Lab03-GreedyStrategy

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2020.

∗ If there is any problem, please contact TA Shuodian Yu.
∗ Name:Yijia Diao Student ID:518030910146 Email: diao_yijia@sjtu.edu.cn

1. There are $n+1$ people, each with two attributes $(a_i, b_i), i \in [0, n]$ and $a_i > 1$. The i-th person can get money worth $c_i = \frac{\prod_{j=0}^{i-1} a_j}{b_i}$. We do not want anyone to get too much. Thus, please design a strategy to sort people from 1 to n, such that the maximum earned money $c_{max} = \max_{1 \le i \le n} c_i$ is minimized. (Note: the 0-th person doesn't enroll in the sorting process, but a_0 always works for each c_i.)

   (a) Please design an algorithm based on greedy strategy to solve the above problem. (Write a pseudocode)

   (b) Prove your algorithm is optimal.

Solution.

(a)

Algorithm 1: Min-Max Earn Money

Input: $(a_i, b_i), i \in [0, n]$.

Output: $P[0, \cdots, n]$ = people's sequence that the maximum earned money $c_{max} = \max_{1 \le i \le n} \frac{\prod_{j=0}^{i-1} a_j}{b_i}$ is minimized.

```
if n = 0 then return P[0] ;
cmp[1, ··· , n] ← 0;
P[0, ··· , n] ← [0, 1, 2, ··· , n];
for i = 1 to n do
    cmp[i] ← a_i × b_i;
Sort P[1, ··· , n] by cmp[1, ··· , n] non-decreasingly;
return P[0, ··· , n];
```

(b) **Proof.**(by Contradiction)
Suppose that the output of Alg.1 is not optimal. Thus, we suppose that the optimal solution is $OP[0, \cdots, n]$.

Definition 1. *$S[i]$ and $S[i+1]$ is an **inverse pair** iff $i \ge 1$ and $cmp_{S[i]} > cmp_{S[i+1]}$.*

Since the output of Alg.1 don't have inverse pair, there must be inverse pairs in $OP[1, \cdots, n]$, suppose they are $OP[i]$ and $OP[i+1]$, $\prod_{k=0}^{i-1} a_k = s$. So the maximum value of $c_{OP[i]}$ and $c_{OP[i+1]}$ is

$$max_{op} = \max\{\frac{s}{b_i}, \frac{s \times a_i}{b_{i+1}}\}$$

if we exchange $OP[i]$ and $OP[i+1]$,other person's money won't change. So the maximum earned money of the two is

$$max_{iop} = \max\{\frac{s}{b_{i+1}}, \frac{s \times a_{i+1}}{b_i}\}$$

Since $\forall i \ge 1$, $a_i \ge 1$, so

$$\frac{s}{b_{i+1}} \le \frac{s \times a_i}{b_{i+1}} \text{ and } \frac{s}{b_i} \le \frac{s \times a_{i+1}}{b_i} \tag{1}$$

Since $cmp_{OP[i]} > cmp_{OP[i+1]}$, which means $a_i b_i > a_{i+1} b_{i+1}$, we have

$$\frac{s \times a_i}{b_{i+1}} > \frac{s \times a_{i+1}}{b_i} \tag{2}$$

from inEqu.1 and inEqu.2, we have $max_{op} > max_{iop}$, which means that after exchanging the inverse pair of the optimal solution, the maximal earned money would decrease. So $OP[0, \cdots, n]$ is not optimal solution. Contradiction. So we can conclude that Alg.1 is the optimal solution.

□

2. **Interval Scheduling** is a classic problem solved by greedy algorithm and we have introduced it in the lecture: given n jobs and the j-th job starts at s_j and finishes at f_j. Two jobs are compatible if they do not overlap. The goal is to find maximum subset of mutually compatible jobs. Tim wants to solve it by sort the jobs in descending order of s_j. Is this attempt correct? Prove the correctness of such idea, or else provide a counter-example.

   Proof. Now prove that the attempt is correct, by contradiction.
   Suppose that Tim's solution is not optimal. Then we suppose that $i_1, i_2, \cdots, i_k (k \leq n)$ is the job selected by Tim's solution; the optimal solution is $j_1, j_2, \cdots, j_{k'} (k' > k)$ with $i_1 = j_1, i_2 = j_2, \cdots, i_r = j_r (r < k)$.
   Since $i_{r+1} \neq j_{r+1}$, and according to Tim's solution, $s_{i_{r+1}} > s_{j_{r+1}}$. If there are some other job(s) in $s_{j_{r+1}}$ and $s_{i_{r+1}}$, they are in Tim's solution, but they cannot in the optimal solution. So $k' \leq k$, contradiction.
   So we can conclude that Tim's solution is optimal. □

3. There are n lectures numbered from 1 to n. Lecture i has duration (course length) t_i and will close on d_i-th day. That is, you could take lecture i **continuously** for t_i days and must finish before or on the d_i-th day. The goal is to find the maximal number of courses that can be taken. (Note: you will start learning at the 1-st day.)

