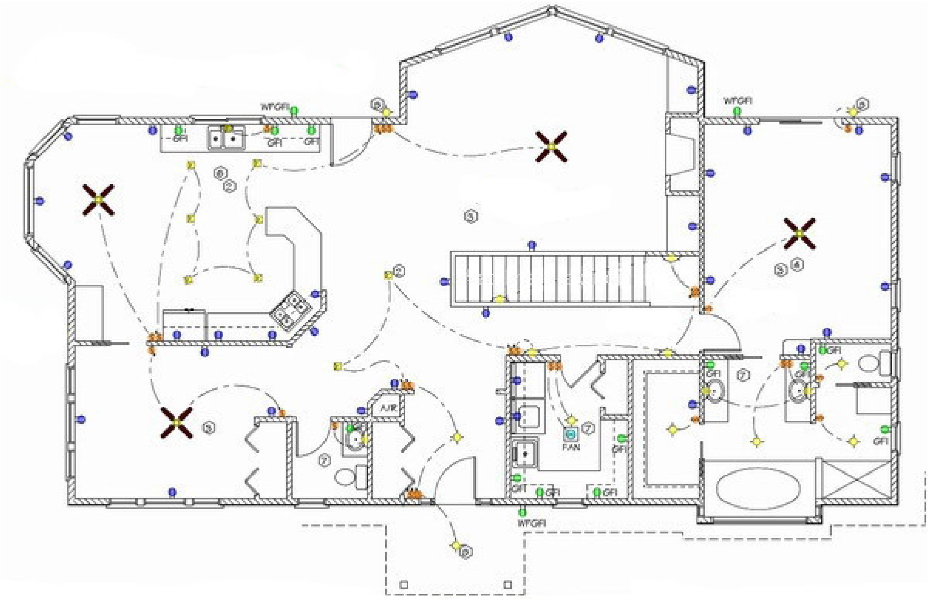
# Crazy Lights Hotel (joc)

Giorgio loves reading before going to sleep, but today he's not at home but in a cheap hotel abroad for a conference. In the *Crazy Lights* hotel, each room has $N$ lights and the corresponding $N$ buttons; if you press the $i$-th button and the $i$-th light was off, it will immediately turn on!



Now Giorgio is in the bed and he wants to start reading: he needs to switch off all the lights but the $K$-th. There is an odd behaviour though: he noticed that by using the $i$-th button not only the $i$-th light turns on but also some other lights turn off! He spent a few minutes to map all the switches to the lights and now he knows exactly what each button does. The management told Giorgio to **never use the $i$-th button if the $i$-th light is already on**, otherwise there will be a peak of power and all the hotel will go black!

Help Giorgio, who wants to start reading as soon as possible: which is the *minimum* number of buttons to press in order to switch off all the lights but the $K$-th one?

> ☞ Among the attachments of this task you may find a template file `joc.*` with a sample incomplete implementation.

## Input

The first line contains two integers: $N$ and $K$, respectively the number of lights and the index of the light that needs to be on (1-based).

The next line contains $N$ values, either 0 or 1. The $i$-th represents the initial state of the $i$-th light (0 means off and 1 means on).

The following $N$ lines describe what each button does. The $i$-th line, relative to the $i$-th button, is composed as follows: an integer $t$ followed by $t$ integers, the indexes (1-based) of the lights that will turn off by pressing this button.

## Output

You need to write a single line with an integer: the minimum number of buttons to press in order to have all the lights turned off but the $K$-th one.

## Constraints

- $2 < N < 20$.
- $1 \leq K \leq N$.
- It's always possible to reach the final configuration.
- Pressing the $i$-th button will turn off all the corresponding lights, no matter if they were on or off.
- Only turned off lights can be turned on.
- The $i$-th button will not turn off the $i$-th light.

## Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

| | |
|---|---|
| – **Subtask 1** (0 points)<br>■□□□□ | Examples. |
| – **Subtask 2** (10 points)<br>■□□□□ | Every button turns off every other light. |
| – **Subtask 3** (10 points)<br>■□□□□ | Every light is turned off by exactly one button. |
| – **Subtask 4** (35 points)<br>■□□□□ | There is a single light on. |
| – **Subtask 5** (25 points)<br>■■■■□ | $N = 3$. |
| – **Subtask 6** (20 points)<br>■■■□□ | No additional limitations. |

## Examples

| input | output |
|---|---|
| 3 3<br>0 1 1<br>2 2 3<br>1 3<br>2 1 2 | 2 |
| 6 3<br>0 0 1 1 0 1<br>2 2 3<br>0<br>3 2 5 6<br>1 1<br>3 1 4 6<br>0 | 3 |

## Explanation

In the **first sample case** there are 3 lights and only the last must be on in the end. To do so the best strategy is to:

- Press the first button, this will turn off the second and the third lights, resulting in `1 0 0`.
- Press the last button, this will turn off the first light (and the second one, but it's already off), resulting in `0 0 1`.

In the **second sample case** there are 6 lights, the best strategy is to press:

- 1-st button, leading to `1 0 0 1 0 1`.
- 5-th button, leading to `0 0 0 0 1 0`.
- 3-rd button, leading to the final configuration `0 0 1 0 0 0`.