A8参考题解

1 Coin changing (CLRS 15-1)

1.1 Greedy algorithm

贪心算法基本上大家都能想到，就是优先用更大面值的硬币来找零：

```python
# 版本一
def makechange1(n):
    quarters = n // 25
    n = n % 25

    dimes = n // 10
    n = n % 10

    nickels = n // 5
    pennies = n % 5

    return (quarters, dimes, nickels, pennies)

# 版本二
def makechange2(n):
    quarters = 0
    dimes = 0
    nickels = 0
    pennies = 0

    while n > 0:
        if n >= 25:
            quarters += 1
            n -= 25
        elif n >= 10:
            dimes += 1
            n -= 10
        elif n >= 5:
            nickels += 1
            n -= 5
        elif n >= 1:
            pennies += 1
            n -= 1

    return (quarters, dimes, nickels, pennies)
```

但是呢，这道题的证明就很难了喔😯可能是这道题最难的一小题了。很多同学都直接省略了这个证明，或者写了一堆看起来不太对的东西……

通常证明贪心算法最优会涉及到两点：

- 利用单调性
- 使用反证法

我们用一个比较笨但是通用的方法来证明。记：

$$\begin{aligned}&\text{denomination: } c_1=1, c_2=5, c_3=10, c_4=25\\&\text{solution given by the greedy algorithm: } x_1, x_2, x_3, x_4\\&\quad\text{such that } \textstyle\sum_{i=1}^4 x_i c_i = n\end{aligned}$$

直接证明贪心算法是正确的会非常困难，我们可以来证明一个局部的正确性：

引理：给定一组还没把钱凑齐的找零钱方案(y_1, y_2, y_3, y_4)使得：

$$\sum_{i=1}^{4} y_i c_i < n, y_i \geq 0$$

假设k为目前能凑的最大面额的硬币，那么在包含当前这组找零钱的方案中的解中，最优解一定会选择k。即：

$$\forall (x_1, x_2, x_3, x_4) \text{ such that } x_i \geq y_i \text{ and } \sum_{i=1}^{4} x_i c_i = n$$
$$\text{the optimal answer } (x_1^*, x_2^*, x_3^*, x_4^*) \text{ satisfies that } x_k^* \geq y_k + 1$$

证明：我们使用反证法，假设最优解不选择k。分情况讨论：

1. $k=1$，此时$1 \leq n - \sum_{i=1}^4 y_i c_i < 5$：该种情况只能选择$k$，故矛盾
2. $k=2$，此时$5 \leq n - \sum_{i=1}^4 y_i c_i < 10$：

   假如最优解中$x_2^* = y_2$，也就是没有选取c_2，仅选取了c_1。此时选取的c_1有$n - \sum_{i=1}^4$个，我们可以构造一组解：$(x_1^* - 5, x_2^* + 1, x_3^*, x_4^*)$满足要求且更优，故推出矛盾
3. $k=3$，此时$10 \leq n - \sum_{i=1}^4 y_i c_i < 25$：

   假如最优解中$x_3^* = y_3$，也就是没有选取c_3，仅选取了c_1和c_2。为了凑齐$n - \sum_{i=1}^4 y_i c_i$，只有以下几种可能：

   - $x_2^* - y_2 \geq 2$，即至少选取了2个c_2，此时$(x_1^*, x_2^* - 2, x_3^* + 1, x_4)$为更优解，矛盾
   - $x_2^* - y_2 \geq 1, x_1^* - y_1 \geq 5$，即至少选取了1个$c_2$，5个$c_1$，此时$(x_1^*, x_2^* - 1, x_3^* - 5, x_4^*)$为更优解，矛盾
   - $x_1^* - y_1 \geq 10$，即至少选取了10个c_1，此时$(x_1^*, x_2^*, x_3^* - 10, x_4^*)$为更优解，矛盾
4. $k=4$，此时$n - \sum_{i=1}^4 y_i c_i \geq 25$：

   同理枚举每种情形（$\binom{3}{1} + \binom{3}{2} + \binom{3}{3} = 2^3 - 1$种可能，可以对一些情形进行合并）即可，此处省略

一旦证明了该局部最优的结构，我们就可以每次都选取那个最大面值的硬币。

1.1.1 同学解答1

1. We always use larger coins as many as possible. Repeat until we have no more coins to change.

Now we prove that the algorithm is optimal.

