OJ 610

Street Directions

Prof. Edson Alves

Faculdade UnB Gama

According to the Automobile Collision Monitor (ACM), most fatal traffic accidents occur on twoway streets. In order to reduce the number of fatalities caused by traffic accidents, the mayor wants to convert as many streets as possible into one-way streets. You have been hired to perform this conversion, so that from each intersection, it is possible for a motorist to drive to all the other intersections following some route.

You will be given a list of streets (all two-way) of the city. Each street connects two intersections, and does not go through an intersection. At most four streets meet at each intersection, and there is at most one street connecting any pair of intersections. It is possible for an intersection to be the end point of only one street. You may assume that it is possible for a motorist to drive from each destination to any other destination when every street is a two-way street.

De acordo com o Automobile Collision Monitor (ACM), a maioria dos acidentes fatais de transito acontecem em vias de mão dupla. Para reduzir o número de fatalidades causadas por acidentes de trânsito, o prefeito quer tornar tantas vias quanto possíveis em vias de mão única. Você foi contratado para esta tarefa, de modo que, para cada interseção entre vias, os motoristas serão capazes de chegar a todas demais interseções por alguma rota.

Você irá receber uma lista de vias (todas de mão dupla) da cidade. Cada via conecta duas interseções, e não atravessa nenhuma interseção. No máximo quatro vias atingem cada interseção, e há no máximo uma via conectando qualquer par de interseções. É possível que uma interseção serja o ponto final de uma única via. Você pode assumir que é possível para um motorista ir a qualquer interseção a partir de qualquer interseção quando todas as vias são de mão dupla.

Input

The input consists of a number of cases. The first line of each case contains two integers n and m. The number of intersections is n $(2 \leq n \leq 1000)$, and the number of streets is m. The next m lines contain the intersections incident to each of the m streets. The intersections are numbered from 1 to n, and each street is listed once. If the pair i j is present, j i will not be present. End of input is indicated by $n = m = 0$.

Entrada

A entrada é composta por uma série de casos de teste. A primeira linha de cada caso de teste contém dois inteiros n e m. O número de interseções é n $(2 \leq n \leq 1000)$, e o número de vias é m. As próximas m linhas contém as interseções conectadas por cada uma das m vias. As interseções são numeradas de 1 a n, e cada via é lista uma única vez. Se o par $i\ j$ está presente, $j\ i$ não estará. O fim da entrada é indicado por $n = m = 0$.

Output

For each case, print the case number (starting from 1) followed by a blank line. Next, print on separate lines each street as the pair i j to indicate that the street has been assigned the direction going from intersection i to intersection j. For a street that cannot be converted into a one-way street, print both i j and j i on two different lines. The list of streets can be printed in any order. Terminate each case with a line containing a single '#' character.

***Note**: There may be many possible direction assignments satisfying the requirements. Any such assignment is acceptable.*

Saída

Para cada caso de teste, imprima o número do caso (começando em 1) seguido de uma linha em branco. Em seguida, imprima em linhas separadas cada via como um par $i\ j$, indicando que foi atribuído à via o sentido que parte da interseção i para a interseção j. Se uma via não puder ser convertida para uma via de mão única, imprima ambos $i\ j$ e $j\ i$ em duas linhas diferentes. A lista de vias pode ser impressa em qualquer ordem. Finalize cada caso com uma linha contendo o caractere '#'.

Nota: Há muitas maneiras distintas de atribuição das vias que satisfazem os requisitos. Qualquer uma delas é aceitável.

Exemplo de entrada e saída

Exemplo de entrada e saída

```
7 10
```

Exemplo de entrada e saída

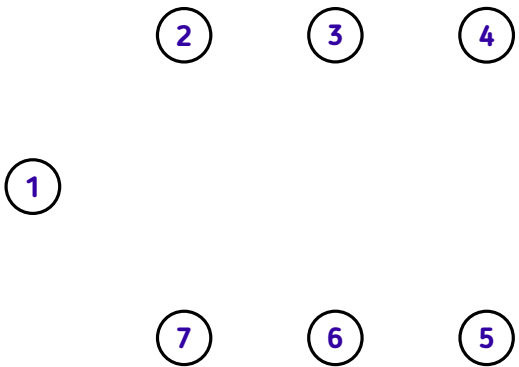
7 10

 # de interseções

Exemplo de entrada e saída

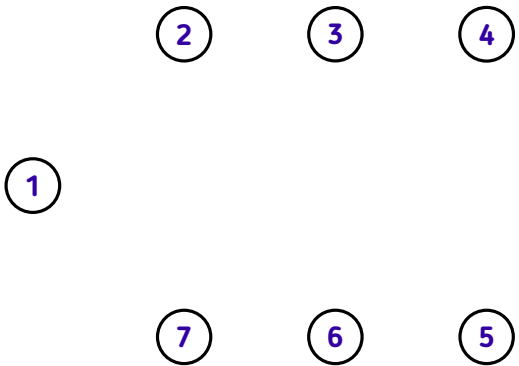
Exemplo de entrada e saída

7 10



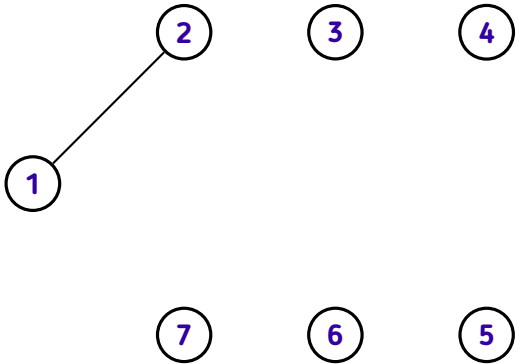
Exemplo de entrada e saída

```
7 10
1 2
```



