])
      return lps (seq, i+1, j-1) + 2;

   // If the first and last characters do not match
   return max( lps(seq, i, j-1), lps(seq, i+1, j) );
}

/* Driver program to test above functions */
int main()
{
    char seq[] = "GEEKSFORGEEKS";
    int n = strlen(seq);
    printf ("The lnegth of the LPS is %d", lps(seq, 0, n-1));
    getchar();
    return 0;
}
```

Output:

```
The lnegth of the LPS is 5
```

Considering the above implementation, following is a partial recursion tree for a sequence of length 6 with all different characters.

```
               L(0, 5)
             /        \ 
            /          \  
        L(1,5)          L(0,4)
       /    \            /    \
      /      \          /      \
  L(2,5)    L(1,4)  L(1,4)  L(0,3)
```

In the above partial recursion tree, L(1, 4) is being solved twice. If we draw the complete recursion tree, then we can see that there are many subproblems which are solved again and again. Since same suproblems are called again, this problem has Overlapping Subprolems property. So LPS problem has both properties (see this and this) of a dynamic programming problem. Like other typical Dynamic Programming(DP) problems, recomputations of same subproblems can be avoided by constructing a temporary array L[][] in bottom up manner.

Dynamic Programming Solution

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

// A utility function to get max of two integers
int max (int x, int y) { return (x > y)? x : y; }

// Returns the length of the longest palindromic subsequence in seq
int lps(char *str)
{
   int n = strlen(str);
   int i, j, cl;
   int L[n][n];  // Create a table to store results of subproblems


   // Strings of length 1 are palindrome of lentgh 1
   for (i = 0; i < n; i++)
      L[i][i] = 1;

    // Build the table. Note that the lower diagonal values of table are
    // useless and not filled in the process. The values are filled in a
    // manner similar to Matrix Chain Multiplication DP solution (See
    // http://www.geeksforgeeks.org/archives/15553). cl is length of
    // substring
    for (cl=2; cl<=n; cl++)
    {
        for (i=0; i<n-cl+1; i++)
        {
            j = i+cl-1;
            if (str[i] == str[j] && cl == 2)
               L[i][j] = 2;
            else if (str[i] == str[j])
               L[i][j] = L[i+1][j-1] + 2;
            else
               L[i][j] = max(L[i][j-1], L[i+1][j]);
        }
    }

    return L[0][n-1];
}

/* Driver program to test above functions */
int main()
{
    char seq[] = "GEEKS FOR GEEKS";
    int n = strlen(seq);
    printf ("The lnegth of the LPS is %d", lps(seq));
    getchar();
    return 0;
}
```

Output:

```
The lnegth of the LPS is 7
```

Time Complexity of the above implementation is O(n^2) which is much better than the worst case time complexity of Naive Recursive implementation.

This problem is close to the Longest Common Subsequence (LCS) problem. In fact, we can use LCS as a subroutine to solve this problem. Following is the two step solution that uses LCS.
1) Reverse the given sequence and store the reverse in another array say rev[0..n-1]
2) LCS of the given sequence and rev[] will be the longest palindromic sequence.
This solution is also a O(n^2) solution.

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

References:
http://users.eecs.northwestern.edu/~dda902/336/hw6-sol.pdf

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings

- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Dynamic Programming

Tweet 1 g+1 0

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

81 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

lucy • 4 days ago

Can we calculate this line out of loop because it require calculate in each iteration..........

```
if (str[i] == str[j] && cl == 2)
L[i][j] = 2;
```

• Reply • Share ›

Ashish Maheshwari • 8 days ago

For naive recursive implementation code, base case 2 is incomplete. it should be:

```
if (i + 1 == j) {
if (seq[i] == seq[j]) return 2;
else return 0;
}
```

• Reply • Share ›

binlong → Ashish Maheshwari • 4 days ago

your else should be 1? for example 12 is not a palindrome, but 1 and 2 both are.

• Reply • Share ›

Ashish Maheshwari → binlong • 4 days ago

oops.. you are right.. i didnt think of this.. thanks :)

• Reply • Share ›

thevagabond85 • a month ago

For those who hate that special treatment given to sub-string of length 2

http://ideone.com/inh1gx

// Memoization

```
int lpsDP(char* str, int n)
{

char LPS[n+1][n+1];
int i,j;
int width; //width of current substring

// first row 0
for(i=0; i<=n; i++)
LPS[0][i] = 0;

// first col 0
for(i=0; i<=n; i++)
```

see more

Reply • Share ›

thevagabond85 • a month ago

FOR those confused with second base case : i+1 == j and want to use as in quick sort etc. here's my code :(LOGIC : no need write w[start] == w[end] twice in the code.)

```
int lpsRecur(char w[], int start, int end)

{

if(start==end)
return 1;

if(start>end)
return 0;

if(w[start] == w[end])
return 2 + lpsRecur(w, start+1, end-1);

// else
return max( lpsRecur(w, start+1, end),
lpsRecur(w, start, end-1
);

}
```

Reply • Share ›

prashant jha • 2 months ago

method-1 http://ideone.com/LNtdWz

method 2 -LCS(s1,rev(s1))

• Reply • Share ›

Mission Peace • 2 months ago

Here is my video on this question https://www.youtube.com/watch?...

