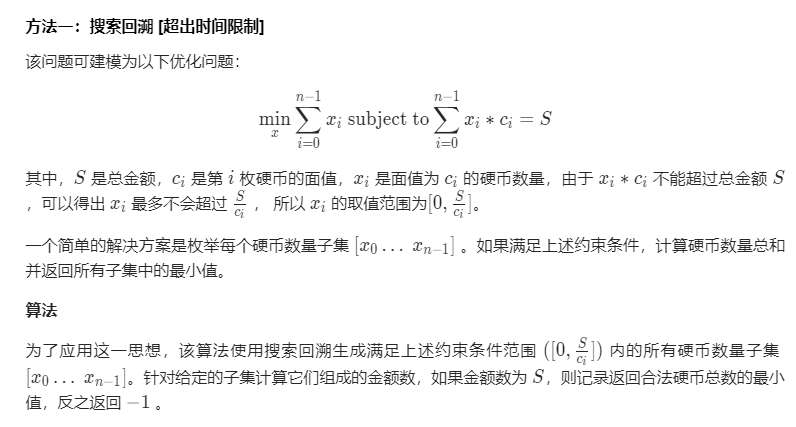
零钱兑换

[力扣官方题解](https://leetcode-cn.com/u/leetcode-solution/)发布于 2020-03-0597.5k**官方**C++JavaPython动态规划

**预备知识**

* 搜索回溯
* 动态规划

 Java

public class Solution {

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

return coinChange(0, coins, amount);

}

private int coinChange(int idxCoin, int[] coins, int amount) {

if (amount == 0) {

return 0;

}

if (idxCoin < coins.length && amount > 0) {

int maxVal = amount / coins[idxCoin];

int minCost = Integer.MAX\_VALUE;

for (int x = 0; x <= maxVal; x++) {

if (amount >= x \* coins[idxCoin]) {

int res = coinChange(idxCoin + 1, coins, amount - x \* coins[idxCoin]);

if (res != -1) {

minCost = Math.min(minCost, res + x);

}

}

}

return (minCost == Integer.MAX\_VALUE)? -1: minCost;

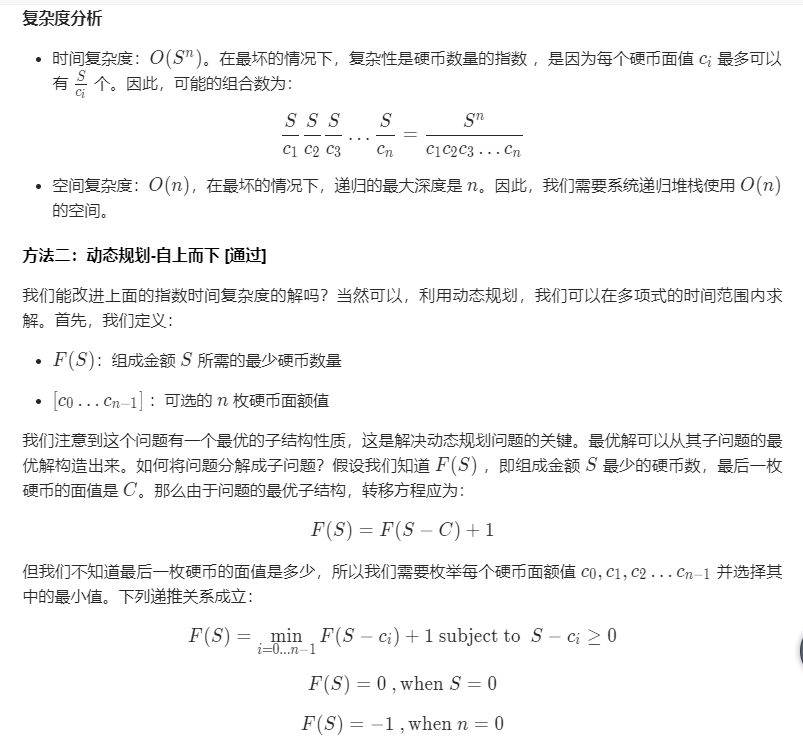
}

return -1;