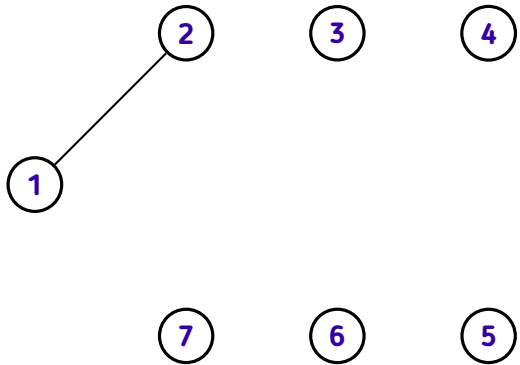
Exemplo de entrada e saída

```
7 10
1 2
```



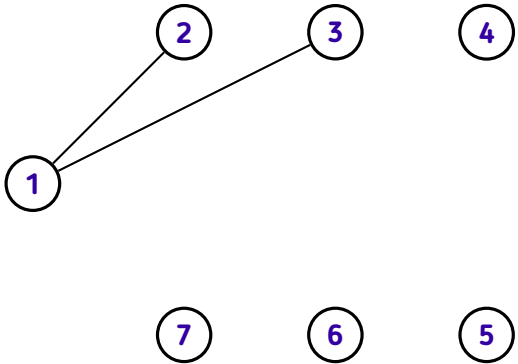
Exemplo de entrada e saída

```
7 10
1 2
1 3
```



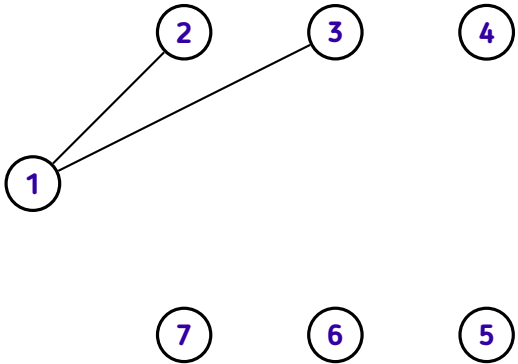
Exemplo de entrada e saída

```
7 10
1 2
1 3
```



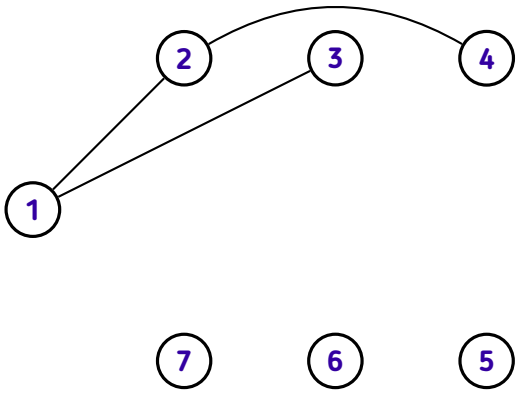
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
```



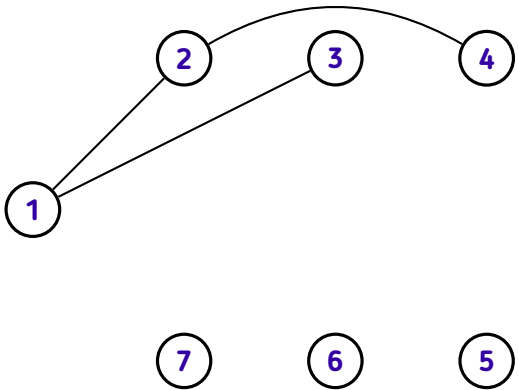
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
```



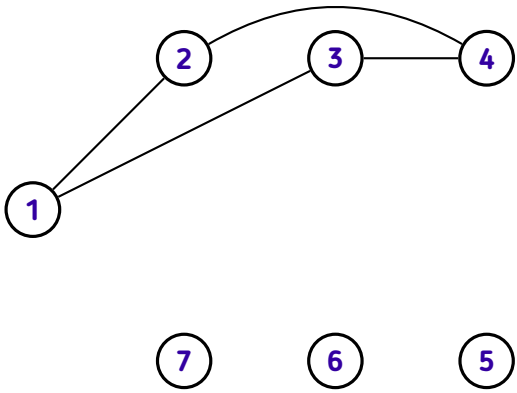
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
```



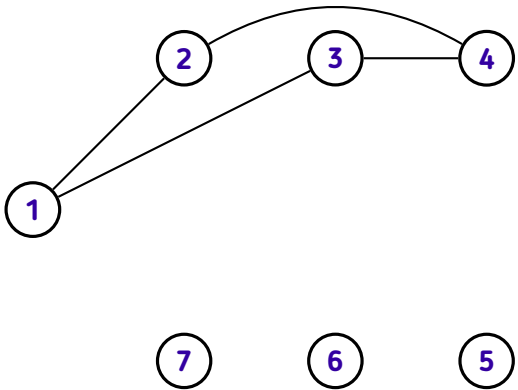
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
```



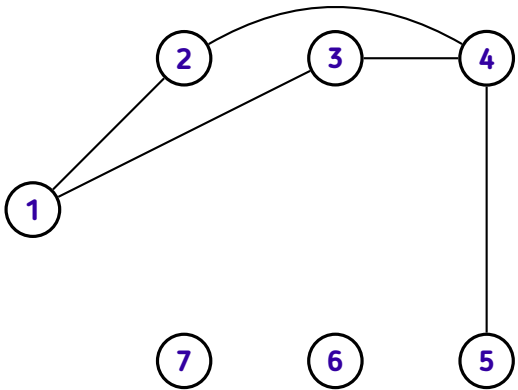
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
```



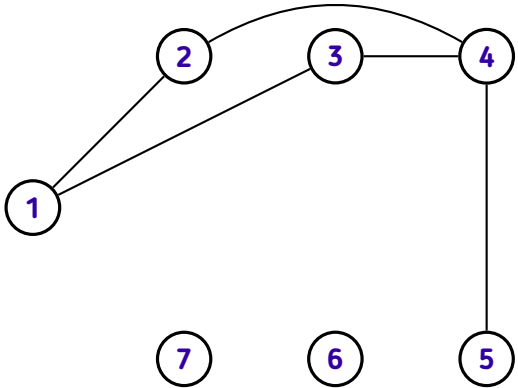
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
```



