Problem G
Optimal Selection

Time limit: 8 seconds

Memory limit: 1024 megabytes

Problem Description

Given n distinct numbers, one can output the k-th smallest number without sorting. This problem is known as *selection problem*, and can be solved in linear time. Some people argue that the existing algorithms for selection is too slow to put into practice. So the conjecture is, every time you encounter an algorithm that uses selection as a building block, you can feel free to replace selection with a fast sorting procedure. However, this conjecture seems not to be true.

Bob is trying to answer whether the above conjecture is valid, but it turns out to be a complicated problem that requires many trial and error experiments. After some failed trials, he decides to first focus on finding the best algorithm to implement selection. That is to find an algorithm that implements selection with the least number of comparisons. We assume that the n input numbers will be all distinct. For $n = 3$ and $k = 1$, an optimal algorithm to solve the selection problem is shown as follows:

```
double selection(double a[3], int k = 1){
    if(a[0] < a[1]){
        if(a[0] < a[2]){
            return a[0];
        }else{
            return a[2];
        }
    }else{
        if(a[1] < a[2]){
            return a[1];
        }else{
            return a[2];
        }
    }
}
```

For every possible $(a[0], a[1], a[2])$, the above selection function uses at most 2 comparisons. Indeed, 2 comparisons is the best possible if the relative order between two input numbers can be inferred only by comparisons, aka *comparison-based algorithms*. So the *complexity* of the selection function for $(n, k) = (3, 1)$ is 2. Now, Bob is considering a more general problem that, the comparison results between some pairs from the input are known before invoking the selection function. The *complexity* of the selection function is defined as the worst case number

of comparisons required, by an optimal comparison-based algorithm, on any possible consistent input array. That is, Bob is wondering what is the complexity of the selection function, given n, k, and some relative order among the n input numbers.

Input Format

Three integers n, k, and ℓ are given in the first line. Then ℓ lines follow. Each of the subsequent ℓ lines contains two integers x and y in $[0, n-1]$ specifying that the relative order between $a[x]$ and $a[y]$ is known to be $a[x] < a[y]$ before invoking the selection function.

Output Format

Given n, k, and the relative order between the given ℓ pairs, output the complexity of the selection function.

Technical Specification

- $1 \leq n \leq 8$.
- $1 \leq k \leq n$.
- $0 \leq \ell \leq n$.
- The relative order between the given ℓ pairs will be consistent.

Sample Input 1

```
3 2 0
```

Sample Output 1

```
3
```

Sample Input 2

```
7 2 5
0 6
3 6
4 6
2 0
0 5
```

Sample Output 2

```
5
```