SPOJ SUBMERGE

Submerging Islands

Prof. Edson Alves

Faculdade UnB Gama

Vice City is built over a group of islands, with bridges connecting them. As anyone in Vice City knows, the biggest fear of vice-citiers is that some day the islands will submerge. The big problem with this is that once the islands submerge, some of the other islands could get disconnected. You have been hired by the mayor of Vice city to tell him how many islands, when submerged, will disconnect parts of Vice City. You should know that initially all the islands of the city are connected.

Vice City foi construída sobre um grupo de ilhas, com pontes conectando-as. Como todos em Vice City sabem, o maior medo dos moradores é que em algum dia as ilhas venham a submergir. O maior problema é que, uma vez que uma ilha afunde, as outras ilhas podem ficar desconectadas. Você foi contratado pelo prefeito de Vice City para dizer a ele quantas ilhas, se submergissem, desconectariam as ilhas de Vice City. Assuma que todas as ilhas da cidade estão inicialmente conectadas.

Input

The input will consist of a series of test cases. Each test case will start with the number N $(1 \leq N \leq 10^4)$ of islands, and the number M of bridges $(1 \leq M \leq 10^5)$. Following there will be M lines each describing a bridge. Each of these M lines will contain two integers U_i, V_i $(1 \leq U_i, V_i \leq N)$, indicating that there is a bridge connecting islands U_i and V_i. The input ends with a case where $N = M = 0$.

Output

For each case on the input you must print a line indicating the number of islands that, when submerged, will disconnect parts of the city.

Entrada

A entrada é composta por uma série de casos de teste. Cada caso de teste irá começar com o número N $(1 \leq N \leq 10^4)$ de ilhas e o número M de pontes $(1 \leq M \leq 10^5)$. Em seguida haverão M linha, cada uma descrevendo uma ponte. Cada uma destas M linha conterá dois inteiros U_i, V_i $(1 \leq U_i, V_i \leq N)$, indicando que há uma ponte conectando as ilhas U_i e V_i. A entrada termina com um caso de teste onde $N = M = 0$.

Saída

Para cada caso da entrada você deve imprimr uma linha indicando o número de ilhas que, se submergidas, desconectariam partes da cidade.

Exemplo de entrada e saída

Exemplo de entrada e saída

```
3 3
```

Exemplo de entrada e saída

Exemplo de entrada e saída

Exemplo de entrada e saída

3 3

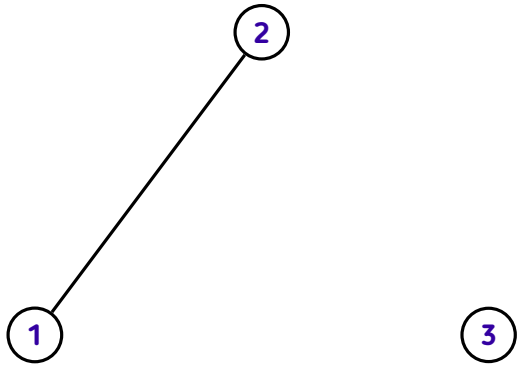
Exemplo de entrada e saída

3 3
1 2

2

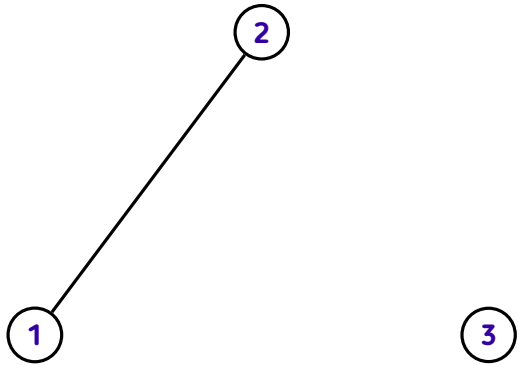
Exemplo de entrada e saída

3 3
1 2



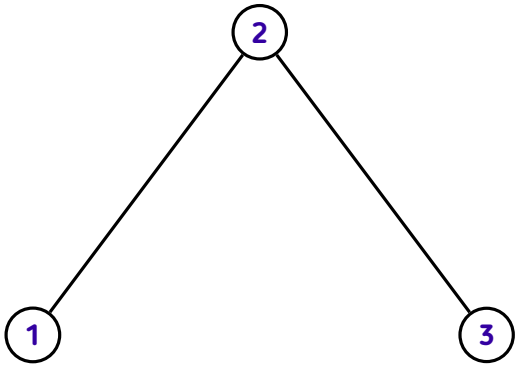
Exemplo de entrada e saída

```
3 3
1 2
2 3
```



