OJ 10505

Montesco vs Capuleto

Prof. Edson Alves

Faculdade UnB Gama

Romeo and Juliet have finally decided to get married. But preparing the wedding party will not be so easy, as it is well-known that their respective families –the Montesco and the Capuleto– are bloody enemies. In this problem you will have to decide which person to invite and which person not to invite, in order to prevent a slaughter.

We have a list of N people who can be invited to the party or not. For every person i, we have a list of his enemies: $E_1, E_2, \ldots, E_p$. The "enemy" relationship has the following properties:

Anti-transitive. ***If a is an enemy of b, and b is an enemy of c, then a is a friend of c. Also, the enemies of the friends of a are his enemies, and the friends of the friends of a are his friends.***

Symmetrical. ***If a is an enemy of b, then b is an enemy of a (although it may not be indicated in his list of enemies).***

Romeu e Julieta decidiram finalmente se casar. Mas os preparativos para a festa do casamento não serão fáceis, uma vez que é bem conhecido que suas respectivas famílias – os Montescos e os Capuletos – são inimigos mortais. Neste problema você deve decidir quais pessoas convidar ou não, para prevenir um massacre.

Nós temos uma listas das N pessoas que podem ser convidadas para a festa ou não. Para cada pessoa i, nos temos uma lista de seus inimigos: $E_1, E_2, \ldots, E_p$. A relação de "inimizade" tem as seguintes propriedades:

Anti-transitiva. **Se a é um inimigo de b, e b é um inimigo de c, então a é amigo de c. Além disso, os inimigos dos amigos de a são seus inimigos, e os amigos dos amigos de a são seus amigos.**

Simétrica. **Se a é inimigo de b, então b é inimigo de a (embora isto possa não estar indicado na lista de seus inimigos).**

One person will accept an invitation to the party if, and only if, he is invited, all his friends are invited and none of his enemies is invited. You have to find the maximum number of people that can be invited, so that all of them accept their invitation.

For instance, if $N = 5$, and we know that: 1 is enemy of 3, 2 is enemy of 1, and 4 is enemy of 5, then we could invite a maximum of 3 people. These people could be 2, 3 and 4, but for this problem we only want the number of people invited.

Uma pessoa aceitará o convite para a festa se, e somente se, ele foi convidado, todos os seus amigos foram convidados e nenhum de seus inimigos foi convidado. Você deve escolhar o número máximo de pessoas que podem ser convidadas, de modo que todos aceitem o convite.

Por exemplo, se $N = 5$ e nós sabemos que: 1 é inimigo de 3, 2 é inimigo de 1, e 4 é inimigo de 5, então nós deveríamos convidar no máximo 3 pessoas. Estas pessoas poderiam ser 2, 3 e 4, mas para este problema nos queremos apenas o número de pessoas convidadas.

Input

The first line of the input file contains the number M of test cases in this file. A blank line follows this number, and a blank line is also used to separate test cases. The first line of each test case contains an integer N, indicating the number of people who have to be considered. You can assume that $N \leq 200$. For each of these N people, there is a line with his list of enemies. The first line contains the list of enemies of person 1, the second line contains the list of enemies of person 2, and so on. Each list of enemies starts with an integer p (the number of known enemies of that person) and then, there are p integers (the p enemies of that person). So, for example, if a person's enemies are 5 and 7, his list of enemies would be: '2 5 7'.

Entrada

A primeira linha da entrada contém o número M de casos de teste. Uma linha em branco seguirá este número, e uma linha em branco também será usada para separar os casos de teste. A primeira linha de cada caso de teste contém um inteiro N, indicando o número de pessoas que foram consideradas. Você pode assumir que $N \leq 200$. Para cada uma destas N pessoas há uma linha com a lista de seus inimigos. A primeira linha contém a lista de inimigos da pessoa 1, a segunda linha contéme a lista de inimigos da pessoa 2, e assim por diante. Cada lista de inimigos inicia com um inteiro p (o número de inimigos conhecidos daquela pessoa) e, em seguida, há p inteiros (os p inimigos daquela pessoa). Por exemplo, se os inimigos da pessoa são 5 e 7, sua lista de inimigos seria: '2 5 7'.

Output

For each test case, the output should consist of a single line containing an integer, the maximum number of people who can be invited, so that all of them accept their invitation.

Saída

Para cada caso de teste a saída deverá consistir em uma única linha contendo um inteiro, o número máximo de pessoas que podem ser convidadas, de modo que todas elas aceitem o convite.

