Timus 1280

Topological Sorting

Prof. Edson Alves

Faculdade UnB Gama

Michael wants to win the world championship in programming and decided to study N subjects (for convenience we will number these subjects from 1 to N). Michael has worked out a study plan for this purpose. But it turned out that certain subjects may be studied only after others. So, Michael's coach analyzed all subjects and prepared a list of M limitations in the form "$s_i\ u_i$" ($1 \le s_i, u_i \le N$; $i = 1, 2, \ldots, M$), which means that subject s_i must be studied before subject u_i.

Your task is to verify if the order of subjects being studied is correct.

Michael quer vencer o campeonato mundial de programação e decidiu estudar N assuntos (de forma conveniente, nós iremos numerar estes assuntos de 1 a N). Michael trabalhou em um plano de estudos para este objetivo. Mas acontece que certos assuntos só devem ser estudados após alguns outros. Assim, o técnico de Michael analisou os assuntos e preparou uma lista de M limitações na forma "$s_i\ u_i$" $(1 \leq s_i, u_i \leq N;\ i = 1, 2, \ldots, M)$, o que significa que o assunto s_i deve ser estudado antes do assunto u_i.

Sua tarefa é verificar se a ordem dos assuntos a serem estudados está correta.

Remark. *It may appear that it's impossible to find the correct order of subjects within the given limitations. In this case any subject order worked out by Michael is incorrect.*

Limitations

$1 \leq N \leq 1000; 0 \leq M \leq 100000.$

Observação. **Pode ser impossível encontrar uma ordem correta para os assuntos que atenda as limitações dadas. Neste caso qualquer ordem de assuntos produzida por Michael estará incorreta.**

Limitações

$1 \leq N \leq 1000; 0 \leq M \leq 100000.$

Input

The first line contains two integers N and M (N is the number of the subjects, M is the number of the limitations). The next M lines contain pairs s_i, u_i, which describe the order of subjects: subject s_i must be studied before u_i. Further there is a sequence of N unique numbers ranging from 1 to N – the proposed study plan.

Output

Output a single word "`YES`" or "`NO`". "`YES`" means that the proposed order is correct and has no contradictions with the given limitations. "`NO`" means that the order is incorrect.

Entrada

A primeira linha contém dois inteiros N e M (N é o número de assuntos, M é o número de limitações). As próximas M linhas contém pares s_i, u_i, os quais descrevem a ordem dos assuntos: o assunto s_i deve ser estudado antes do assunto u_i. A seguir há uma sequência de N inteiros únicos no intervalo de 1 a N – o plano de estudo proposto.

Saída

Imprima uma única palavra: "`YES`" ou "`NO`". "`YES`" significa que a ordem proposta está correta e que não há contradições com as limitações dadas. "`NO`" significa que a ordem está incorreta.

Exemplo de entrada e saída

Exemplo de entrada e saída

```
5 6
```

Exemplo de entrada e saída

5 6

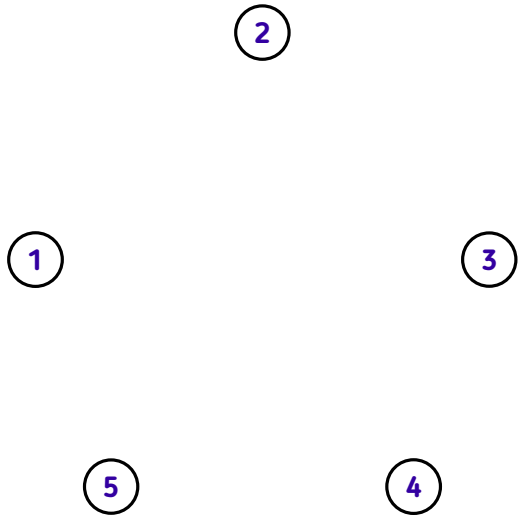
 # de assuntos

Exemplo de entrada e saída

5 6

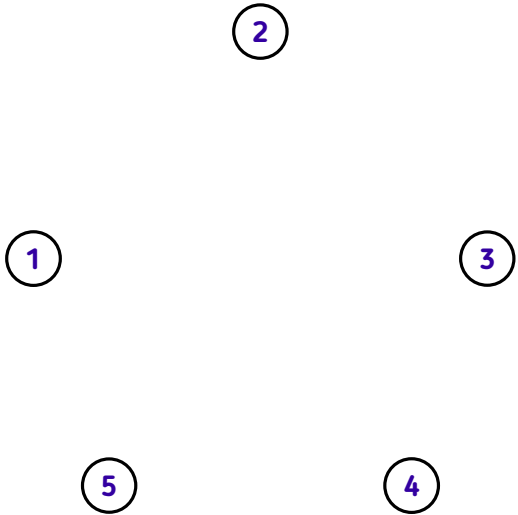
Exemplo de entrada e saída

```
5 6
```



Exemplo de entrada e saída

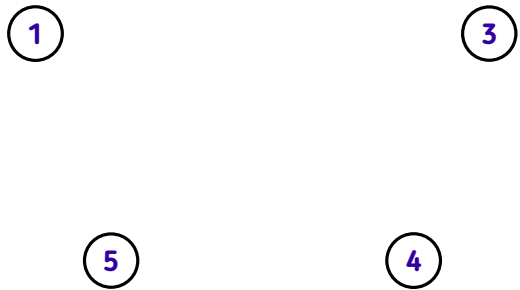
```
5 6
1 3
```



Exemplo de entrada e saída

```
5 6
1 3
```

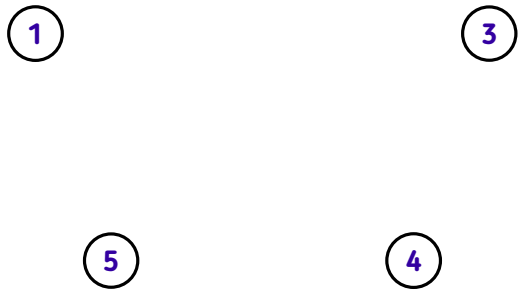
s_i



Exemplo de entrada e saída

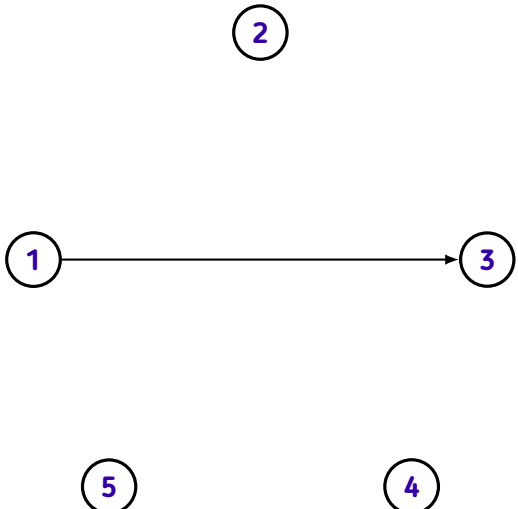
```
5 6
1 3
```

u_i



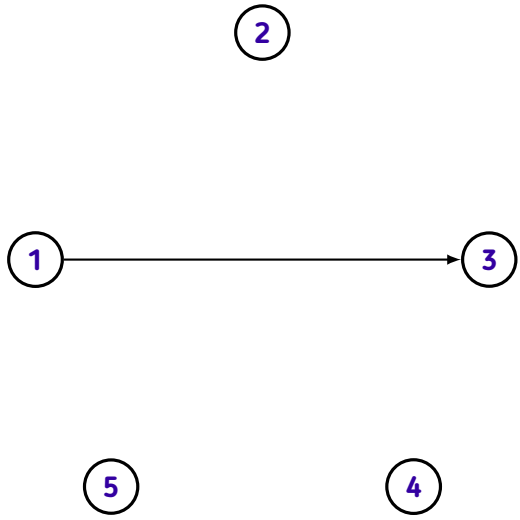
Exemplo de entrada e saída

```
5 6
1 3
```