   Please design an algorithm based on greedy strategy to solve it. You could use the data structrue learned on Data Structrue course. You need to write pseudo code and prove its correctness.

   Solution. The algorithm is below:

Algorithm 2: Maximal Number of Courses

Input: $(t_i, b_i), i \in [1, n]$.
Output: $count$ = the maximal number of courses that can be taken.

```
1  C[1, ··· , n] ← [1, 2, ··· , n];
2  Sort C[1, ··· , n] by (d_i − t_i), i ∈ [1, n] non-increasingly;
3  ltime ← (d_{C[1]} − t_{C[1]});
4  if ltime ≤ 0 then return 0;
5  count ← 1;
6  if n > 2 then
7      for i = 2 to n − 1 do
8          if ltime − t_{C[i]} ≥ d_{C[i+1]} − t_{C[i+1]} > 0 then
9              count ← count + 1;
10             ltime ← ltime − t_{C[i]};
11 if n > 1 and ltime − t_{C[n]} > 0 then count ← count + 1 ;
12 return count;
```

Proof.(by Contradiction)
Assume that Alg.2 is not optimal.
Then we suppose that the course's sequence(ordered by start time non-increasingly)of Alg.2 is $i_1, \cdots, i_k (k \leq n)$, the optimal solution's is $j_1, \cdots, j_{k'} (k' > k)$; and we have $i_1 = j_1, \cdots, i_r = j_r (r < k)$.
According to Alg.2, we arrange the latest start course before the already arranged. So we have $begintime_{i_{r+1}} > begintime_{j_{r+1}}$. Since there may be courses that could be arranged between time $begintime_{j_{r+1}}$ and $begintime_{i_{r+1}}$, so $k \geq k'$, contradiction.
We can conclude that Alg.2 is optimal. □

4. Let $S_1, S_2, \ldots, S_n$ be a partition of S and $k_1, k_2, \ldots, k_n$ be positive integers. Let $\mathcal{I} = \{I : I \subseteq S, |I \cap S_i| \leq k_i \text{ for all } 1 \leq i \leq n\}$. Prove that $\mathcal{M} = (S, \mathcal{I})$ is a matroid.

Proof.

(1) **hereditary**:
Since $\forall B \in \mathcal{I} \Leftrightarrow \forall I : I \subseteq S$ and $|I \cap S_i| \leq k_i$ for all $1 \leq i \leq n$, and $\forall J \subseteq I : |J \cap S_i| \leq |I \cap S_i|$, we have:

$$\forall J \subseteq I : J \subseteq S \text{and} |J \cap S_i| \leq k_i \text{ for all } 1 \leq i \leq n$$

which is equivalent to $\forall A, \forall B : A \subset B, B \in \mathcal{I} \Rightarrow A \in \mathcal{I}$

(2) **Exchange property**:
Assume $\forall A \in \mathcal{I}, \forall B \in \mathcal{I}$ and $|A| < |B|$, for all $1 \leq i \leq n$, we discuss by cases below:

i. $(B \backslash A) \subseteq S_i$:
we have $|(B \backslash A) \cap S_i| = |S_i| \leq k_i$. So $\exists x \in B \backslash A, |(A \cup \{x\}) \cap S_i| \leq |S_i| \leq k_i$.

ii. $(B \backslash A) \not\subset S_i$:
$\exists x_0 \in (B \backslash A) \backslash S_i : x_0 \in B \backslash A,$ and $|(A \cup \{x_0\}) \cap S_i| = |A \cap S_i| + |x_0 \cup S_i| = |A \cap S_i| \leq k_i$

So we have $\exists x \in B \backslash A, |(A \cup \{x\}) \cap S_i| \leq |S_i| \leq k_i$. Since$A \in S$ and $B \in S \Rightarrow A \cup \{x\} \in S$, we have $A \cup \{x\} \in \mathcal{I}$.

So we can conclude that $\mathcal{M} = (S, \mathcal{I})$ is a matroid. □

Remark: You need to include your .pdf and .tex files in your uploaded .rar or .zip file.