Exemplo de entrada e saída

Exemplo de entrada e saída

```
3
```

Exemplo de entrada e saída

3

 # de casos de teste

Exemplo de entrada e saída

```
3
```

Exemplo de entrada e saída

3
5

Exemplo de entrada e saída

3

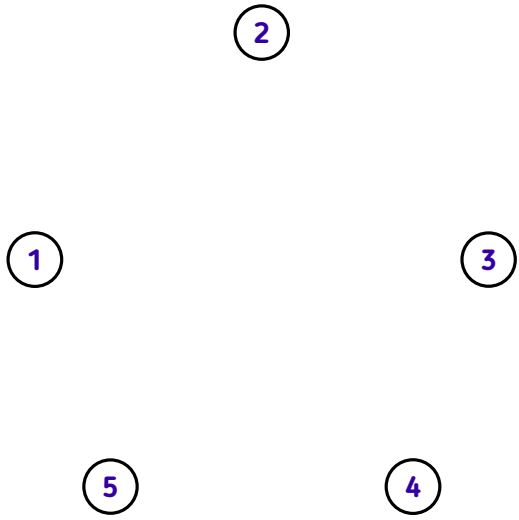
5

 # de pessoas consideradas

Exemplo de entrada e saída

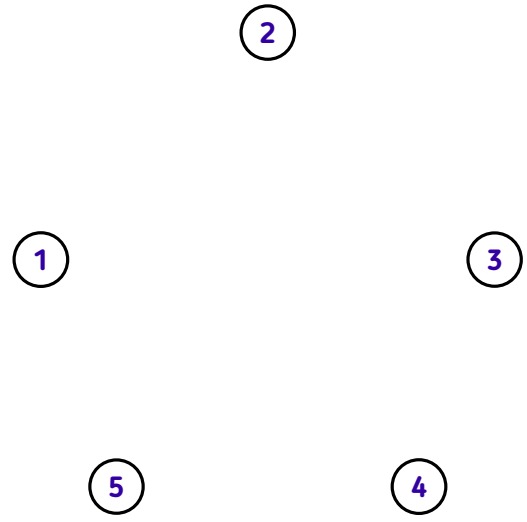
3

5



Exemplo de entrada e saída

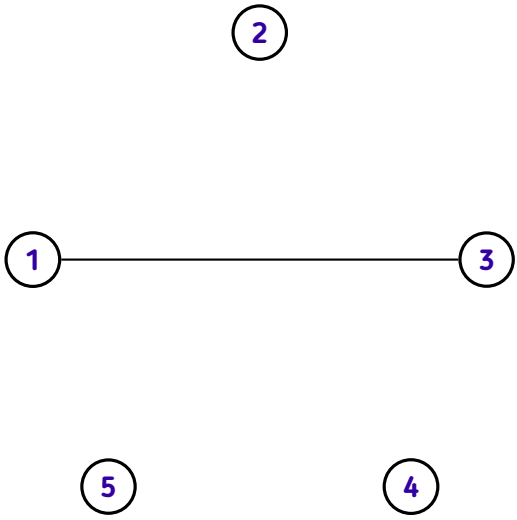
```
3

5
1 3
```



Exemplo de entrada e saída

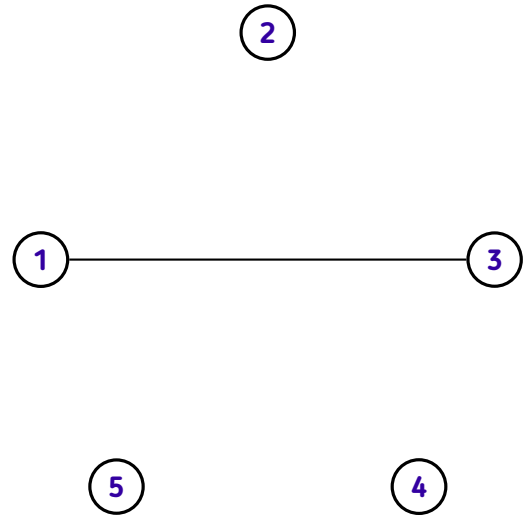
3

5
1 3



Exemplo de entrada e saída

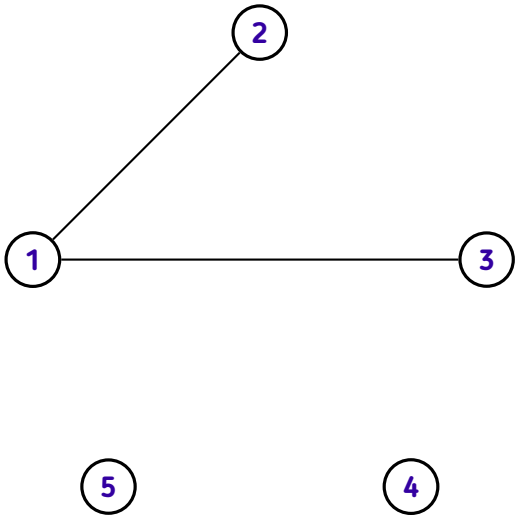
```
3

5
1 3
1 1
```



Exemplo de entrada e saída

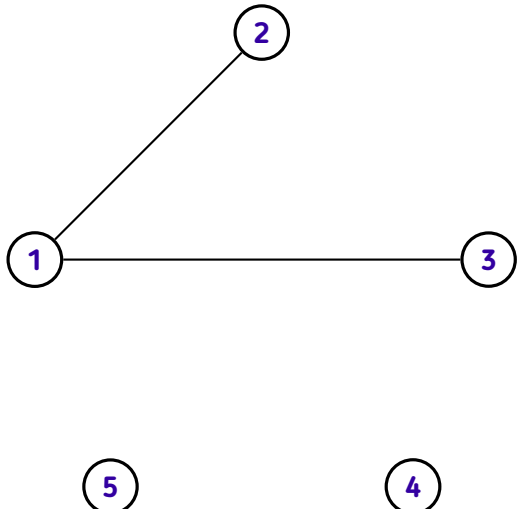
3

5
1 3
1 1



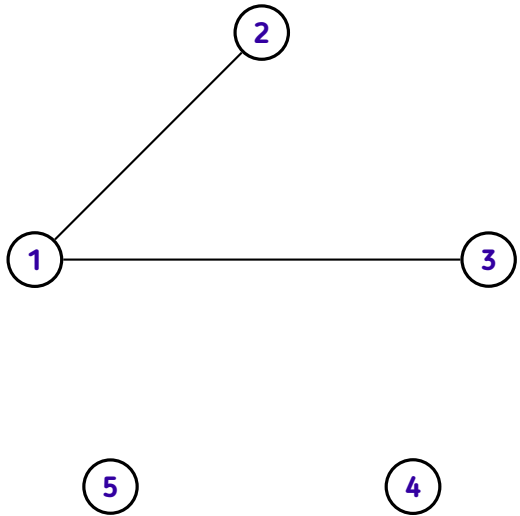
Exemplo de entrada e saída

```
3

5
1 3
1 1
0
```



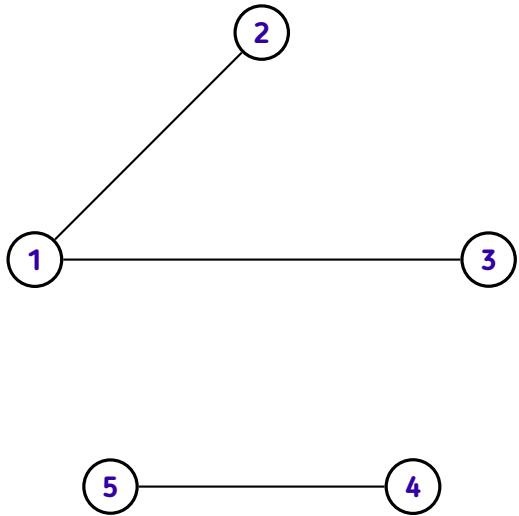
Exemplo de entrada e saída

```
3

5
1 3
1 1
0
1 5
```



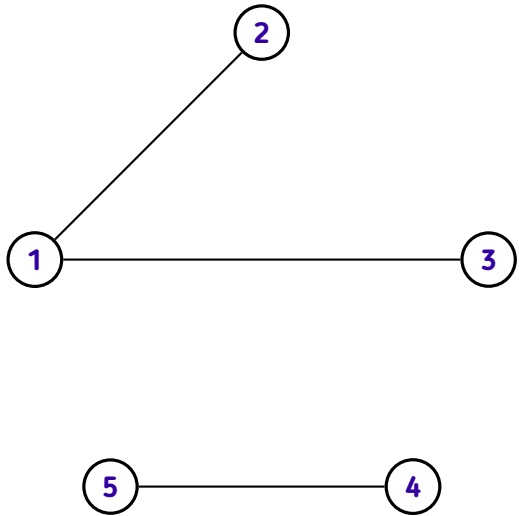
Exemplo de entrada e saída

```
3

5
1 3
1 1
0
1 5
```



