硬币选择问题

原创 张峻博 LOA算法学习笔记 2021-01-14 22:14

01 问题描述

给定m种不同面值的硬币和目标金额n。假设每种面值的硬币数量没有限制，问有多少种方法能凑出目标金额。

02 问题分析

对于原问题，我们可以分解成多个子问题，即m种硬币中任意使用1种硬币能凑成目标金额的方法、任意使用2种硬币能凑成目标金额的方法、...、任意使用m-1种硬币能凑成目标金额的方法、使用所有的m种硬币能凑成目标金额的方法。但是直接进行枚举子问题的数量将是指数级。

所以我们对m种硬币引入**“序”**（看成一个数组coin），从数组最后往前判断当前剩余金额的拼凑用不用当前的这种硬币。这种方法包含了上述所有子问题的情况，而我们只需要判断用不用当前的这种硬币即可。

假设指针i指向硬币数组的第i枚，当前剩余金额为j，OPT[i][j]表示用前i枚硬币拼凑金额j的方法数。用不用当前第i枚硬币与剩余金额j有关，分两种情况：

1) j<coin[i]

第i枚硬币面值大于剩余金额j，第i枚硬币一定用不到。即：

$$OPT[i][j]=OPT[i-1][j]$$

2) j≥coin[i]

第i枚硬币面值小于剩余金额j，可以选择用或不用第i枚硬币。即：

$$OPT[i][j]=OPT[i-1][j]+OPT[i][j-coin[i]]$$

边界考虑：

1) 对于任意金额j（j>0），如果没有硬币可用，则拼凑方法为0，即：

$$OPT[0][j]=0$$

2) 对于任意的硬币，表示金额0的方法有1种，即：

$$OPT[i][0]=1$$

递推关系式为：

$$OPT[i][j]=\begin{cases}OPT[i-1][j] & i>0, j<coin[i]\\ OPT[i-1][j]+OPT[i][j-coin[i]] & i>0, j\geq coin[i]\\ 0 & i=0, j>0\\ 1 & i\geq 0, j=0\end{cases}$$

03 正确性证明

循环不变量：OPT[i][j]表示前i枚硬币可以凑出金额j的方法数。

初始化：对于任意的硬币面值，凑出金额0的方法有1种，初始化OPT[i][0]=1；金额大于0，没有硬币则一定凑不出来，方法数为0，初始化OPT[0][j]=0。

维护：假设当前考察前i枚硬币凑出金额j的情况。如果j小于第i枚硬币的面值，则第i枚硬币一定用不到，即等于前i-1枚硬币凑出金额j的方法，OPT[i][j]=OPT[i-1][j]。当j大于等于第i枚硬币的面值，我们可以选择用第i枚硬币或者不用第i枚硬币。总的方法为这两种情况的和，OPT[i][j]=OPT[i-1][j]+OPT[i][j-coin[i]]。循环不变量成立。

终止：考察前m枚硬币拼凑金额n的情况。如果第m枚硬币的面值大于n，则只取决于前m-1枚硬币，即OPT[m][n]=OPT[m-1][n]。如果第m枚硬币的面值小于等于n，则可以选择用或不用第m枚硬币，即OPT[m][n]=OPT[m-1][n]+OPT[m][n-coin[m]]。循环不变量成立。

04 代码实现

```
int CoinChange(vector<int> Coin, int number)
{
    int OPT[Coin.size()+1][number+1];    // OPT[i][j] 前i个 硬币 能表示 j 的 方法数量

    for (int j = 1; j <= number; j++)
    {
        OPT[0][j] = 0;
    }
    for (int i = 0; i <= Coin.size(); i++)
    {
        OPT[i][0] = 1;
    }

    for (int i = 1; i <= Coin.size(); i++)
    {
        for (int j = 1; j <= number; j++)
        {
            if(j >= Coin[i - 1])
                OPT[i][j] = OPT[i-1][j] + OPT[i][j - Coin[i - 1]];
            else
                OPT[i][j] = OPT[i-1][j];
        }

    }
    return OPT[Coin.size()][number];
}
```

（注：Coin数组中Coin[i-1]表示第i枚硬币的面值）

05 总结

此问题与背包问题相似，当子问题的数量为指数级时，我们可以引入“序”，在多步决策过程中有序的对候选项进行判断。引入“序”相当于把每步决策中从候选项集合里选择哪个候选项的**多分叉选择问题**变成了每步决策中对当前固定候选项用或不用的**二分叉选择问题**。结构的引入减少了分析问题的复杂性，便于程序的实现。

买房子是消费降级么 #1880

水库之声

“懂了”到底是“Get it”还是“Got it”？瞬间暴露英文水平！

爱学习英语口语