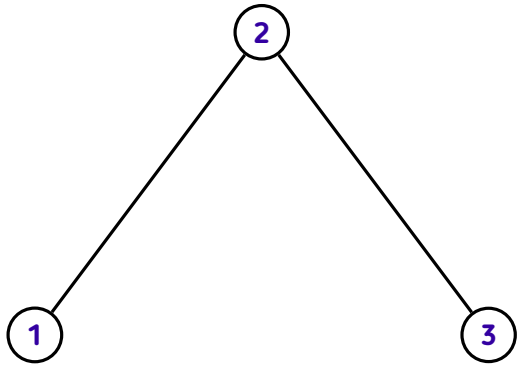
Exemplo de entrada e saída

```
3 3
1 2
2 3
```



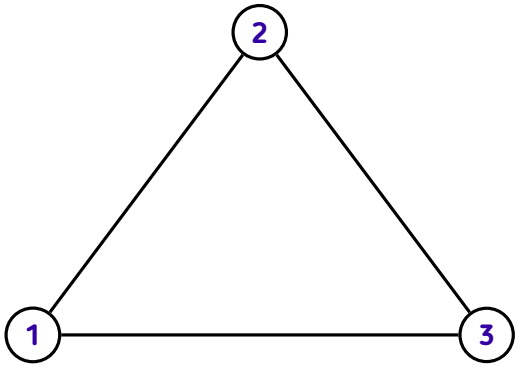
Exemplo de entrada e saída

```
3 3
1 2
2 3
1 3
```



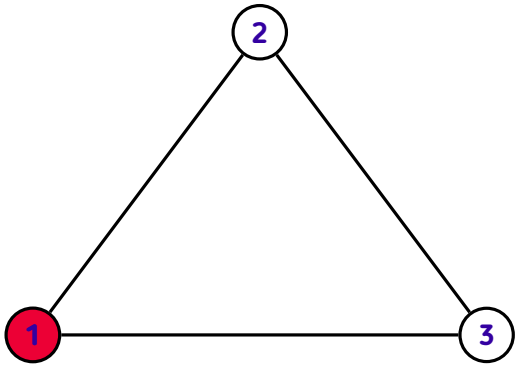
Exemplo de entrada e saída

```
3 3
1 2
2 3
1 3
```



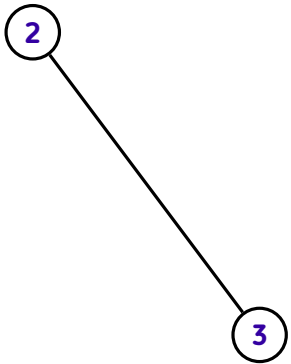
Exemplo de entrada e saída

```
3 3
1 2
2 3
1 3
```



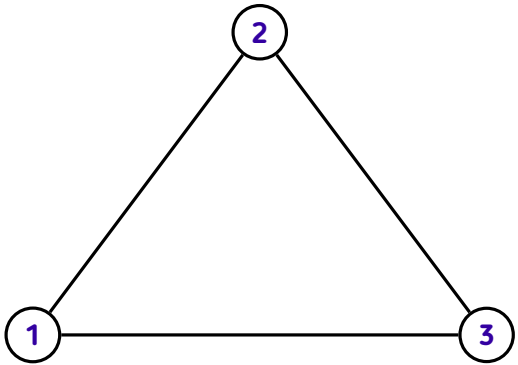
Exemplo de entrada e saída

```
3 3
1 2
2 3
1 3
```



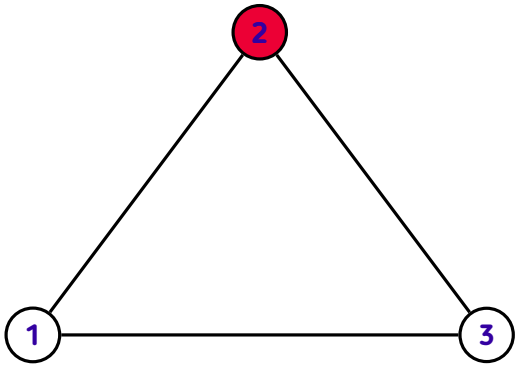
Exemplo de entrada e saída

```
3 3
1 2
2 3
1 3
```