}

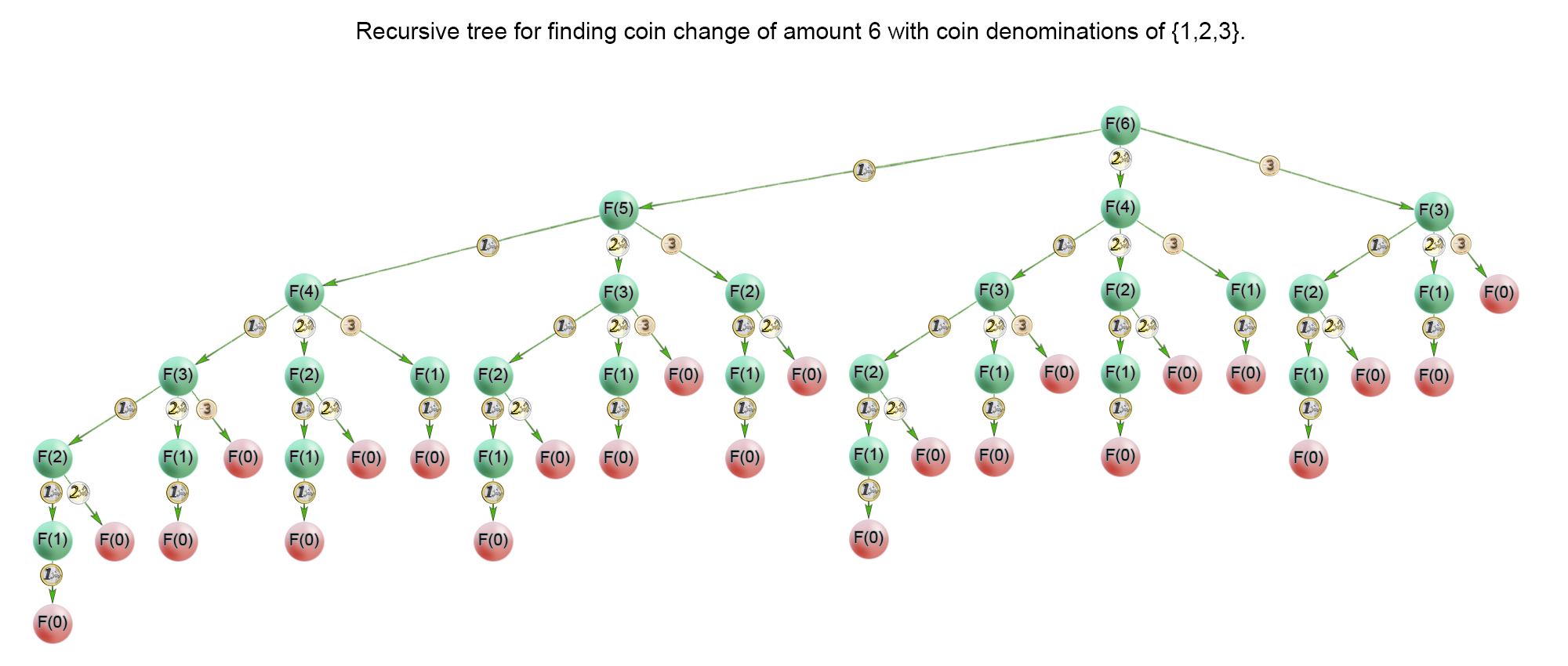
}

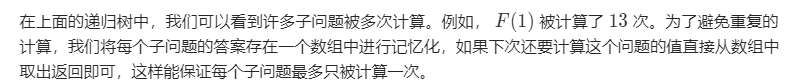
// Time Limit Exceeded



F(S)可看成是（c1, c1, c2, c3, c1）这样组成的，最后一枚硬币，不是最后一种硬币，可能最后一枚的面值之前取过，一次取一枚，区别于一次得到一种硬币的数量

最后一枚，①可能是某一种，②可能是0，③可能不存在，-1 （不一定有这样的最后）



* Java

public class Solution {

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

if (amount < 1) {

return 0;

