



13

More ▾ Next Blog»

Create Blog Sign In

# 水中的鱼

zhang lei

Add to circles

Followers list is private

Monday, March 4, 2013

## [LeetCode] Palindrome Partitioning II, Solution

Given a string *s*, partition *s* such that every substring of the partition is a palindrome.  
Return the minimum cuts needed for a palindrome partitioning of *s*.

For example, given *s* = "aab",

Return 1 since the palindrome partitioning ["aa", "b"] could be produced using 1 cut.

[» Solve this problem](#)

### [Thoughts]

凡是求最优解的，一般都是走DP的路线。这一题也不例外。首先求dp函数，

定义函数

$D[i, n]$  = 区间 $[i, n]$ 之间最小的cut数， $n$ 为字符串长度

```
a b a b b b a b b a b a
      i                     n
```

如果现在求 $[i, n]$ 之间的最优解？应该是多少？简单看一看，至少有下面一个解

```
a b a b b b a b b a b a
      i         j j+1 n
```

此时  $D[i, n] = \min(D[i, j] + D[j+1, n]) \quad i \leq j < n$ 。这是个二维的函数，实际写代码时维护比较麻烦。所以要转换成一维DP。

如果每次，从 $i$ 往右扫描，每找到一个回文就算一次DP的话，就可以转换为

$D[i]$  = 区间 $[i, n]$ 之间最小的cut数， $n$ 为字符串长度，则，

$D[i] = \min(1 + D[j+1]) \quad i \leq j < n$

有个转移函数之后，一个问题出现了，就是如何判断 $[i, j]$ 是否是回文？每次都从 $i$ 到 $j$ 比较一遍？太浪费了，这里也是一个DP问题。

定义函数

$P[i][j]$  = true if  $[i, j]$ 为回文

那么

$P[i][i] = \text{str}[i] == \text{str}[i] \quad \&\& \quad P[i+1][j-1]$ ;

基于以上分析，实现如下：

```
1: int minCut(string s) {
2:     int len = s.size();
3:     int D[len+1];
4:     bool P[len][len];
5:     //the worst case is cutting by each char
6:     for(int i = 0; i <= len; i++)
7:         D[i] = len-i;
8:     for(int i = 0; i < len; i++)
9:         for(int j = 0; j < len; j++)
10:            P[i][j] = false;
11:     for(int i = len-1; i >= 0; i--){
12:         for(int j = i; j < len; j++){
13:             if(s[i] == s[j] && (j-i<2 || P[i+1][j-1])){
14:                 P[i][j] = true;
15:                 D[i] = min(D[i], D[j+1]+1);
16:             }
17:         }
18:     }
19:     return D[0]-1;
```

### 最近阅读

- Orleans: Distributed Virtual Actors for Programmability and Scalability
- Swift: A Storage Architecture for Large Objects
- Swift: Using Distributed Disk Striping to Provide High I/O Data Rates
- Software-defined Storage - IBM

### Google Plus

[Lei Zhang](#)

### About Me



zhang lei

Follow 0

小磊哥

[View my complete profile](#)

### The Book

[Coding Puzzles](#)

Follow @codingtmd

### Visit Map



### Labels

- Algorithm (138)
- BFS (1)
- binary tree (7)
- Bit Operation (1)

```
20: }
```

或者可以考虑使用回溯+剪枝，比如：

```
1: int minCut(string s) {
2:     int min = INT_MAX;
3:     DFS(s, 0, 0, min);
4:     return min;
5: }
6: void DFS(string &s, int start, int depth, int& min)
7: {
8:     if(start == s.size())
9:     {
10:         if(min > depth-1)
11:             min = depth-1;
12:         return;
13:     }
14:     for(int i = s.size()-1; i >= start; i--) //find the biggest palindrome first
15:     {
16:         if(isPalindrome(s, start, i))
17:         {
18:             DFS(s, i+1, depth+1, min);
19:         }
20:         if(min != INT_MAX) //if get result, then stop search
21:         break;
22:     }
23: }
24: bool isPalindrome(string &s, int start, int end)
25: {
26:     while(start < end)
27:     {
28:         if(s[start] != s[end])
29:             return false;
30:         start++; end--;
31:     }
32:     return true;
33: }
```

