作业: 硬币找零问题

钟琦
混合 2103　3210103612

2023 年 12 月 23 日

1　问题描述

有一定数目的面值为 1、5、10、25 美分的硬币，输入这些硬币的数目和需要的总金额，输出最少需要的硬币总数目。注意到硬币数目的限制，可能存在无解的情况，此时可以规定输出-1。

2　算法思路

本题采用动态规划的方法，对于所需要的总金额 n 美分，设最少需要的硬币数目为 dp[n]。则根据动态规划的思路，n 美分是由“(n-1) 美分 +1 美分”或“(n-5) 美分 +5 美分”或“(n-10) 美分 +10 美分”或“(n-25) 美分 +25 美分”组成的，所需硬币数 dp[n] 等于前者 ((n-1) 或 (n-5) 或 (n-10) 或 (n-25) 美分）所需硬币数再加 1。所以为使 dp[n] 最小，组成 (n-1)、(n-5)、(n-10)、(n-25) 美分需要的硬币总数目也要最少，且在这四种可能性中选择最少的硬币总数目。

因此，dp[n]=min(dp[n-1],dp[n-5],dp[n-10],dp[n-25])+1，初始条件为 dp[0]=0，i<0 时 dp[i] 不存在，可忽略。

进一步考虑硬币数目有限这一细节，产生的影响主要有两方面。一是由于硬币数量不够无法纳入动态规划：比如现有 k 个 1 美分硬币，考虑 n 美分由 (n-1) 美分 +1 美分组成，前面 (n-1) 美分中已用了 k 个 1 美分硬币，则受到 1 美分硬币数目限制无法再使用 (n-1) 美分 +1 美分的策略，需要将 dp(n-1) 从 dp[n]=min(dp[n-1],dp[n-5],dp[n-10],dp[n-25])+1 中剔除，即 dp[n]=min(dp[n-5],dp[n-10],dp[n-25])+1；二是由于特定种类的硬币数目不够导致问题无解：比如有 1、5、10、25 美分硬币分别有 0，0，2，0 个，那对于 n=15 美分，2 个 10 美分的硬币无论如何组合都无法组成 15 美分，会导致问题无解，此时规定返回-1。

展示部分求解最少硬币数的函数伪代码：

Algorithm 1: 部分 MinCoinNumber 函数

```
Data: 总金额 Amount，各类硬币数量 coins，dp[i]
// dp[i].num 表示所需最少硬币数
// dp[i].number[t] 表示最少硬币策略中所需第 t 类硬币数量
// 0、1、2、3 类分别对应 1 美分、5 美分、10 美分、25 美分
Result: 最少所需硬币数 MinCoinNumber
for i ← 1 to Amount do
    for t ← 1 to 3 do
        // t=0,1,2,3 四种策略
        // T={1,5,10,25}
        if i>= 第 t 类硬币面值 T[t] 且 dp[i − T[t]] 有解 then
            // 保证 Amount=i-T[t] 的问题是有解的，否则无意义
            if dp[i − T[t]].number[t] < coins[t] then
                // 保证第 t 类硬币还有剩余，否则该策略行不通
                // n[t] 表示第 t 种策略下可能的 dp
                n[t].num = dp[i − T[t]] + 1;
                n[t].number[t] = dp[i − T[t]] + 1;
                for j! = t do
                    n[t].number[j] = dp[i − T[t]];
                end
            end
        end
    end
    dp[i].num = min(n[t].num);
    // 选择硬币数最少的策略
end
```

3 复杂度分析

3.1 时间复杂度

设要求总金额为 N 情况下的最少硬币数，在利用动态规划方法计算 $dp[N]$ 的过程中，依次计算了 $dp[1] \sim dp[N]$ 这 N 种情况，在每种情况中，遍历了减少 1 美分、5 美分、10 美分、25 美分这四种情况，且每种情况都是 $O(1)$ 级别的平凡的判断、复制与比较，所以总时间复杂度为 $O(N)$。

