



数据结构与算法 (Python版)

找零兑换问题的递归解法

陈斌 北京大学 gischen@pku.edu.cn

找零兑换问题：递归解法

- ❖ 我们来找一种**肯定**能找到最优解的方法
贪心策略是否有效依赖于具体的硬币体系
- ❖ 首先是确定**基本结束条件**，兑换硬币这个问题最简单直接的情况就是，需要兑换的找零，其**面值正好等于某种硬币**
如找零25分，答案就是1个硬币！



25¢

找零兑换问题：递归解法

❖ 其次是**减小问题的规模**，我们要对每种硬币尝试1次，例如美元硬币体系：

找零减去1分(penny)后，求兑换硬币最少数量（递归调用自身）；

找零减去5分(nikel)后，求兑换硬币最少数量

找零减去10分(dime)后，求兑换硬币最少数量

找零减去25分(quarter)后，求兑换硬币最少数量

上述4项中**选择最小**的一个。

$$\text{numCoins} = \min \begin{cases} 1 + \text{numCoins}(\text{originalamount} - 1) \\ 1 + \text{numCoins}(\text{originalamount} - 5) \\ 1 + \text{numCoins}(\text{originalamount} - 10) \\ 1 + \text{numCoins}(\text{originalamount} - 25) \end{cases}$$

找零兑换问题：递归解法代码

```
def recMC(coinValueList,change):  
    minCoins = change  
    if change in coinValueList:  
        return 1  
    else:  
        for i in [c for c in coinValueList if c <= change]:  
            numCoins = 1 + recMC(coinValueList,change-i)  
            if numCoins < minCoins:  
                minCoins = numCoins  
        return minCoins  
  
print(recMC([1,5,10,25],63))
```

最小规模，直接返回

调用自身

减小规模：
每次减去一种硬币面值
挑选最小数量

找零兑换问题：递归解法分析

❖ 递归解法虽然能解决问题，但其最大的问题是：**极！其！低！效！**

对63分的兑换硬币问题，需要进行67,716,925次递归调用！

在我这台笔记本电脑上花费了40秒时间得到解：
6个硬币

```
import time
```

```
print(time.clock())  
print(recMC([1, 5, 10, 25], 63))  
print(time.clock())
```

```
/Library/Frameworks/Python.framework
```

```
0.050163
```

```
6
```

```
40.899408
```