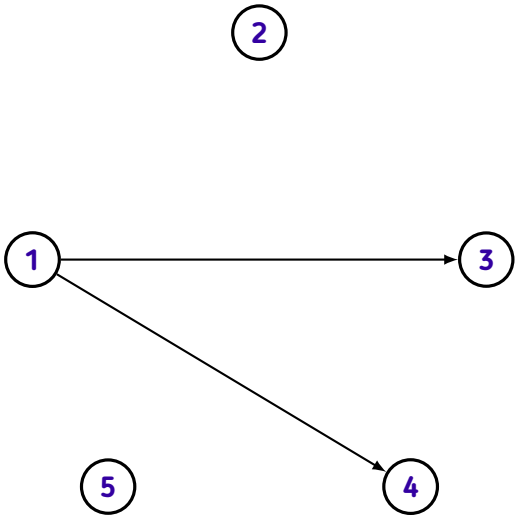
Exemplo de entrada e saída

```
5 6
1 3
1 4
```



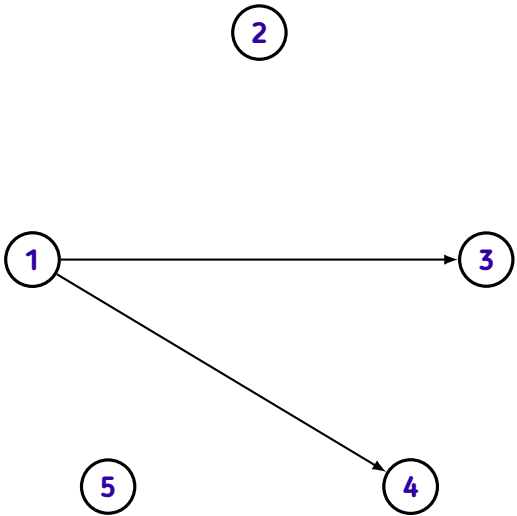
Exemplo de entrada e saída

```
5 6
1 3
1 4
```



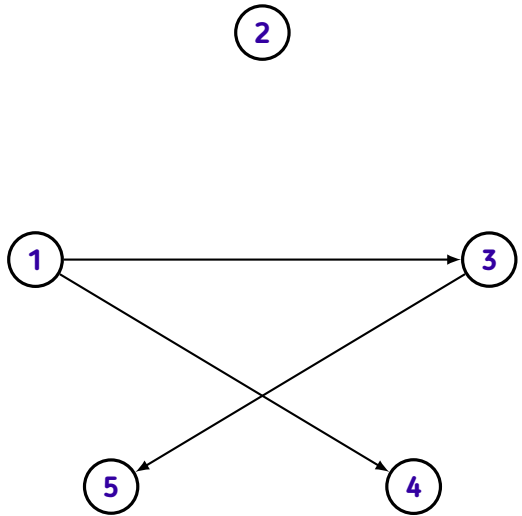
Exemplo de entrada e saída

```
5 6
1 3
1 4
3 5
```



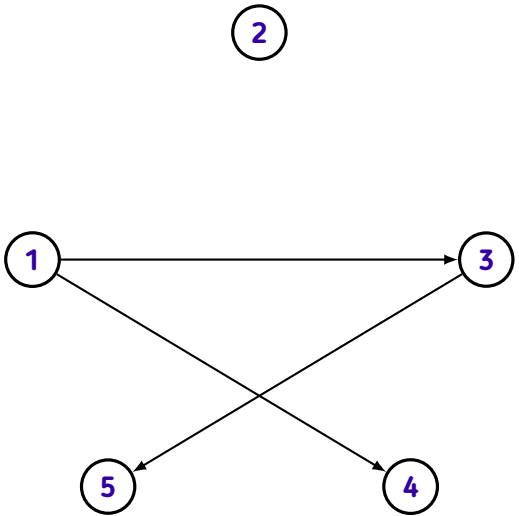
Exemplo de entrada e saída

```
5 6
1 3
1 4
3 5
```



