CCF CSP 计算机软件能力认证

CCF CSP

第 34 次认证

时间：2024 年 6 月 2 日 13:30 ~ 17:30

题目名称	矩阵重塑（其一）	矩阵重塑（其二）	文本分词	货物调度	哥德尔机
题目类型	传统型	传统型	传统型	传统型	传统型
输入	标准输入	标准输入	标准输入	标准输入	标准输入
输出	标准输出	标准输出	标准输出	标准输出	标准输出
每个测试点时限	1.0 秒	1.0 秒	1.5 秒	2.0 秒	5.0 秒
内存限制	512 MiB	512 MiB	512 MiB	512 MiB	1024 MiB
子任务数目	10	20	10	20	20
测试点是否等分	是	是	是	是	是

矩阵重塑（其一）（reshape1）

【题目背景】

矩阵（二维）的重塑（reshape）操作是指改变矩阵的行数和列数，同时保持矩阵中元素的总数不变。

【题目描述】

矩阵的重塑操作可以具体定义为以下步骤：

设原矩阵为 $\mathbf{M}$，其维度为 $n \times m$，即有 n 行和 m 列。新矩阵为 $\mathbf{M}'$，其维度为 $p \times q$。重塑操作要满足 $n \times m = p \times q$，这保证了元素的总数不变。

1. **线性化原矩阵**：按照行优先的顺序，将原矩阵 $\mathbf{M}$ 的元素转换成一个长度为 $n \times m$ 的一维数组 $\mathbf{A}$。这意味着你先读取 $\mathbf{M}$ 的第 0 行元素，然后是第 1 行，依此类推，直到最后一行。
2. **填充新矩阵**：使用一维数组 $\mathbf{A}$ 中的元素按照行优先的顺序填充新矩阵 $\mathbf{M}'$。首先填充 $\mathbf{M}'$ 的第 0 行，直到该行有 q 个元素，然后继续填充第 1 行，直到所有 p 行都被填满。

给定原矩阵中的一个元素的位置 (i, j)（$0 \le i < n$ 且 $0 \le j < m$），我们可以找到这个元素在被线性化后的一维数组 $\mathbf{A}$ 中的位置 k（$0 \le k < n \times m$），然后确定它在新矩阵 $\mathbf{M}'$ 中的位置 (i', j')（$0 \le i' < p$ 且 $0 \le j < q$）。它们之间满足如下数学关系：

$$i \times m + j = k = i' \times q + j'$$

给定 $n \times m$ 的矩阵 $\mathbf{M}$ 和目标形状 p、q，试将 $\mathbf{M}$ 重塑为 $p \times q$ 的矩阵 $\mathbf{M}'$。

【输入格式】

从标准输入读入数据。

输入共 $n + 1$ 行。

输入的第一行包含四个正整数 n、m 和 p、q。

接下来依次输入原矩阵 $\mathbf{M}$ 的第 0 到第 $n - 1$ 行，每行包含 m 个整数，按列下标从 0 到 $m - 1$ 的顺序依次给出。

【输出格式】

输出到标准输出。

输出共 p 行，每行 q 个整数，表示重塑后的矩阵 $\mathbf{M}'$。输出格式与输入相同，即依次输出 $\mathbf{M}'$ 的第 0 行到第 $p - 1$ 行；行内按列下标从 0 到 $q - 1$ 的顺序输出，且两个整数间仅用一个空格分隔。

【样例 1 输入】

```
2 3 3 2
1 2 3
4 5 6
```

【样例 1 输出】

```
1 2
3 4
5 6
```

【样例 2 输入】

```
2 2 1 4
6 6
6 6
```

【样例 2 输出】

```
6 6 6 6
```

【子任务】

全部的测试数据满足：

- n、m 和 p、q 均为正整数且 $n \times m = p \times q \leq 10^4$；
- 输入矩阵中每个元素的绝对值不超过 1000。

【提示】

评测环境仅提供各语言的标准库，特别地，不提供任何线性代数库（如 numpy、pytorch 等）。

矩阵重塑（其二）（reshape2）

【题目背景】

矩阵转置操作是将矩阵的行和列交换的过程。在转置过程中，原矩阵 $\mathbf{A}$ 的元素 a_{ij} 会移动到转置后的矩阵 $\mathbf{A}^T$ 的 a_{ji} 的位置。这意味着 $\mathbf{A}$ 的第 i 行第 j 列的元素在 $\mathbf{A}^T$ 中成为了第 j 行第 i 列的元素。

