

9 九章算法 第四章 互动 划分型，博弈型和背包型动态规划

笔记本: DP Note

创建时间: 10/26/2019 2:24 PM

更新时间: 10/30/2019 3:27 PM

作者: tanziqi1756@outlook.com

URL: https://www.lintcode.com/problem/copy-books/description?_from=ladder...

必做

单选题

有15个硬币排成一条线。两个参赛者轮流从右边依次拿走1或2个硬币，直到没有硬币为止。拿到最后一枚硬币的人获胜。请问先拿的参赛者获胜还是后拿的参赛者获胜？

(A)

先拿的参赛者获胜

(B)

后拿的参赛者获胜

(C)

无法确定

提交

我不会

这个题把15换成3，答案是没有变的。

解析：后手必胜，那么后手策略是什么呢？是这样的，如果先手拿1个，后手就拿2个，如果先手拿2个，后手就拿1个。一直保持这样的策略，就可以保证后手一定获胜。

必做

多选题

下列哪些是回文串？

(A)

aba

(B)

abba

(C)

a

(D)

空字符串

已回答

答错了，好可惜。正确答案是ABCD，有64%的同学答对了，要加油了。

解析：“回文串”是一个正读和反读都一样的字符串，所以上面四个选项都是回文串。

必做

多选题

假如有一些物品, 每个物品有一定重量. 给你一个背包, 它的装载重量有限. 那么你想要最大的重量物品, 你应该用什么方法呢?

- (A) 按照重量从小到大排序, 先放轻的再放重的
- (B) 按照重量从大到小排序, 先放重的再放轻的
- (C) dfs搜索, 把所有可以放入背包的物品方案求一遍
- (D) 用神奇的动态规划来解决这个问题

已回答

是有点难吗, 答错了呢。正确答案是 C D , 有39%的同学超过了你, 但是千万不要气馁。

解析: 这是很经典的一类背包问题, 可行型的背包问题. AB选项都可以很容易地举出反例. C的选项是可以完成这个, 但是直接的搜索需要指数复杂度时间。

DFS搜索的复杂度为 $O(2^n)$, 左孩子表示当前不加入背包, 右孩子表示加入背包

必做

多选题

上面一个问题的子问题是什么? (N为物品数量, M为物品重量)

- (A) N-1个物品可以拿走的最大重量是多少?
- (B) 1个物品可以拿走的最大重量是多少?
- (C) M-1的重量可以拿走多少个物品?
- (D) 1的重量可以拿走多少个物品?

已回答

没错~正确答案就是 A , 你对这个知识点的掌握超过了71%的同学

解析: 子问题是指与原问题性质相同, 规模更小的问题, CD与原题目的性质已经不同, 而B选项1个物品可以拿的重量是已知不是问题。

尝试做完全平方数

279. Perfect Squares

难度 中等

221



收藏

分享

切换为中文

关注

题目描述

评论 (201)

题解 (53) ^{New}

提交记录

Given a positive integer n , find the least number of perfect square numbers (for example, 1, 4, 9, 16, ...) which sum to n .

Example 1:

Input: $n = 12$

Output: 3

Explanation: $12 = 4 + 4 + 4$.

Example 2:

Input: $n = 13$

Output: 2

Explanation: $13 = 4 + 9$.

执行结果: 通过 [显示详情](#)

执行用时: 1949 ms, 在所有 java 提交中击败了 5.00% 的用户

内存消耗: 35.6 MB, 在所有 java 提交中击败了 41.13% 的用户

炫耀一下:

进行下一个挑战:

丑数 II 中等

提交时间	提交结果	执行用时	内存消耗	语言
2 分钟前	通过	1949 ms	35.6 MB	Java
3 分钟前	通过	1937 ms	35.3 MB	Java

```
1 class Solution {
2     public int numSquares(int n) {
3         int[] dp = new int[n+1];
4
5         dp[0] = 0;
6         // 转移方程: dp[i] = min(dp[i-1] + 1, dp[a] + dp[b])
7         for (int i = 1; i <= n; i++) {
8             dp[i] = dp[i-1] + 1;
9             for (int j = 1; j < i; j++) {
10                 if (j*j == i) {
11                     dp[i] = 1;
12                 }
13                 else {
14                     dp[i] = Math.min(dp[i], dp[j]+dp[i-j]);
15                 }
16             }
17         }
18
19         // System.out.println(Arrays.toString(dp));
20         return dp[n];
21     }
22 }
```

可以继续优化
 $dp[a] + dp[b]$

执行结果: 通过 [显示详情](#)

执行用时: 38 ms, 在所有 java 提交中击败了 78.11% 的用户

内存消耗: 35.6 MB, 在所有 java 提交中击败了 41.13% 的用户

炫耀一下:

进行下一个挑战:

丑数 II 中等

提交时间	提交结果	执行用时	内存消耗	语言
2 分钟前	通过	38 ms	35.6 MB	Java

```
1 class Solution {
2     public int numSquares(int n) {
3         int[] dp = new int[n+1];
4
5         dp[0] = 0;
6         // 转移方程: dp[i] = min(dp[i-1] + 1, dp[i-j*j]+1)
7         // 式中, j重新遍历dp
8         for (int i = 1; i <= n; i++) {
9             dp[i] = dp[i-1] + 1;
10            for (int j = 1; j*j <= i; j++) {
11                dp[i] = Math.min(dp[i], dp[i-j*j]+1);
12            }
13        }
14
15        // System.out.println(Arrays.toString(dp));
16        return dp[n];
17    }
18 }
```

必做

单选题

有多少种方式把N表示小于K个完全平方数之和？这个问题的转移方程和答案是？

A

```
f[i][k] = f[i - j * j][k - 1] + 1
return f[n][k]
```

B

```
f[i][k] += f[i - j * j][k - 1]
return f[n][k]
```

C

```
f[i][k] = f[i - j * j][k - 1] + 1
return f[n][0] + f[n][1] + ... + f[n][k]
```

D

```
f[i][k] += f[i - j * j][k - 1]
return f[n][0] + f[n][1] + ... + f[n][k]
```

已回答

答错了，好可惜。正确答案是D，有34%的同学答对了，要加油了。

解析：f[i][k]表示的是数字i可以拆分成k的方案数，它就应该是所有 $i - j * j$ 的并且由k-1组成的方案数之和，而如果是等于前面加一的话，比如 $17 = 1 + 16 = 4 + 13$ 这样的，只是等于前面加一，对于1+16和4+13我们就只会保留一种，然后我们要求小于k的方案数，结果就是拆分成1,2,3...k个数之和。

分割回文串II Leetcode132

132. Palindrome Partitioning II

难度 困难

74



收藏

分享

切换为中文

关注

题目描述

评论 (50)

题解 (18) ^{New}

提交记录

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*.

Example:

Input: "aab"

Output: 1

Explanation: The palindrome partitioning ["aa","b"] could be produced using 1 cut.

```

1  class Solution {
2      public int minCut(String ss) {
3          int N = ss.length();
4          if( N == 0 ) {
5              return 0;
6          }
7          char[] s = ss.toCharArray();
8
9          boolean[][] isPalin = isPalindrome(s);
10
11         int[] dp = new int[N+1];
12
13         // State: dp[i] means the first how many cuts needed for substring(0, i-1)
14         // State transfer equation: dp[i] = min(dp[j]+1) where j = 0, 1, 2 ... , i
15         // initialize:
16         dp[0] = 0;
17         // transfer equation:
18         for( int i = 1; i < N+1; i++ ) {
19             dp[i] = Integer.MAX_VALUE;
20             for( int j = 0; j < i; j++ ) {
21                 if( isPalin[j][i-1] ) {
22                     dp[i] = Math.min(dp[i], dp[j] + 1);
23                 }
24                 // 注意一个字符也是回文串
25             }
26         }
27         return dp[N] - 1; // Every time you cut one more time actually.
28     }
29
30     // Record whether substring(i, j) is a palindrome.
31     private boolean[][] isPalindrome(char[] s) {
32         int N = s.length;
33         // initial values are false.
34         boolean[][] isPalin = new boolean[N][N];
35         // 中心扩展法
36         int left;
37         int right;
38
39         // Record whether substring(i, j) is a palindrome.
40         private boolean[][] isPalindrome(char[] s) {
41             int N = s.length;
42             // initial values are false.
43             boolean[][] isPalin = new boolean[N][N];
44             // 中心扩展法
45             int left;
46             int right;
47             for( int mid = 0; mid < N; mid++ ) {
48                 // odd-length palindrome
49                 left = mid;
50                 right = mid;
51                 while( left >= 0 && right < N && s[left] == s[right] ) {
52                     isPalin[left][right] = true;
53                     left--;
54                     right++;
55                 }
56
57                 // even-length palindrome
58                 left = mid - 1;
59                 right = mid;
60                 while( left >= 0 && right < N && s[left] == s[right] ) {
61                     isPalin[left][right] = true;
62                     left--;
63                     right++;
64                 }
65             }
66             return isPalin;
67         }
68     }
69 }

```


↓ i

S[0] a a c b c a a d c

0 1 2 3 4 5 6 7 8

f[i]

初始化 f[0] = 0

f[1] = 1 substring(j, i-1) 是回文串
在 S[0] 后面切一下

f[2] = 1 sub(j, i-1) j=0, 1 时, 都是回文串
f[2] = min(f[0] + 1, f[1] + 1) = 1

f[3] = 2 sub(j, i-1) 这时 j=0 就不是回文串了
f[3] = min(f[1] + 1, f[2] + 1) = 2

...