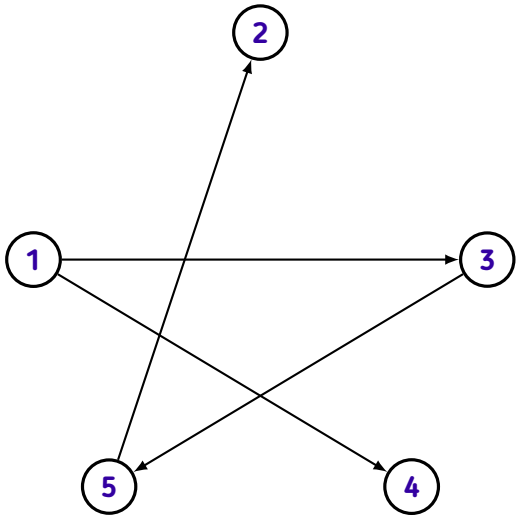
Exemplo de entrada e saída

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



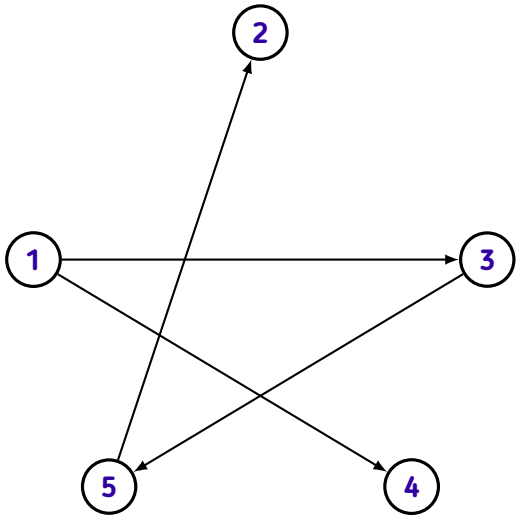
Exemplo de entrada e saída

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



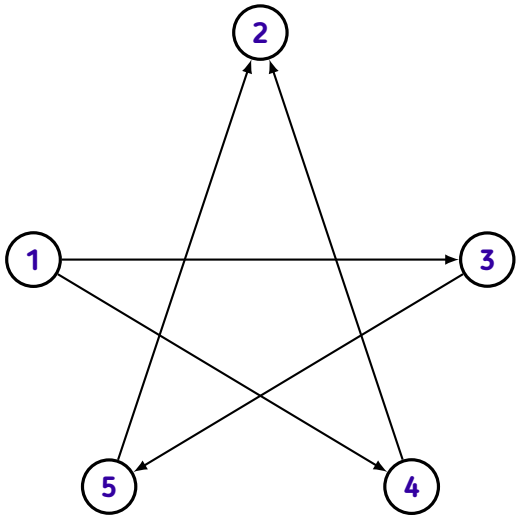
Exemplo de entrada e saída

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



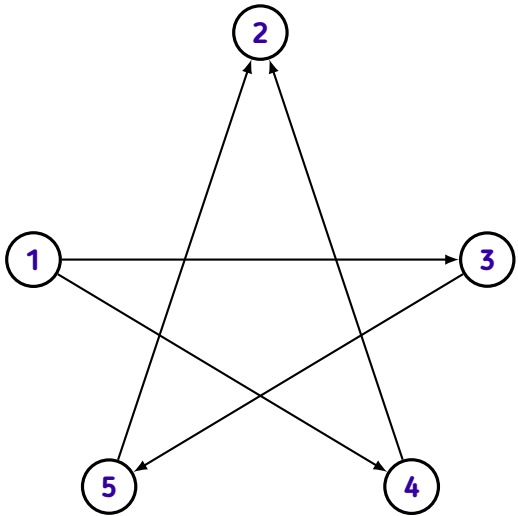
Exemplo de entrada e saída

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



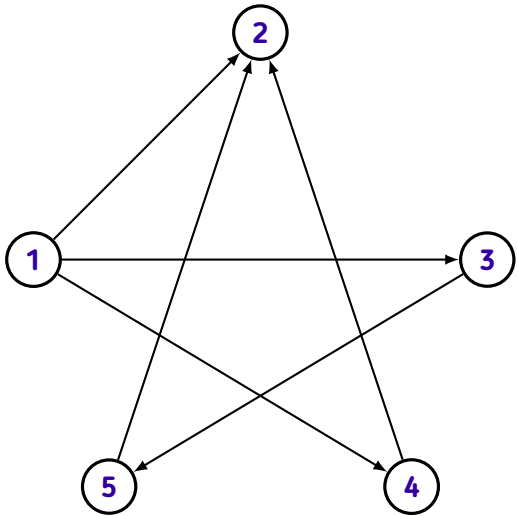
Exemplo de entrada e saída

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



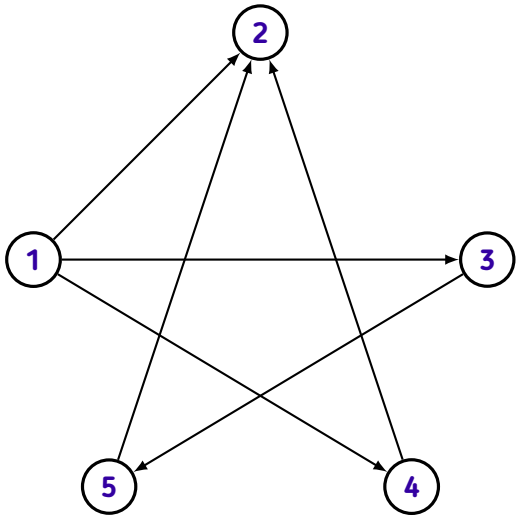
Exemplo de entrada e saída

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



Exemplo de entrada e saída

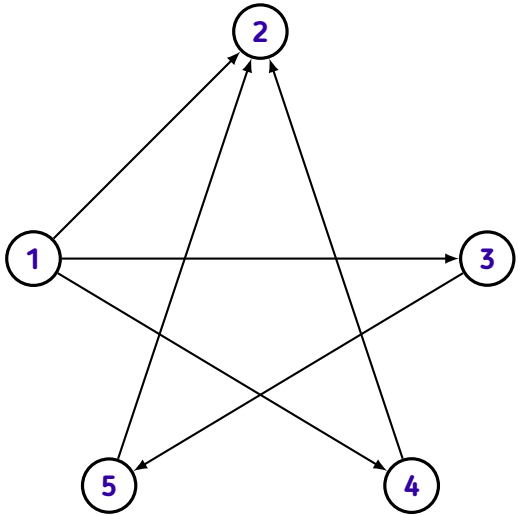
```
5 6
1 3
1 4
3 5
5 2
4 2
1 2
1 3 4 5 2
```



Exemplo de entrada e saída

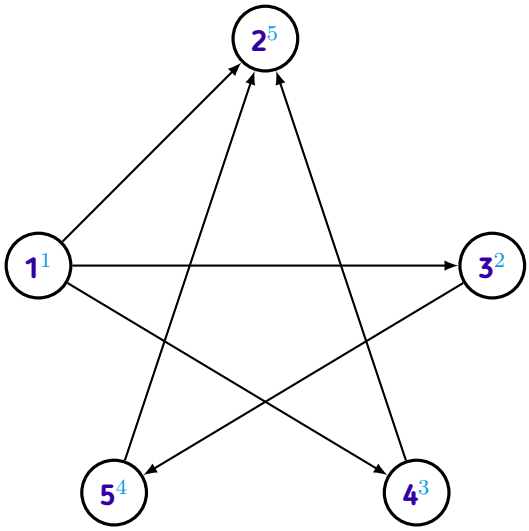
```
5 6
1 3
1 4
3 5
5 2
4 2
1 2
1 3 4 5 2
```

