Homework04 2021.11.07

1

假定你希望兑换外汇，你意识到与其直接兑换，不如进行多种外币的一系列兑换，最后兑换到你想要的那种外币，可能会获得更大收益。假定你可以交易 n 种不同的货币，编号为 $1, 2, \ldots, n$ ，兑换从 1号货币开始，最终兑换为 n 号货币。对每两种货币i和j，给定汇率r_{ij}。意味着你如果有 d 个单位的货币 i ，可以兑换 dr_{ij} 个单位的货币 j 。进行一系列的交易需要支付一定的佣金，金额取决于交易的次数。令c_k表示k次交易需要支付的佣金。证明：如果对所有 $k = 1, 2, \ldots, n, c_k = 0$，那么寻找最优兑换序列的问题具有最优子结构。然后请证明：如果佣金 c_k 为任意值，那么问题不一定具有最优子结构

- 假设S_{ij}为从货币i到货币j的兑换序列，希望求的是这样一个序列使得最终对换得到的货币j最多。假设A_{ij}就是这样一个序列，包含货币k。由于最优解包含货币k，我们得到两个子问题：寻找从货币i到货币k的最优兑换序列A_{ik}，以及从货币k到货币j的最优兑换序列A_{kj}。而A_{ij}必然包括这两个子问题的最优解，因为假设可以找到一个兑换序列A'_{ik}，使A'_{ik}比A_{ik}兑换得到的货币k多，则可以将A'_{ik} 而不是A_{ik}作为最优解的一部分，这样就构造出一个兑换序列A'_{ij}，最终兑换得到的货币j比A_{ij}多，这与A_{ij}是最优序列矛盾。因此有$A_{ij} = A_{ik} \cup A_{kj}$。

  因此$c_k = 0$时，寻找最优兑换序列的问题具有最优子结构。

- 而在考虑佣金c_k的情况下，最优解的结构就与兑换的次数相关，此时原问题的最优解就不一定包含其子问题的最优解。举一个例子，$n = 5$，兑换矩阵为

  $$\begin{bmatrix} 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

  $c_k = [0, +\infty, +\infty, 0]$

  则从货币1到货币5的最优兑换序列为$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$，包含子问题从货币1到货币3，但是从货币1到货币3的最优兑换序列为$1 \rightarrow 3$，故原问题的最优解就不一定包含其子问题的最优解。

  因此在考虑佣金c_k的情况下，问题不一定具有最优子结构。

2

设计一个高效的算法，对实数线上给定的一个点集$x_1, x_2, \ldots, x_n$, 求一个单位长度闭区间的集合，包含所有给定的点，并要求此集合最小。证明你的算法是正确的。

算法思想：首先对点集进行升序排序得到$y_1, y_2, \ldots, y_n$，借助贪心的思想，从左到右进行区间遍历。属于此单位区间内的点即可消去。

算法伪代码：

```
y = sort(x)
n = y.length
num = 0          //区间的数量
for i = 1 to n
    num++
    up = y[i] + 1
    while y[i] <= up
        i++      //向后遍历，是否落入当前的单位区间
return
```

正确性证明：

证明算法的贪心选择性质，即存在一个最优解，数轴上"最左"的单位闭区间为$[x, x+1]$，其中x是点集中最小的数。

假设对于任意一个最优解，

- 如果已经包含了$[x, x+1]$，则证明结束；
- 如果不包含$[x, x+1]$，设这个最优解的"最左"的单位闭区间为$[y, y+1]$，显然$y < x$（因为要包含x），从而$y+1 < x+1$，又因为x是点集中最小的点，所以用$[x, x+1]$包含了$[y, y+1]$中所有点集中的数，得到的仍然是一个最优解。

3

一位公司主席正在向 Stewart 教授咨询公司聚会方案。公司的内部结构关系是层次化的，即员工按主管-下属关系构成一棵树，根结点为公司主席。人事部按"宴会交际能力"为每个员工打分，分值为实数。为了使所有参加聚会的员工都感到愉快，主席不希望员工及其直接主管同时出席。
公司主席向 Stewart 教授提供公司结构树，采用左孩子右兄弟表示法（参见课本 10.4 节）描述。每个节点除了保存指针外，还保存员工的名字和宴会交际评分。设计算法，求宴会交际评分之和最大的宾客名单。分析算法复杂度。

算法思想：

对于以p为根的子树，假设p有k个子节点分别是$p_{cd1}, p_{cd2}, \ldots, p_{cdk}$，p的交际评分为$score[p]$。根据p是否参会，可以分为两种情况。若p不参加宴会，则设以p为根的子树的最大交际评分之和为$s_0[p]$；若p参加宴会，则设以p为根的子树的最大交际评分之和为$s_1[p]$