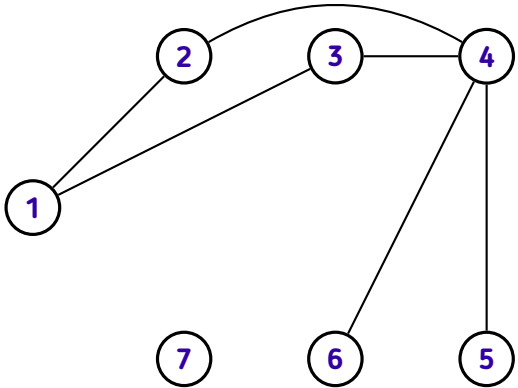
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
```



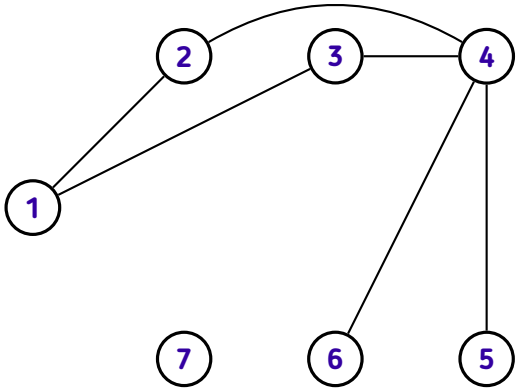
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
```



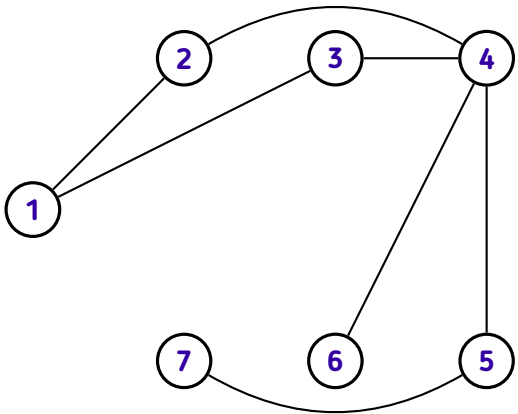
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
```



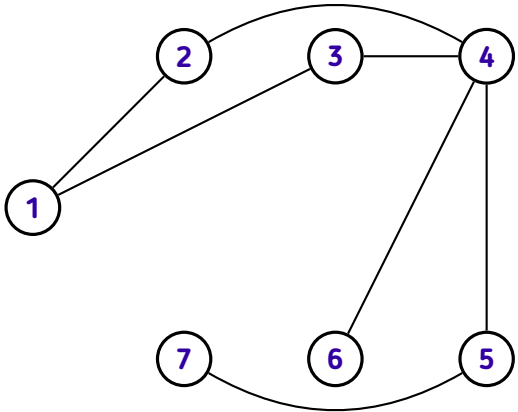
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
```



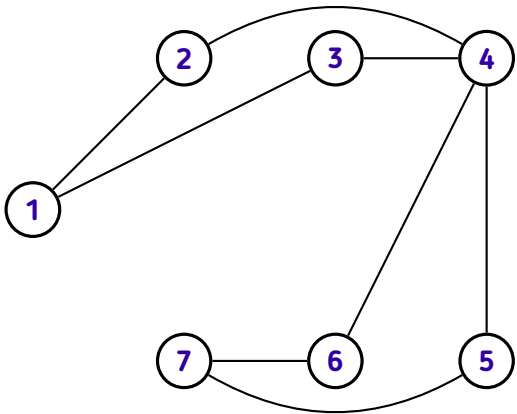
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
```



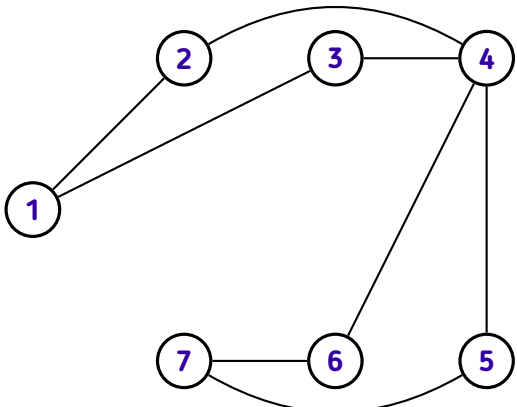
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
```



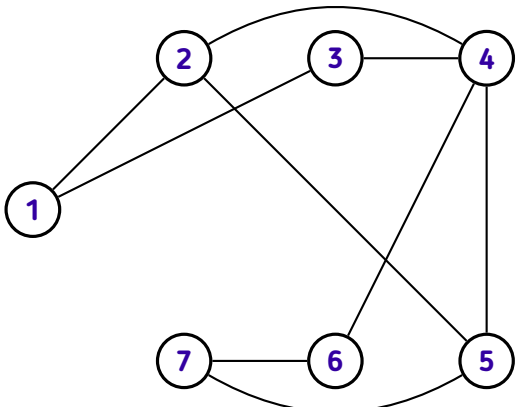
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
```



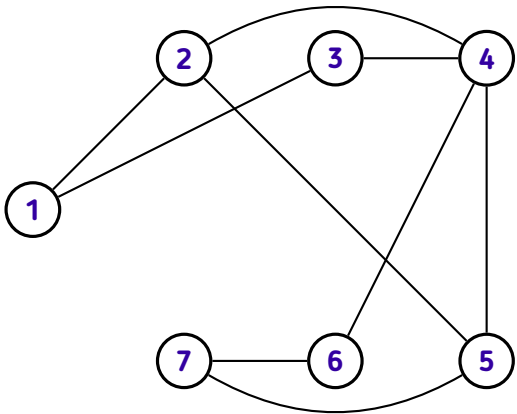
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
```