Lintcode

437. Copy Books

中文 ☒ English

Given `n` books and the `i`-th book has `pages[i]` pages. There are `k` persons to copy these books.

These books list in a row and each person can claim a continuous range of books. For example, one copier can copy the books from `i`-th to `j`-th continuously, but he can not copy the 1st book, 2nd book and 4th book (without 3rd book).

They start copying books at the same time and they all cost 1 minute to copy 1 page of a book. What's the best strategy to assign books so that the slowest copier can finish at earliest time?

Return the shortest time that the slowest copier spends.

Example

Example 1:

Input: `pages = [3, 2, 4]`, `k = 2`

Output: 5

Explanation:

First person spends 5 minutes to copy book 1 and book 2.

Second person spends 4 minutes to copy book 3.

Example 2:

Input: `pages = [3, 2, 4]`, `k = 3`

Output: 4

Explanation: Each person copies one of the books.

Challenge

O(nk) time

Notice

The sum of book pages is less than or equal to 2147483647

Input test data (one parameter per line)

How to understand a testcase? ◀

[3,2,4]

2

Accepted

⚡ Powered by LintCode FlashJudge

Input

Show Difference ☒

```
[3,2,4,1,5]  
3
```

Your stdout

```
[0, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647]  
[0, 3, 5, 9, 10, 15]  
[0, 3, 3, 5, 5, 9]  
[0, 3, 3, 4, 5, 5]
```

Output


```
5
```

Expected

```
5
```

Accepted

⚡ Powered by LintCode FlashJudge

 100%

100% test cases passed

Total runtime 562 ms

```

1 import java.util.Arrays;
2 /*
3 Author: Ziqi Tan
4 */
5 public class Solution {
6     public static int copyBooks(int[] pages, int k) {
7         // write your code here
8
9         int N = pages.length;
10        if( N == 0 ) {
11            return 0;
12        }
13        if( k > N ) {
14            k = N;
15        } // because we need N guys at most to copy N books
16
17        int[][] dp = new int[k+1][N+1];
18        // State: f[k][j] means
19        // the time needed for the first k copiers to copy the first j books.
20
21        // State transfer equation:
22        // f[k][j] = min(max(f[k-1][i], A[i] + ... + A[j-1]))
23        // where i = 0, 1, 2 ... j
24
25        // initialize
26        dp[0][0] = 0;
27        // 前k个人完成0本书
28        for( int _k = 1; _k < k+1; _k++ ) {
29            dp[_k][0] = 0;
30        }
31        // 前0个人不可能完成任何一本书
32
33        for( int j = 1; j < N+1; j++ ) {
34            dp[0][j] = Integer.MAX_VALUE;
35        }
36        // 前1个人直接累计求和
37        dp[1][1] = pages[0];
38        for( int j = 2; j < N + 1; j++ ) {
39            dp[1][j] = dp[1][j-1] + pages[j-1];
40        }
41
42        // state transfer
43        for( int _k = 2; _k < k + 1; _k++ ) {
44            for( int j = 1; j < N + 1; j++ ) {
45                dp[_k][j] = Integer.MAX_VALUE;
46                // 第_k个抄写员, 抄写第j本书到第i本书 where i < j
47                // 从后往前遍历
48                // 第一步的运算: 第_k个抄写员, 不抄书, 前_k-1个抄写员, 抄写前j本书
49                int timeFor_k = 0;
50                for( int i = j; i >= 0; i-- ) {
51                    dp[_k][j] = Math.min(dp[_k][j], Math.max(timeFor_k, dp[_k-1][i]));
52                    // max(timeFor_k, time for the first _k-1 copiers)
53                    if( i > 0 ) {
54                        timeFor_k += pages[i-1];
55                    }
56                }
57            }
58        }
59
60        /*for( int i = 0; i < k+1; i++ ) {
61            System.out.println(Arrays.toString(dp[i]));
62        }*/
63
64        return dp[k][N];
65    }
66    public static void main(String[] args) {
67        // TODO Auto-generated method stub
68        int[] pages = new int[]{1,9,7,3,4,7};
69        System.out.println(copyBooks(pages, 3));
70    }
71 }
72

```


	3	2	4	1	5	
	0	1	2	3	4	5
0	0	∞	∞	∞	∞	∞
1	0	3	5	9	10	15
2	0	3	3	5	5	9
3	0	3	3	4	5	5

$$\begin{aligned}
 f[2][4] &= \text{前2个人抄前4本书} \\
 &= \min \left(\begin{aligned}
 &\max(0, 3+2+4+5) = 10 \\
 &\max(1, 3+2+4) = 9 \\
 &\max(1+4, 3+2) = 5 \\
 &\max(1+4+2, 3) = 7 \\
 &\max(1+4+2+3, 0) = 10
 \end{aligned} \right) \\
 &= 5
 \end{aligned}$$