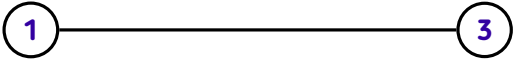
Exemplo de entrada e saída

3 3
1 2
2 3
1 3



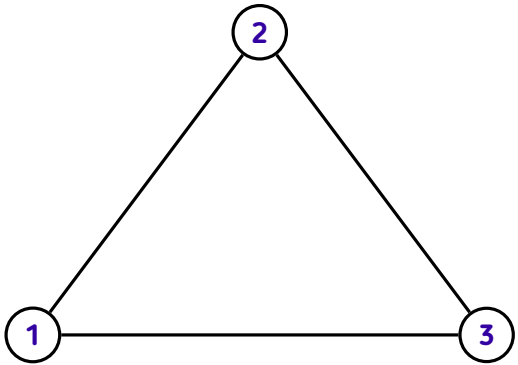
Exemplo de entrada e saída

```
3 3
1 2
2 3
1 3
```



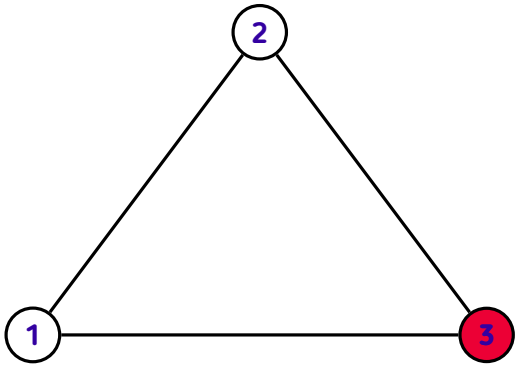
Exemplo de entrada e saída

```
3 3
1 2
2 3
1 3
```



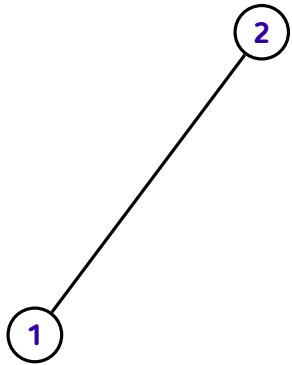
Exemplo de entrada e saída

```
3 3
1 2
2 3
1 3
```