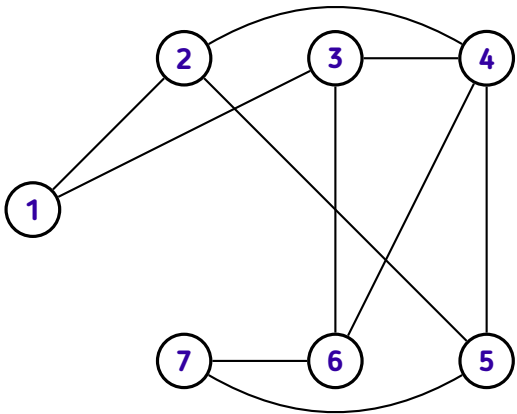
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



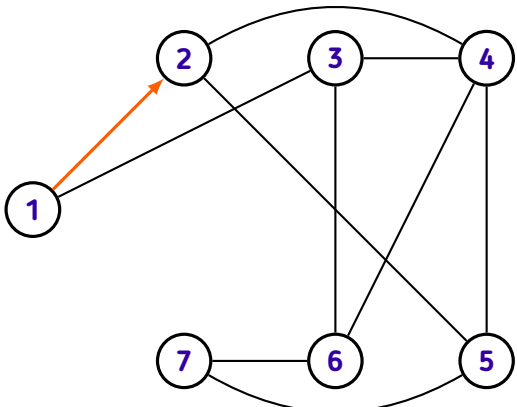
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



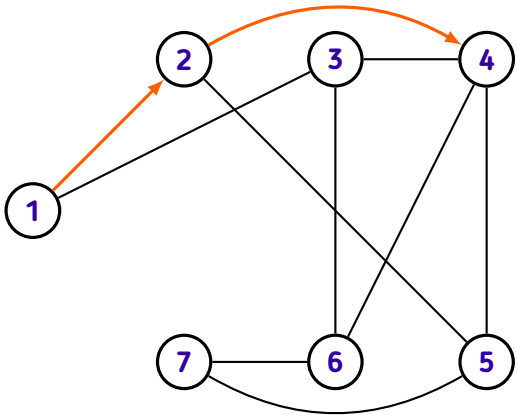
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



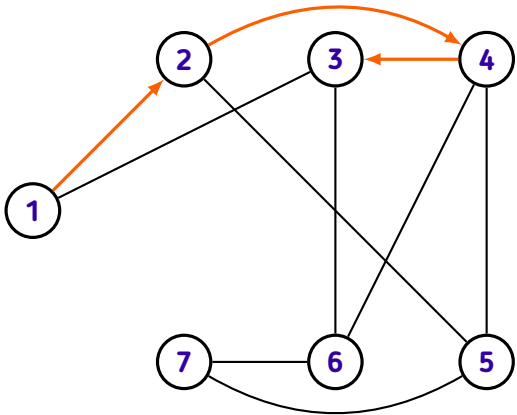
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



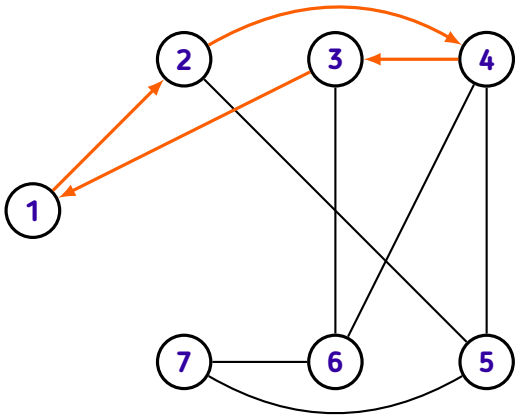
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



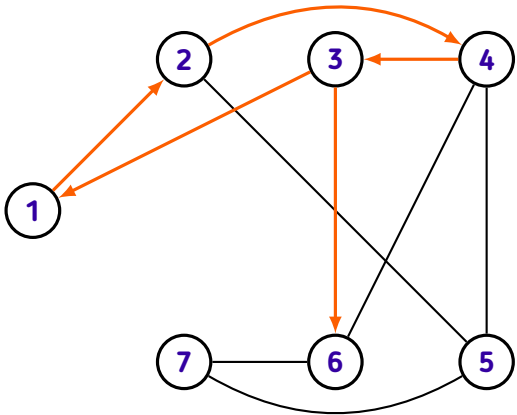
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



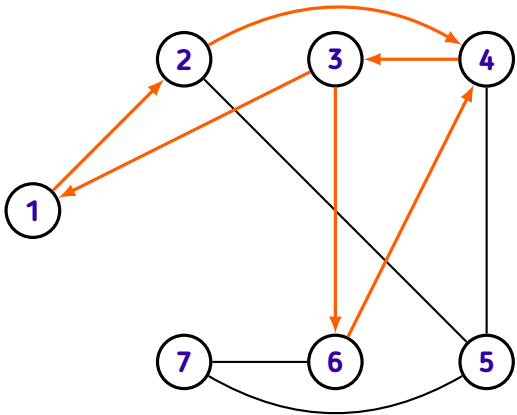
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



