Problem H

Random Walking

The Army of Coin-tossing Monkeys (ACM) is in the business of producing randomness. Good random numbers are important for many applications, such as cryptography, online gambling, randomized algorithms and panic attempts at solutions in the last few seconds of programming competitions.

Recently, one of the best monkeys has had to retire. However, before he left, he invented a new, cheaper way to generate randomness compared to directly using the randomness generated by coin-tossing monkeys. The method starts by taking an undirected graph with 2^n nodes labelled $0, 1, \ldots, 2^n - 1$. To generate k random n-bit numbers, they will let the monkeys toss n coins to decide where on the graph to start. This node number is the first number output. The monkeys will then pick a random edge from this node, and jump to the node that this edge connects to. This new node will be the second random number output. They will then select a random edge from this node (possibly back to the node they arrived from in the last step), follow it and output the number of the node they landed on. This walk will continue until k numbers have been output.

During experiments, the ACM has noticed that different graphs give different output distributions, some of them not very random. So, they have asked for your help testing the graphs to see if the randomness is of good enough quality to sell.

They consider a graph good if, for each of the n bits in each of the k numbers generated, the probability that this bit is output as 1 is greater than 25% and smaller than 75%.

Input specifications

The input will consist of several data sets. Each set will start with a line consisting of three numbers k, n, e separated by single spaces, where k is the number of n-bit numbers to be generated and e is the number of edges in the graph ($1 \leq k \leq 100$, $1 \leq n \leq 10$ and $1 \leq e \leq 2000$). The next e lines will consist of two space-separated integers v_1, v_2 where $0 \leq v_1, v_2 < 2^n$ and $v_1 \neq v_2$. Edges are undirected and each node is guaranteed to have at least one edge. There may be multiple edges between the same pair of nodes.

The last test case will be followed by a line with $k = n = e = 0$, which should not be processed.

Output specifications

For each input case, output a single line consisting of the word `Yes` if the graph is good, and `No` otherwise.

Sample input

```
10 2 3
0 3
1 3
2 3
5 2 4
0 1
0 3
1 2
2 3
0 0 0
```

Output for sample input

```
No
Yes
```