J – Jumbled Stacks

Time limit: 1 s   Memory limit: 256 MiB

We are given a set of n cards, labelled from 1 to n, which are distributed into k stacks $S_1, S_2, \ldots, S_k$. Each stack has a limited capacity: the i-th stack, S_i, can contain at most C_i cards. The only way we can manipulate these cards is by taking the top card of a stack and moving it to the top of some other stack (as long as this wouldn't exceed the capacity of the destination stack).

Using a sequence of such moves, we would like to rearrange the cards so that the first few stacks (0 or more) with the smallest indices are filled to capacity, the stack immediately after them is not filled to capacity (and may even be empty) and all stacks after that are completely empty. Moreover, if we stack together all the stacks from S_1 at the bottom to S_k at the top, the cards should be ordered from smallest to largest, with 1 at the bottom and n at the top.

It is guaranteed that $n \le \left(\sum_{i=1}^{k} C_i\right) - \max_{1\le i\le k} C_i$.

Suppose we had $n = 6$ cards on $k = 3$ stacks, with capacities $C_1 = 4$, $C_2 = C_3 = 3$, and with the following initial state: $S_1 = [2, 3, 0, 0]$ (from bottom to top; 0 indicates an empty slot), $S_2 = [4, 1, 6]$, $S_3 = [5, 0, 0]$. Then the desired end state is $S_1 = [1, 2, 3, 4]$, $S_2 = [5, 6, 0]$ and $S_3 = [0, 0, 0]$.

Input data

The first line contains two integers, n (the number of cards) and k (the number of stacks), separated by a space. The remaining k lines describe the initial state of the stacks; the i-th of these lines describes S_i and contains $C_i + 1$ integers, separated by spaces. The first of these integers is C_i (the capacity of the stack S_i), the rest of them are the labels of the cards on S_i, from bottom to top. If the stack S_i contains fewer than C_i cards (it could even be empty), the last few integers in the line will be 0.

Input limits

- $1 \le n \le 100$
- $3 \le k \le 100$
- $1 \le C_i \le n$

Output data

Print a sequence of moves that bring the stacks into the desired end state. For each move, output a line containing two integers, separated by a space: first the number of the stack from which the card is being moved and then the number of the stack to which it is being moved (the stacks are numbered from 1 to k; the destination stack must not be the same as the source stack). The number of moves must not exceed 10^5. After the end of the sequence of moves, print a line containing "`0 0`" (without the quotation marks). If there are several possible solutions, you may output any of them.

Example

Input

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

Output

```
2 3
2 3
1 2
1 2
3 1
2 1
2 1
3 2
3 1
2 3
1 3
2 1
3 2
3 2
0 0
```

Comment

This is the example discussed earlier in the problem statement. The sample output shows a sequence of 14 moves which bring the stacks into the desired end state.