ordem proposta



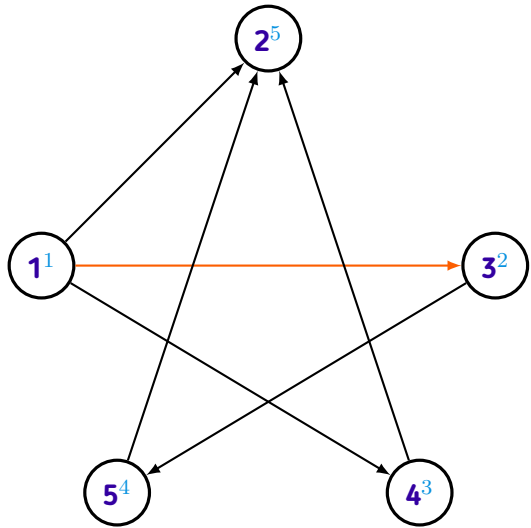
Exemplo de entrada e saída

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



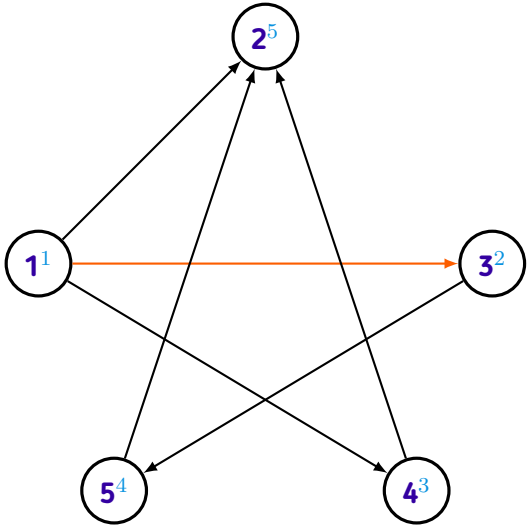
Exemplo de entrada e saída

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



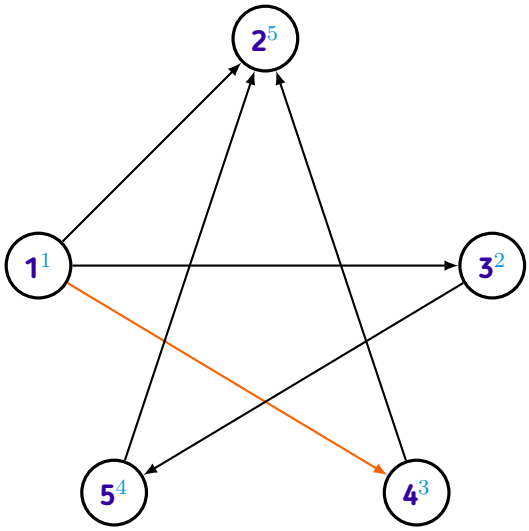
Exemplo de entrada e saída

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



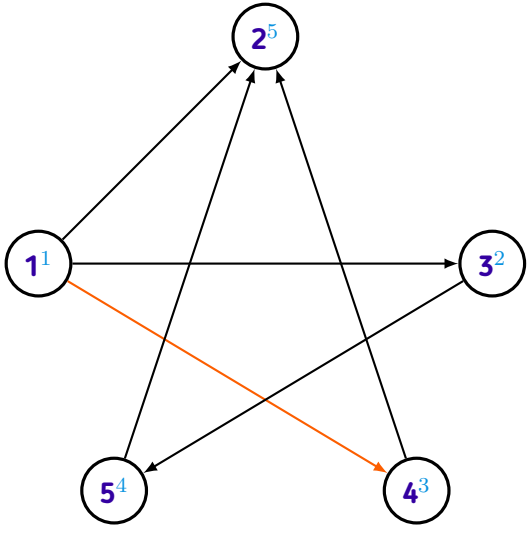
Exemplo de entrada e saída

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



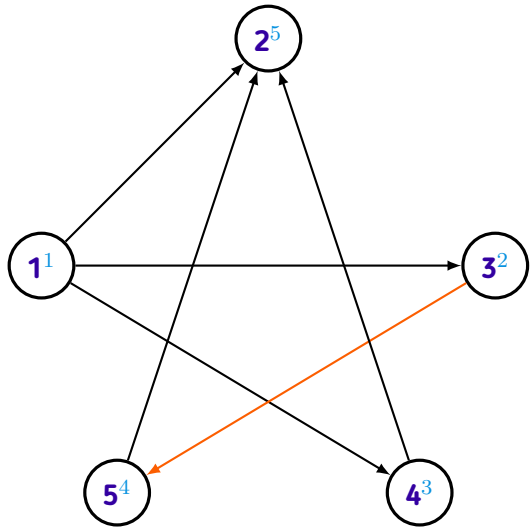
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5
5 2
4 2
1 2
1 3 4 5 2
```



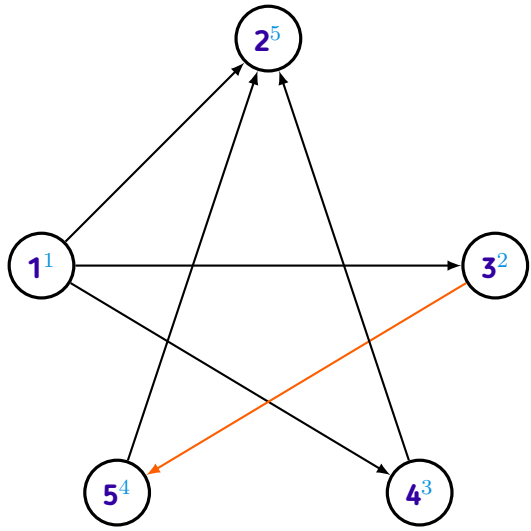
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5
5 2
4 2
1 2
1 3 4 5 2
```



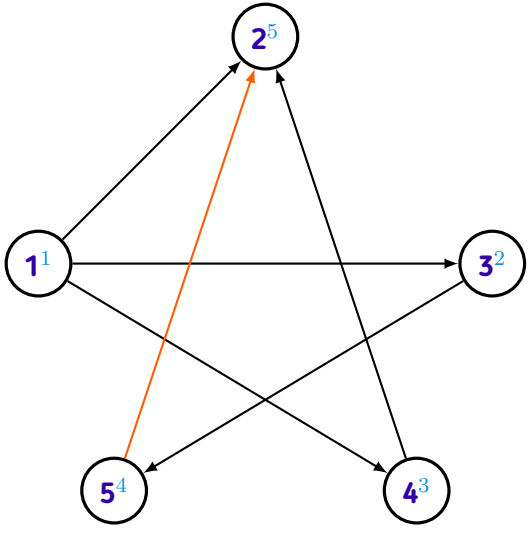
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5 ✔
5 2
4 2
1 2
1 3 4 5 2
```



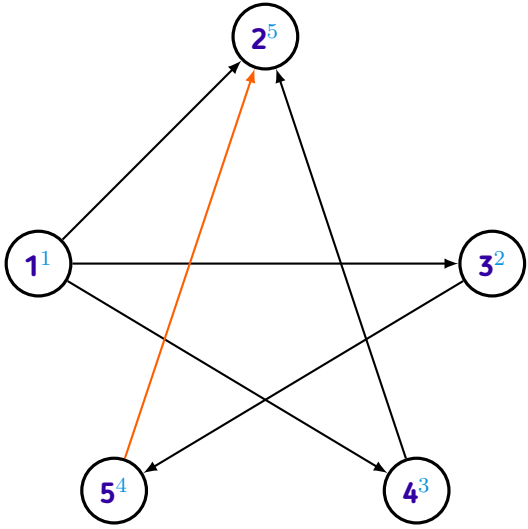
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5 ✔
5 2
4 2
1 2
1 3 4 5 2
```