例如，有矩阵 $\mathbf{A}$ 如下：

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

它的转置矩阵 $\mathbf{A}^T$ 会是：

$$\mathbf{A}^T = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$$

矩阵转置在线性代数中是一个基本操作，广泛应用于各种数学和工程领域。

【题目描述】

给定 $n \times m$ 的矩阵 $\mathbf{M}$，试编写程序支持以下查询和操作：

1. **重塑操作** p、q：将当前矩阵重塑为 $p \times q$ 的形状（重塑的具体定义见上一题）；
2. **转置操作**：将当前矩阵转置；
3. **元素查询** i、j：查询当前矩阵第 i 行 j 列的元素（$0 \le i < n$ 且 $0 \le j < m$）。

依次给出 t 个上述查询或操作，计算其中每个查询的结果。

【输入格式】

从标准输入读入数据。

输入共 $n + t + 1$ 行。

输入的第一行包含三个正整数 n、m 和 t。

接下来依次输入初始矩阵 $\mathbf{M}$ 的第 0 到第 $n - 1$ 行，每行包含 m 个整数，按列下标从 0 到 $m - 1$ 的顺序依次给出。

接下来输入 t 行，每行包含形如 `op a b` 的三个整数，依次给出每个查询或操作。具体输入格式如下：

- 重塑操作：`1 p q`
- 转置操作：`2 0 0`
- 元素查询：`3 i j`

【输出格式】

输出到标准输出。

每个查询操作输出一行，仅包含一个整数表示查询结果。

【样例 1 输入】

```
3 2 3
1 2
3 4
5 6
3 0 1
1 2 3
3 1 2
```

【样例 1 输出】

```
2
6
```

【样例 2 输入】

```
3 2 5
1 2
3 4
5 6
3 1 0
2 0 0
3 1 0
1 3 2
3 1 0
```

【样例 2 输出】

```
3
2
5
```

初始矩阵：$\begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}$，$(1,0)$ 位置元素为 3；

转置后：$\begin{bmatrix}1 & 3 & 5\\2 & 4 & 6\end{bmatrix}$，$(1,0)$ 位置元素为 2；

重塑后：$\begin{bmatrix}1 & 3\\5 & 2\\4 & 6\end{bmatrix}$，$(1,0)$ 位置元素为 5。

【子任务】

80% 的测试数据满足：

- $t \leq 100$；

全部的测试数据满足：

- $t \leq 10^5$ 且其中转置操作的次数不超过 100；
- n、m 和所有重塑操作中的 p、q 均为正整数且 $n \times m = p \times q \leq 10^4$；
- 输入矩阵中每个元素的绝对值不超过 1000。

【提示】

- 对于 $n \times m$ 的矩阵，虽然转置和重塑操作都可以将矩阵形态变为 $m \times n$，但这两种操作通常会导致不同的结果。
- 评测环境仅提供各语言的标准库，特别地，不提供任何线性代数库（如 `numpy`、`pytorch` 等）。

文本分词（tokenizer）

【题目背景】

西西艾弗岛大数据中心正在如火如荼地开展大语言模型的研究工作。众所周知，计算机在执行机器学习的任务时，更适合处理数字的数据。对于大语言文本的处理，最好的方式是将文本转化为数字，然后再进行处理。通常，对输入数据进行整理后，需要将其按照一定的规则进行编码，以便计算机能够更好地处理。

这一转换过程是通过词汇表进行的。词汇表是一个包含了所有可能出现的词的列表。对于一个给定的文本，可以按照该表格将文本转化为一个数字的序列。而词表也需要根据文本的特点进行设计，以便更好地反映文本的特点。

【题目描述】

词汇表包括一系列的字符串，可以用于将输入的文本转换为数字序列。这里，我们认为输入文本事先经过一定的处理，将字母统一转换为了小写字母。词汇表的生成过程是一个迭代的过程。首先将文本按照一定的规则切分为单词序列，并统计全部单词的出现频率。然后将单词拆分为单字母字符串，组成初始的词汇表。接下来根据词汇表中的词汇接连出现在单词中的频率，将词汇反复合并，组成更长的词汇加入到词汇表中。词汇表的具体生成过程如下：

首先，输入文本中所有的单词和其出现的频率。然后，统计其中所有的字符，将其按照字典序排序，并将这些字符作为单字母字符串加入到词汇表中。同时，将输入的单词相应切分为词汇序列。

例如，输入下列词组和频率：

```
cut 15
cute 10
but 6
execute 3
```

则执行完上述过程后，词汇表中包含了 b、c、e、t、u、x 这些单字母字符串，而输入的词组被切分为：

```
c u t 15
c u t e 10
b u t 6
e x e c u t e 3
```

接下来，统计词汇表中，两个词汇组成的“词汇对”相连出现的频率，并选取出现次数最多的那一组拼接为一个字符串加入词汇表中。如果存在多个这样的“词汇对”，则再按照如下优先级顺序选取：