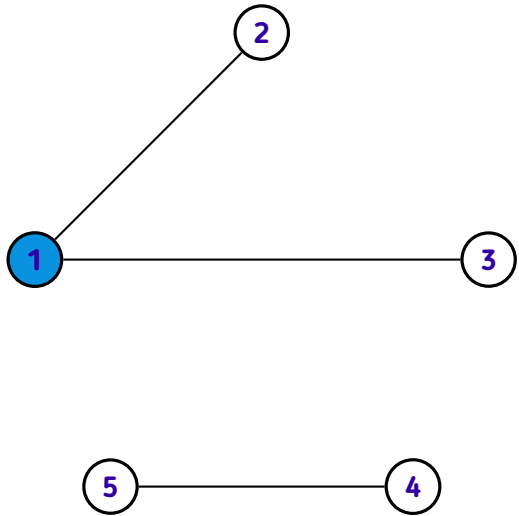
Exemplo de entrada e saída

```
3

5
1 3
1 1
0
1 5
0
```



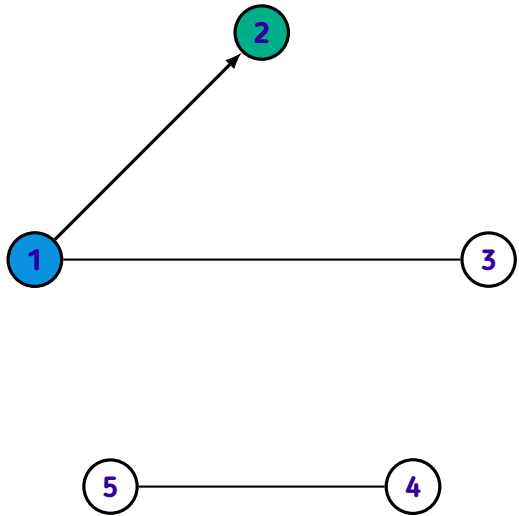
Exemplo de entrada e saída

```
3

5
1 3
1 1
0
1 5
0
```



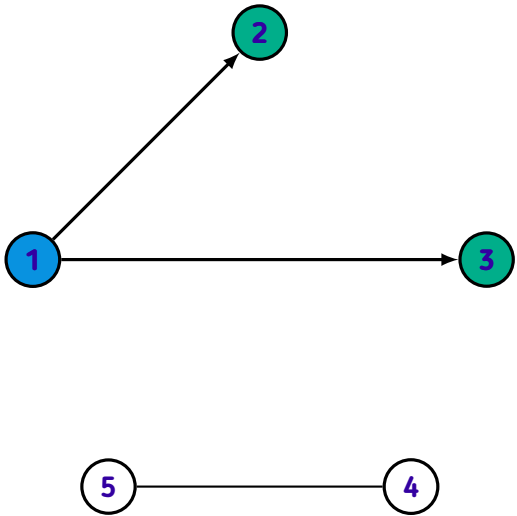
Exemplo de entrada e saída

```
3

5
1 3
1 1
0
1 5
0
```



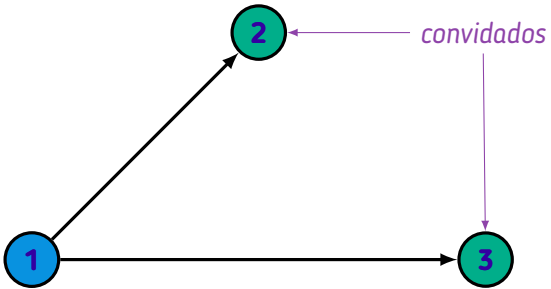
Exemplo de entrada e saída

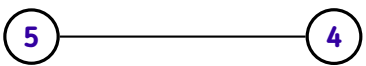
```
3

5
1 3
1 1
0
1 5
0
```



Exemplo de entrada e saída

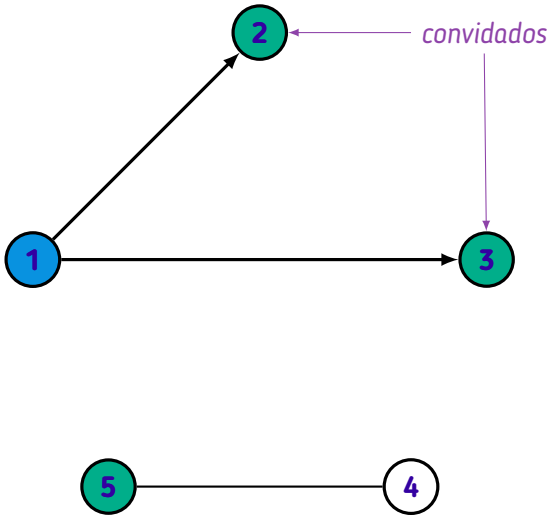
```
3

5
1 3
1 1
0
1 5
0
```





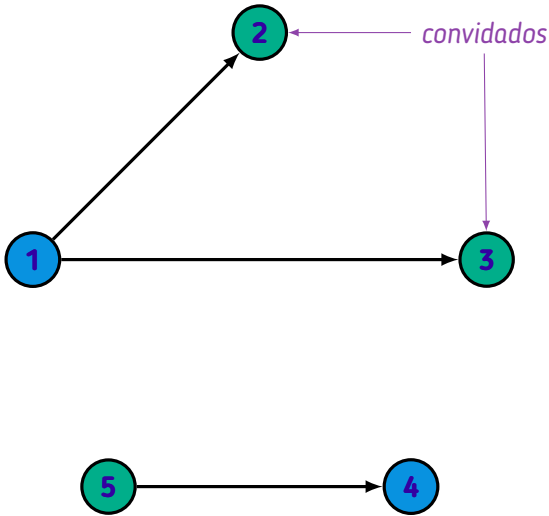
Exemplo de entrada e saída

```
3

5
1 3
1 1
0
1 5
0
```



