Leetcode\_322\_CoinChange\_找零钱\_Medium

# Leetcode\_322\_CoinChange\_找零钱\_Medium

## 题目介绍

\* 难度：Medium

\* <https://leetcode.com/problems/coin-change/description/>

\* 题目介绍：

\* You are given coins of different denominations and a total amount of money amount.

\* Write a function to compute the fewest number of coins that you need to make up that amount.

\* If that amount of money cannot be made up by any combination of the coins, return -1.

\* Example 1:

\* Input: coins = [1, 2, 5], amount = 11

\* Output: 3

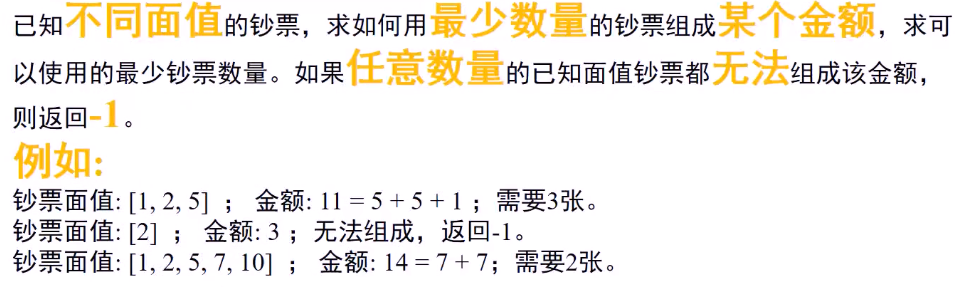
\* Explanation: 11 = 5 + 5 + 1

\* Example 2:

\* Input: coins = [2], amount = 3

\* Output: -1

\* Note:You may assume that you have an infinite number of each kind of coin.



## 思路分析

\* 思路分析：动态规划思想：

\* 定义dp数组，长度为amount+1；dp[0]=0;其他初始化为-1.

**\* dp[n] = min{dp[n-c\_1],dp[n-c\_2],......,dp[n-c\_n]} + 1;**

**\* dp[0] = 0**

\* 编程实现困难：n从1到amount，依次找到对应dp[n];

\* 由于需要每次遍历coins，判断i < coins[j]，若成立直接continue；

\* 否则进入判断是否是有更小coin数，若dp[i - coins[j]] == -1，认为不可行；

\* 否则，可行选取最小的coins。由于初始时dp[i]为-1，所以需要更新的条件是

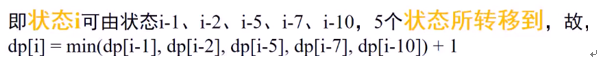
\* dp[i] == -1 || dp[i] > dp[i - coins[j]]+1。