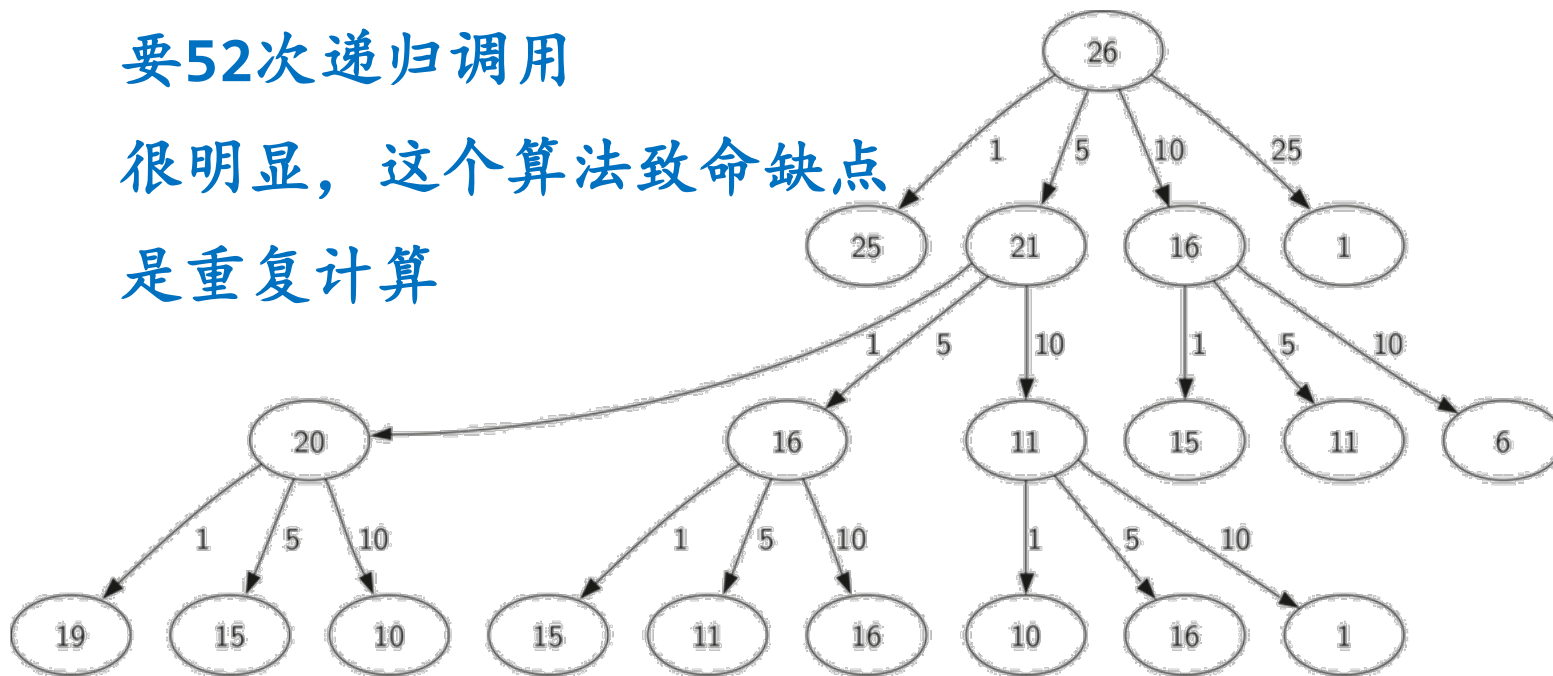
找零兑换问题：递归解法分析

❖ 以26分兑换硬币为例，看看递归调用过程 (377次递归的一小部分)

我们发现一个重大秘密，就是重复计算太多！

例如找零15分的，出现了3次！而它最终解决还要52次递归调用

很明显，这个算法致命缺点是重复计算



找零兑换问题：递归解法改进

❖ 对这个递归解法进行改进的关键就在于**消除重复计算**

我们可以用一个表将计算过的**中间结果保存**起来，在计算之前查表看看是否已经计算过

❖ 这个算法的中间结果就是**部分找零**的最优解，在递归调用过程中已经得到的最优解被记录下来

在递归调用之前，先查找表中是否已有**部分找零**的最优解

如果有，**直接返回**最优解而不进行递归调用

如果没有，才进行递归调用

找零兑换问题：递归解法改进代码

```
1 def recDC(coinValueList, change, knownResults):
2     minCoins = change
3     if change in coinValueList: # 递归基本结束条件
4         knownResults[change] = 1 # 记录最优解
5         return 1
6     elif knownResults[change] > 0:
7         return knownResults[change] # 查表成功, 直接用最优解
8     else:
9         for i in [c for c in coinValueList if c <= change]:
10             numCoins = 1 + recDC(coinValueList, \
11                                 change - i, knownResults)
12             if numCoins < minCoins:
13                 minCoins = numCoins
14                 # 找到最优解, 记录到表中
15                 knownResults[change] = minCoins
16     return minCoins
17
18 print(recDC([1, 5, 10, 25], 63, [0]*64))
```


找零兑换问题：递归解法改进

- ❖ 改进后的解法，极大减少了递归调用次数
对63分兑换硬币问题，仅仅需要221次递归调用
是改进前的三十万分之一，瞬间返回！

```
17  import time
18
19  memo = [0] * 64
20  print(time.clock())
21  print(recDC([1, 5, 10, 25], 63, memo))
22  print(time.clock())
23
24  print(memo)
```

↑	/Library/Frameworks/Python.framework
	0.046622
↓	6
	0.046793
↕	[0, 1, 0, 0, 0, 1, 2, 3, 4, 5, 1,