}

return coinChange(coins, amount, new int[amount]);

}

private int coinChange(int[] coins, int rem, int[] count) {

if (rem < 0) {

return -1;

}

if (rem == 0) {

return 0;

}

if (count[rem - 1] != 0) {

return count[rem - 1];

}

int min = Integer.MAX\_VALUE;

for (int coin : coins) {

int res = coinChange(coins, rem - coin, count);

if (res >= 0 && res < min) {

min = 1 + res;

}

}

count[rem - 1] = (min == Integer.MAX\_VALUE) ? -1 : min;

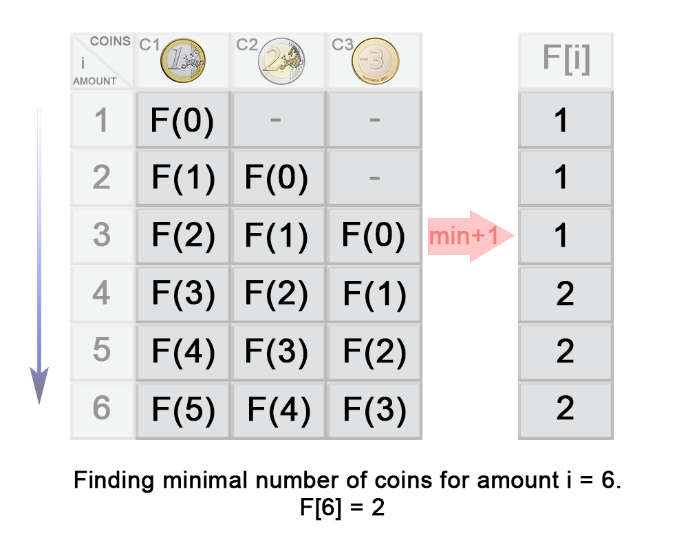
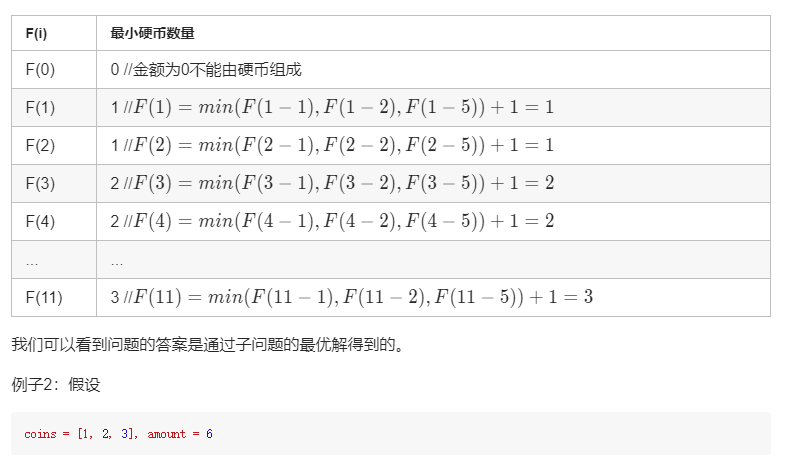
return count[rem - 1];

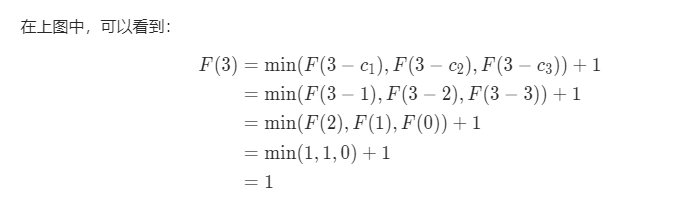
}

}

最坏的情况全是1组成s，计算F(s),F(s-1),F(s-2)…F(0)个





* Java

class Solution:

def coinChange(self, coins: List[int], amount: int) -> int:

dp = [float('inf')] \* (amount + 1)

dp[0] = 0

for coin in coins:

for x in range(coin, amount + 1):

dp[x] = min(dp[x], dp[x - coin] + 1)

return dp[amount] if dp[amount] != float('inf') else -1

