Algorithms @tutorialhorizon

- Home
- Contribute
- Arrays
- Linked List
- Recursion
- Dynamic Programming
- Backtracking
- Binary Tree
- Trees
- Difficulty Level
- Interviews
- MISC

DIFFICULTY LEVEL / EXPERT / GOOGLE INTERVIEW / TOP COMPANIES

2

Dynamic Programming — Split the String into Minimum number of Palindromes.

BY SJ · APRIL 10, 2016

Objective:

FOLLOW:

To search type and hit enter

SUBSCRIBE FOR NEW POSTS (NO SPAMS!!)

Enter your email address to subscribe to this blog and receive notifications of new posts by email.

Join 300 other subscribers

Email Address

Subscribe

RECENT POSTS

You are given a large string. You need to cut the string into chunks such that each substring that you get is a palindrome. Remember that each 1 length string is always a palindrome. You need to find the minimum number of cuts that you need to make such that each substring is a palindrome.

Example:

```
String x = "xabaay"
5 cuts makes all the
substrings palindrome
: x, a, b, a, a, y
4 cuts makes all the
substrings palindrome
: x, a, b, aa, y
3 cuts makes all the
substrings palindrome
: x, aba, a, y
Output: 3 cuts
```

Approach:

Using Recursion:

We need to try all the cuts which makes all the substrings palindrome and then we will choose the minimum number of cuts required.

Find the last repeating character in a given string.

Find the first non repeating character in a given string

Find the first repeating character in a given string

K-Means Algorithm

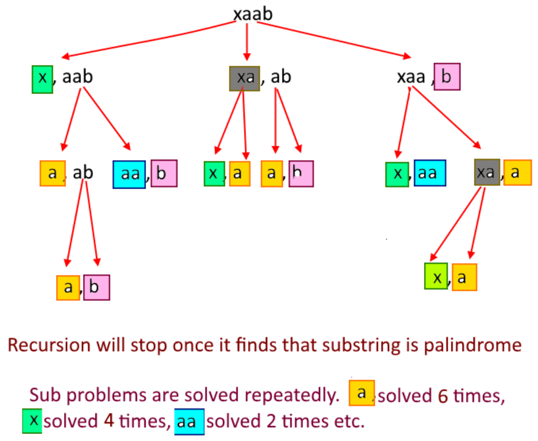
k-Nearest Neighbors

Find longest Snake sequence in a given matrix

Dynamic Programming — Count all paths in 2D Matrix with Obstructions in it

FOLLOW ME ON TWITTER

1. Check if entire string is palindrome, if yes then return 0 (no cuts required).
2. If step 1 fails means it’s not a palindrome then split the string into two parts in every possible way (ex: String is “xaab” then all possible splits are “x, aab” , “xa, ab”, “xaa, b”) and solve these two parts recursively till substring not found to be a palindrome. Each time you make a split, add 1 to number of cuts.
3. Choose the minimum cuts in step 2.
4. See the diagram for more understanding.



Recursion will stop once it finds that substring is palindrome

Sub problems are solved repeatedly. a solved 6 times, x solved 4 times, aa solved 2 times etc.

Time Complexity: If there are n characters in string then n-1 cuts can be made and for every cut we two options, whether cut or not. so time complexity will be $2^{(n-1)}$.

Recursion Code:

Tweets by @tutorialhorizon

Tutorial Horizon
@tutorialhorizon

Find the last repeating character in a given string. algorithms.tutorialhorizon.com/find-the-last-…

Find the last rep…
Objective: Given …
algorithms.tutoria…

30 May

Tutorial Horizon
@tutorialhorizon

Find the first non repeating character in a given string algorithms.tutorialhorizon.com/find-the-first…

Find the first non…
Objective: Given …
algorithms.tutoria…

27 May

Tutorial Horizon
@tutorialhorizon

Copy of How to read Wall Street Journal for free online?: youtu.be/Ytpbrv2BfXM?a via @YouTube

13 Mar

Embed View on Twitter

TOP POSTS & PAGES

Binary Search Tree Complete Implementation.