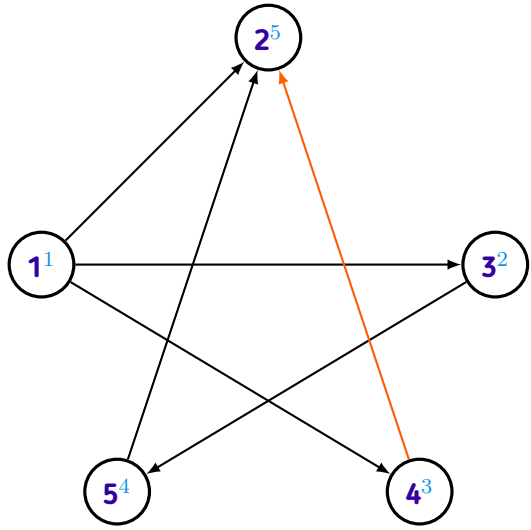
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5 ✔
5 2 ✔
4 2
1 2
1 3 4 5 2
```



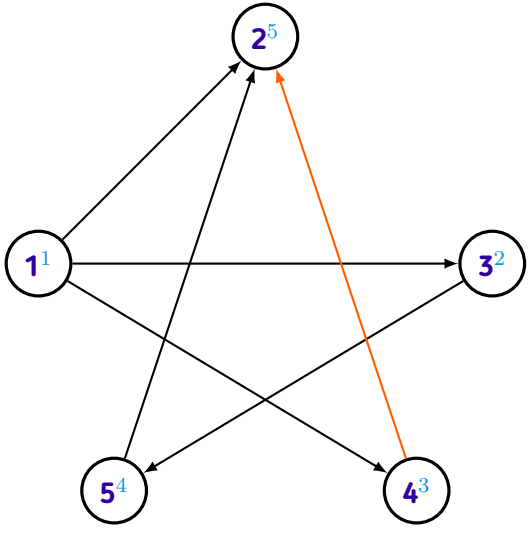
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5 ✔
5 2 ✔
4 2
1 2
1 3 4 5 2
```



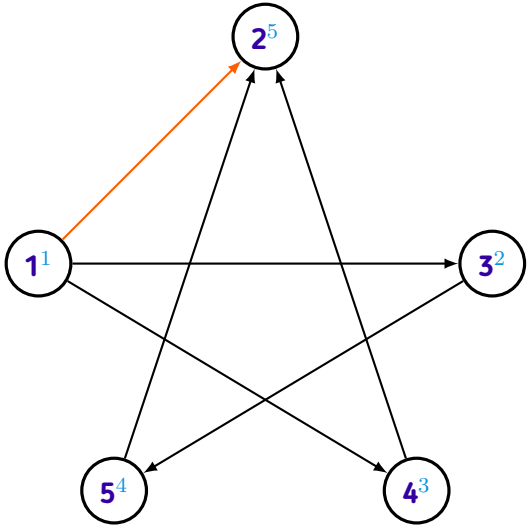
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5 ✔
5 2 ✔
4 2 ✔
1 2
1 3 4 5 2
```



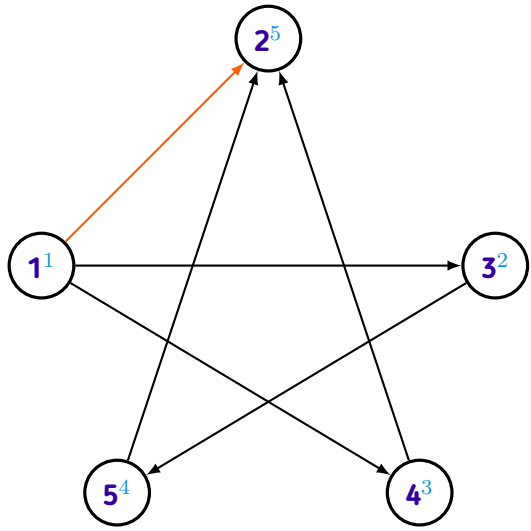
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5 ✔
5 2 ✔
4 2 ✔
1 2
1 3 4 5 2
```



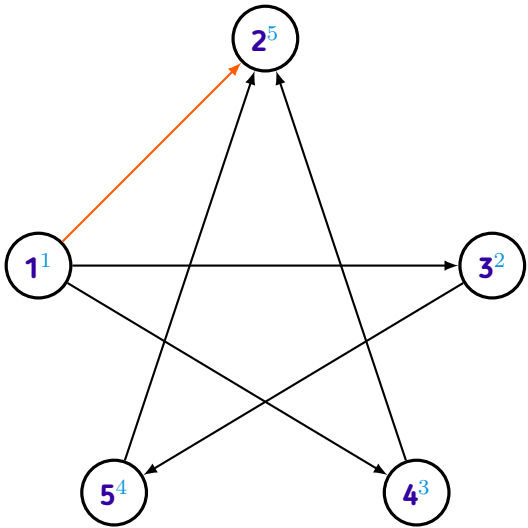
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5 ✔
5 2 ✔
4 2 ✔
1 2 ✔
1 3 4 5 2
```



Exemplo de entrada e saída

5 6
1 3 ✔
1 4 ✔
3 5 ✔
5 2 ✔
4 2 ✔
1 2 ✔
1 3 4 5 2



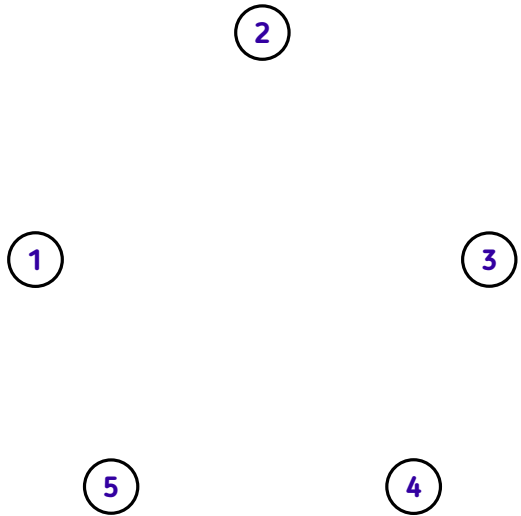
Exemplo de entrada e saída

Exemplo de entrada e saída

```
5 6
```

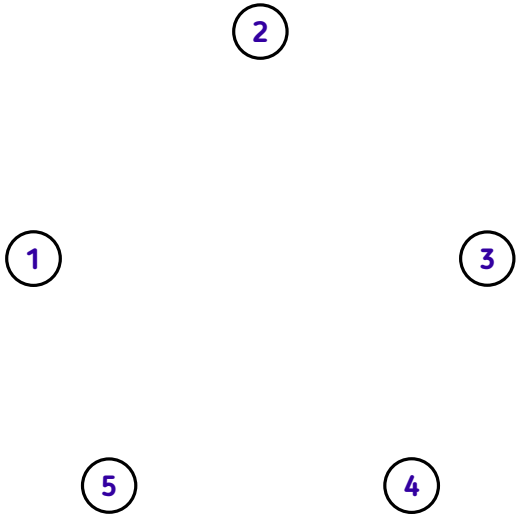
Exemplo de entrada e saída

```
5 6
```



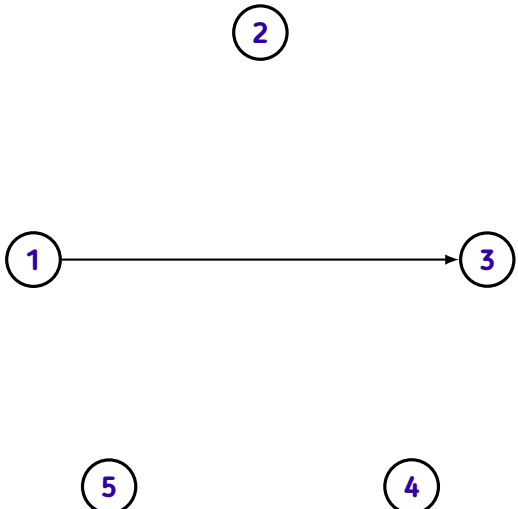
Exemplo de entrada e saída

