

The best programs are written so that computing machines can perform them quickly and so that human beings can understand them clearly. A programmer is ideally an essayist who works with traditional aesthetic and literary forms as well as mathematical concepts, to communicate the way that an algorithm works and to convince a reader that the results will be correct. — Donald E. Knuth

Newsletter Subscription

Subscription closed

Algorithms

Introduction

Arrays and Strings

Linked List

Stacks and Queues

Trees

Dynamic Programming

Bitwise Tricks

Backtracking

Sorting

Miscellaneous

Dynamic Programming

Problem :-

Splitting a String into minimum number of **palindromes**.

Given a string, find the minimum no. of palindromes into which the string can be broken.

Consider the string "**civic**" (String is Palindrome by itself, so min no. of palindromes = 1)

The string "**abcde**" cannot be splitted into less than **5** palindromes (each character is a palindrome)

Consider the string "**civicacaramanamaracacammacdeified**".

It can be broken into 4 palindromes "**civic**" + "**acaramanamaraca**" + "**cammac**" + "**deified**"

Solution :-

Suppose the size of the string is **n**.

We maintain a 2D array **palins [n][n]** where **palins [i][j]** stores the minimum number of palindromes in the string **str [i ... j]** i.e from index 'i' to index 'j' of the string. So, the final solution of breaking the string of size **n** into minimum no. of palindromes is given by **palins [0][n - 1]**.

Let us determine the structure of the optimal solution.

Let **palins (i , j)** denotes the min no. of palindromes in the string **str (i ... j)**.

We will try to break the string at all possible positions **k** such that **i <= k < j** and sum up the minimum no. of palindromes in **palins (i , k)** and **palins (k , j)**.

Thus, the structure of optimal solution is determined recursively as :

palins (i , j) = palins (i , k) + palins (k , j) for all **i <= k < j**

Finally, **palins (0 , n - 1)** gives the required solution. See the implementation below.

```

1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4
5  // check if a string is palindrome, given the start and end index of the str
6  bool isPalindrome(char str[], int i, int j) {
7      while (i < j) {
8          if (str[i++] != str[j--])
9              return false;
10     }
11     return true;
12 }
13
14 /* Find the minimum no. of palindromes in the input string i.e the
15    string has to be broken down into min. no. of palindromes
16    Suppose palins[i][j] -> min no. of palindromes in the string str[i...j]
17    Then, palins[0][n-1] is the final solution
18    Standard Recursive Solution :-
19    palins(i,j) = min(palins(i,k) + palins(k,j)) for i <= k < j
20 */
21 int minPalindromes(char str[]) {
22     int i, j, k;
23     int substr_size; //substring size
24     int n = strlen(str);
25     // Matrix where cell (i,j) stores the minimum no. of palindromes in str[i...j]
26     int palins[n][n];
27
28     // iterating over all possible substring sizes
29     for (substr_size = 1; substr_size <= n; substr_size++) {
30         // 'i' and 'j' tracks the start and end index of the current substring
31         for (i = 0; i <= n - substr_size; i++) {
32             j = i + substr_size - 1;
33             if (isPalindrome(str, i, j)) {
34                 // substring is a palindrome
35                 palins[i][j] = 1;
36             }
37             else {
38                 // search for palindromes within the substring
39                 // worst case (min. no of palindromes = length of the substring)
40                 // each character is a palindrome

```

```

41         palins[i][j] = substr_size;
42         // find min no. of palindromes within substring
43         for (k = i; k < j; k++) {
44             // try splitting substring at each position
45             int sum = palins[i][k] + palins[k+1][j];
46             if (sum < palins[i][j])
47                 palins[i][j] = sum;
48         }
49     }
50 }
51 }
52 // return min no. of palindromes in the input string
53 return palins[0][n-1];
54 }
55
56 // main
57 int main() {
58     char str[] = "civicacaramanamaracacammadeified";
59     //civic + acaramanamaraca + cammac + deified (4 palindromes)
60     int min_palin = minPalindromes(str);
61     cout<<"\nMinimum no. of palindromes :: "<<min_palin;
62     cout<<endl;
63     return 0;
64 }

```

[Civics](#)
[College student jobs](#)
[Concepts](#)
[Education Grant](#)
[Inaccuracies](#)
[infolinks](#)
[Back](#) | [Next](#)

All problems on Dynamic Programming

- * [Find the nth term of fibonacci series](#)
- * [Evaluate combination\(n, r\)](#)
- * [Solve the Edit-Distance problem](#)
- * [Longest Common Subsequence \(LCS \) problem](#)
- * [Given a set of coin denominations, find the minimum number of coins required to make a change for a target value](#)
- * [Longest Increasing Subsequence \(LIS \) problem](#)
- * [Unbounded Knapsack problem](#)
- * [0/1 Knapsack problem](#)
- * [Splitting a string into minimum number of palindromes](#)

Privacy Policy

csegeek.com does not use cookies to track any user information. No personal information is collected. However, users can optionally register for the email newsletter to remain updated with the website content. This website displays ads via ad-networking mediums which might contain cookies. This website may contain links to other sites and is not responsible for the contents or privacy policies of such websites.

Copyright © 2013 csegeek.com | All rights reserved.

The content is copyrighted to csegeek.com.

All contents of this website are a property of csegeek.com may not be redistributed in any way. Failure to do so is the violation of copyright laws.

The contents are provided without any warranty and may contain inaccuracies or errors. All risk related to the use of the contents is borne entirely by the user.