3.2 空间复杂度

向量 dp 需要开长度为 N 的空间，其余向量的长度都是有限常数大小的，且数量也是有限常数个，所以空间复杂度也为 $O(N)$。

4 测试说明

4.1 测试程序可行性

输入总金额 Amount，1、5、10、25 美分硬币数目 CoinLimit，输出 solution。

1. 输入：Amount=0， CoinLimit={1,1,1,1}

   输出：solution=0

   分析：金额 0 需要 0 个硬币，符合

2. 输入：Amount=40， CoinLimit={0,0,4,1}

   输出：solution=4

   分析：金额 40 若有 1 个 25 美分的硬币，则剩下的 4 个 10 美分硬币无法组成 15 美分，所以金额 40 只能由 4 个 10 美分硬币组成，符合

3. 输入：Amount=40， CoinLimit={0,1,4,1}

   输出：solution=3

   分析：与 2 不同的是多了一个 5 美分硬币，则 40=5+10+25 是最少硬币数的情况，符合

4. 输入：Amount=102， CoinLimit={1,100,100,100}

   输出：solution=-1

   分析：102 模 5 余 2，所以至少需要 2 个 1 美分硬币，而提供的情况只有 1 个 1 美分硬币，所以没有解，返回-1，符合

5. 输入：Amount=10000， CoinLimit={100,2,1,399}

   输出：solution=407

   分析：为使得所需硬币数较少，应该尽可能多地使用 25 美分硬币，假设全用，则 $25*399 = 9975$，还差 25 美分，同理，进可能多使用 10 美分、5 美分硬币，$5*2+1*10 = 20$，所以还需要 5 个 1 美分硬币，solution= $399+1+2+5 = 407$，符合

6. 输入：Amount=10000， CoinLimit={1249,250,250,200}

   输出：solution=-1

   分析：$1249*1+250*5+250*10+200*25 = 9999 < 10000$，所以无解，返回-1，符合

4.2 测试计算效率

1. 当各类硬币均充足时，CoinLimit=1000000000,1000000000,1000000000,1000000000，现令 Amount 从 0∼ 1000000，步长为 1000，输出 1000 个运行时长，导出为表格（见附件 1.xlxs），下为生成的散点图：

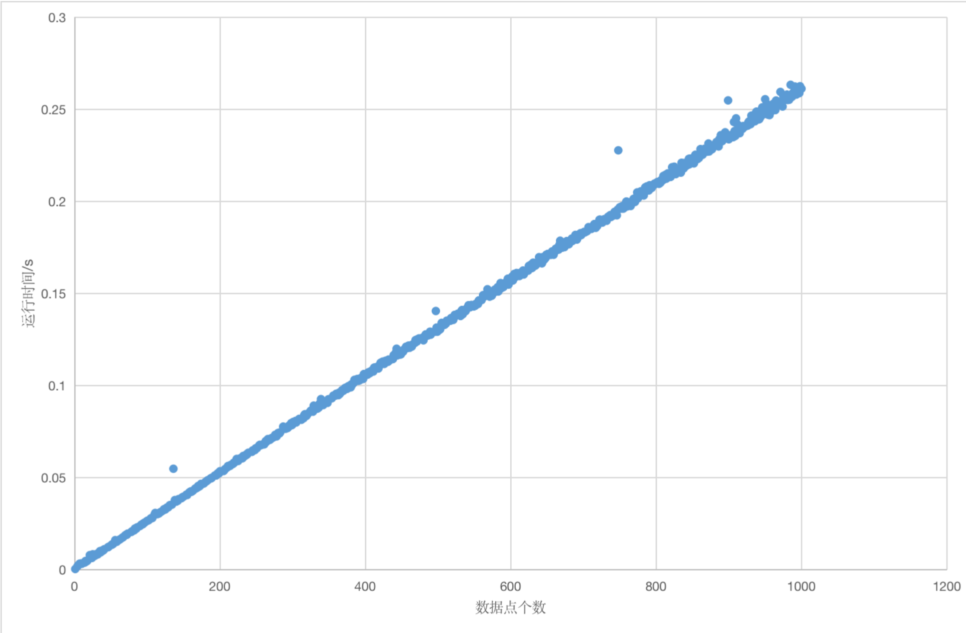

图 1: 运行时间与总金额关系图

由图可知，运行时间与总金额关系曲线大致为直线，因此时间复杂度为 $O(N)$，与理论结果相同。

2. 当 1 美分硬币不充足时，CoinLimit=1000,1000000000,1000000000,1000000000，Amount 以与 1 相同的状态增加，下为生成的散点图：