• Reply • Share ›

Cracker • 2 months ago

http://algods-cracker.blogspot...

• Reply • Share ›

Nobody • 3 months ago

If we use Memoization instead of DP,how would its running time be,as compared to the DP solution?Because I find it really hard to code the DP solution.Please help.

• Reply • Share ›

Aditya Goel Nobody • 3 months ago

You can use Memoization. Time and space complexity remains the same for Tabulation & Memoization in most problems. Only difference being Memoization is Top-Bottom(using recursion) and Tabulation is Bottom-up(using iteration).

• Reply • Share ›

alkaphalswal • 3 months ago

how to solve it if length of string is of order 10^6.
we cant take L[n][n] it that case

• Reply • Share ›

kal • 4 months ago

I think the following relation for the optimal substructure is wrong:

L(0, n-1) = L(1, n-2) + 2

It gives wrong answers when first and last items are similar but the rest of the string inside of it is not. For example , for string "ZCABAZ" it gives answer of 5, but in fact it should be 3.

The second solution that you suggest of using LCS algorithm will work correctly.

• Reply • Share ›

kal kal • 4 months ago

Looks like I misunderstood the definition of "Palindrome Subsequence". It may not be contiguous.

1 • Reply • Share ›

coderABC • 6 months ago

Counter example to DP solution "BBABCB"

The current solution returns "5" as the palindrome length.

Corrected solution: Note the extra check on line 8.
1. for (cl=2; cl<=n; cl++)
2. {
3. for (i=0; i<n-cl+1; i++)="" 4.="" {="" 5.="" j="i+cl-1;" 6.="" if="" (str[i]="=" str[j]="" &&="" cl="=" 2)="" 7.="" l[i][j]="2;" 8.="" else="" if="" (str[i]="=" str[j]="" &&="" (cl="=" l[i+1][j-1]="" +="" 2=""))="" 9.="" l[i][j]="L[i+1][j-1]" +="" 2;="" 10.="" else="" 11.="" l[i][j]="max(L[i][j-1]," l[i+1][j]);="" 12.="" }="" 13.}="">

^ | v • Reply • Share ›

Aditya ➦ coderABC • 6 months ago

For " BBABCB". The largest palindrome is "BBABB" whose length is 5.

1 ^ | v • Reply • Share ›

coderABC ➦ coderABC • 6 months ago

8. else if (str[i] == str[j] && (cl == L[i+1][j-1] + 2))

^ | v • Reply • Share ›

Guest • 7 months ago

The second solution given doesn't work. Counterexample: EXAMPLEXYELPMAXE
The approach as given will find a "palindrome" of length 7.

^ | v • Reply • Share ›

Guest • 7 months ago

@GeeksforGeeks ,
Is this problem can also solved like given string is str1="BBABCBCAB", and reverse of the str1 is str2="BCABCBABB". Now find the the greatest common subsequence like in the following post of geeksforgeeks?

http://www.geeksforgeeks.org/d...

^ | v • Reply • Share ›

lucy ➦ Guest • 3 months ago

Look question it may not continues

^ | v • Reply • Share ›

anon ➦ Guest • 6 months ago

Yeah it can be done that way.

^ | v • Reply • Share ›

adam ➦ anon • 5 months ago

It is true that the length of the longest palindromic subsequence equals the length of LCS(x,reverse(x)), but it is *not* the case that every longest common subsequence between x and rev(x) is a palindrome. E.g., in x=ABCAB, the subsequence ACB is common to x and rev(x).

2 • Reply • Share ›

darkside • 8 months ago

We can optimize the code by removing the following code
if (str[i] == str[j] && cl == 2)
L[i][j] = 2;
the algo is still the same
values of l[][] for the string "geeks"
1 1 2 2 2
0 1 2 2 2
0 0 1 1 1
0 0 0 1 1
0 0 0 0 1

• Reply • Share ›

Shahil • 9 months ago

The recursive solution to the problem is wrong.The answer returned to the string "XHAX" is 3. But the actual answer is 0.

• Reply • Share ›

Shahil Shahil • 9 months ago

sorry actual answer should be 1.

• Reply • Share ›

Vishesh Shahil • 8 months ago

the answer is right it should be 3 only as u see first and last X would be picked and then we have a choice of H or A to pick. This is possible as we are talking of the palindromic subsequence.

• Reply • Share ›

Vãîbhåv Joshî • 10 months ago

DP code in java
http://ideone.com/qzxrWs

recursive Code in java
http://ideone.com/xD4H7s

• Reply • Share ›

Red • 10 months ago

Using a condition //

if(i>j)

return 0;

//

would make it possible to eliminate the 2 character condition in the recursive program, right?