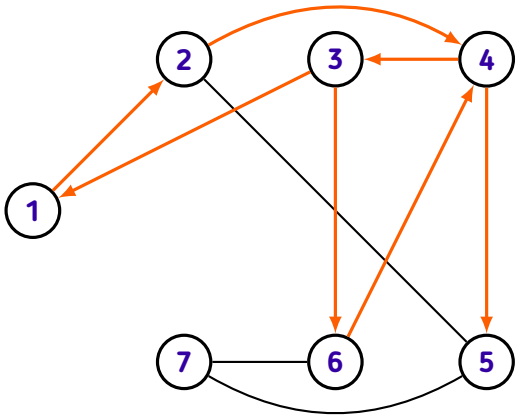
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



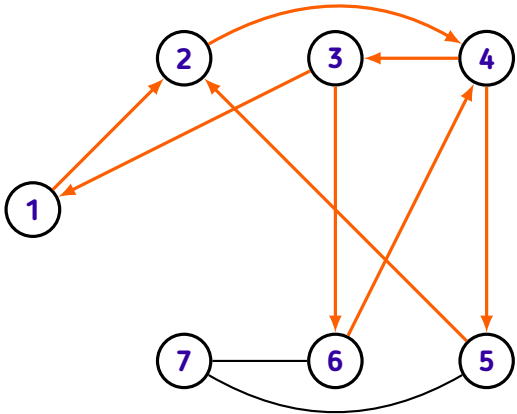
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



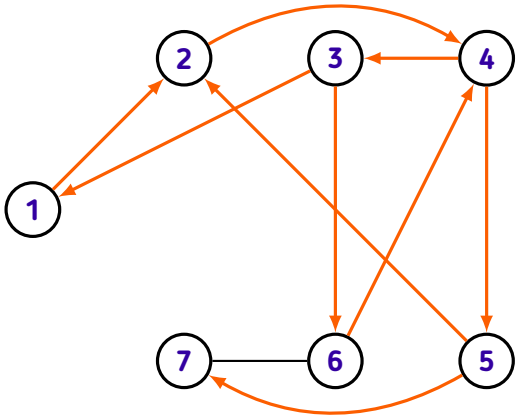
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



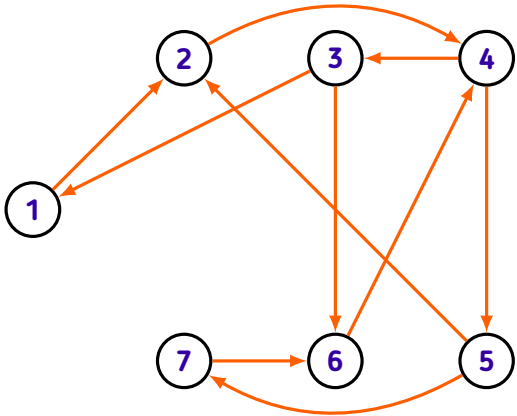
Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



Exemplo de entrada e saída

```
7 10
1 2
1 3
2 4
3 4
4 5
4 6
5 7
6 7
2 5
3 6
```



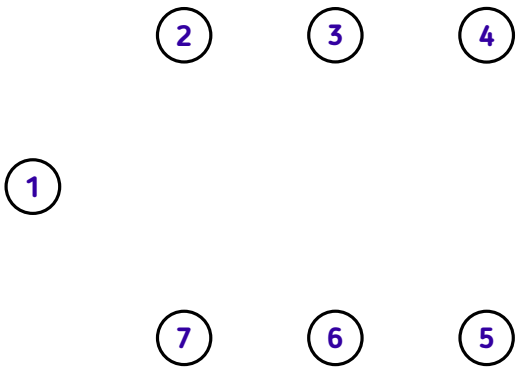
Exemplo de entrada e saída

Exemplo de entrada e saída

```
7 9
```

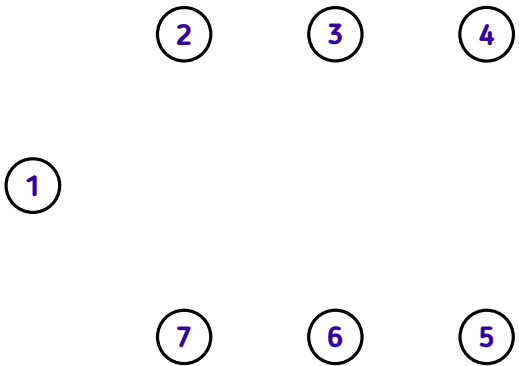
Exemplo de entrada e saída

7 9



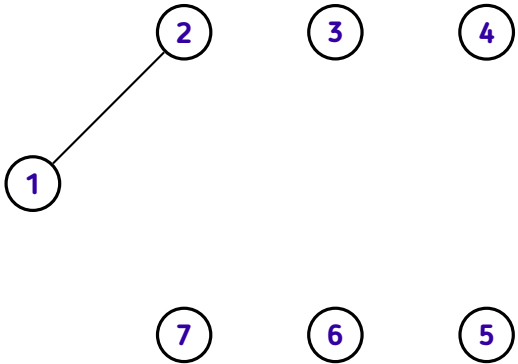
Exemplo de entrada e saída

```
7 9
1 2
```



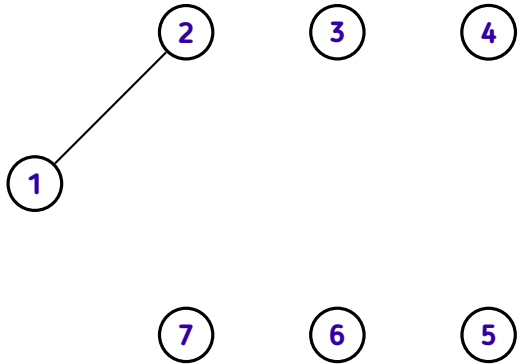
Exemplo de entrada e saída

7 9
1 2



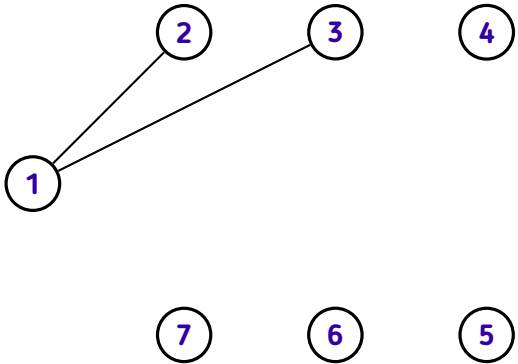
Exemplo de entrada e saída

```
7 9
1 2
1 3
```



