Dynamic Programming | Set 17 (Palindrome Partitioning)

Given a string, a partitioning of the string is a *palindrome partitioning* if every substring of the partition is a palindrome. For example, “aba|b|bbabb|a|b|aba” is a palindrome partitioning of “ababbbabbababa”. Determine the fewest cuts needed for palindrome partitioning of a given string. For example, minimum 3 cuts are needed for “ababbbabbababa”. The three cuts are “a|babbbab|b|ababa”. If a string is palindrome, then minimum 0 cuts are needed. If a string of length n containing all different characters, then minimum n-1 cuts are needed.

Solution
This problem is a variation of Matrix Chain Multiplication problem. If the string is palindrome, then we simply return 0. Else, like the Matrix Chain Multiplication problem, we try making cuts at all possible places, recursively calculate the cost for each cut and return the minimum value.

Let the given string be str and minPalPartion() be the function that returns the fewest cuts needed for palindrome partitioning. following is the optimal substructure property.

```
// i is the starting index and j is the ending index. i must be passed as 0 and j as n-1
minPalPartion(str, i, j) = 0 if i == j. // When string is of length 1.
minPalPartion(str, i, j) = 0 if str[i..j] is palindrome.

// If none of the above conditions is true, then minPalPartion(str, i, j) can be
// calculated recursively using the following formula.
minPalPartion(str, i, j) = Min { minPalPartion(str, i, k) + 1 +
                                 minPalPartion(str, k+1, j) }
                           where k varies from i to j-1
```

Google™ Custom Search

GeeksforGeeks
53,524 people like GeeksforGeeks.

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

Following is Dynamic Programming solution. It stores the solutions to subproblems in two arrays P[][] and C[][], and reuses the calculated values.

```
// Dynamic Programming Solution for Palindrome Partitioning Problem
#include <stdio.h>
#include <string.h>
#include <limits.h>

// A utility function to get minimum of two integers
int min (int a, int b) { return (a < b)? a : b; }

// Returns the minimum number of cuts needed to partition a string
// such that every part is a palindrome
int minPalPartion(char *str)
{
    // Get the length of the string
    int n = strlen(str);

    /* Create two arrays to build the solution in bottom up manner
       C[i][j] = Minimum number of cuts needed for palindrome partitio
                 of substring str[i..j]
       P[i][j] = true if substring str[i..j] is palindrome, else false
       Note that C[i][j] is 0 if P[i][j] is true */
    int C[n][n];
    bool P[n][n];

    int i, j, k, L; // different looping variables

    // Every substring of length 1 is a palindrome
    for (i=0; i<n; i++)
    {
        P[i][i] = true;
        C[i][i] = 0;
    }

    /* L is substring length. Build the solution in bottom up manner b
       considering all substrings of length starting from 2 to n.
       The loop structure is same as Matrx Chain Multiplication proble
       See http://www.geeksforgeeks.org/archives/15553 )*/
    for (L=2; L<=n; L++)
    {
        // For substring of length L, set different possible starting
        for (i=0; i<n-L+1; i++)
        {
            j = i+L-1; // Set ending index

            // If L is 2, then we just need to compare two characters
```

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

```
            // If L is 2, then we just need to compare two characters.
            // need to check two corner characters and value of P[i+1]
            if (L == 2)
                P[i][j] = (str[i] == str[j]);
            else
                P[i][j] = (str[i] == str[j]) && P[i+1][j-1];

            // IF str[i..j] is palindrome, then C[i][j] is 0
            if (P[i][j] == true)
                C[i][j] = 0;
            else
            {
                // Make a cut at every possible localtion starting fro
                // and get the minimum cost cut.
                C[i][j] = INT_MAX;
                for (k=i; k<=j-1; k++)
                    C[i][j] = min (C[i][j], C[i][k] + C[k+1][j]+1);
            }
        }
    }

    // Return the min cut value for complete string. i.e., str[0..n-1]
    return C[0][n-1];
}

// Driver program to test above function
int main()
{
   char str[] = "ababbbabbababa";
   printf("Min cuts needed for Palindrome Partitioning is %d",
           minPalPartion(str));
   return 0;
}
```

Output:

```
Min cuts needed for Palindrome Partitioning is 3
```

Time Complexity: O(n^3)

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

705 Subscribe

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 38 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 42 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

Related Tpoics:

- Print all possible words from phone digits
- 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

3 Tweet 0 2

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

30 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Ankit Chaudhary · 7 days ago