```
5 6
1 3
```



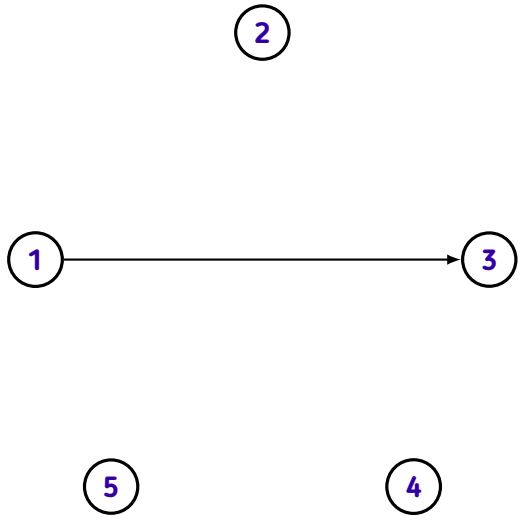
Exemplo de entrada e saída

```
5 6
1 3
```



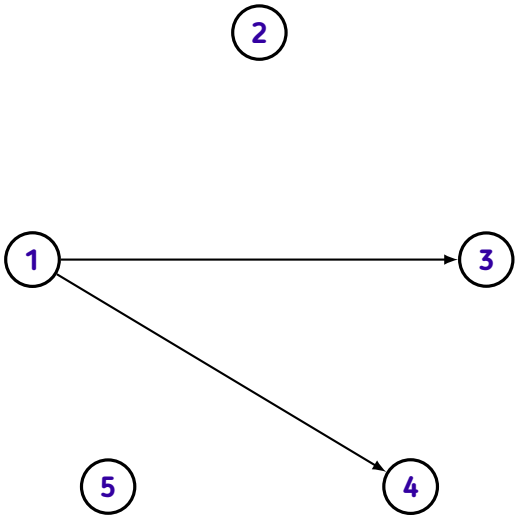
Exemplo de entrada e saída

```
5 6
1 3
1 4
```



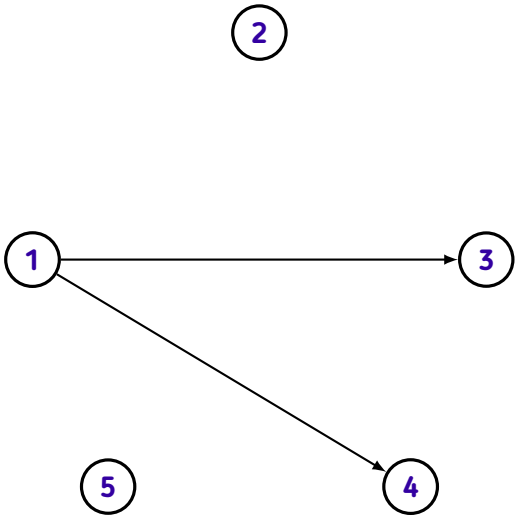
Exemplo de entrada e saída

```
5 6
1 3
1 4
```



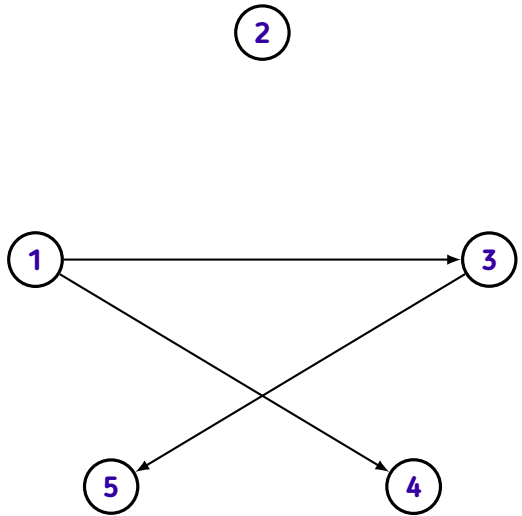
Exemplo de entrada e saída

```
5 6
1 3
1 4
3 5
```



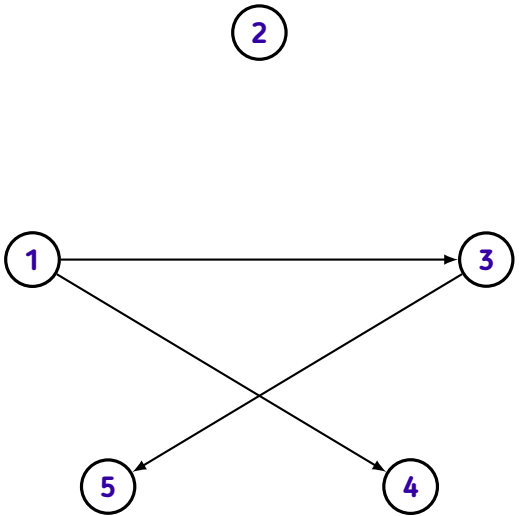
Exemplo de entrada e saída

