Maxpath

SEASON 8 – THIRD ROUND

An undirected graph with N nodes and M bidirectional edges is given. A simple path is a sequence of $K \geq 1$ nodes $V_1, V_2, \ldots, V_K$, such that $V_i \neq V_j$ for $i \neq j$ and there exists an edge between V_i and V_{i+1} for $i=1, 2, \ldots, K-1$.

We define the value of a simple path $V_1, V_2, \ldots, V_K$, to be $\sum_{i=1}^{K} i \times V_i$. Write a program **maxpath** which finds a path with as large value as possible.

Input

The first line of the input file `maxpath.in` contains two positive integers N and M – the number of nodes and the number of edges in the graph, respectively. The next M lines contain two integers each, representing the edges of the graph. It's guaranteed that there are no self-loops or duplicate edges.

Output

On the first line of the output file `maxpath.out` print a single positive integer K – the number of nodes in the path found by your program. On the next K lines, print the number of the current node in the path.

Scoring

If the output doesn't follow the format or the printed nodes don't form a simple path, you will receive 0 points for the test.

Otherwise, you will receive $score \times (\frac{yours+1}{best+1})^2$ points, where *score* is the number of points the test is worth, *yours* is the value of the path you found and *best* is the greatest value of a path among all participants for the given test.

Maxpath

SEASON 8 – THIRD ROUND

Constraints

Portion of tests	Constraints on N and M
30%	$N = 100, M \in [\frac{N(N-1)}{40}; \frac{N(N-1)}{5}]$
30%	$N = 1\,000, M \in [\frac{N(N-1)}{40}; \frac{N(N-1)}{10}]$
40%	$N = 100\,000, M \in [200\,000;\ 500\,000]$

In each of the three groups in the table above, half of the test cases will be generated with algorithm 1 and the other half – with algorithm 2, mentioned below.

Test generation

Two algorithms are used for generating the graphs:

- *Algorithm 1:* We generate a tree by assigning to each node a random parent with smaller number (except for node 1). We add edges *(x,y)* to the obtained graph, as long as they are not already in the graph, until the total number of edges becomes *M*. After that, the nodes' numbers are shuffled randomly.
- *Algorithm 2:* We generate a random number T_1 from *1* to *N*, then a random number T_2 from T_1+1 to *N* and so on, until T_k becomes equal to *N*. We form *K* paths (meaning that we connect the nodes with edges in the given order): $\{1, 2, \ldots, T_1\}$, $\{T_1+1, T_1+2, \ldots, T_2\}$, …, $\{T_{k-1}+1, \ldots, T_k=N\}$. From each path, except for the first one, a random node is chosen and an edge between it and a random node from the previous paths is added. We add edges *(x,y)* to the obtained graph, as long as they are not already in the graph, until the total number of edges becomes *M*. After that, the nodes' numbers are shuffled randomly.

Time limit: 5 s

Memory limit: 256 MB

Maxpath

SEASON 8 – THIRD ROUND

Sample test

Input (maxpath.in)	Output (maxpath.out)
5 5 1 2 2 3 2 5 2 4 3 5	4 4 2 3 5

The proposed output is a path of value 37.