I think it can be done in O(n^2).

```
bool palin[1000][1000];
void preprocess(char *str,int n){ // O(n^2)
for(int i=0;i<n;i++) palin[i][i]="true;" for(int="" l="2;L&lt;n;L++){" for(int="" i="0;i+l
palin[i][j]="(str[i]==str[j])" &&="" palin[i+1][j-1];="" }="" }="" bool="" ispalin(int=""
partition(char="" *str,int="" low,int="" high){="" if(low="">=high || isPalin(low,hig
if(dp[low][high]!=-1) return dp[low][high];
for(int i=low;i<=high;i++){
dp[low][high]=min(dp[low][high], partition(str,low,i) + partition(str,i+1,high)+1);
}
return dp[low][high];
}

int main(){
char str[1000];
scanf("%s",str);
int ans=partition(str);

}
```

NB · 24 days ago

What about this approach :

a) Find the longest common substring between the string and its reverse. This
present in the string.

b) Remove this palindrome from the original string and repeat the process

c) base case is when string size =1 or when the entire string being looked at is

d) The the number of palindrome strings is x, then x -1 cuts are needed.

Eg :

String : abbacdcef Longest Pal : abba
String : cdcef Longest Pal : cdc
String : ef Longest Pal : e
String : f Longest Pal : f

No. of cuts = No of 'Longest. pal' -1 = 3

Susnata Roy · 2 months ago

have a look at the dp solution below... O(n^2) .. similar to idea of word break pr

http://ideone.com/sYkYFo

Word break problem:

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

darshini · 5 months ago

how can we get the value of p[i+1][j-1] as we are currently calculating p[i][j].

s poonia · 5 months ago

this wont work for -ive numbers . try the following code. it following code. it will corner cases like 0 and -1 :

int numofone(int n){

int count=0;

```
while(n){

n=n&(n-1);

count++;

}

return count;

}

int mymethod(int x){
```

see more

1

s poonia s poonia • 5 months ago

wrong que... this is ans for http://www.geeksforgeeks.org/n...

Sumit Monga • 7 months ago

a solution in O(n^2) where we first check for a substring starting at i and endin
using this table we calculate the minimum no. of continuous palindromes upto
a 1D array is used so space complexity is also better .Here is the code:

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

int no_of_pal_parts(char * str,int n)
{

int i,j,k,c;
```

bool is_pal[n][n];
memset(is_pal,0,sizeof(is_pal));
for(i = n-1; i >= 0; i--)
{
for(j = i; j < n; j++)
{
if(i== j)

see more

4

trying · 8 months ago

a | babbbab | babab | a

I think this is the minimum cuts.

trying trying · 8 months ago

sorry i did not see the whole stuff.

equation · 10 months ago

i think there is any bug in the logic...
because for

```
char str[] = "ababbabbababa"
```

the output should be 1 as ababbabbaba|ba
but the code is giving 2.

@geeksforgeeks please correct me if m wrong.....

1

GeeksforGeeks ➦ equation • 10 months ago

The output 2 seems to be correct. Please note that ba is not a palindrc ababbabbaba**aa**, we get 1 because aa is a palindrome.

Pankaj Goyal • 10 months ago

can smone tell me what is the time complexity of recursive method? O(n*(2^n

abhishek08aug • a year ago

Intelligent :D

zyfo2 • a year ago

can be done in O(n^2) using dp. check leetcode for details
so basically, you don't need variable starting point. just start from 0 instead of

1

Amit • a year ago

```
 #include"stdio.h"
#include "malloc.h"
int min(int a, int b,int c)
{
        if(a<b)
        {
                return a<c?a:c;
        }
        else
                return b<c?b:c;
```

```
}
int isPalindrome(char str[],int start,int end)
{
        while(start<end)
        {
                if(str[start] != str[end])
                        return 0;
                start++;
```

see more

Vishal Johri • a year ago

I thought of an alternate algo using Divide and Conquer..

Break string into 2 parts and merge them into 1 if after merging they are palinc

int merge(int lb1,int ub1,int lb2,int ub2) : merges only if after merging they are p
=0..

void part(int lb,int ub) //send intially lb=0 ub=n-1...
{
static int merge_count=0;
mid=(lb+ub)/2;
part(lb,mid);
part(mid+1,ub);
merge_count=merge_count+merge(lb,mid,mid+1,ub);

}
merge_count counts number of mergings done..
Min number of cuts = merge_count (Think it carefully!)....

Vishal Johri ➦ Vishal Johri • a year ago

Sorry some changes...

1. Insert a special character '*' in between each character initially to signify cuts...
At merging,remove this cut...else this cut(*) remains..

2. At the end,count the number of '*' cuts ==> x
Then, min number of cuts = x;

^ | ⌄ •

sush • a year ago

This implementation uses just one array for storing

