算法作业第五章：贪心算法

> 梁鑫嵘 200111619

1. （30分）在任务安排问题中，假定我们不再一直选择最早结束的活动，而是选择最晚开始的活动，前提是仍然与之前选出的所有活动兼容。描述如何利用这一方法设计贪心算法，并**证明**算法会产生最优解。

   如何利用这一方法设计贪心算法：

   1. 将所有活动按照开始时间从大到小排序。
   2. 每次选择未选择的活动中开始时间最晚的那一个活动，并且使得其与之前选出的活动兼容。
   3. 取完满足条件的所有活动，则得到一个最优解。

   伪代码：

   > 输入：开始时间数组S，结束时间数组F，其中假设$s_1 \geq s_2 \geq \cdots s_n$已经排好序。

$$
\begin{aligned}
&GREEDY\_ACTIVITY\_SELECTOR(S, F) \\
&n \leftarrow S.length \\
&A \leftarrow 1 \\
&j \leftarrow 1 \\
&\textbf{for } i \leftarrow 2 \textbf{ to } n \textbf{ do} \\
&\quad \textbf{if } f_i \leq s_i \textbf{ then} \\
&\qquad A \leftarrow A \cup \{i\} \\
&\qquad j \leftarrow i \\
&\textbf{return } A
\end{aligned}
$$

   证明这个算法具有最优解：

   引理1： 设$S = \{1, 2, \cdots, n\}$是n个活动集合，$[s_i, f_i]$为活动起止时间，而且$s_1 \geq s_2 \geq \cdots \geq s_n$。$S$的活动选择问题的某个优化解包括活动1。

   引理1证明：

   设A为一个优化解，按照**开始时间从大到小**排序A中的活动，设第1个活动为k，第2个活动为j。

   1. 如果$k = 1$，引理成立。
   2. 如果$k \neq 1$，使$B = A - \{k\} \cup \{1\}$，则$f_j \leq s_k \leq s_1$，$\therefore B$中的活动也是相容的。
   3. $\because |B| = |A| \therefore B$是一个优化解，而且包括活动1，引理得证。

   引理2： 设$S = \{1, 2, \cdots, n\}$是n个活动集合，$[s_i, f_i]$为活动起止时间，而且$s_1 \geq s_2 \geq \cdots \geq s_n$。设$A$是$S$的调度问题的一个优化解而且包括活动1，则$A' = A - \{1\}$是$S' = \{i \in S | f_i \leq s_1\}$的调度问题的优化解。

   引理2证明：

   1. 显然，A'中活动是相容的。所以仅仅需要证明A'是最大的。
   2. 假设A'不是最大的，则存在一个S'的活动选择的优化解B'而且$|B'| > |A'|$。
   3. 令$B = \{1\} \cup B'$，对于$\forall i \in S', s_i \geq f_i, B$中的活动相容，$B$是$S$的一个解。
      $\because |A| = |A'| + 1, |B| = |B'| + 1 > |A'| + 1 = |A|$，与$A$最大矛盾。
   4. 引理2得证。

引理3：设$S=\{1,2,\cdots,n\}$是n个活动的集合，其中$s_1\geq\cdots\geq s_n, f_0=0, l_i$是集合$S_i=\{j\in S|f_j\leq s_{i-1}\}$中具有最小开始时间的活动。设$A$是$S$的包含活动1的优化解，则$A=\cup_{i=1}^{k}\{l_i\}$。

引理3证明：对$|A|$做数学归纳。

1. 当$|A|=1$，又引理1，命题成立。
2. 当$|A|<k$，命题成立。
3. 当$|A|=k$，由引理2，$A=\{1\}\cup A_1, A_1$是$S_2=\{j\in S|f_j\leq s_1\}$的优化解，由归纳假设得知$A_1=\cup_{i=1}^{k}\{l_i\},\therefore A=\cup_{i=1}^{k}\{l_i\}$。

由引理3，这个算法具有最优解。

2. （30分）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。
   1. 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4种面额的硬币。