Posted by [zhang lei](#) at 7:57 PM



Labels: [Algorithm](#), [DP](#), [LeetCode](#), [review](#)

- [Brute-force](#) (2)
- [design](#) (1)
- [DFS](#) (3)
- [DP](#) (16)
- [facebook](#) (1)
- [google](#) (1)
- [hash](#) (1)
- [LeetCode](#) (121)
- [Math](#) (1)
- [recursion](#) (4)
- [review](#) (22)
- [sort](#) (1)
- [SRM](#) (6)
- [TopCoder](#) (7)
- [two-pointer](#) (5)
- [Ubuntu](#) (1)
- [Yahoo](#) (1)
- [二分搜索](#) (7)
- [二叉树](#) (15)
- [动态规划](#) (20)
- [双指针](#) (16)
- [图](#) (1)
- [字符串处理](#) (8)
- [实现题](#) (7)
- [数组](#) (8)
- [模拟](#) (23)
- [解题报告](#) (22)
- [贪心](#) (1)
- [递归](#) (32)
- [递归+剪枝](#) (2)
- [链表](#) (10)

#### Blog Archive

- ▼ [2013](#) (102)
  - [December](#) (4)
  - [November](#) (12)
  - [August](#) (3)
  - [June](#) (2)
  - [May](#) (4)
  - [April](#) (4)
  - ▼ [March](#) (14)
    - [\[LeetCode\] Path Sum, Solution](#)
    - [\[Interview\] Serialize and De-serialize a tree](#)
    - [\[LeetCode\] Convert Sorted Array to Binary Search T...](#)
    - [\[LeetCode\] Same Tree, Solution](#)
    - [\[LeetCode\] Unique Binary Search Trees II, Solution...](#)
    - [\[LeetCode\] Unique Binary Search Trees, Solution](#)
    - [\[LeetCode\] Remove Duplicates from Sorted List II, ...](#)
    - [\[LeetCode\] Search a 2D Matrix, Solution](#)
    - [\[LeetCode\] Merge Two Sorted Lists, Solution](#)
    - [\[LeetCode\] Longest Valid Parentheses, Solution](#)
    - [\[LeetCode\] Two Sum, Solution](#)
    - [\[LeetCode\] Palindrome Partitioning II, Solution](#)
    - [\[LeetCode\] Palindrome Partitioning, Solution](#)
    - [\[LeetCode\] Surrounded Regions, Solution](#)
  - [February](#) (7)
  - [January](#) (52)
- [2012](#) (50)
- [2007](#) (1)

8 comments



Add a comment as Hongchuan Wei

Top comments ▾



**Bingkun Guo** 1 week ago · Shared publicly  
赞这个双重DP

+1 · Reply



**Zhu DZ** 1 year ago · Shared publicly  
DFS的算法有点小bug。搜索策略是找到第一个满足s[1..i] s[i+1...j].....s[k+1, n]的cut，但这个不一定是最小的。应该把第20、21行去了。比如对于“ababbbabbaba”，mi babbbab b aba”，而用这个算法得到的solution是4: “aba bbb abba b a”

但去了之后还是会超时。  
所以还是要用dp。

Translate

+1 · Reply



**zhang lei** 11 months ago  
是的，当时直接从Palindrome Partitioning拷过来。没有细细审查。应该把20,21行删掉

Translate



**Jianzhang Ma** 8 months ago · Shared publicly  
如果把DP第七行改成 len-1-i是不是更好理解？

Translate

+1 +1 · Reply



**Wei Fang** 5 months ago  
是的，那样的话，return D[0]就行了

Translate



**王川** 8 months ago · Shared publicly  
非常感谢提供这有用的资料！

Translate

+1 · Reply



**Auburn Tiger** 7 months ago · Shared publicly  
回溯+剪枝会超时的啊

Translate

+3 +1 · Reply



**sixin li** 7 months ago (edited) · Shared publicly  
表示感激, leetcode上面的都没有解释!

+1 +1 · Reply

[Post a Comment](#)

[Newer Post](#)

[Home](#)

[Older Post](#)

Subscribe to: [Post Comments \(Atom\)](#)