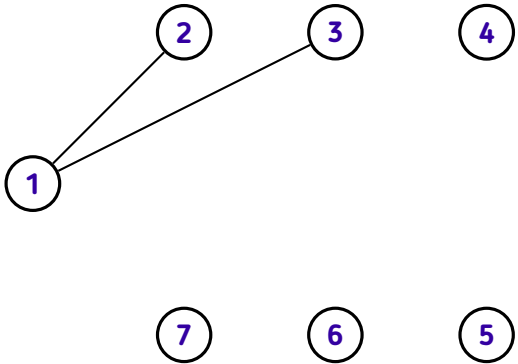
Exemplo de entrada e saída

```
7 9
1 2
1 3
```



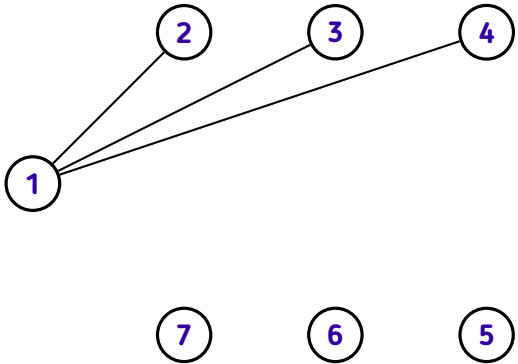
Exemplo de entrada e saída

```
7 9
1 2
1 3
1 4
```



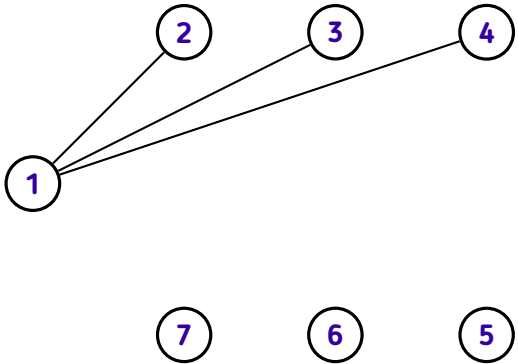
Exemplo de entrada e saída

```
7 9
1 2
1 3
1 4
```



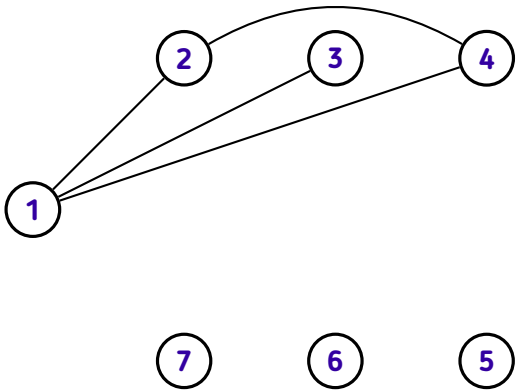
Exemplo de entrada e saída

```
7 9
1 2
1 3
1 4
2 4
```



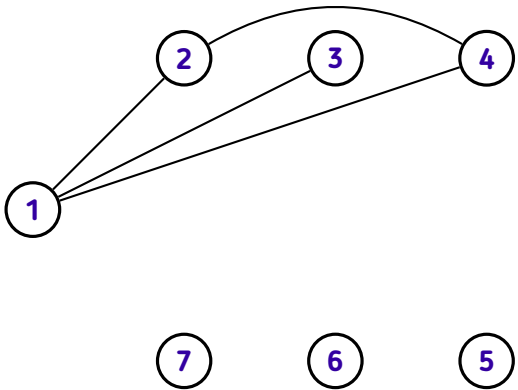
Exemplo de entrada e saída

```
7 9
1 2
1 3
1 4
2 4
```



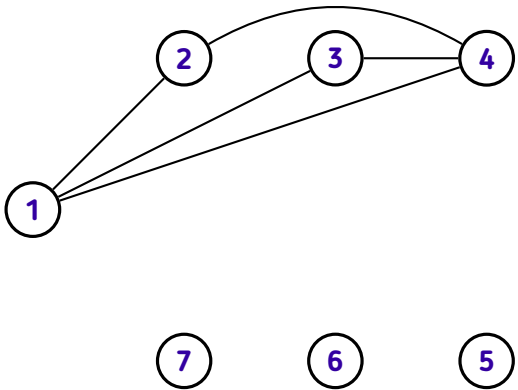
Exemplo de entrada e saída

```
7 9
1 2
1 3
1 4
2 4
3 4
```



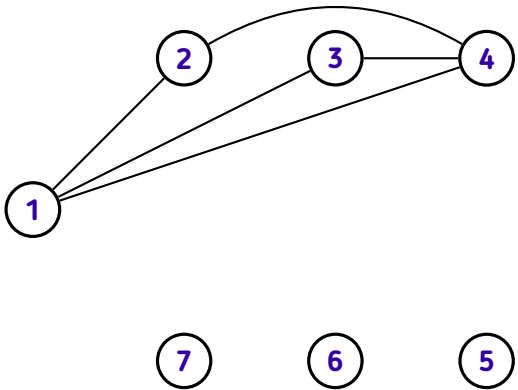
Exemplo de entrada e saída

```
7 9
1 2
1 3
1 4
2 4
3 4
```



