E Chem Is A Third Try——by Arthur

题目描述

“轰——”伴随着一声巨响，Lucifer•Tang 跟着自己的实验室一起，被轰上了天——
Thor（震惊状，斜四十五度向上指去）：“(⊙o⊙)看——灰机——”
Arthur（期待状，仰望星空——）：“在哪在哪=w=”
于是二人开开心心地开始整（GUA）理（FEN）Lucifer•Tang 的遗物。
有趣的事情出现了……
==========
Thor 和 Arthur 发现，Lucifer•Tang 对其遗产进行了加密工作，密码为 n 个长度不等的序列，其序列长度分别为 A1,A2,A3…,An。机（YU）智（CHUN）的二人很快便知道了破解的方案，很简单，就是按照某种顺序，每一次交换第 i 个序列与第 j 个序列的位置。
于是从来便已勤（LAN）劳（DUO）著称的二人便将执行破解的工作交给了你。

输入

多组测试数据。
对于每组测试数据，第一行为两个整数 n，m（1≤n≤1000，0≤m≤100000），用空格隔开，表示序列的个数，以及破解需要进行的交换次数。
第二行为 n 个整数 A1,A2,A3…,An(1≤Ai≤50000,SUM(Ai)<=10000000)，用空格隔开，为每一个序列的长度。
接下来 n 行，第 i 行有 Ai 个整数，用空格隔开，为第 i 个序列的元素值。
再接下来 m 行，每行两个整数 x,y，表示交换第 x 个序列与第 y 个序列。

输出

对于每组数据，输出 n 行，每行一个序列，为经过 m 次交换之后的最终结果。
具体见样例。

输入样例

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

1 5
2 5
3 4

输出样例

---

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

HINT

数据量较大，请使用 scanf 与 printf

解题分析

观察数据范围不难发现，数据量大得令人发指……由于序列最多有一千个，每个序列中的元素最多有 50000 个，这样如果开二维数组，与 C 题相同，也是根本开不下的。
然而又发现了一个非常有趣的现象，就是这些序列的元素总个数不超过一千万，而实际上给一千万个数组元素分配内存是可以的 0v0.
这样不难想出，这一千个序列的长度一定有不相同的，如果都按五万开的话，会浪费极多的内存，对于本题大概是 400%吧……
于是怎么办？指针！而且是指针数组。课前给大家讲解的指针数组，就是用来做这个的，基础部分有什么不明白的可以参考辅助文档或询问助教，在这里不再赘述。
好了进入正题
声明一个指针数组代替二维数组，与二维数组不同的是，它每一行的元素数量是不确定的，而且是可以人为指定的。很多同学不知道对于指针数组应该如何开辟内存，其实很简单，用一个 for 循环呗（就像你对普通的一维数组赋值一样喽）：

```
int *p[1001],num[1001],n;

for(int i=0;i<n;i++)

    p[i]=new int[num[i]];
```

请注意这里的格式是严格确定的，后面的中括号里是你给这一个指针指定多少个元素，只可以有一个中括号，而不是说二维数组就两个中括号，三维数组就三个中括号。
这样第一个内存问题就解决了。

接下来的问题是交换，注意操作数是10^5级的，而如果朴素的交换（一个元素一个元素地交换），每一次交换都是 O(n)级的，这里在最极端的情况下(n=50000)。你的处理就变成了10^{10}级的了，注意计算机平均一秒计算一亿次，这样你这个程序一秒钟之内是跑不完的。
这里有两种方法，一种使用指针，另一种不使用，但是两种采用的都是指针的思想。

方法①：头指针交换

之前讲过数组是一个指向起始地址的指针以及一串连续的内存。
而我们的指针数组都是指向每一行的起始地址的指针。
因此为何不对这些指针进行交换呢？这样每一次交换只需要进行两次交换，一次是头指针的交换，一次是序列长度的交换。
指针的交换方法如下（不针对此题）：

```
int *p,*q,*temp;
temp=p;    //temp 指向 p 指向的地址
p=q;       //p 的地址转为指向 q 指向的地址
q=temp;    //q 指向的地址变成 temp 指向的地址，也
             就是 p 原来指向的地址，从而实现交换
```

方法②——没节操的序号

初始的时候，直接给这个序列按照从上到下的顺序标号为 1、2、3、…、n，存储在一个数组里，每一次交换的时候只交换这个序号数组里对应位置元素的值，而不改变原有的序列的顺序，最后按照这里的序号顺序输出对应的序列即可。这里实际上采用的也是一种指针的思想，因为从这个序号数组向二维数组的行的映射与指针实际上是差不多的。
最后输出的时候是这样的：

```
int No[1001];
for(int i=0;<n;i++){
    for(int j=0;j<num[ No[i] ];j++)
        printf("%d ",p[ No[i] ][ j ]);
    printf("\n");
}
```

这样交换的超时问题我们也解决了。
最后一步，内存问题②，由于多组测试数据将造成内存浪费，这个很简单，只需要在每一组测试数据处理完毕之后，将这一轮开辟的内存再销毁掉就可以了。

参考代码（仅提供指针交换版）

C++:

```
#include<cstdio>
#include<iostream>
using namespace std;
int *array[1001],N,M,num[1001],x,y,*p,q;
int main()
{
    while(scanf("%d%d",&N,&M)!=EOF)
    {
            for(int i=0;i<N;i++)
            {
                scanf("%d",&num[i]);
                array[i]=new int[num[i]];
            }
            for(int i=0;i<N;i++)
                for(int j=0;j<num[i];j++)
                   scanf("%d",&array[i][j]);
            for(int i=0;i<M;i++)
            {
                scanf("%d%d",&x,&y);
                p=array[x-1];array[x-1]=array[y-1];array[y-1]=p;
                q=num[x-1];num[x-1]=num[y-1];num[y-1]=q;
            }
            for(int i=0;i<N;i++)
            {
                    for(int j=0;j<num[i];j++)
                       printf("%d ",array[i][j]);
                    printf("\n");
            }
     }
}
```

C:

```
#include<cstdio>
#include<cstdlib>
int *array[1001],N,M,num[1001],x,y,*p,q;
int main()
{
    while(scanf("%d%d",&N,&M)!=EOF)
    {
             for(int i=0;i<N;i++)
             {
                 scanf("%d",&num[i]);
                 array[i]=(int *)malloc(sizeof(int)*num[i]);
             }
             for(int i=0;i<N;i++)
                 for(int j=0;j<num[i];j++)
                    scanf("%d",&array[i][j]);
             for(int i=0;i<M;i++)
             {
                 scanf("%d%d",&x,&y);
                 p=array[x-1];array[x-1]=array[y-1];array[y-1]=p;
                 q=num[x-1];num[x-1]=num[y-1];num[y-1]=q;
             }
             for(int i=0;i<N;i++)
             {
                     for(int j=0;j<num[i];j++)
                        printf("%d ",array[i][j]);
                     printf("\n");
             }
            for(int i=0;i<N;i++)
                  free(array[i]);
    }
}
```