Exemplo de entrada e saída

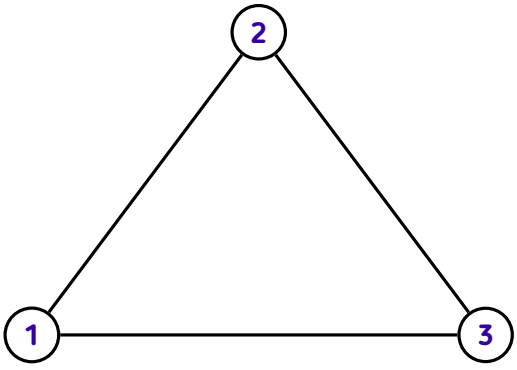
```
3 3
1 2
2 3
1 3
```



Exemplo de entrada e saída

3 3
1 2
2 3
1 3

0



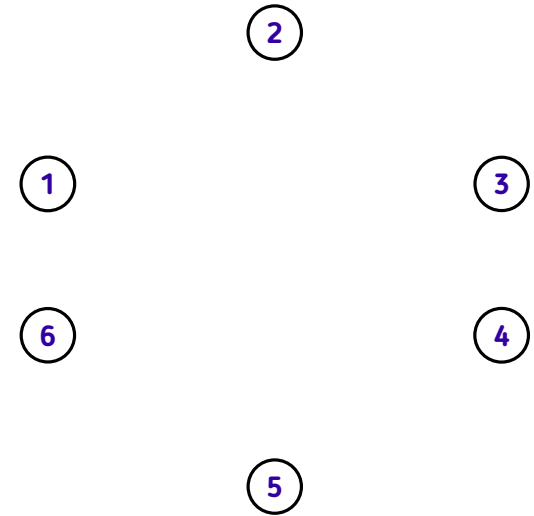
Exemplo de entrada e saída

Exemplo de entrada e saída

```
6 8
```

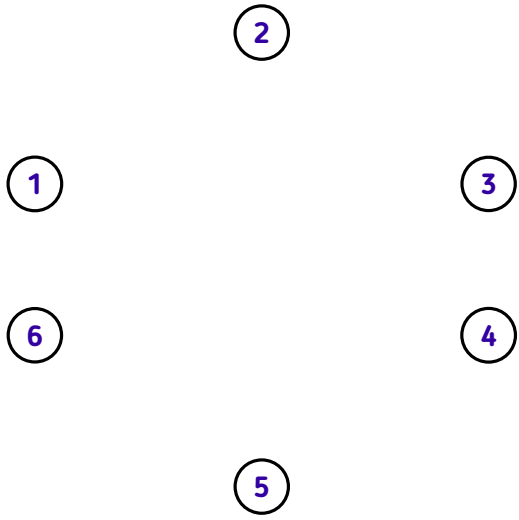
Exemplo de entrada e saída

6 8



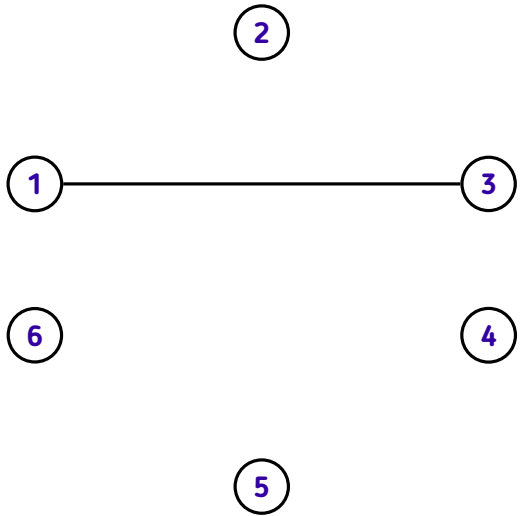
Exemplo de entrada e saída

```
6 8
1 3
```



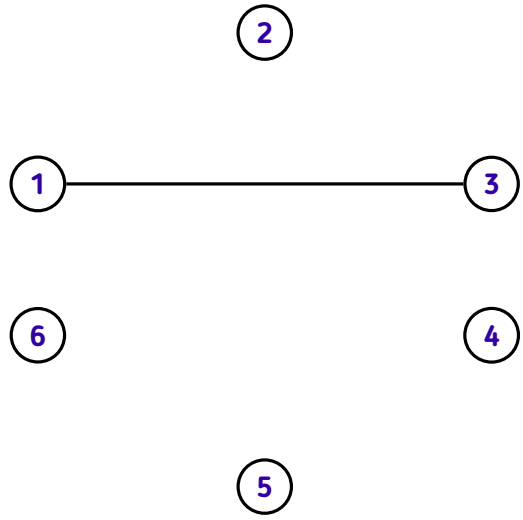
Exemplo de entrada e saída

6 8
1 3



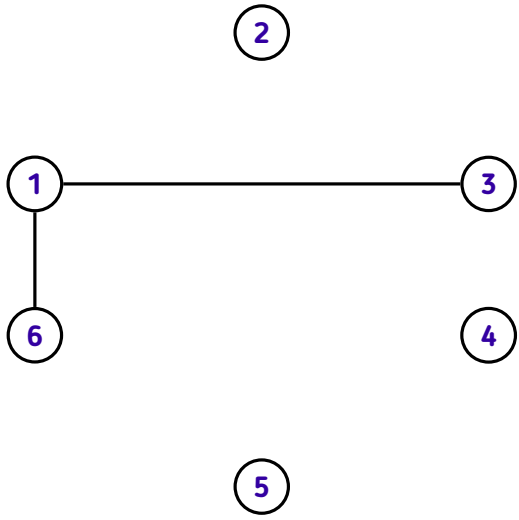
Exemplo de entrada e saída

```
6 8
1 3
6 1
```