All Articles

Reverse a Linked List

Backtracking - N Queens Problem

Dynamic Programming - Coin Change Problem

```
public int splitRecursion(String x)
                if(x=="" || isPalin
//                      System.out.
                        return 0;
                }else{
                        int cuts =
                        for (int i
                                cut
                        }
                        return cuts
                }
        }
        public boolean isPalindrome
                int n = s.length();
                  for (int i=0;i<(n
                     if (s.charAt(i
                         return fal
                     }
                  }
                  return true;
        }
```

splitRecursion.java hosted with by view raw GitHub

We can reduce it by **dynamic programming**.

Using Dynamic Programming:

As we have see in the diagram above many problems are solved repeatedly. So we can apply the **Top-down approach**.

We will use Hash Map and store the solution of sub problems. So every time we make a cut, we check whether we have already solved the sub problem by checking its entry in Hash Map,

Introduction To Backtracking Programming

Print All Paths from Top left to bottom right in Two Dimensional Array

Dynamic Programming - Subset Sum Problem

Binary Min - Max Heap

Dynamic Programming - Count all paths from top left to bottom right of a mXn matrix

if yes then use it and if not then solve it and store it in HashMap for future use.

Now this way every problem will be solved only once. Time Complexity will be number of sub problems so it will **O(N^2)**.

NOTE: We have compared the running time of recursion and dynamic programming in the output.

Complete Code:

```
import java.util.HashMap;

public class SplitPalindrome {

        static HashMap<String,Integ

        public int splitDP(String x
                if(x=="" || isPalin
//                      System.out.
                        return 0;
                }else{
                        int cuts =
                        for (int i










                                if(

                                }el


                                }
                                cut
                        }
                        return cuts
                }
        }

        public int splitRecursion(S
                if(x=="" || isPalin
//                      System.out.
                        return 0;
                }else{
                        int cuts =
                        for (int i
                                cut
                        }
                        return cuts
                }
        }
        public boolean isPalindrome
                int n = s.length();
                  for (int i=0;i<(n
                     if (s.charAt(i
                         return fal
                     }
                  }
                  return true;
        }

        public static void main(Str
                String a = "cdcdddc
                SplitPalindrome s =
                long startTime = Sy
                System.out.println(
                long stopTime = Sys
                long elapsedTime =
                System.out.println(
                startTime = System.
                System.out.println(
                stopTime = System.c
                elapsedTime = stopT
                System.out.println(

        }

}
```

SplitPalindrome.java hosted with **view raw** by **GitHub**

Output:

```
Recursion- Cuts
Required: 3
Recursion- Time
Taken(ms): 345

Dynamic Programming-
Cuts Required: 3
Dynamic Programming-
Time Taken(ms): 2
```

NOTE: As you can see the Dynamic Programming is way faster than Recursion.

Related Posts:

- Dynamic Programming — Maximum Product Cutting Problem.
- Dynamic Programming — Rod Cutting Problem
- Dynamic Programming — Minimum Coin Change Problem
- Dynamic Programming — Minimum Numbers are Required Whose Square Sum is Equal To a Given…

- Introduction To Dynamic Programming — Fibonacci Series
- Dynamic Programming — Longest Palindromic Subsequence
- Dynamic Programming — Edit Distance Problem
- Dynamic Programming — Longest Common Substring
- Dynamic Programming — Coin Change Problem
- Dynamic Programming — Stairs Climbing Puzzle

Share this:

LinkedIn | Google | Print

Tumblr | Facebook | Twitter

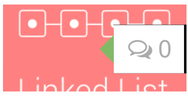
Email

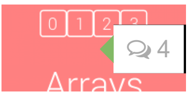
Related

Dynamic Programm... — Longest Palindromic Subsequen...	Dynamic Programm... - Longest Common Substring	Dynamic Programm... - Edit Distance Problem
In "Amazon Questions"	In "Arrays"	In "Amazon Questions"

Tags: Dynamic Programming

YOU MAY ALSO LIKE...





Alternate Splitting

Find longest

Find the Kth

of a given Linked List

Snake sequence in a given matrix

Smallest/ Largest Element in an Array

giraffe0813

Thanks for your posts,it’s really really helpful~~

Harry Patton

good algorithm; however, function isPalindrome is o(n) so the DP is o(n*3). In addition to that, it uses recursion. http://www.geeksforgeeks.org/dynamic-programming-set-17-palindrome-partitioning/ shows a better o(n*2) solution without recursion.

Powered by WordPress. Theme by Alx.

214 queries in 0.791 seconds.