322. Coin Change

Description | Hints | Submissions | Discuss | Solution

Pick One

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.

```java
public class L322 {
/*
 * 这道题目不能用回溯(DFS),因为会超时。（这个自己尝试过）
 * 所以本道题目采取的是利用动态规划
 */
      public int coinChange(int[] coins, int amount) {

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

          int [] dp = new int [amount + 1];

          for(int i = 1; i <= amount; i ++) {
              dp[i] = Integer.MAX_VALUE;
              for(int j = 0; j < coins.length; j ++) {
                  if(i >= coins[j] && dp[i - coins[j]] != Integer.MAX_VALUE) {
                      //这个就是动归方程，原来都是用coins作为目标，现在用amount作为目标，方法值得借鉴
                       dp[i] = Math.min(dp[i], dp[i - coins[j]] + 1);
                  }
              }
          }
          return dp[amount] == Integer.MAX_VALUE ? -1 : dp[amount];
      }
```