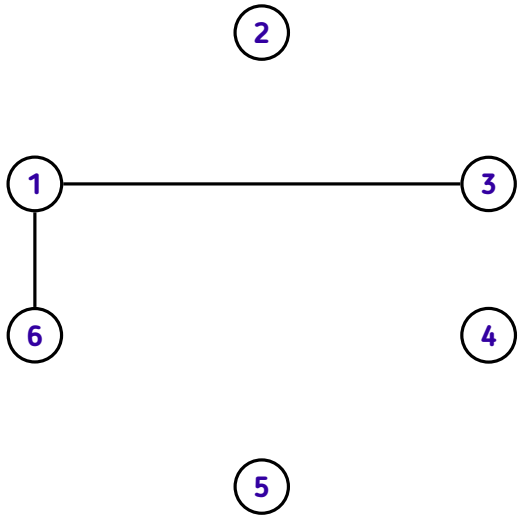
Exemplo de entrada e saída

6 8
1 3
6 1



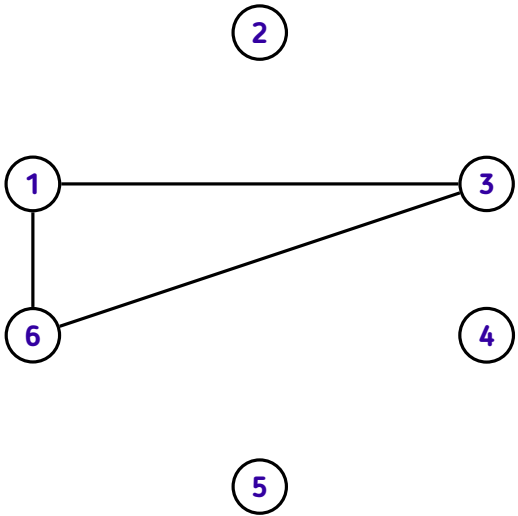
Exemplo de entrada e saída

```
6 8
1 3
6 1
6 3
```



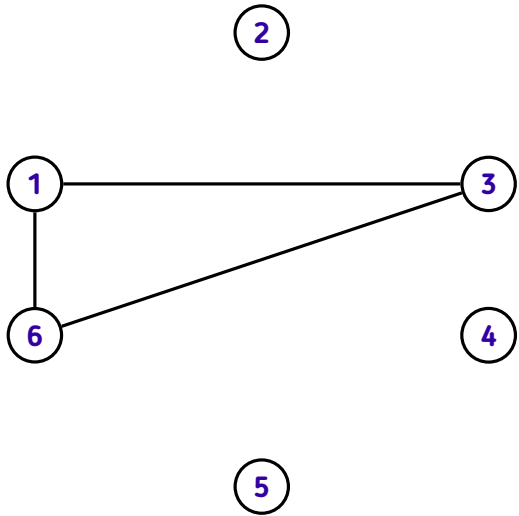
Exemplo de entrada e saída

```
6 8
1 3
6 1
6 3
```



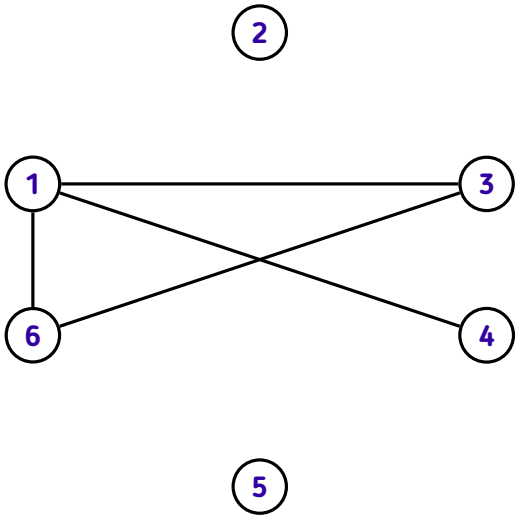
Exemplo de entrada e saída

```
6 8
1 3
6 1
6 3
4 1
```