- 选取拼接后的字符串长度最短的那一组；
- 选取“词汇对”中前一个词汇长度最短的那一组；
- 选取拼接后的字符串字典序最小的那一组。

同时生成对应的合并规则（即将选出的“词汇对”合并成一个词汇），并按照该规则将所有输入单词的词汇序列按从前到后的顺序依次加以合并。

例如，在上述单词列表中词汇组合 `<c, u>` 在单词 `cut` 、`cute` 和 `execute` 中分别出现了 15、10 和 3 次。相应统计全部的“词汇对”出现次数如下：

```
c u 28
u t 34
t e 13
b u 6
e x 3
x e 3
e c 3
```

于是，将 `ut` 加入词汇表中，并生成合并规则 `<u, t>`，可得到词汇表 `b`、`c`、`e`、`t`、`u`、`x`、`ut`。同时，将输入的单词切分为：

```
c ut 15
c ut e 10
b ut 6
e x e c ut e 3
```

上述过程可以重复进行。例如，继续统计“词汇对”出现的频率如下：

```
c ut 28
ut e 13
b ut 6
e x 3
x e 3
e c 3
```

这时，将 `cut` 加入词汇表中，并生成合并规则 `<c, ut>`，可得到词汇表 `b`、`c`、`e`、`t`、`u`、`x`、`ut`、`cut`。同时，将输入的单词切分为：

```
cut 15
cut e 10
b ut 6
e x e cut e 3
```

词汇表的生成，需要重复进行上述过程，直到词汇表达到指定的长度，或者所有输入的单词都被合并为一个词汇。

此外需要注意，一种特殊情况是选取的“词汇对”由两个相同的词汇组成。例如按“词汇对”<a, a> 进行合并时，由于从前到后的顺序要求，序列 a a a 会被合并为 aa a，而序列 a a a a 则会被合并为 aa aa。

【输入格式】

从标准输入读入数据。

输入的第一行包含两个正整数，n 和 m，分别表示输入的单词的数量和期望的词汇表长度。

接下来的 n 行，每行包含一个非空字符串 s 和一个正整数 f，表示输入的单词和其出现的频率。其中，s 中只包含小写字母。

【输出格式】

输出共 m 行，每行包含一个字符串，按照加入词汇表的顺序输出词汇表中所有词汇。

【样例 1 输入】

```
4 8
cut 15
cute 10
but 6
execute 3
```

【样例 1 输出】

```
b
c
e
t
u
x
ut
cut
```

【样例 1 解释】

该样例即为题目描述中所举的例子。

【子任务】

对 20% 的数据，有 $n \leq 200$，m 恰好等于输入单词中出现的所有字母的个数。

对 40% 的数据，有 $n \leq 200$，$m \leq 200$。

对 80% 的数据，有 $n \leq 2000$，$m \leq 2500$。

对 100% 的数据，有 $n \leq 10000$，$m \leq 5000$ 且大于等于输入单词中出现的所有字母的个数，s 的长度 $|s| \leq 25$，输入单词的总频率（f 的总和）不超过 10^6。文本取材于真实的英文著作。

货物调度（trade）

【题目描述】

西西艾弗岛上共有 n 间物流仓库，小 P 目前有 m 件货物存放其中。为了获得至少为 v 的现金，小 P 需要选取一些货物卖出。

已知货物信息如下，第 i 件（$0 \leq i < m$）货物：

- 存放在第 t_i 间仓库中（$0 \leq t_i < n$）；
- 价值为 a_i，即选择卖出该货物可获得 a_i 的现金。

但在调货出库时也需要支付一些费用，对于第 j 间（$0 \leq j < n$）仓库：

- 只要调用了该仓库的货物（至少一件），就需要支付 b_j 的**基本费用**；
- 如果调用了该仓库中共 k 件货物，则还需要支付 $k \times c_j$ 的**计件费用**。

小 P 的最终目标是获得至少为 v 的现金，即要求卖出的货物总价值减去总费用的结果大于或等于 v。在满足该目标的前提下，试为小 P 规划一种花费最小的卖货方案。

【输入格式】

从标准输入读入数据。

输入的第一行包含三个正整数 n、m 和 v。

接下来 n 行输入仓库信息，其中第 j 行（$0 \leq j < n$）包含两个整数 b_j 和 c_j。

接下来 m 行输入货物信息，其中第 i 行（$0 \leq i < m$）包含两个整数 a_i 和 t_i。

【输出格式】

输出到标准输出。

仅输出一个整数，表示完成目标前提下的最小花费。

【样例 1 输入】

```
2 3 15
2 1
3 2
10 0
20 1
15 0
```

【样例 1 输出】

```
4
```

【样例 1 解释】

最优方案：选取货物 0 和 2，二者均在 0 号仓库，总花费为 $2+2\times1=4$。
选取货物 1 也刚好能满足要求（$20-3-1\times2\geq15$），但花费更多。
单独选取货物 0 或 2 均不能满足要求。

