1. 题目
2. 题干

给定正整数 n，找到若干个完全平方数（比如 1, 4, 9, 16, ...）使得它们的和等于 n。你需要让组成和的完全平方数的个数最少。

1. 示例

示例 1:

输入: n = 12

输出: 3

解释: 12 = 4 + 4 + 4.

示例 2:

输入: n = 13

输出: 2

解释: 13 = 4 + 9.

1. 题解
2. 思路

动态规划问题，循序渐进解决问题。首先，要写一个函数判断当前的n是否是完全平方数，当n是完全平方数时，那么组成n的完全平方数个数就是 1。dp的典型思路是选取最近的一个完全平方数然后加上剩余的数，然而这样做是无法包括所有情况的。

如，当n = 12时，最近的完全平方数是9，那么剩余的数是3，组成3的完全平方个数为3个，那么n = 12时组成12的完全平方数个数是4。然而，12可以用3个4组成，那么只需要3个完全平方数就可以组成。说明需要对12之前的所有完全平方数进行遍历，求得最小的一个。

那么，我们就需要在创建一个完全平方数数组记录出现的完全平方数。当n为非完全平方数时，遍历之前的完全平方数数组，寻找组合数最小的一个。

1. 代码实现

Java :

class Solution {

    public int numSquares(int n) {

        int[] dp = new int[n+1];

        int[] dou = new int[n+1];

        int minVal;

        for(int i = 1, j = 1; i <= n; ++i){

            if(isDou(i)){

                dp[i] = 1;

                dou[j++] = i;

            }else{

                minVal = dp[dou[1]] + dp[i-dou[1]];

                for(int k = 1; k < j; ++k){

                    minVal = Math.min(dp[dou[k]] + dp[i-dou[k]], minVal);

                }

                dp[i] = minVal;

            }

        }

        return dp[n];

    }

    public boolean isDou(int n){

        for(int i = 1; i <= n ; ++i){

            if( i \* i == n){

                return true;

            }

        }

        return false;

    }

}