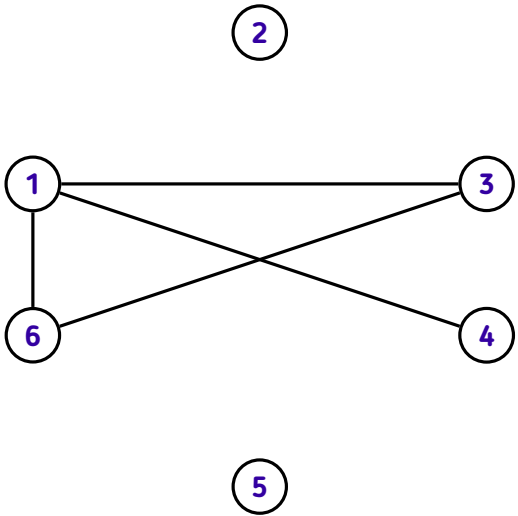
Exemplo de entrada e saída

```
6 8
1 3
6 1
6 3
4 1
```



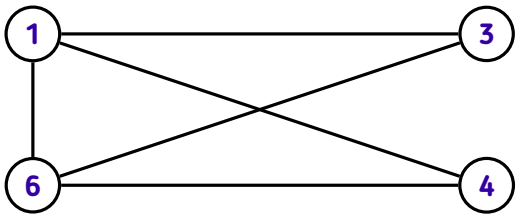
Exemplo de entrada e saída

```
6 8
1 3
6 1
6 3
4 1
6 4
```



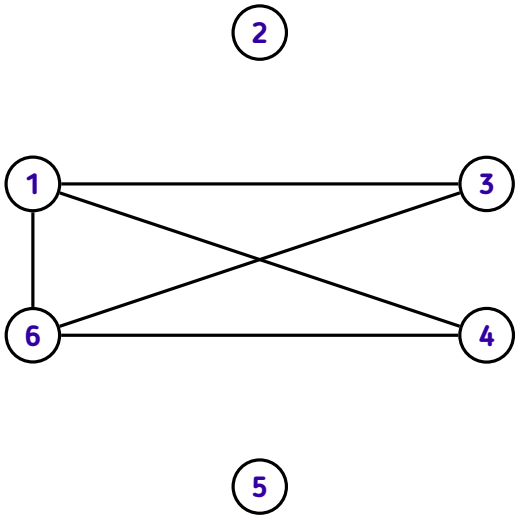
Exemplo de entrada e saída

```
6 8
1 3
6 1
6 3
4 1
6 4
```