```
 int minCuts(char *str)
{
        int n=strlen(str),i,j,l;
    int t[n][n];
        memset(t,-1,n*n*sizeof(int));
        for(i=0;i<n-1;++i)
        {
                t[i][i]=0;
                if(str[i]==str[i+1])
                t[i][i+1]=0;
                else
                t[i][i+1]=1;
        }
        t[i][i]=0;
        for(l=3;l<=n;++l)
        {
```

see more

^ | ⌄ •

HLS • 2 years ago

```
 void pc(char a[],int n)
{
int m[n+1];
for(int i=0;i<=n;i++)
m[i]=i;
m[0]=0;
for(int i=0;i<=n;i++)
{
for(int j=i-1;j>=0;j--)
{
if(palindrom(a,j,i))
{if(j-1>=0) m[i]=m[j-1]+1; else m[i]=m[j];}

}

m[i]=(m[i]<m[i-1]+1)?(m[i]):(m[i-1]+1);
}

cout<<m[n];
}
```

Abhishek Gayakwad • 2 years ago

This is a normal recursive program which works perfectly, why we need to sol
problem ?

[sourcecode language="JAVA"]
private static int minPalPartition(char[] s, int i, int j) {
if (i == j) return 0;

```
if (isPalindrome(s, i, j)) return 0;
int min = Integer.MAX_VALUE;
for (int x = i; x < j; x++) {
min = Math.min(min, minPalPartition(s, i, x) + 1 + minPalPartition(s, x + 1, j));
}
return min;
}
```

trying Abhishek Gayakwad • 8 months ago

if somebody does not know or understand matrix chain multiplication a
obvious.

Amit Abhishek Gayakwad • a year ago

You are not storing solutions to sub-problems anywhere, if you draw th
repeated computations.

lohith • 2 years ago

```
import java.util.HashMap;


public class MinimumDivisionsForPalindrome {

        public static void main(String str[]){
                String st0 = "cba";
                String st1 = "ababbbabbababa";
                String st2 = "ababbcgdgcbbaba";
                String st3 = "cgdgcabababbbabb";
                //int p = partition(st1);
                System.out.println(FindMinimumSubPalindromes(0, st0.l
                System.out.println(FindMinimumSubPalindromes(0, st1.l
                System.out.println(FindMinimumSubPalindromes(0, st2.l
                System.out.println(FindMinimumSubPalindromes(0, st3.l
```

see more

GeeksforGeeks • 2 years ago

Following is Naive recursive solution for the same problem

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

// A utility function to get minimum of two integers
int min (int a, int b) { return (a < b)? a : b; }

// A utility function to check whether str[i..j] is
// palindrome or not
bool isPal (char *str, int i, int j)
{
   if (i == j)
     return true;

   if (str[i] == str[j] && isPal(str, i+1, j-1))
```

see more

Neevan ➦ GeeksforGeeks • a year ago

@GeeksforGeeks : Can u please remove keyword 'dynamic programn
from question...because I want to guess/try my best before you see ur

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

Neevan ➦ Neevan • a year ago

I am sorry to wrong keyword above.... before I see your solutior

rock_hammer ➦ GeeksforGeeks • 2 years ago

```
[sourcecode language="c++"]
#include<iostream>
using namespace std;
bool palindrom(char a[],int j,int i);
void pc(char a[],int n);
int main()
{
char a[100];
cin>>a;

pc(a,strlen(a)-1);

system("pause");
}

void pc(char a[],int n)
{
int m[n+1];
```

for(int i=0;i<=n;i++)

see more

rock_hammer rock_hammer • 2 years ago

m(i): min no of cuts for string a(0,i)

recursive eqn for dp::

m(j)= min(
m(j-i-1)+1: if a(i,j) is palindrom
where 0<=i<j ,
m(j-1)+1

)

rock_hammer rock_hammer • 2 years ago

this is n^2 soln..
for any prob please comment :)

Avik rock_hammer • a year ago

@rock hammer.......your soln takes O(n^3) as well sinc
and inside dat u call palindrome(..) which takes O(n),so
O(n^3)..correct me if I m wrong