- p不参加宴会

  在这种情况下，p的任意一个子节点既可以选择参加宴会（$attend[p_{cd}] = 1$），也可以选择不参加宴会（$attend[p_{cd}] = 0$），因此有$s_0[p] = \sum_{i=1}^{k} max\{s_0[p_{cdi}], s_1[p_{cdi}]\}$
- p参加宴会

  在这种情况下，p的所有子节点都不会参加宴会，因此有$s_1[p] = score[p] + \sum_{i=1}^{k} s_0[p_{cdi}]$

算法伪代码：

```
COMPANY-PARTY(root, score)
    NewArray s0[] and s1[]
    NewArray attend[]
    COMPANY-PARTY-RECURSION(root, score, s0, s1, attend)
    s = max(s0[r], s1[r])
    return s and attend

COMPANY-PARTY-RECURSION(p, score, s0, s1, attend)
```

```
    if p == NULL
        return
    else
        s0[p] = 0
        s1[p] = score[p]
        for i = 1 to p.child_num
            COMPANY-PARTY-RECURSION(p.childi, score, s0, s1, attend)
            s0[p] += max(s0[p.childi], s1[p.childi])
            s1[p] += s0[p.childi]
        if(s0[p]>s1[p])
            attend[p] = 0
        else
            attend[p] = 1

COMPANY-PARTY-PRINT(p, attend, parent_attend)
    if p == NULL
        return
    if parent_ttend == 0
        cur_attend = attend[p]
        if cur_attend == 1
            print p
    else
        cur_attend = 0
    for i = 1 to p.child_num
        COMPANY-PARTY-PRINT(p.childi, attend, cur_attend)
```

算法分析:

上述算法对树的每一个结点都遍历了一次，且对每个结点的处理也是常数时间。因此算法的时间复杂度为$\Theta(n)$

4

考虑用最少的硬币找 n 美分零钱的问题。假定每种硬币的面额都是整数。设计贪心算法求解找零问题，假定有 25 美分、 10 美分、 5 美分和 1 美分四种面额的硬币。证明你的算法能找到最优解。

算法思想： 设可供找零的硬币序列为$c_1, c_2, ..., c_m$（本题中$m = 4$)

设找零的硬币数为n，找零的硬币序列为$i_1, i_2, ..., i_n$，对于i_k $k \in (1, n)$，有

$i_k = max\{c_i | c_i \leq total - \sum_{l=1}^{k-1} i_l\}$ $k \in (1, n)$

即每一步找零都尽可能取最大的面额。

正确性证明:

取最优解中与上述算法得到的找零序列$i_1, i_2, ..., i_n$从第一个元素起连续相同元素最多的最优解序列$j_1, j_2, ..., j_m$，此处$m \leq n$。那么存在r，当满足$l < r$时，都有$i_l = j_l$，而$i_{l+1} \neq j_{l+1}$。

令$total'$为在第$r+1$次找零之前，剩余需要找零的面额，由于贪心算法会选择尽量大的面额来找零，所以一定有$i_{r+1} > j_{r+1}$

- 在剩余的$j_{r+1}, ... j_m$中，若存在$j_l = i_{r+1}$；那么互换j_l与j_{r+1}，这样既不影响最优解的正确性，同时使得调换后的新序列与贪心算法序列有更长的相同子序列。这与取出的最优解是最长的相同序列的假设不符。故在剩余的最优解子序列。因此在剩余的$j_{r+1}, ... j_n$中，任意$j_l \neq i_{r+1}$
- 在剩余的$j_{r+1}, ... j_m$中，若存在$j_l > i_{r+1}$；那么i_{r+1}就不是小于$total'$的最大可选找零面值，与贪心算法的思想不符。因此在剩余的$j_{r+1}, ... j_n$中，任意$j_l < i_{r+1}$

另外，在最优解序列中，面额5最多只有1个，否则可用面额10来代替。同理有，面额1最多有4个；面额10最多两个；面额25数量不限。从而我们发现，只使用面额1的最优解，最多能表示4；只使用面额1、5的最优解，最多能表示9；只使用面额1、5、10的最优解，最多能表示24。即：在本题的币值中，不使用面额大于等于c的硬币就无法表示大于等于c的面额。

综上可以推出：$i_{r+1} > \sum_{l=r+1}^{m} j_l$。这与两者剩余值都为$total'$矛盾

因此，这样的r不存在，必然存在一个最优解序列与贪心算法得到的序列相同。