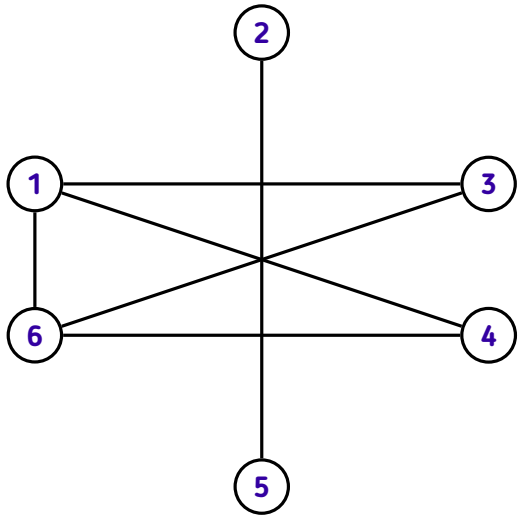
Exemplo de entrada e saída

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



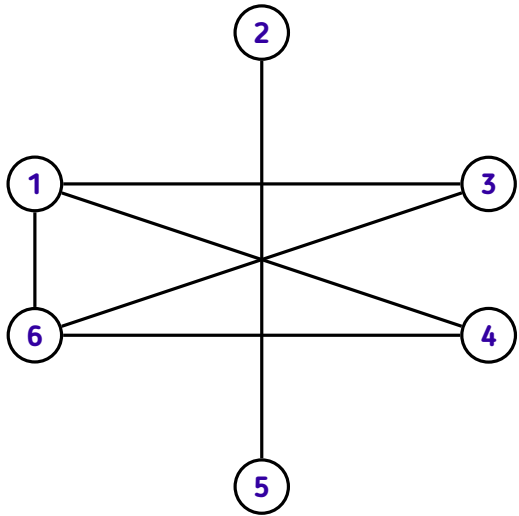
Exemplo de entrada e saída

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



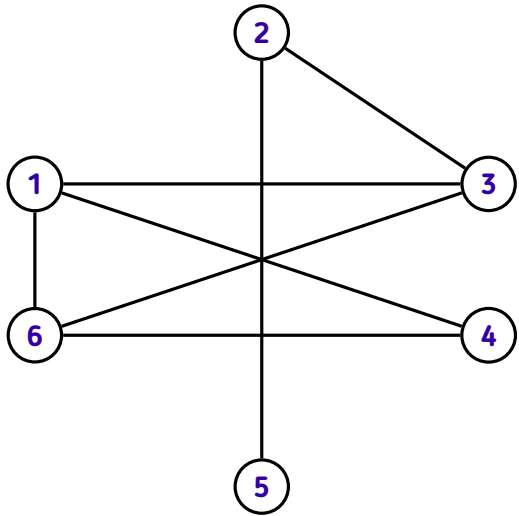
Exemplo de entrada e saída

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



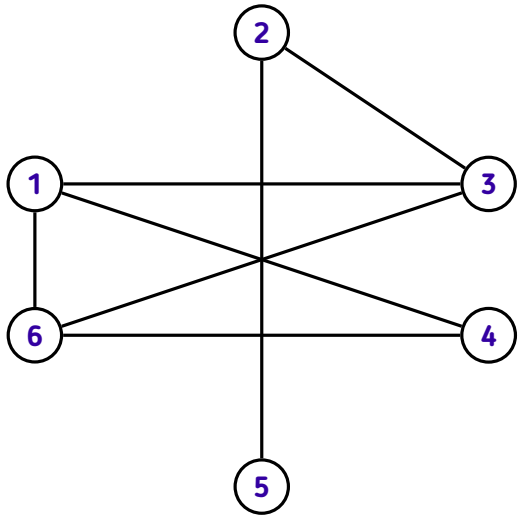
Exemplo de entrada e saída

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



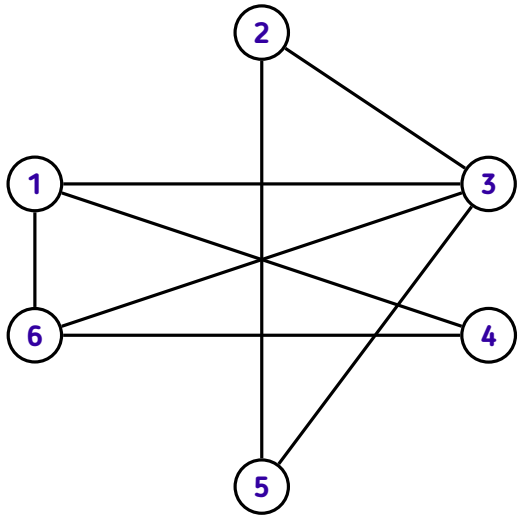
Exemplo de entrada e saída

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



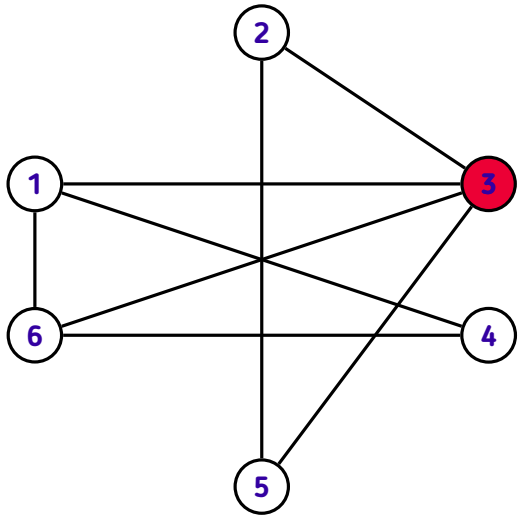
Exemplo de entrada e saída

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



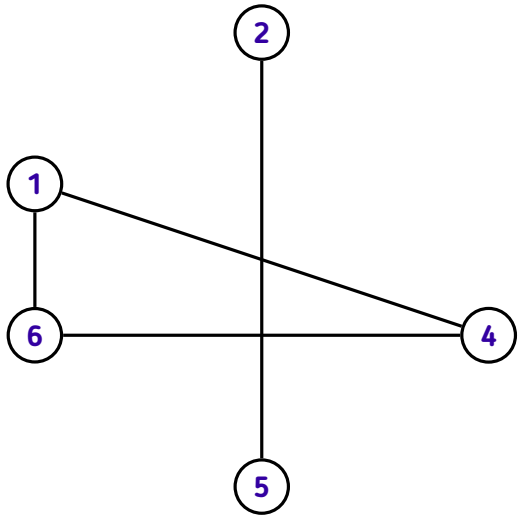
Exemplo de entrada e saída

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



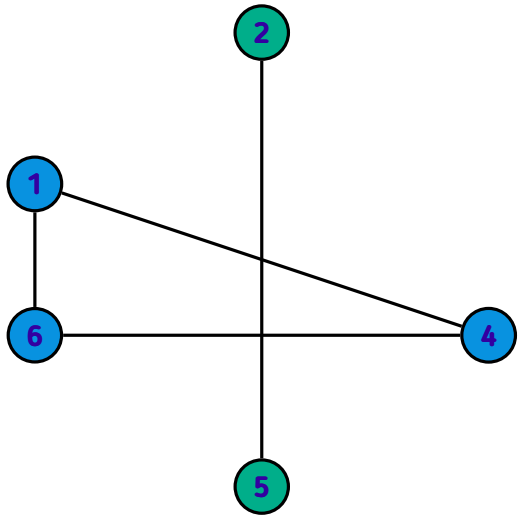
Exemplo de entrada e saída

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



Exemplo de entrada e saída

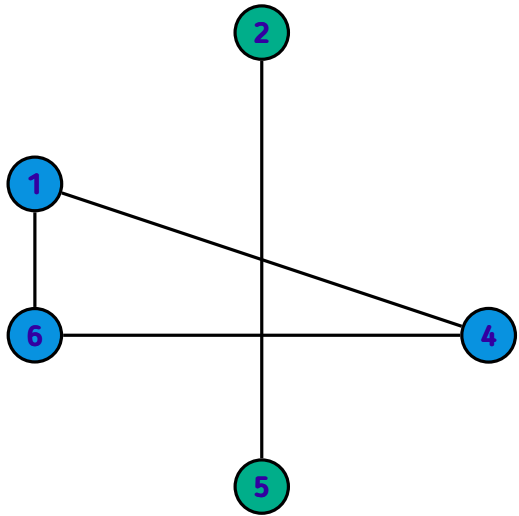
```
6 8
1 3
6 1
6 3
4 1
6 4
5 2
3 2
3 5
```