• Reply • Share ›

Karshit Jaiswal • a year ago

Using LCS :
http://ideone.com/JQe9aJ

3 • Reply • Share ›

ANA Karshit Jaiswal • 10 months ago

can you explain your approach ?

• Reply • Share ›

Karshit Jaiswal ANA • 10 months ago

Reverse the string and apply LCS.

• Reply • Share ›

Goku Karshit Jaiswal • 10 months ago

fails for case CBACB
LCS for string and it's reverse is CAB but CAB is not a palindrome.

• Reply • Share ›

Karshit Jaiswal Goku • 10 months ago

NO !!! It does not fails.
CAC is also one of the LCS (which is a palindrome)

• Reply • Share ›

Vivek Chicharito Zakwalia • a year ago

In the problem statement how LPS of BBABCBCAB is BABCBAB... it should be BABCCBAB and thus ur code should return '8'?

• Reply • Share ›

Archit Vivek Chicharito Zakwalia • a year ago

i think you should not consider all the permutation of string.
Check palindrom in string from index 0 to len-1 without changing the order of characters.

• Reply • Share ›

Vivek Chicharito Zakwalia Archit • a year ago if order isnt changing then how it will generate BABCBAB ?

• Reply • Share ›

Archit Vivek Chicharito Zakwalia • a year ago

string is BBABCBCAB you should take char at index 0,2,3,4,5,7,8 ,,,and you are taking 0,2,3,4,(6,5),7,8
,,order of characters is changed.
5 is coming after 6.

• Reply • Share ›

Vivek Chicharito Zakwalia Archit • a year ago

esa kya... o.O (Y)

• Reply • Share ›

renu • a year ago

the brute force approach has a solution of o(n^3) not exponential cxity..right?

• Reply • Share ›

prashant jha • a year ago

my c++ code in exponential time 0(2^n)
reduce it into polynomial time by 2d array and memorisation which is quite easy.....see for recursive part
http://ideone.com/Z4qbMP

• Reply • Share ›

prashant • a year ago

my c++ code but naive recursive approach in 0(2^n) complexity
get into polynomial time by using 2-d array and memorisation
http://ideone.com/Z4qbMP

• Reply • Share ›

Guest • a year ago

At last in this post u have mentioned that we can use LCS as a subroutine to solve this problem,which is not correct everytime..let me explain with example

For example
S=original string
S'=reverse of orginal string

S = “caba”, S’ = “abac”.
The longest common substring between S and S’ is “aba”, which is the answer.but now Let’s try another example:
S = “abacdfgdcaba”, S’ = “abacdgfdcaba”.
The longest common substring between S and S’ is “abacd”. Clearly, this is not a valid

palindrome.

3 • Reply • Share ›

loconet Guest • a year ago

> The longest common substring between S and S' is "abacd"

The problem here is that you are using "longest common substring". The solution by the poster is using LCS, which is Longest Common *Subsequence*. That is completely different.

A longest common subsequence of "abacdfgdcaba" and "abacdgfdcaba" is [a, b, a, c, d, f, d, c, a, b, a] ..which as you can see is a palindrome.

3 • Reply • Share ›

pinto • a year ago

This code in the article is incorrect and misleading. Please either fix the code or remove it the page.

correct code is something like:

public class Palindrome {

public static boolean find(int[][] memo, char[] in, int i, int j) {
if (i == j) {
memo[i][j] = 1;
return true;
}

if (in[i] == in[j]) {
if (find(memo, in, i+1, j-1)) {
memo[i][j] = memo[i+1][j-1] + 2;
return true;
} else {
memo[i][j] = 0;

see more

4 • Reply • Share ›

PAN SINGH • a year ago

```
#include <iostream>
#include <string>
#include <string.h>
using namespace std;
```

```
string S = "GEEKSFORGEEKS";

int dp[100][100];

int maxPalindrome(int i, int j)
{
    int length;

    if(dp[i][j] != -1)
        return dp[i][j];

    if(i == j)
```

see more

3 • Reply • Share ›

Guest • 2 years ago

How to print the longest string ?

1 • Reply • Share ›

Kaidul Islam Sazal • 2 years ago

How to print the longest palindromic string?

1 • Reply • Share ›

Kaidul Islam Sazal Kaidul Islam Sazal • 2 years ago

Ignore it! I am able to do it :)

• Reply • Share ›

rohit • 2 years ago

for (cl=2; cl<=n; cl++)
{
for (i=0; i<n-cl+1; i++)="" {="" j="i+cl-1;">

• Reply • Share ›

rohit • 2 years ago

for (cl=2; cl<=n; cl++)
{
for (i=0; i<n-cl+1; i++)="" {="" j="i+cl-1;" any="" one="" help="" me="" what="" is="" this="" part="" of="" code="" doing="">

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site Privacy

-
-
-
-
    - 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 · 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
-