

Lab03-Matroid & Dynamic Programming

CS2308-Algorithm and Complexity, Xiaofeng Gao, Spring 2022.

* If there is any problem, please contact TA Hongjie Fang.

* Name:----- Student ID:----- Email: -----

1. Let S be a finite set and let S_1, S_2, \dots, S_n be a partition of S into non-empty disjoint subsets. Define structure (S, \mathcal{I}) by the condition that

$$\mathcal{I} = \{I \mid I \subseteq S, |I \cap S_i| \leq k_i \text{ for } i = 1, 2, \dots, n\}$$

where k_1, k_2, \dots, k_n are positive integers. Does (S, \mathcal{I}) form a matroid? Why or why not? (*Hint: you may start from a simple case that $k_1 = k_2 = \dots = k_n = 1$.*)

2. Given n matrices A_1, A_2, \dots, A_n and $(n+1)$ positive integers m_1, m_2, \dots, m_{n+1} . The i -th matrix A_i ($1 \leq i \leq n$) has the size of $m_i \times m_{i+1}$. Our goal is to calculate $A_1 \times A_2 \times \dots \times A_n$. For any two matrix $P_{p \times q}$ and $Q_{q \times r}$, the computational cost of $P \times Q$ is $p \cdot q \cdot r$. Due to the combination law of matrix multiplication, we can arbitrarily add parentheses to change the calculation order without changing the final result. Since different methods of adding parentheses have different computational costs, please design an algorithm to minimize the total computational costs. You need to briefly describe your algorithm and write it in pseudo-code, then analyze the time complexity and space complexity of your algorithm.

Example. A_1 is a 4×2 matrix, A_2 is a 2×3 matrix, A_3 is a 3×1 matrix. If we calculate $A_1 \times A_2 \times A_3$ as the original order, the total computational costs are $4 \times 2 \times 3 + 4 \times 3 \times 1 = 36$. But if we calculate it as $A_1 \times (A_2 \times A_3)$, the total computational costs are $2 \times 3 \times 1 + 4 \times 2 \times 1 = 14$. In this example, your algorithm should output the minimum total computational cost 14.

3. You are given an $m \times n$ matrix A with integer elements. Initially, you are located at the top-left corner $(1, 1)$, and in each step, you can only move right or down in the matrix. Among all possible paths starting from the top-left corner $(1, 1)$ and ending in the bottom-right corner (m, n) , find the path with the maximum non-negative product. The product of a path is the product of all integers in the grid cells visited along the path.
 - (a) If all elements in A are non-negative integers, design an algorithm to solve the problem. Your algorithm should return the maximum non-negative product as well as the optimal path. Write the algorithm in pseudo-code and analyze its time complexity.
 - (b) If the elements in A are arbitrary integers, design an algorithm to solve the problem. Your algorithm should return the maximum non-negative product. Specially, if the maximum product is negative, your algorithm should return -1 . Briefly describe your algorithm and analyze its time complexity.

Example. An example of the problem is illustrated in Fig. 1. The path marked in blue is the optimal path, whose product is $1 \times 1 \times (-2) \times (-4) \times 1 = 8$.

1	-2	1
1	-2	1
3	-4	1

Figure 1: An Example of the Maximum Non-negative Product Path Problem

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