证明. Suppose we have n cents to change, coin with value v cents is the largest to make a change. We use k coins(at most) with value v cents, then we have $n - kv$ cents to change. We will show that these k coins will be used in the optimal solution, S.

We suppose to the contrary that there exists a optimal solution S' using less than k coins. It is clear that S' cannot use any coin with value larger than v cents, which is contrary to the premise. Thus, there exists some coin with value v' cents, which is less than v cents, used more S' than in S. However, we have known that $2v' \leq v$. Thus if we reduce the number of coin with value v by 1, the number of other coins will increase by at least 2, which is certainly not the optimal solution. □

2. Since $c > 1$ and $c \in \mathbb{N}$, the condition $2v' \leq v$ still holds. Thus the proof is still valid.

这个证明的问题在于一次仅考虑两枚硬币，这种情况下不一定是对的，比方说如果硬币面值是$11, 5, 1$。尽管满足条件，但是这种情况下贪心不一定对。如果要凑15块钱，贪心给出的是$11, 1, 1, 1, 1$，而最优则是$5, 5, 5$。

1.1.2 同学解答2

反证法

假设本题最优解为a_1个25美分，b_1个10美分，c_1个5美分，d_1个1美分

而该算法的解a_2个25美分，b_2个10美分，c_2个5美分，d_2个1美分不是最优解

即

$25a_1 + 10b_1 + 5c_1 + d_1 = 25a_2 + 10b_2 + 5c_2 + d_2$

$a_1 + b_1 + c_1 + d_1 < a_2 + b_2 + c_2 + d_2$

因为若$d_1 \geq 5$，则可以改为将c_1加1而d_1减5，此时更优，因此$d_1 < 5$，显然d_2也 < 5

故$d_1 = d_2 = n\%5$

$\therefore 5a_1 + 2b_1 + c_1 = 5a_2 + 2b_2 + c_2$

进而$4a_1 + b_1 > 4a_2 + b_2$

因为a_2取的是可能的最大值

若$a_1 = a_2$，则$b_1 < b_2, c_1 > c_2$与不等式不符

若$a_1 < a_2$，则$a1$每比a_2小1，b_1至多比b_2大2.5，同样无法满足不等式

故该算法的解即为最优解

这是一个比较聪明且高效的反证法。

1.1.3 同学解答3

To prove the correctness of the algorithm, we need to prove that to make change for n cents, a largest official value coin that is less or equal to n *(largest coin)* is in all optimal choices. Assume that the coin isn't in any optimal choices:

1. n<5 : the largest coin is 1 and is trivially in any optimal choices.
2. 5≤n<10 : the largest coin is 5 and is not in any optimal choices. Therefore, there must exists more than 5 pennies, which can be substituted by a nickel, then these choices excluding nickels can't be optimal, hence making a contradiction.
3. 10≤n<25 : the largest coin is 10 and is not in any optimal choices. Therefore, there must exists at least 2 nickels or 10 cents, which can be substituted by a dime, then these choices excluding nickels can't be optimal, hence making a contradiction.
4. n≥25 : the largest coin is 25 and is not in any optimal choices. Therefore, there must exists at least 2 dimes, 5 nickels or 25 cents, which can be substituted by a quarter, then these choices excluding nickels can't be optimal, hence making a contradiction.

Hence, the greedy choice makes sense, we just need to pick a largest coin each turn.