```
5 6
1 3
1 4
3 5
```



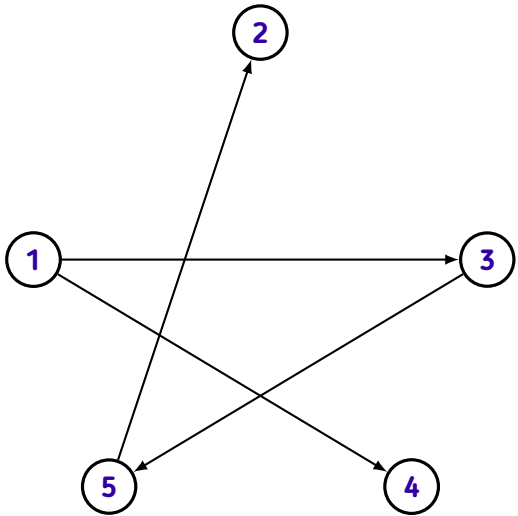
Exemplo de entrada e saída

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



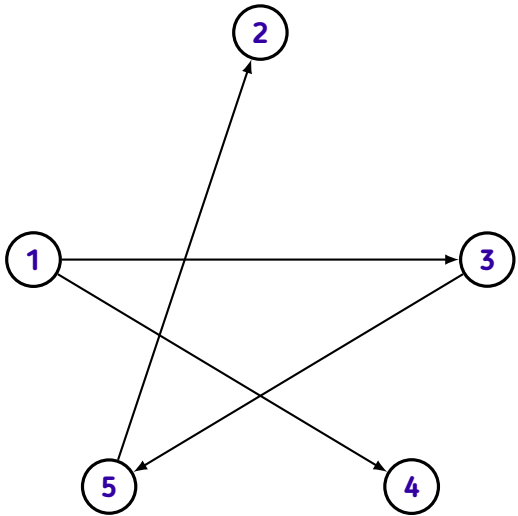
Exemplo de entrada e saída

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



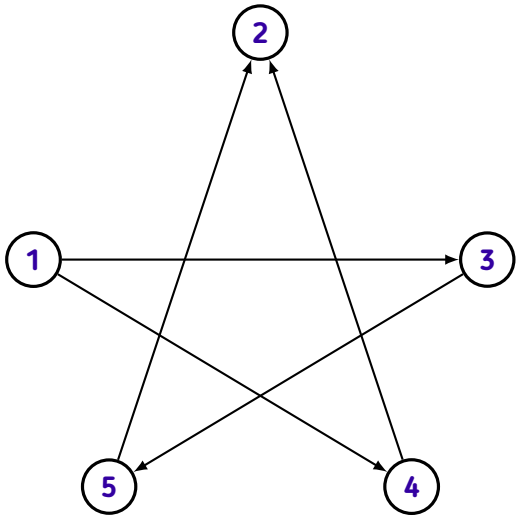
Exemplo de entrada e saída

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



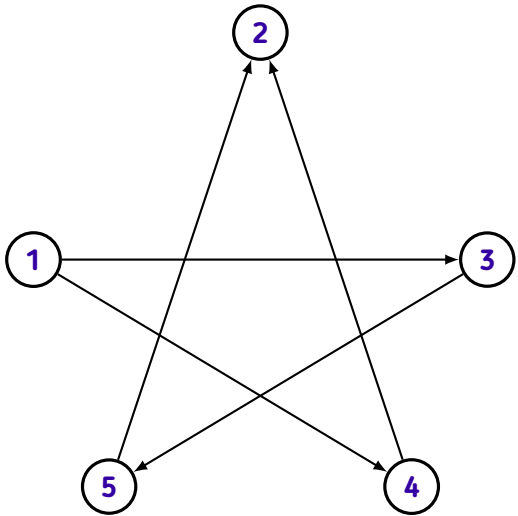
Exemplo de entrada e saída

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



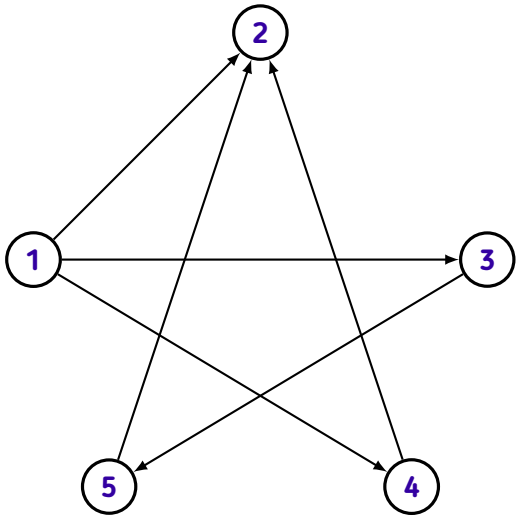
Exemplo de entrada e saída

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



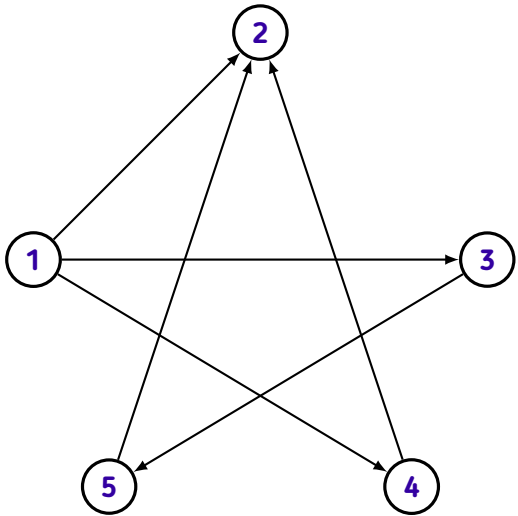
Exemplo de entrada e saída

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



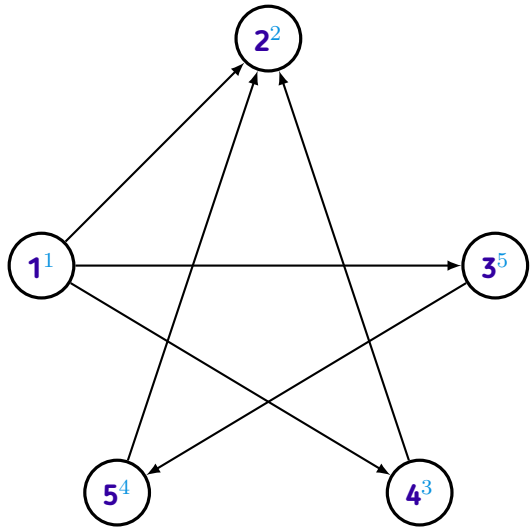
Exemplo de entrada e saída

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



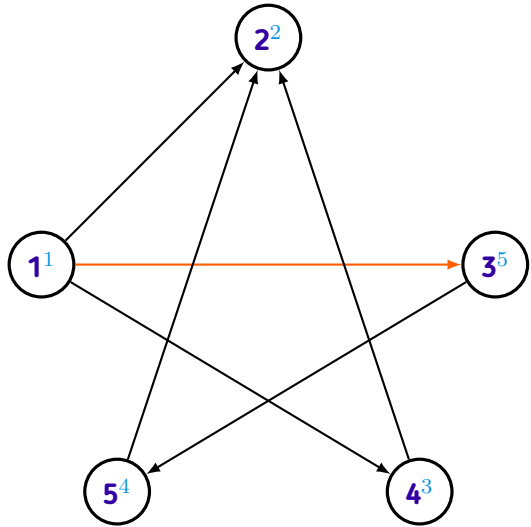
Exemplo de entrada e saída

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



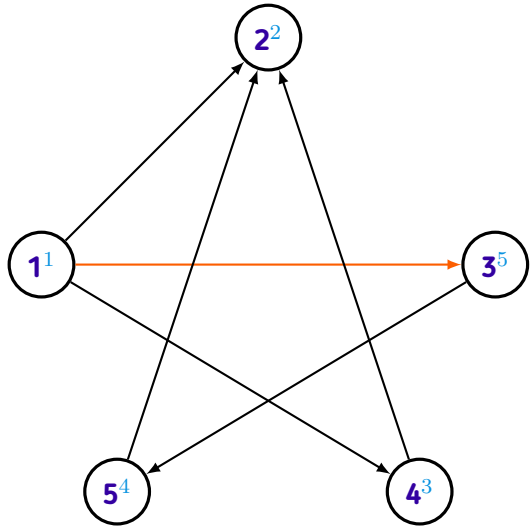
Exemplo de entrada e saída

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



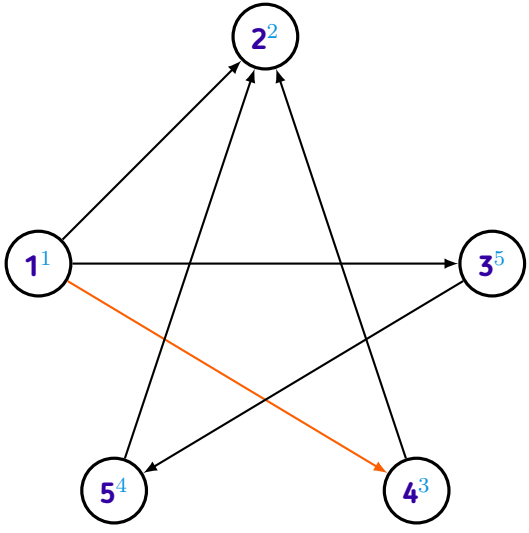
Exemplo de entrada e saída

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



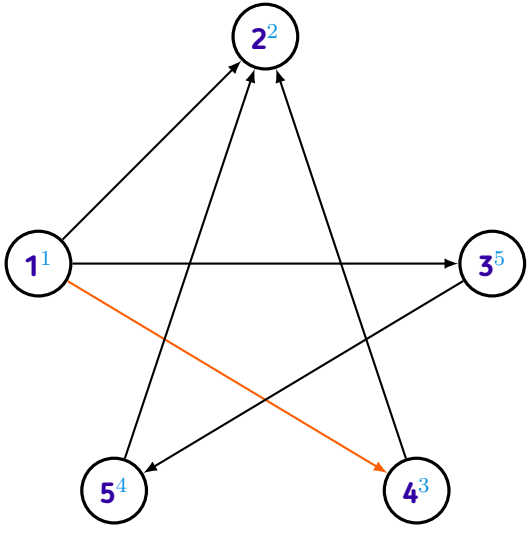
Exemplo de entrada e saída

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



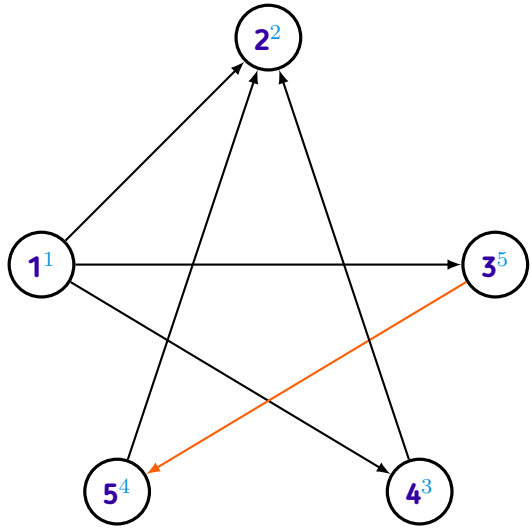
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5
5 2
4 2
1 2
1 2 4 5 3
```



