Lab03-GreedyStrategy

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

∗ If there is any problem, please contact TA Shuodian Yu.
∗ Name:Hanzhang Yang Student ID:518030910022 Email: linqinluli@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.

   (a).

Input: Array $A[a_0, a_1, \cdots, a_n]$, array $B[b_0, b_2, \cdots, b_n]$

Output: sorted array that makes max $c_i = \frac{\prod_{j=0}^{i-1} a_j}{b_i}$ minimized

1 $M = [1, 2, \cdots, n]$;
2 Sort M by $A[M[i]] \times B[M[i]]$ so that
$A[M[1]] \times B[M[1]] \le A[M[2]] \times B[M[2]] \le \cdots \le A[M[n]] \times B[M[n]]$;
3 **Output** M;

(b).

Solution. Consider two people next to each other. (a_i, b_i), (a_{i+1}, b_{i+1}).

Assume it's best when a_i is before a_{i+1}. We can find that exchanging a_i and a_{i+1} won't affect other people.

$S = \prod_{j=0}^{i-1} a_j$

case 1 : a_i is before a_{i+1}

$c_i = \frac{S}{b_i}$

$c_{i+1} = \frac{S \times a_i}{b_{i+1}}$

case 2 : a_{i+1} is before a_i

$c_{i+1} = \frac{S}{b_{i+1}}$

$c_i = \frac{S \times a_{i+1}}{b_i}$

we can get

$$max\{\frac{S}{b_i}, \frac{S \times a_i}{b_{i+1}}\} \le max\{\frac{S}{b_{i+1}}, \frac{S \times a_{i+1}}{b_i}\}$$

$\frac{S}{b_i} \le \frac{S \times a_{i+1}}{b_i}$, $\frac{S \times a_i}{b_{i+1}} \ge \frac{S}{b_{i+1}}$

Thus $\frac{S \times a_i}{b_{i+1}} \le \frac{S \times a_{i+1}}{b_i} \quad \rightarrow a_i b_i \le a_{i+1} b_{i+1}$

Therefore it's best for all people to sort by $a_i \times b_i$ □

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.

   Yes.

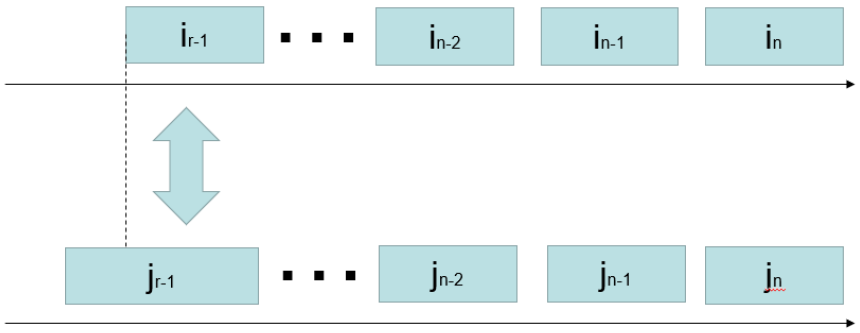
   Proof. (By contradiction) Assume greedy is not optimal.

   Let $i_n, i_{n-1}, \cdots, i_k$ denote set of jobs selected by greedy.

   Let $j_n, j_{n-1}, \cdots, j_m$ denote set of jobs in an optimal solution with $i_j = j_n, \cdots, i_r = j_r$ for the largest possible value of r.

   The i_{r-1} is selected by Greedy, so job i_{r-1} begins after j_{r-1}.

   Thus we change j_{r-1} to i_{r-1}. The solution of OPT is still feasible and optimal, but contradicts the maximality of r.



   Therefore we can conclude that greedy 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. By induction.

   Basis step. When $n = 1$, greedy=opt.

   Induction Hypothesis. Assume when $n = k$ greedy is opt.

   Proof of Induction Step. For the $k + 1$ lecture, if $time + t_i \leq d_i$. Greedy increases 1 the same as opt; When $time + t_i > d_i$, greedy choose smaller one between the lecture owning the largest time of all selected lectures and the $k + 1_{th}$ lecture. (Assume the largest one is on the m_{th}.) If exchange happened, since $d_{k+1} \leq d_m$ and $t_{k+1} < t_m$, number of lectures won't change and the *time* is smaller than before. Thus no matter whether the exchange would happen, the number and the time is sitll the best case.

   Therefore greedy is opt. □

Input: $A[(t_1, d_1), (t_2, d_2), \cdots, (t_n, d_n)]$
Output: maximal number of courses that can be taken

```
1  sort A that d_1 ≤ d_2 ≤ ··· ≤ d_n
2  creat a max-heap M
3  M.push(A[1]);
4  time = t_1;
5  num = 1;
6  for i = 2 to n do
7      if time + t_i > d_i then
8          if M.top().t > t_i then
9              time = time − M.top().t + t_i;
10             M.pop();
11             M.push(A[i]);
12             num++;
13         else
14             continue;
15     else
16         num + +;
17         time = time + t_i
18         M.push(A[i]);
19 output num;
```

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. Hereditary:

Assume $A \subset B, B \in \mathcal{I}$.

Since $A \subset B$,$B \subseteq S, |B \cap S_i| \leq k_i$ for all $1 \leq i \leq n$, $A \subseteq S, |A \cap S_i| \leq k_i$ for all $1 \leq i \leq n$

Thus $A \in \mathcal{I}$

Exchange Property:

Assume $A, B \in \mathcal{I}$, $|A| < |B|$ and $x \in B \backslash A$.

For $A \subseteq S,\ B \subseteq S$, $A \cup \{x\} \subseteq S$.

case 1: $B \subseteq S_i$, $k_i \geq |B|$

Thus $|A \cup \{x\}| = |A| + 1 \leq |B| \leq k_i$, $A \cup \{x\} \in \mathcal{I}$.

case 2: $B \nsubseteq S_i$

Make $x \in B \backslash S_i$. Thus $|A \cup \{x\}| = |A| \leq k_i$, $A \cup \{x\} \in \mathcal{I}$.

$\mathcal{M} = (S, \mathcal{I})$ is a matroid.

□

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