【样例 2 输入】

```
5 3 15
2 1
1 1
3 2
4 2
1 5
10 1
10 1
10 1
```

【样例 2 输出】

```
3
```

【样例 2 解释】

小 P 所有货物均存放在仓库 1 中，任取两件货物卖出即可满足要求（$10+10-1-2\times1\geq15$）。

【子任务】

30% 的数据满足：

- $m\leq15$

另有 40% 的数据满足：

- $a_i\leq20$

全部的数据满足：

- $0<n,m\leq1000$
- $0<b_j,c_j\leq20$
- $0<a_i\leq1000$
- $0<v\leq10^6$ 且保证至少存在一种可满足要求的卖货方案。

哥德尔机（Goedel）

【题目背景】

ReLU 函数是机器学习中常用的一个激活函数，其定义为：$\mathrm{ReLU}(x) = \max(0, x)$。

【题目描述】

在西西艾弗岛上有一台基于哥德尔机原理设计的通用人工智能机器，小 C 是负责维修这台机器的机器人。

有一天小 C 发现这个机器在一个算法部分上遇到了计算瓶颈，这个算法是这样的：

机器内部维护了一个神经网络，这个神经网络的权重是一个二维矩阵 V，并且权重是一个整数。

即对于每个二维坐标 (x, y)，矩阵在该位置的权重是 $V(x, y)$，初始权重为 0。

神经网络会进行 n 轮学习操作，每轮学习会给出参数 x_1, x_2, y_1, y_2, v，然后对每个满足 $x_1 \le x \le x_2,\ y_1 \le y \le y_2$ 的 (x, y)，将该位置对应的神经网络权重 $V(x, y)$ 修改为 $v + \mathrm{ReLU}(V(x, y) - v)$；

在所有学习操作之后，神经网络的参数定下来不变了，紧接着有 m 次神经网络推理操作，每次推理操作给出 x_1, x_2, y_1, y_2，查询对应子矩阵范围内最大的神经网络权重，即 $\max\limits_{\substack{x_1 \le x \le x_2 \\ y_1 \le y \le y_2}} V(x, y)$。

小 C 发现机器在枚举这个问题优秀的算法时卡住了，目前只枚举出了一个很暴力的算法，为了让机器可以快点启动，他决定自己写好这个问题的算法来降低其启动常数，你能帮帮他吗？

【输入格式】

从标准输入读入数据。

输入的第一行包含两个整数 n, m；

接下来 n 行每行五个整数 x_1, x_2, y_1, y_2, v，依次表示每次学习操作的参数；

接下来 m 行每行四个整数 x_1, x_2, y_1, y_2，依次表示每次推理操作的参数。

【输出格式】

输出到标准输出。

共 m 行，依次表示每次查询操作的答案。

【样例 1 输入】

```
5 5
1 3 2 3 3
4 5 2 5 1
3 5 1 2 1
2 5 3 4 4
1 4 3 4 2
1 5 2 5
1 5 2 5
1 5 1 5
1 4 1 5
2 5 1 3
```

【样例 1 输出】

```
4
4
4
4
4
```

【样例 2 输入】

```
10 10
3 10 7 7 10
1 10 9 9 3
4 6 7 7 7
1 8 5 5 1
6 8 1 1 1
1 3 8 8 2
2 10 10 10 9
1 10 6 6 4
1 8 9 9 4
5 9 9 9 2
1 9 2 2
2 10 1 10
```

```
2 10 6 9
2 2 1 4
2 10 8 10
7 10 1 10
1 8 1 9
1 8 5 7
3 7 5 8
2 10 1 7
```

【样例 2 输出】

```
0
10
10
0
9
10
10
10
10
10
```

【子任务】

对于 10% 的数据，满足 $1 \leq n, m \leq 100$。

对于另外 10% 的数据，满足 $1 \leq n, m \leq 10^3$。

对于另外 20% 的数据，满足 $1 \leq n, m \leq 10^4$。

对于另外 20% 的数据，满足 $1 \leq n, m \leq 5 \times 10^4$。

对于另外 10% 的数据，满足 $1 \leq n \leq 10^3$。

对于 100% 的数据，满足 $1 \leq n, m \leq 5 \times 10^5$，对每个修改或查询操作，满足 $1 \leq x_1 \leq x_2 \leq n$，$1 \leq y_1 \leq y_2 \leq n$ 对每个修改操作，满足 $1 \leq v \leq n$，所有数值为整数。