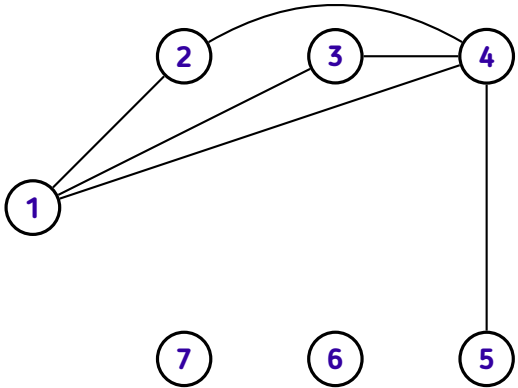
Exemplo de entrada e saída

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



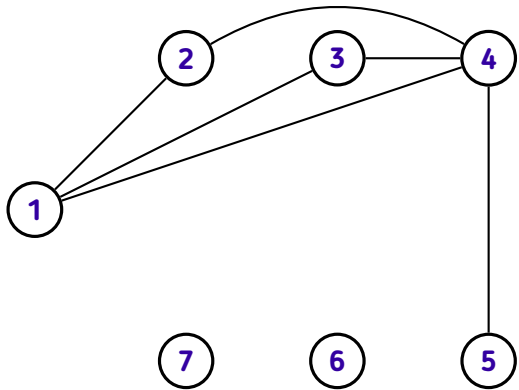
Exemplo de entrada e saída

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



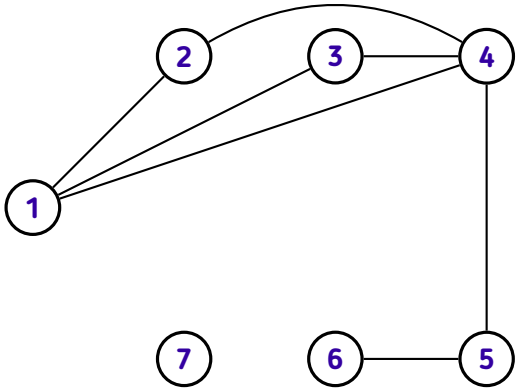
Exemplo de entrada e saída

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



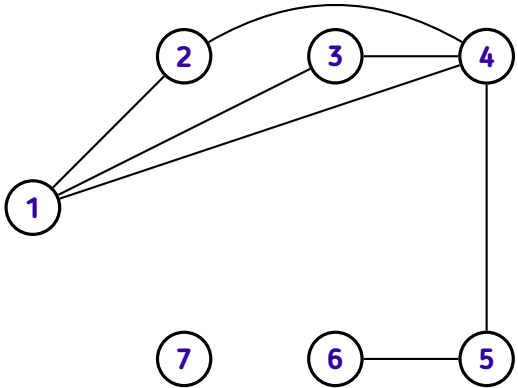
Exemplo de entrada e saída

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



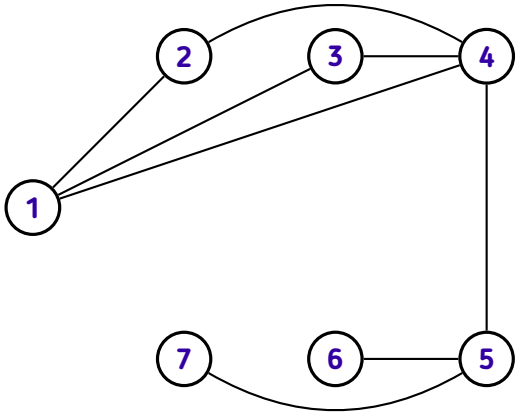
Exemplo de entrada e saída

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



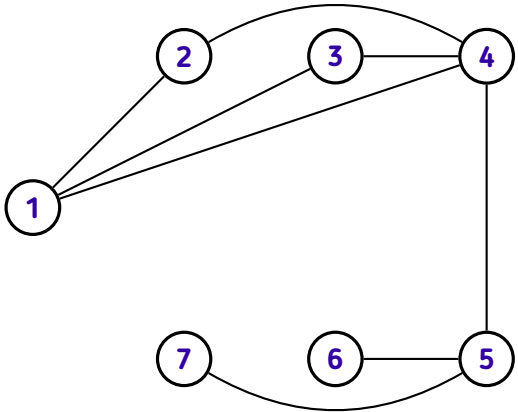
Exemplo de entrada e saída

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



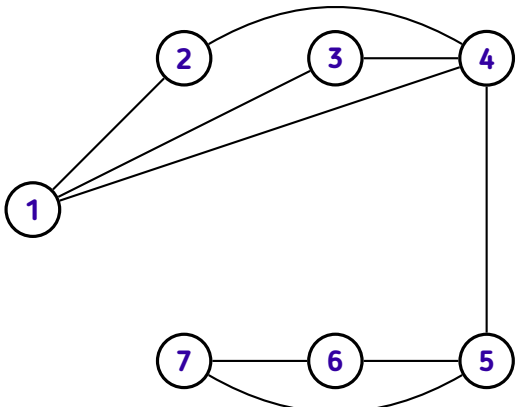
Exemplo de entrada e saída

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



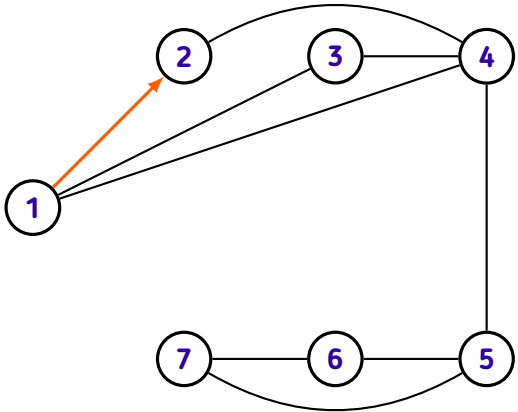
Exemplo de entrada e saída

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



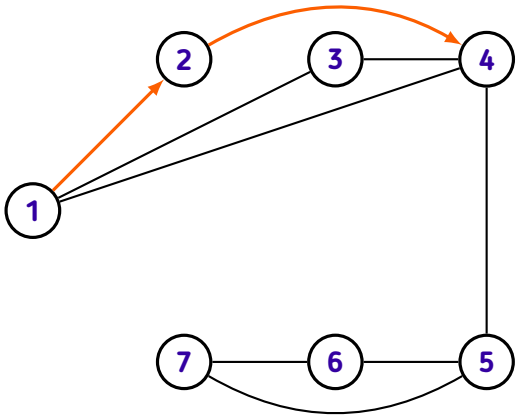
Exemplo de entrada e saída

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



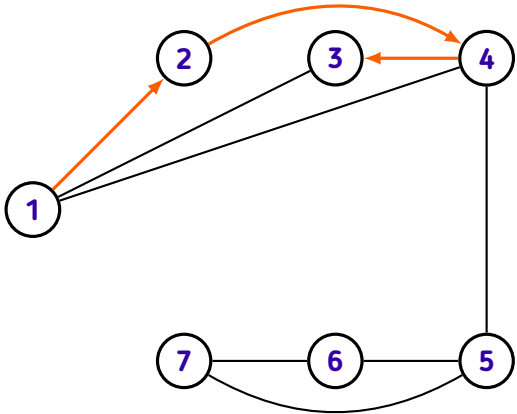
Exemplo de entrada e saída

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



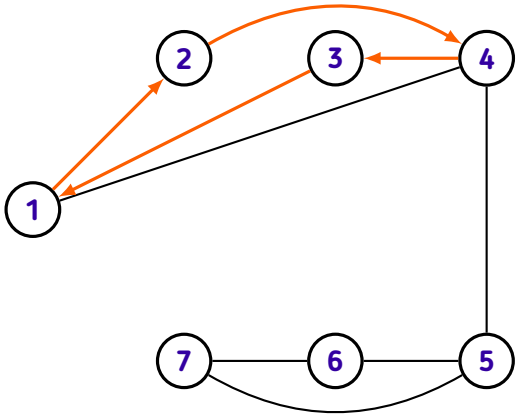
Exemplo de entrada e saída

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



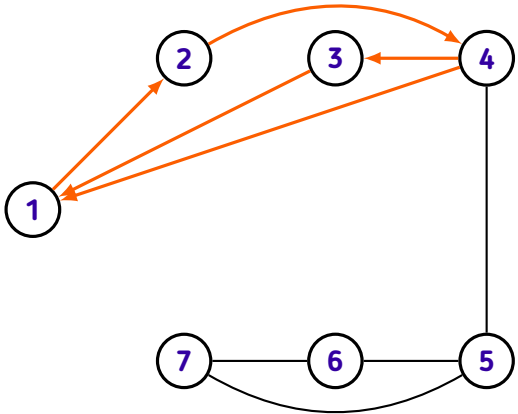
Exemplo de entrada e saída

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



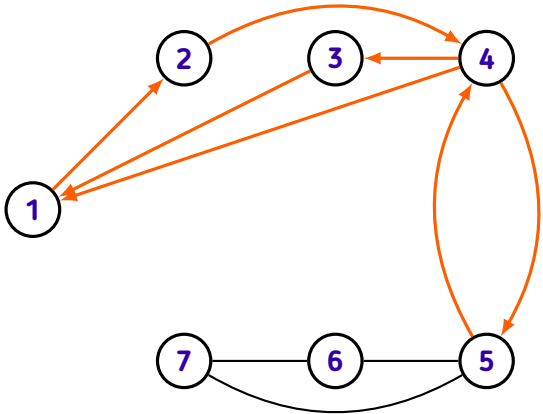
Exemplo de entrada e saída

