7603 Independent Edges

Let G be a simple undirected graph. Two vertices of G are said to be *adjacent* if they are connected by an edge, and two edges of G are said to be *adjacent* if they share a vertex. In the graph shown in Figure 1(a), vertices 3 and 4 are adjacent because there is an edge (3,4) connecting them; edges (3,4) and (3,5) are adjacent because they share a vertex 3. A subset of vertices of G is called an *independent vertex set* if no two vertices in the subset are adjacent; also, a subset of edges of G is called an *independent edge set* if no two edges in the subset are adjacent. The size of a maximum independent vertex set, an independent vertex set of largest possible size, of G is called the *vertex independence number* and denoted by $\alpha(G)$; analogously, the size of a maximum edge independent set of G is called the *edge independence number* and denoted by $\upsilon(G)$.

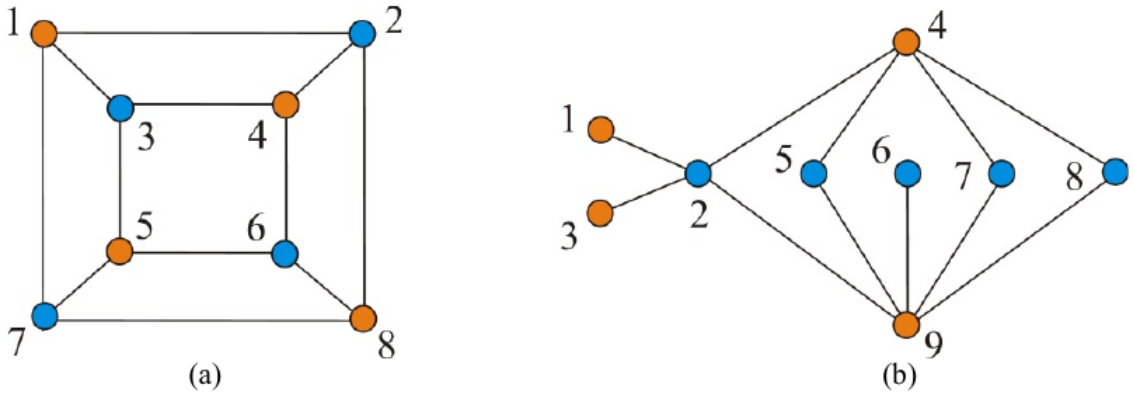

Figure 1. Two graphs G_1 and G_2 : (a) graph G_1, where $\alpha(G_1) = 4$ and $\upsilon(G_1) = 4$; (b) graph G_2, where $\alpha(G_2) = 6$ and $\upsilon(G_1) = 3$.

The Independent Edge Set problem is the problem of finding a maximum independent edge set of a graph. The problem is well-studied, so many algorithms for the problem have been designed and implemented, where most of the algorithms run in time polynomial to the size of the graph. Sometimes, we expect something more than the output of a computer program, so as to be sure of the correctness of the output. This motivates the study of so-called certifying algorithms.

When the user gives X as an input and the program outputs Y, the user usually has no way of knowing whether Y is a correct output on input X or it has been compromised by a bug. A *certifying algorithm* is an algorithm that produces, with each output, a *certificate* Z that the particular output has not been compromised by a bug. By inspecting the certificate, either manually or by use of a program, the user can convince him/her that the output is correct, or reject the output as buggy. The process of checking Z can be automated with a *checker*, which is an algorithm for verifying that Z proves that Y is a correct output for X.

Let us think of the certificate that should be produced by a certifying algorithm for the Independent Edge Set problem on a bipartite graph. Here, we say that a graph is *bipartite* if its vertex set can be divided into two disjoint sets U and V in such a way that every edge connects a vertex in U to one in V. The graph of Figure 1(a) is bipartite, where the vertex set is partitioned into two sets of orange-colored vertices and blue-colored vertices; the same follows for the graph of Figure 1(b). If the input graph is bipartite, we can utilize König's theorem which states that $\upsilon(G) + \alpha(G) = n$ for any bipartite graph G with n vertices. This suggests that if the input graph is bipartite, its maximum independent vertex set can serve as a good certificate. Once we have identified an independent edge set of size k and an independent vertex set of size $n - k$ in a bipartite graph G with n vertices, then it

is evident that $\upsilon(G) = k$ and $\alpha(G) = n - k$, i.e., the independent edge set and the independent vertex set found are both maximum possible.

Given a bipartite graph G, your task is to devise a certifying algorithm for finding a maximum independent edge set of G. The algorithm should produce, in addition to an output Y, a certificate Z which has been described above. It is assumed that the graph G has n vertices that are indexed from 1 to n. It may be easy to write a checker that determines if no two edges in Y are adjacent, and determines if no two vertices in Z are adjacent, and finally accepts the output if $|Y| + |Z| = n$.

Input

The input file contains several test cases, each of them as described below.

Your program is to read from standard input. The first line contains two positive integers n and m, respectively, representing the numbers of vertices and edges of the input graph, where $n \leq 1,000$ and $m \leq 50,000$. It is followed by m lines, each contains two positive integers u and v that represent an edge between vertex u and vertex v of the input graph. The input graph is a bipartite graph that is not necessarily connected.

Output

For each test case, the output must follow the description below.

Your program is to write to standard output. The first line should contain an integer, k, indicating the size of a maximum independent edge set of the input graph. In the following k lines, each contains an edge of the maximum independent edge set. Then, a certificate produced by your algorithm should follow: a line containing an integer, k', representing the size of the certificate is followed by a line containing the members of the certificate in ascending order.

Sample Input

```
8 12
1 2
2 8
8 7
7 1
3 4
4 6
6 5
5 3
1 3
2 4
5 7
6 8
9 11
4 2
4 5
4 7
4 8
9 2
9 5
9 6
9 7
9 8
```

```
3 2
2 1
```

Sample Output

```
4
2 4
8 6
7 5
1 3
4
1 4 5 8
3
3 2
5 4
9 6
6
1 3 5 6 7 8
```