Lab03-GreedyStrategy

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

∗ If there is any problem, please contact TA Shuodian Yu.

∗ Name:Haotian Xue   Student ID:518021910506   Email: xavihart@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 $c_{max} = \max\limits_{1 \le i \le n} c_i$ 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

Input: The pair list $T = \{(a_0, b_0), (a_1, b_1), ..., (a_n, b_n)\}$
Output: $min\ c_{max}$

$min = +\infty$, $mul = 1$
Sort pair list T by $a_i \times b_i$ in an increasing order.
for $i = 1$ *to* n **do**
  $tmp = \frac{mul}{b_i}$
  if $tmp < min$ **then**
    $min = tmp$
  $mul = mul \times a_i$
return min

Proof. We can proof the greedy algorithm by contradiction.

If the optimal answer doesn't satisfy the increasing order of $a_i \times b_i$, we think about the adjacent pair, normally T_i, T_{i+1} for $i \in \{0, 1, ..., n-1\}$, where $T_i = (a_i, b_i)$ and $T_{i+1} = (a_{i+1}, b_{i+1})$.

Then in the optimal order, there exists an integer i, such that $a_i b_i > a_{i+1} b_{i+1}$. We note $\prod_{j=0}^{i-1} a_j = p$, then we can calculate the costs: $c_i = \frac{p}{b_i}$, $c_{i+1} = \frac{pa_i}{b_{i+1}}$. From $a_i > 1$ we can get:

$$max\{c_i, c_{i+1}\} = max\{\frac{p}{b_i}, \frac{pa_i}{b_{i+1}}\} = \frac{p \times max\{a_i b_i, b_{i+1}\}}{b_i b_{i+1}}$$

If we reverse the order of T_i and T_{i+1}, we can get(It is obvious that only c_i and c_{i+1} is changed):

$$max\{c_i^{'}, c_{i+1}^{'}\} = max\{\frac{p}{b_{i+1}}, \frac{pa_{i+1}}{b_i}\} = \frac{p \times max\{a_{i+1} b_{i+1}, b_i\}}{b_i b_{i+1}}$$

Since $a_i b_i > a_{i+1} b_{i+1} > b_{i+1}$ and $b_i < b_i a_i$, so:

$$max\{a_{i+1} b_{i+1}, b_i\} < a_i b_i = max\{a_i b_i, b_{i+1}\}$$

Which is contradictory with the assumption. So we can conlcude that the optimal order will have the T sorted with the methods in the algorithm. □

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. Tim's algorithm is right, we can proof it by assuming it is not optimal.

   We define the job sequence in Tim's algorithm $T = T_1, T_2, ..., T_k$, and the sequence of the optimal method is $R = R_1, R_2, ..., R_o$, and r is the largest number that $R_i = T_i, i \in \{1, 2, ..., r\}$ and $R_{r+1} \neq T_{r+1}$, where T_i is a tuple with $T_i[0]$ is the start time and $T_i[1]$ the end time.

   Since in Tim's algorithm, $T_i[0] > T_j[0]$ for $i < j$. So we have $T_i[0] < R_i[0]$. If we replace R_{r+1} with T_{r+1} in the optimal result, it actually doesn't influence the rest arrangement $R_{r+2}, R_{r+3}, ..., R_o$. And this is contradictory with the definition of r. In that way, Tim got the right algorithm. m □

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.

4. Let $S_1, S_2, \dots, S_n$ be a partition of S and $k_1, k_2, \dots, k_n$ 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. To prove that $\mathcal{M} = (S, \mathcal{I})$ is a matroid, we need to prove the hereditary and exchange property of it.

   Hereditary: For any $A \subset B$, $B \in \mathcal{I}$. Since $B \subset S$, then $A \subset S$. For $i \in \{1, 2, ..., n\}$ we have $|B \cap S_i| \leq k_i$, note that $|B \cap S_i| > |A \cap S_i|$, so $|A \cap S_i| \leq k_i$. Then we get $A \in \mathcal{I}$.

   Exchange Property:For $A, B \in \mathcal{I}$, $|A| < |B|$, firstly we divide A by the partition of S, namely: $A = A_1 \cup A_2 \cup ... \cup A_n$, where $A_i \cap A_j = \emptyset$ for $i \neq j \in \{1, 2, ..., n\}$, $A_i \cap S_j = \emptyset$ for $i \neq j$, which means $A_i \cap S_i$.

   In the same way, we get a partition of B: $B = B_1 \cup B_2 \cup ... \cup B_n$. Since $|B| > |A|$, there must exist i, such that: $k_i \geq |B_i| > |A_i|$. So we can find $x \in B$, $x \notin A$, such that $|A'| = |A_i \cup \{x\}| \leq k_i$, so $|A' \cap S_i| = |A_i| \leq k_i$, which satisfy the exchange property of a Matroid.

   Conclusion: Considering the two properties above, we can safely 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.