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



Exemplo de entrada e saída

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



Solução

Solução

- ⋆ **Para cada estudante a são necessárias, no máximo, $n + 1$ verificações**

Solução

- ⋆ **Para cada estudante a são necessárias, no máximo, $n + 1$ verificações**
- ⋆ **Estas verificações podem ser feitas por meio de uma DFS**

Solução

- ⋆ **Para cada estudante a são necessárias, no máximo, $n + 1$ verificações**
- ⋆ **Estas verificações podem ser feitas por meio de uma DFS**
- ⋆ **Há n estudantes distintos**

Solução

- ⋆ **Para cada estudante a são necessárias, no máximo, $n + 1$ verificações**
- ⋆ **Estas verificações podem ser feitas por meio de uma DFS**
- ⋆ **Há n estudantes distintos**
- ⋆ **A complexidade da solução é $O(n^2)$**

```
void dfs(int u, int p, int& next, set<edge>& streets)
{
    dfs_low[u] = dfs_num[u] = next++;

    for (auto v : adj[u]) {
        if (not streets.count(edge(u, v)) and not streets.count(edge(v, u)))
            streets.insert(edge(u, v));

        if (not dfs_num[v]) {

            dfs(v, u, next, streets);

            if (dfs_low[v] > dfs_num[u])
                streets.insert(edge(v, 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]);
    }
}
```

```
set<edge> solve(int N)
{
    memset(dfs_num, 0, (N + 1)*sizeof(int));
    memset(dfs_low, 0, (N + 1)*sizeof(int));

    set<edge> streets;

    for (int u = 1, next = 1; u <= N; ++u)
        if (not dfs_num[u])
            dfs(u, u, next, streets);

    return streets;
}
```