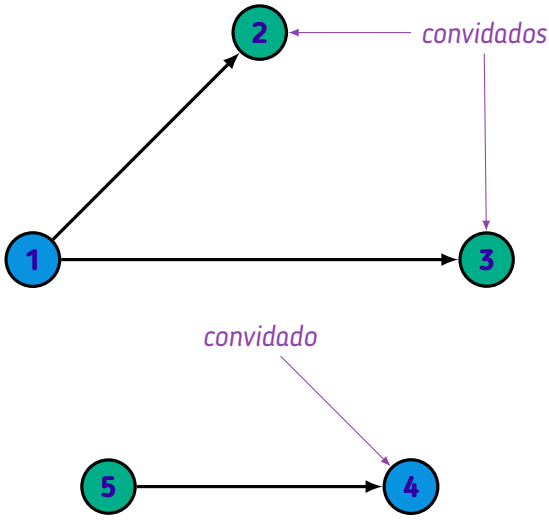
Exemplo de entrada e saída

```
3

5
1 3
1 1
0
1 5
0
```



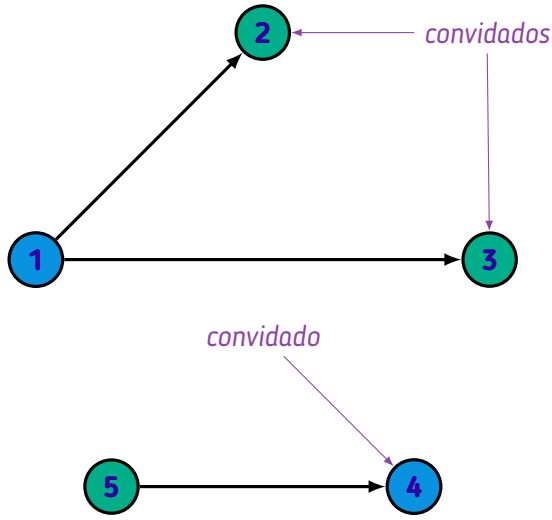
Exemplo de entrada e saída

```
3

5
1 3
1 1
0
1 5
0
```



Exemplo de entrada e saída

```
3

5
1 3
1 1
0
1 5
0
```



3

Exemplo de entrada e saída

```
3
```

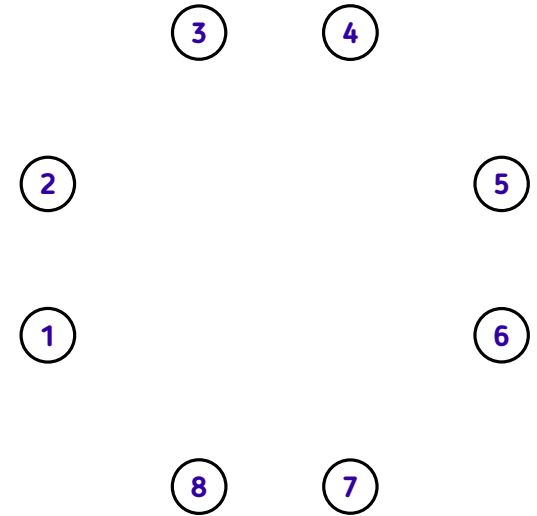
Exemplo de entrada e saída

3
8

Exemplo de entrada e saída

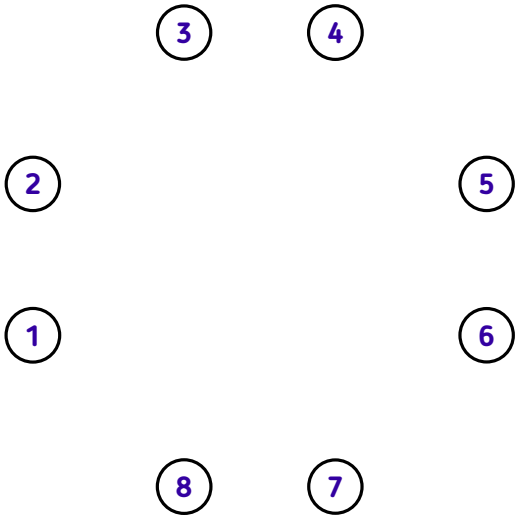
3

8



Exemplo de entrada e saída

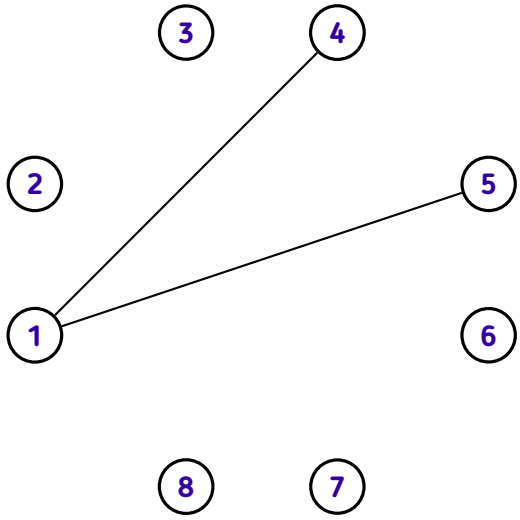
```
3

8
2 4 5
```



Exemplo de entrada e saída

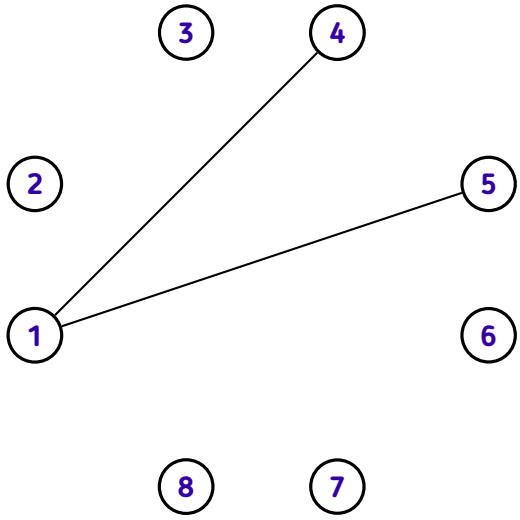
3

8
2 4 5



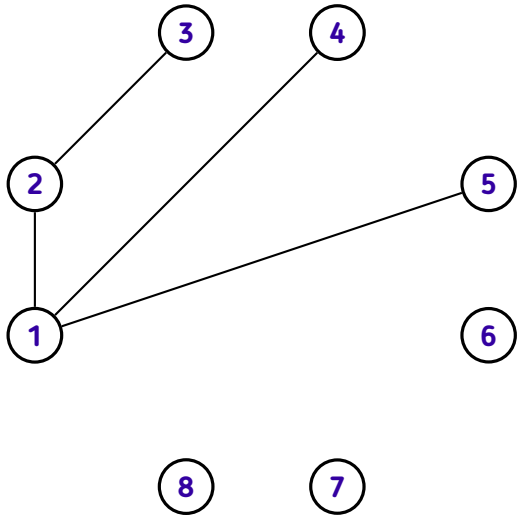
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
```



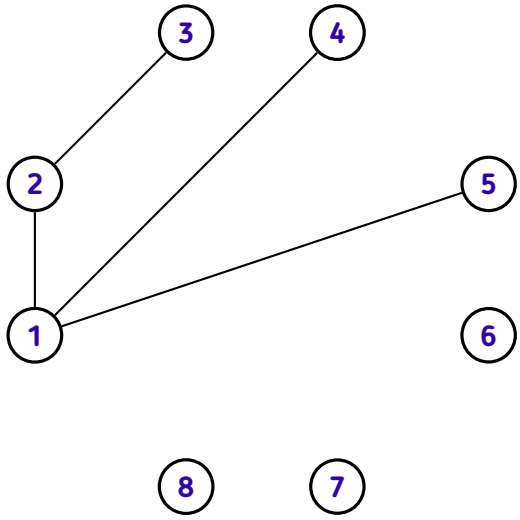
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
```