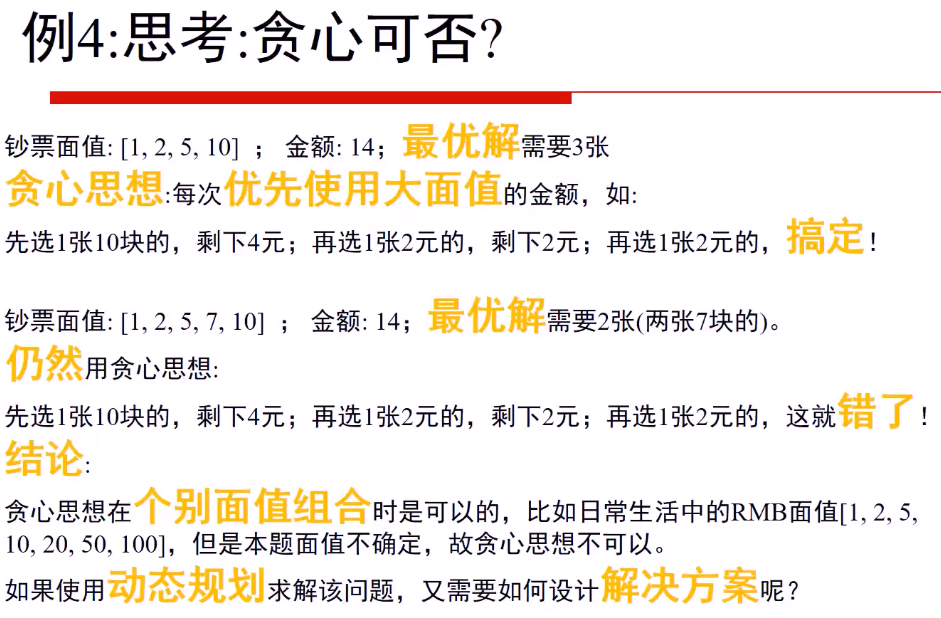
\*

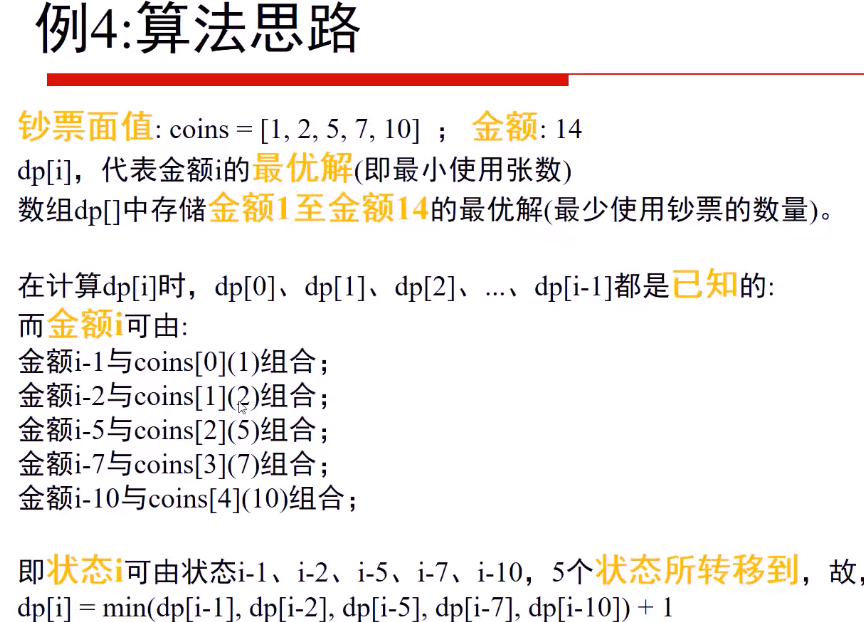
\* 改进：每次都要遍历coins，复杂度高。

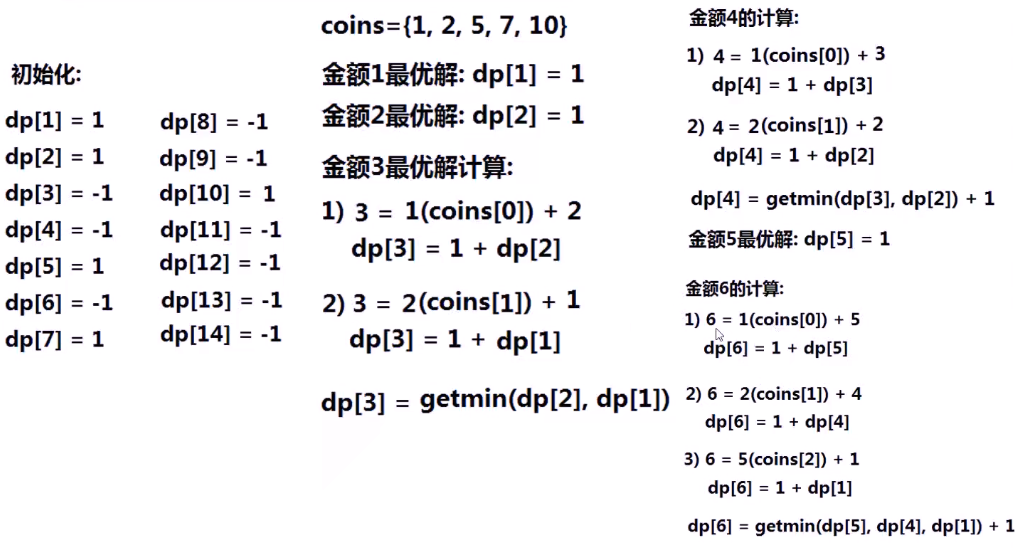
\* 首先对coins排序，然后遇到i<coins[j]直接break；

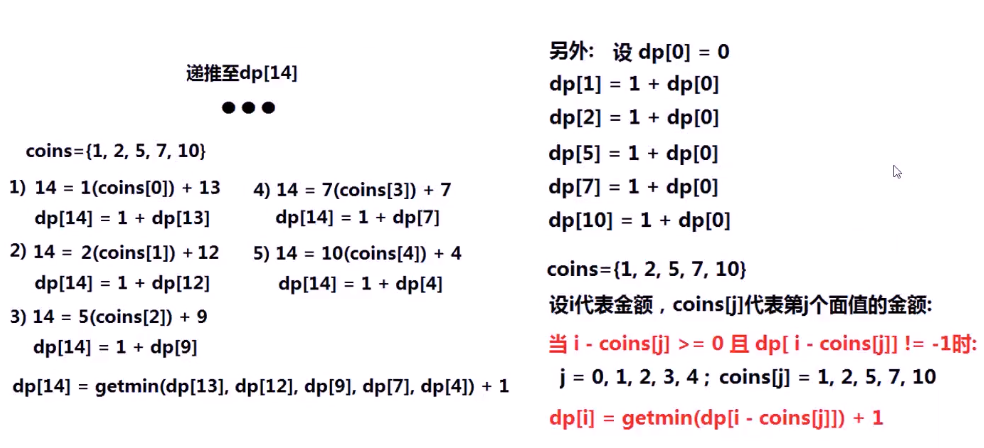
\* 排序后，就可以由continue变为break。











## Java代码

public int **coinChange**(int[] coins, int amount) {

if(coins == null|| coins.length == 0) return -1;

int[] dp = new int[amount+1];//存放各个amount下需要的coins个数

for(int i = 1;i < amount+1;i++) dp[i] = -1;//初始化为-1

dp[0] = 0;

// Arrays.sort(coins);

for(int i = 1;i <= amount;i++){//更新dp[1]->dp[amount]

//实现 dp[n] = min{dp[n-c\_1],dp[n-c\_2],......,dp[n-c\_n]} + 1;

for(int j = 0;j < coins.length;j++){

if(i < coins[j]) continue;//因为coins不一定是排序的，所以用continue

// if(i < coins[j]) break;//coins必须是从小到大顺序，才break，否则continue

if (dp[i - coins[j]] != -1){//等于-1表示无法找到coins

if(dp[i] == -1 || dp[i] > dp[i - coins[j]]+1) {

dp[i] = dp[i - coins[j]] + 1;

}

}

}

}

return dp[amount];//返回结果

}