      思路：

      1. $\because$ 面额越大的硬币，平均价格对应的硬币数量越少
      2. $\therefore$ 应该尽量选择面额大的硬币
      3. 算法表示为：$D_{i,1\leq i\leq 4}=\{25,10,5,1\}, S_{i,1\leq i\leq 4}$表示第$i$种面额选择的硬币数量，则$S_i=\lfloor\frac{n-\sum_{i=1}^{j-1}S_j\times D_j}{D_i}\rfloor$

      伪代码：

$$
\begin{aligned}
&GIVE\_CHANGE(D,n)\\
&S\leftarrow[]\\
&left\leftarrow n\\
&\textbf{for } i\leftarrow 1 \textbf{ to } D.length \textbf{ then}\\
&\quad S_i\leftarrow\lfloor\frac{left}{D_i}\rfloor\\
&\quad left\leftarrow left-S_i\times D_i\\
&\textbf{return } S
\end{aligned}
$$

   2. 设计一组硬币面额，使得贪心算法不能保证的到最优解。这组硬币面额中应该包含1美分，使得对每个零钱值都存在找零方案。

      当$D=\{4,3,1\}, n=6$，如果使用贪心法，得到答案为：$[4,1,1]$，而最优解应该为：$[3,3]$。

3. （40分）编程题：柠檬水找零

题目描述：

在柠檬水摊上，每一杯柠檬水的售价为 5 美元。

顾客排队购买你的产品，（按账单 bills 支付的顺序）一次购买一杯。

每位顾客只买一杯柠檬水，然后向你付 5 美元、10 美元或 20 美元。你必须给每个顾客正确找零，也就是说净交易是每位顾客向你支付 5 美元。

注意，一开始你手头没有任何零钱。

如果你能给每位顾客正确找零，返回 true ，否则返回 false 。

提示：

`0 <= bills.length <= 10000`

`bills[i]` 不是 5 就是 10 或是 20

示例 1：

输入：`[5,5,5,10,20]`

输出：`true`

> 解释：
>
> 前 3 位顾客那里，我们按顺序收取 3 张 5 美元的钞票。
>
> 第 4 位顾客那里，我们收取一张 10 美元的钞票，并返还 5 美元。
>
> 第 5 位顾客那里，我们找还一张 10 美元的钞票和一张 5 美元的钞票。
>
> 由于所有客户都得到了正确的找零，所以我们输出 true。

示例 2：

输入：`[5,5,10]`

输出：`true`

示例 3：

输入：`[10,10]`

输出：`false`

示例 4：

输入：`[5,5,10,10,20]`

输出：`false`

> 解释：
>
> 前 2 位顾客那里，我们按顺序收取 2 张 5 美元的钞票。
>
> 对于接下来的 2 位顾客，我们收取一张 10 美元的钞票，然后返还 5 美元。
>
> 对于最后一位顾客，我们无法退回 15 美元，因为我们现在只有两张 10 美元的钞票。
>
> 由于不是每位顾客都得到了正确的找零，所以答案是 false。

要求：运用**贪心思想**作答，请写出**分析过程**，并用一种语言（最好是C++或JAVA）实现你的思路，上交作业时请将**代码一并提交**，代码粘贴在交作业的word里面，复杂度尽可能低。

思路：

1. 根据题目2，每次都从手上有的零钱中找到尽可能大的面值凑足找补。
2. 因为零钱的面额是固定的而且很少，所以我们可以只记录零钱面值的数量。
3. 当模拟找补零钱的过程发现找不到零钱的时候，直接返回`false`，否则到最后模拟完成后返回`true`。

代码：

```
bool lemonadeChange(vector<int> &bills) {
  int five = 0, ten = 0;
  for (const auto &x : bills) {
    int ret = x - 5;
    if (ret == 0) {
      five++;
    } else if (ret == 5) {
      ten++;
      if (five == 0) return false;
      five--;
    } else if (ret == 15) {
```

```
      if (five && ten) {
        five--;
        ten--;
      } else if (five >= 3) {
        five -= 3;
      } else
        return false;
    }
  }
  return true;
}
```