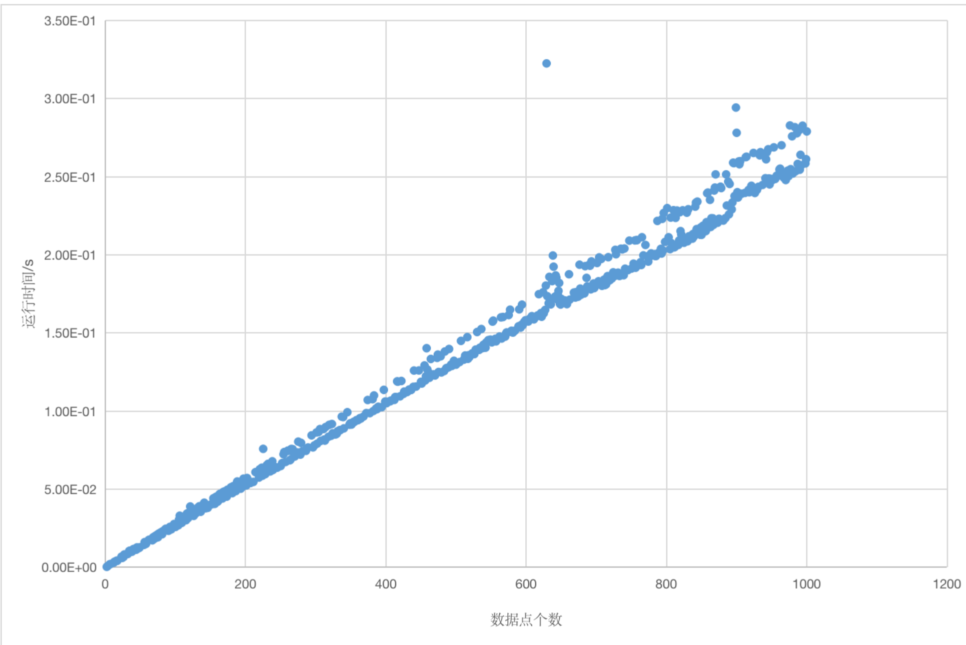

图 2: 运行时间与总金额关系图

由图可知，运行时间与总金额关系曲线虽然有所分叉，都均大致为直线，因此时间复杂度为 $O(N)$，与理论结果相同。

3. 当全部硬币不充足时，CoinLimit=1000,1000,1000,1000,Amount 以与 1 相同的状态增加，下为生成的散点图：

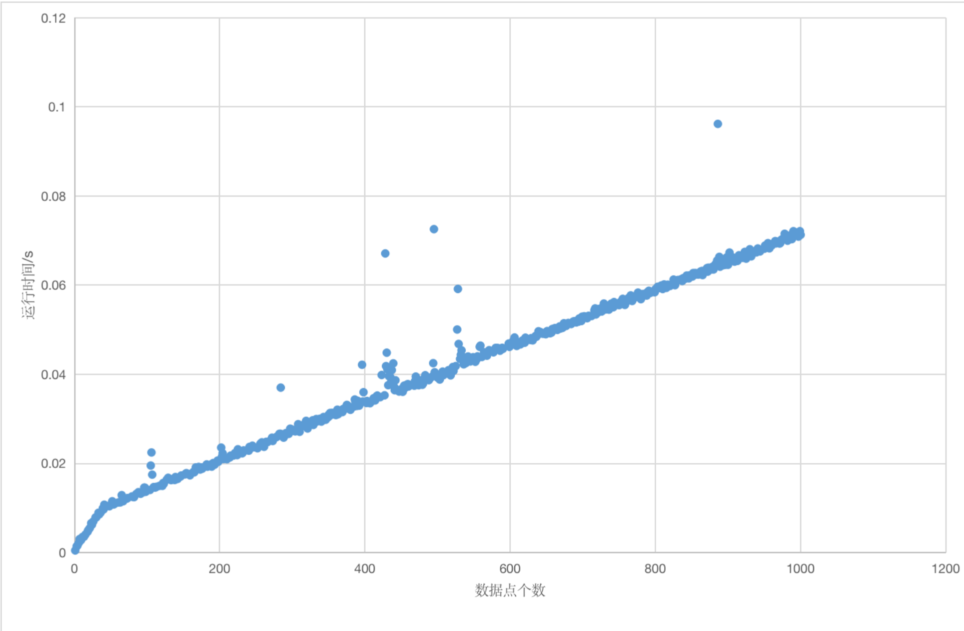

图 3: 运行时间与总金额关系图

有图可知，运行时间与总金额关系曲线大致为两段斜率不同的直线组成，前一段表明硬币数目较为充足，后一段表明硬币数目不充足，但均为直线，因此时间复杂度为 $O(N)$，与理论结果相同。

综上，可以得出对一般性的用例，时间复杂度为 $O(N)$。