这位同学的讨论看起来很简短，细看发现在枚举情况的时候有漏，比方说第三个case中"*there must exists at least 2 nickels or 10 cents*(此处应为*Pennies*)"，这里就考虑少了1个nickel加5个pennies的情况。

1.2 Denominations that are powers of c

这道题有一个比较方便的做法就是用c进制来做。我们假设用贪心算法得出来的解是：

$$(x_0, x_1, \ldots, x_k), \text{where} \sum_{i=0}^{k} c^i x_i = n \text{ and } x_i < c \text{ for } i = 0, 1, \ldots, k-1$$

此时n的c进制表示刚好就是$(x_k x_{k-1} \ldots x_1 x_0)_c$。假设存在其它最优解$(x'_0, x'_1, \ldots, x'_k)$，由$n$的$c$进制唯一可知必然存在一个$0 \le j \le n-1$使得$x'_j \ge c$。此时我们直接进位得到另外一个解：

$$x'_j := x'_j - c$$
$$x'_{j+1} := x'_{j+1} + 1$$

此时总硬币个数少了$c - 1 > 0$，这与$(x'_0, x'_1, \ldots, x'_k)$为最优解矛盾。故贪心算法解出来的解是最优解。

1.3 Counterexample

这个太简单了，略

1.4 $O(nk)$-time algorithm

我们可以用动态规划解该问题（只要是有上课的同学基本都会）：

$$\begin{aligned}
&k \text{ different coin denominations} : 1 = c_1 < c_2 < \cdots < c_k \\
&f(n,k) : \text{minimum coins needed to make changes for } n \text{ cents using } c_1, c_2, \ldots, c_k \\
&\text{base case: } f(n,1) = n \\
&\text{recurrence: } f(n,k) = \min\{f(n,k-1), f(n-c_k,k)\}, k > 1
\end{aligned}$$

2 Yen's improvement to Bellman-Ford (CLRS 22-1)

2.1 G_f and G_b

这道题害挺简单的，简单写一下G_f的证明，G_b同理就不写了：

- 要证明G_f是无环的，等价于证明：

  $$\forall u, v \in V. \exists \text{a path from } u \text{ to } v \Rightarrow \nexists \text{a path from } v \text{ to } u$$

  不妨假设v_i到v_j有路径$\{e_1, \ldots, e_m\}$，且v_j到v_i也有路径$\{e_{m+1}, \ldots, e_n\}$。那么v_i可以通过$\{e_1, \ldots, e_m, e_{m+1}, \ldots, e_n\}$回到自身。由于$e_1, \ldots, e_m, e_{m+1}, \ldots, e_n \in E_f$，故：

  $$\forall v_s v_t \in \{e_1, \ldots, e_m, e_{m+1}, \ldots, e_n\}. s < t$$

  由单调性得到：

  $$i < i$$

  矛盾，故u到v的路径与v到u的路径不能同时存在，得证。

- 证明$\langle v_1, v_2, \ldots, v_{|V|} \rangle$为$G_f$的一个拓扑排序，等价于证明

  $$\forall v_i v_j \in G_f. i < j$$

  而此为已知，故得证

2.2 Correctness of this algorithm

由于G没有负环，故对于任意从s可达的节点v，s到v存在最短路径且长度小于等于$|V|$。不妨记这条路径为$p = (e_1, e_2, \ldots, e_m)$, $m \leq n$. 此时我们将路径p中相邻且相同方向（此处相同方向指顶点编号要么都是递增，要么都是递减）的边进行合并，即如果$e_i, e_{i+1} \in E_f$或者$e_i, e_{i+1} \in E_b$，我们合并e_i, e_{i+1}：

$$(e_1, e_2, \ldots, e_i, e_{i+1}, \ldots, e_m) \rightarrow (e_1, e_2, \ldots, e_{i-1}, p_{i,i+1}, e_{i+2}, \ldots, e_m)$$

重复合并直到任意两条相邻的路径是反向的，此时我们得到（对子路径名进行重新编排）：

$$p = (p_1, p_2, \ldots, p_k)$$

我们知道$k \leq |V|$，而在DAG中按照拓扑排序可以直接得到最短路径。

如果$p_1 \in E_f$，那么算法运行如下：

- 1st pass：p_1, p_2
- 2nd pass：p_3, p_4
- ...

如果$p_1 \in E_b$，那么算法运行如下：

- 1st pass：p_1
- 2nd pass：p_2, p_3
- ...

综合两种情况，需要$\lceil |V|/2 \rceil$就能把$p_1, p_2, \ldots, p_k$全部松弛。

2.3 Asymptotic running time

由于要做$\lceil |V|/2 \rceil$个pass，每个pass要遍历所有边，也就是$|E|$，故渐进复杂度还是$O(|E||V|)$.