Graph

SEASON 8 – SECOND ROUND

We are given an undirected connected graph with **N** vertices and **M** edges.

A **subgraph** is defined as a pair of a subset of vertices and a subset of edges, such that the endpoints of all edges in the subset are in the subset of vertices.

We want to check if there exists a subgraph of the given graph that satisfies the following conditions:

1) It is **biconnected**. This means that if we erase any **vertex or edges**, the subgraph will remain connected.

2) It is **bipartite**. This means that it is possible to color the vertices in two colors so that the endpoints of the edges have different colours.

3) The number of vectices in this subgraph is greater than or equal to 3.

Write a program that checks whether such a subgraph exists. If so, you should also find an example of such a subgraph.

Input

The first line of the input file `graph.in` contain ***N*** and ***M*** – the number of vertices and the number of edges of the given graph.

The **i**-th of the next **M** lines contains **u[i]** and **v[i]** – the information about the **i**-th edge of the graph. This means that there is a undirected edge between vertices **u[i]** and **v[i]**.

It is guaranteed that the given graph is connected and it has no loops **(u[i] ≠ v[i])**. Also it is guaranteed that there is at most one edge between every pair of vertices in the graph.

Output

On the first line of the output file `graph.out` отпечатайте един ред с print **„Yes“** if there exists a subgraph that satisfies the above-mentioned conditions. Overwise print **„No“**.

If such a subgraph exists, on the second line print the numbers **P** and **K** – the number of vertices and edges in the subgraph respectivly. On the **i**-th of next **K** lines print the **a[i]** и **b[i]** – the endpoints of the **i**-th edge in the subgraph. If there are multiple correct answers, you can print any.

Constraints

$2 \le N \le 10^5$

$1 \le M \le 2 * 10^5$

Time limit: 1 sec
Memory limit: 256 MB

Graph

SEASON 8 – SECOND ROUND

Examples

Input (graph.in)	Output (graph.out)
4 5 1 2 2 3 3 4 1 3 2 4	Yes 4 4 1 2 3 4 1 2 1 3 2 4 3 4
3 3 1 2 2 3 3 1	No