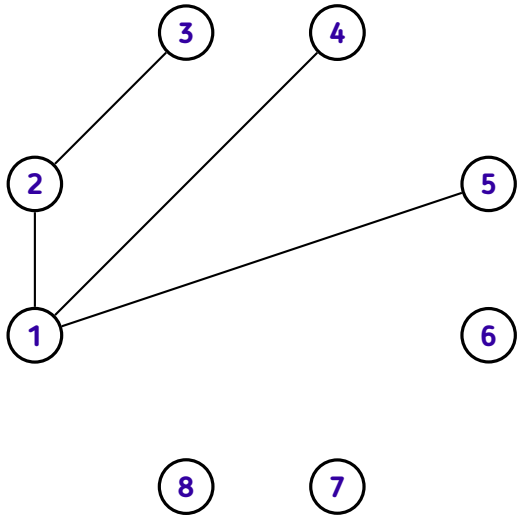
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
```



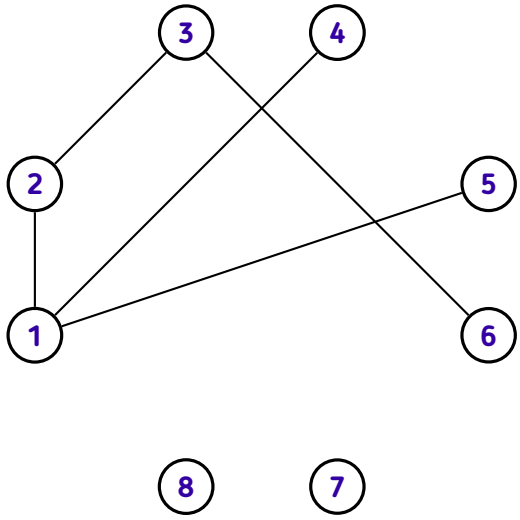
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
```



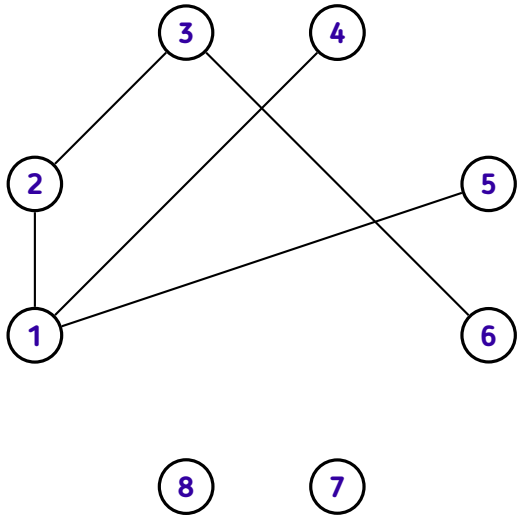
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
```



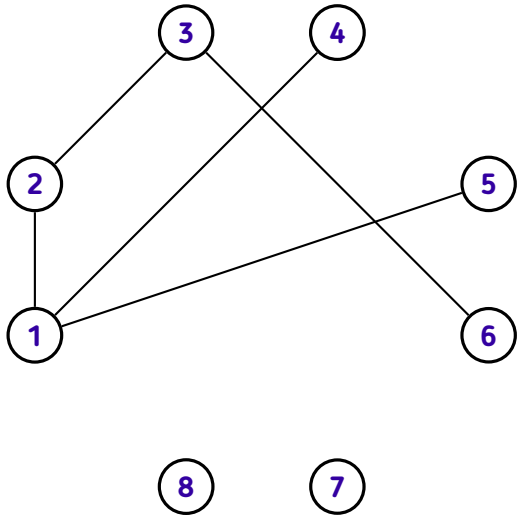
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
```



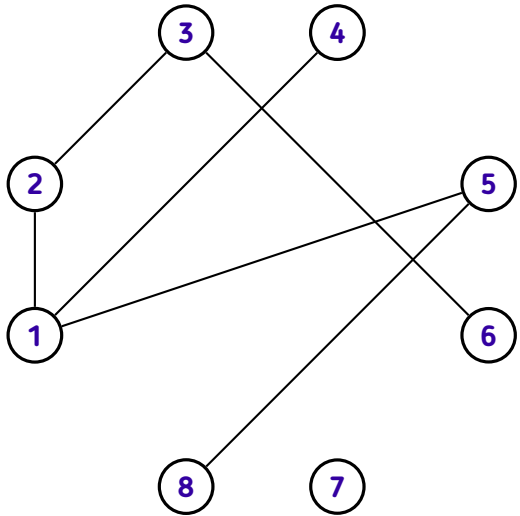
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



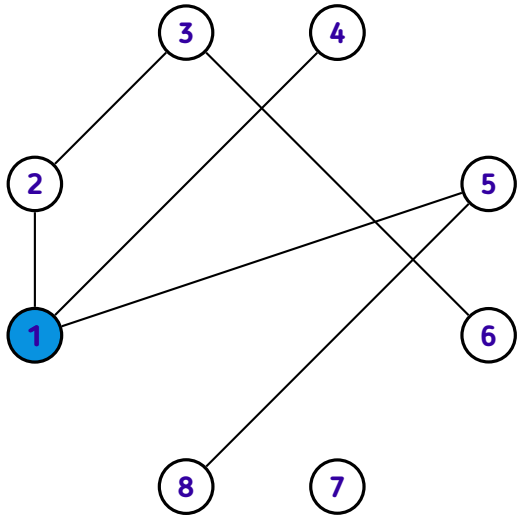
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