Exemplo de entrada e saída

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

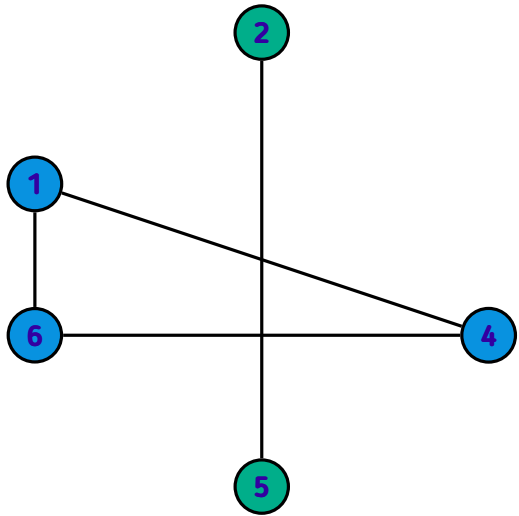
1



Exemplo de entrada e saída

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

1



Solução

Solução

Basta contar os pontos de articulação!

```
int dfs(int u, int p, int& next, set<int>& points)
{
    int children = 0;
    dfs_low[u] = dfs_num[u] = next++;

    for (auto v : adj[u])
        if (not dfs_num[v]) {
            ++children;

            dfs(v, u, next, points);

            if (dfs_low[v] >= dfs_num[u])
                points.insert(u);

            dfs_low[u] = min(dfs_low[u], dfs_low[v]);
        } else if (v != p)
            dfs_low[u] = min(dfs_low[u], dfs_num[v]);

    return children;
}
```

```
size_t solve(int N)
{
    memset(dfs_num, 0, (N + 1)*sizeof(int));
    memset(dfs_low, 0, (N + 1)*sizeof(int));

    set<int> points;

    for (int u = 1, next = 1; u <= N; ++u)
        if (not dfs_num[u])
        {
            auto children = dfs(u, u, next, points);

            if (children == 1)
                points.erase(u);
        }

    return points.size();
}
```