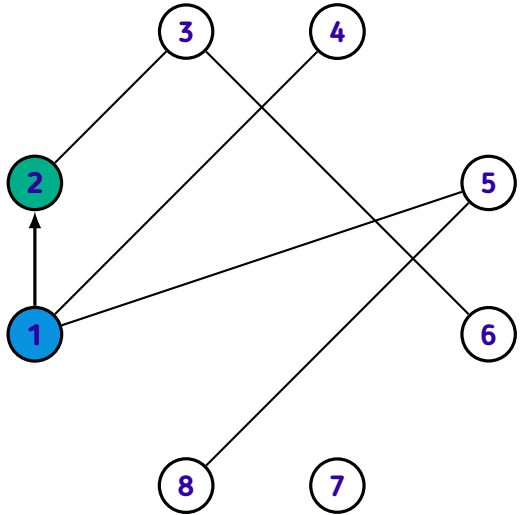
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



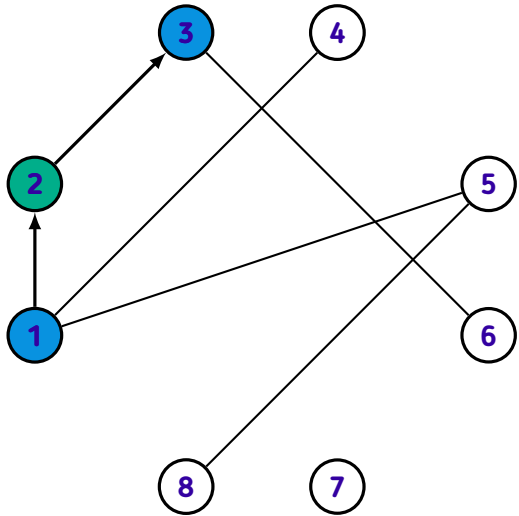
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



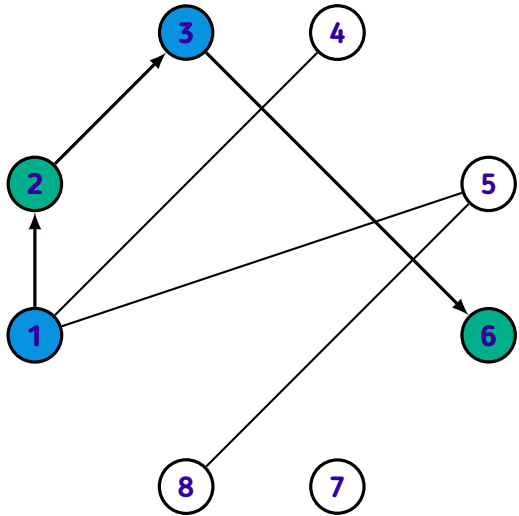
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



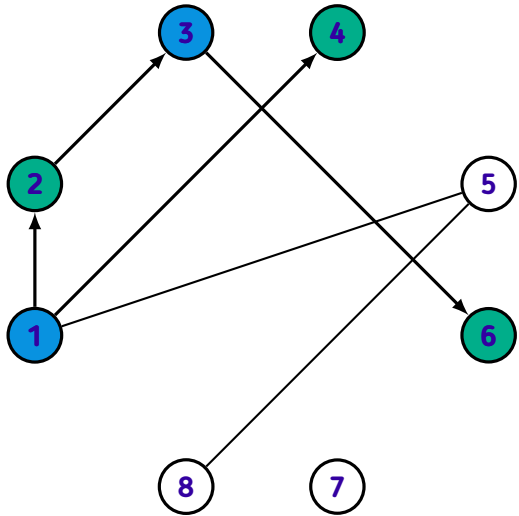
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



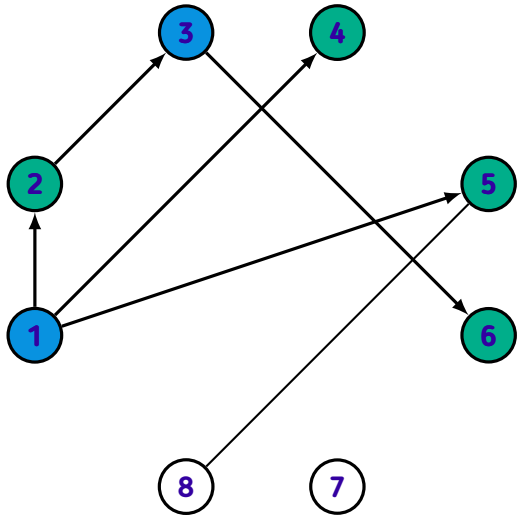
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



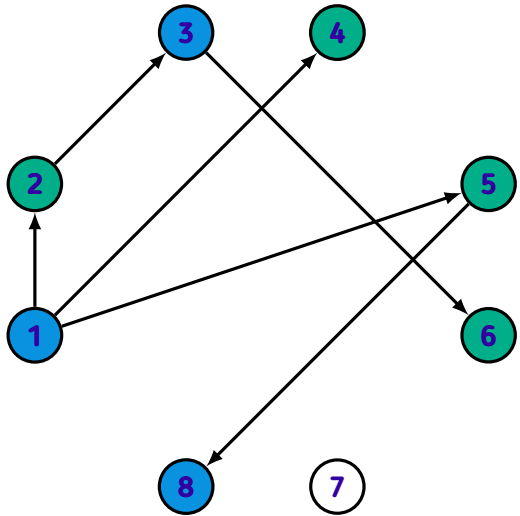
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



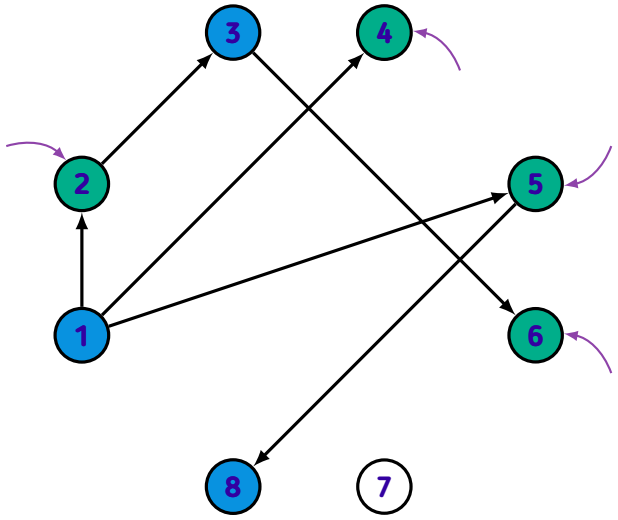
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



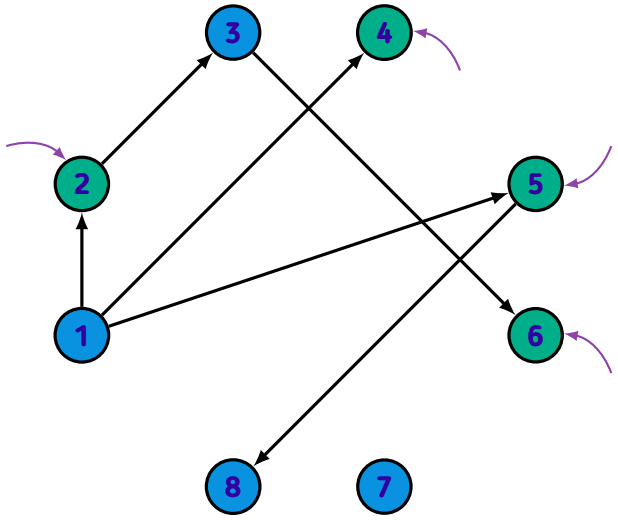
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



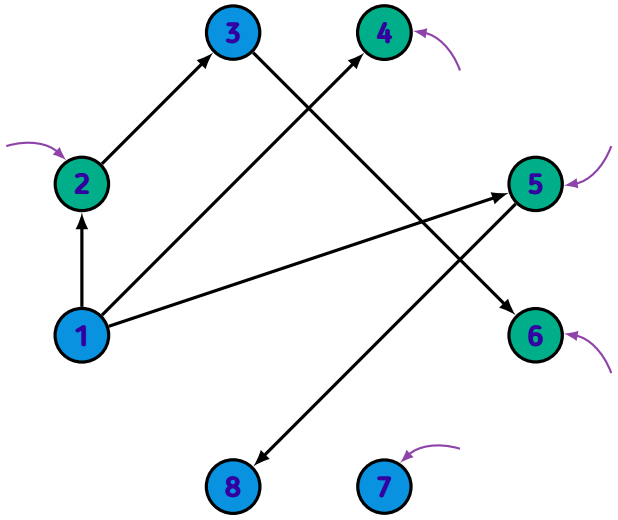
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



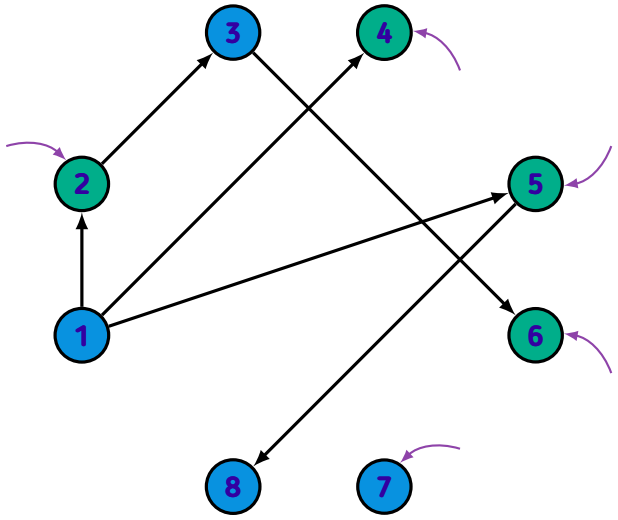
Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5
```



Exemplo de entrada e saída

```
3

8
2 4 5
2 1 3
0
0
0
1 3
0
1 5  ⟶  5
```



Exemplo de entrada e saída

```
3
```

Exemplo de entrada e saída

```
3
```

```
3
```

Exemplo de entrada e saída

3 3	2

Exemplo de entrada e saída

```
3
```

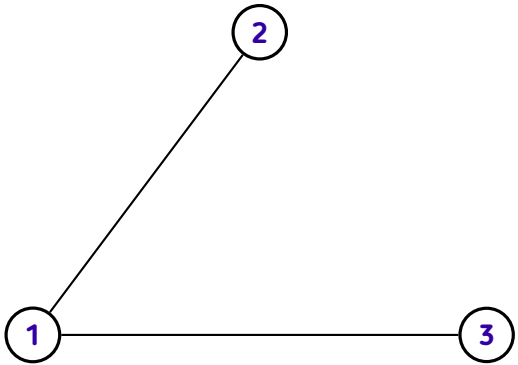
```
2
```

```
3
2 2 3
```

Exemplo de entrada e saída

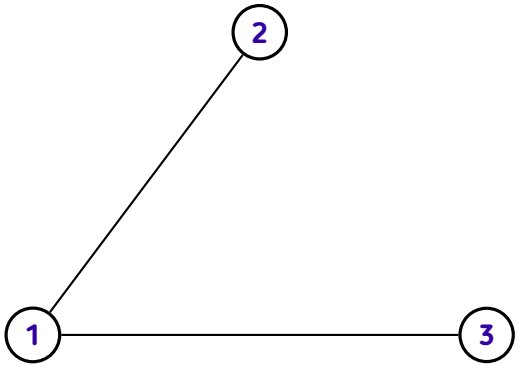
3

3
2 2 3



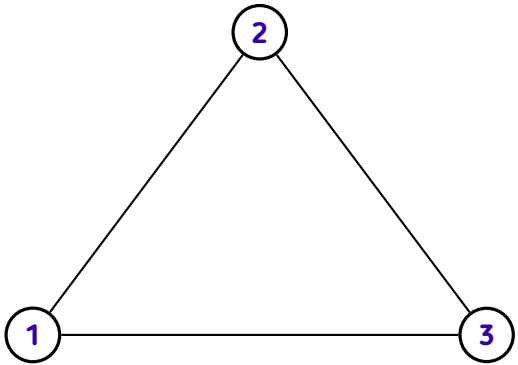
Exemplo de entrada e saída

```
3

3
2 2 3
1 3
```



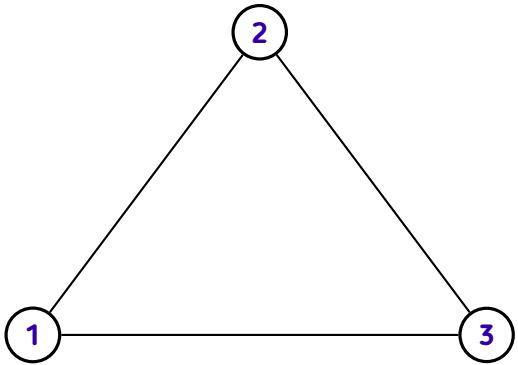
Exemplo de entrada e saída

```
3

3
2 2 3
1 3
```



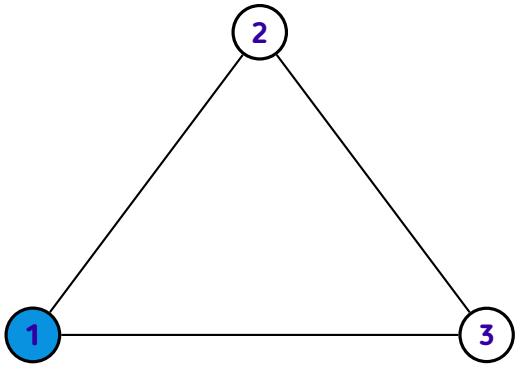
Exemplo de entrada e saída

```
3

3
2 2 3
1 3
1 1
```



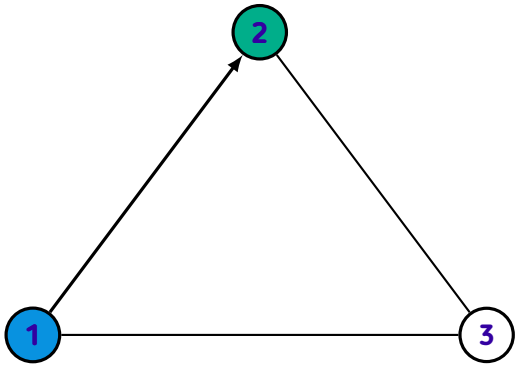
Exemplo de entrada e saída

```
3

3
2 2 3
1 3
1 1
```



Exemplo de entrada e saída

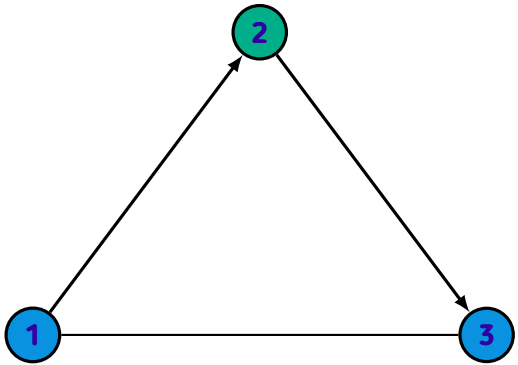
```
3

3
2 2 3
1 3
1 1
```



Exemplo de entrada e saída

3

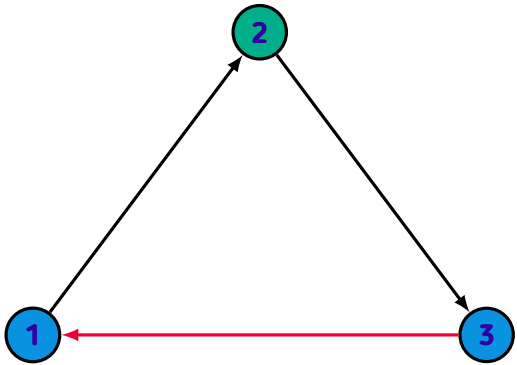
3
2 2 3
1 3
1 1



Exemplo de entrada e saída

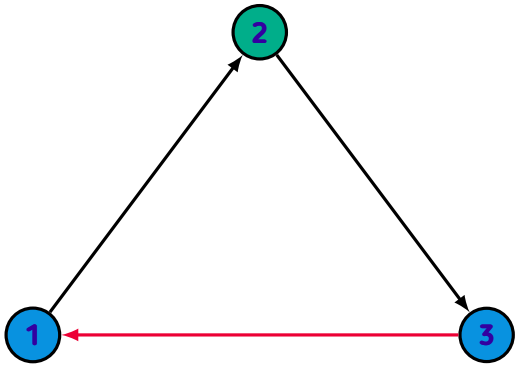
3

3
2 2 3
1 3
1 1



Exemplo de entrada e saída

```
3

3
2 2 3
1 3
1 1
```

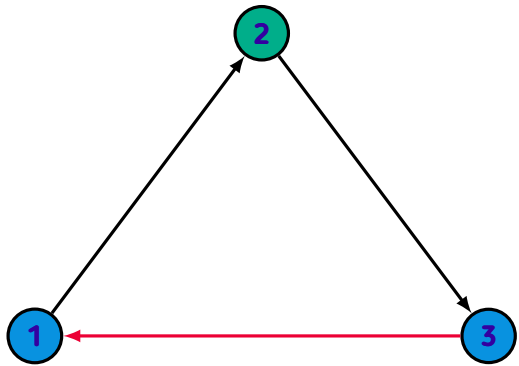


O componente não é bipartido!

Exemplo de entrada e saída

```
3

3
2 2 3
1 3
1 1
```

→ 0



O componente não é bipartido!

Solução

Solução

- ⋆ **Determine, para cada componente conectado C, se ele é bipartido ou não**

Solução

- ⋆ **Determine, para cada componente conectado C, se ele é bipartido ou não**
- ⋆ **Em caso afirmativo, a resposta deve ser incrementada em** $\max(B, G)$

Solução

- ⋆ **Determine, para cada componente conectado C, se ele é bipartido ou não**
- ⋆ **Em caso afirmativo, a resposta deve ser incrementada em $\max(B, G)$**
- ⋆ **Se C não é bipartido, nenhum elemento do componente pode ser convidado**

Solução

- ⋆ **Determine, para cada componente conectado C, se ele é bipartido ou não**
- ⋆ **Em caso afirmativo, a resposta deve ser incrementada em $\max(B, G)$**
- ⋆ **Se C não é bipartido, nenhum elemento do componente pode ser convidado**
- ⋆ **Atenção: A entrada pode ter nós com índice fora de $[1, N]$!**

Solução

- ⋆ **Determine, para cada componente conectado C, se ele é bipartido ou não**
- ⋆ **Em caso afirmativo, a resposta deve ser incrementada em $\max(B, G)$**
- ⋆ **Se C não é bipartido, nenhum elemento do componente pode ser convidado**
- ⋆ **Atenção: A entrada pode ter nós com índice fora de $[1, N]$!**
- ⋆ **Neste caso, estes nós devem ser ignorados**

```
int coloring(int s) {
    int blue = 0, green = 0;
    queue<int> q; q.push(s);
    color[s] = 1; ++blue;

    while (not q.empty()) {
        auto u = q.front(); q.pop();

        for (auto v : adj[u])
            if (color[v] == -1)
            {
                color[v] = 1 - color[u];
                blue += color[v]; green += (1 - color[v]);
                q.push(v);
            } else if (color[v] == color[u])
                return 0;
    }

    return max(blue, green);
}
```

```
int solve(int N)
{
    auto ans = 0;

    for (int u = 1; u <= N; ++u)
        if (color[u] == -1)
            ans += coloring(u);

    return ans;
}
```