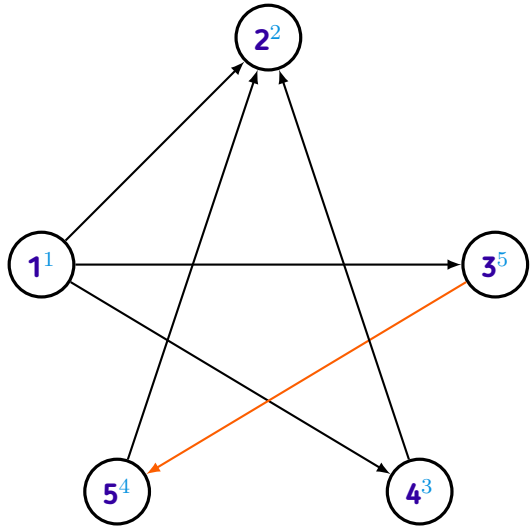
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5
5 2
4 2
1 2
1 2 4 5 3
```



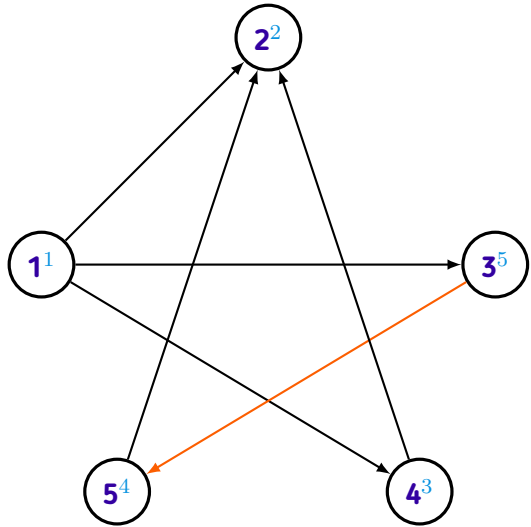
Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5 ✖
5 2
4 2
1 2
1 2 4 5 3
```



Exemplo de entrada e saída

```
5 6
1 3 ✔
1 4 ✔
3 5 ✖
5 2
4 2
1 2
1 2 4 5 3
        ↓
       NO
```



Solução

Solução

- **O problema consiste em determinar se a ordenação proposta por Michael é ou não topológica**

Solução

- O problema consiste em determinar se a ordenação proposta por Michael é ou não topológica

- Assim, não é preciso de fato gerar uma ordenação topológica: basta verificar se a ordenação proposta atende às limitações dadas

Solução

- O problema consiste em determinar se a ordenação proposta por Michael é ou não topológica

- Assim, não é preciso de fato gerar uma ordenação topológica: basta verificar se a ordenação proposta atende às limitações dadas

- Um dicionário permite identificar, de forma eficiente, a posição de uma tarefa de acordo com a ordenação

Solução

- O problema consiste em determinar se a ordenação proposta por Michael é ou não topológica

- Assim, não é preciso de fato gerar uma ordenação topológica: basta verificar se a ordenação proposta atende às limitações dadas

- Um dicionário permite identificar, de forma eficiente, a posição de uma tarefa de acordo com a ordenação

- Se alguma limitação for violada, a resposta será `NO`

Solução

⋆ **O problema consiste em determinar se a ordenação proposta por Michael é ou não topológica**

⋆ **Assim, não é preciso de fato gerar uma ordenação topológica: basta verificar se a ordenação proposta atende às limitações dadas**

⋆ **Um dicionário permite identificar, de forma eficiente, a posição de uma tarefa de acordo com a ordenação**

⋆ **Se alguma limitação for violada, a resposta será** `NO`

⋆ **Complexidade:** $O(N + M)$

```
string solve(int N, const vector<int>& xs)
{
    unordered_map<int, int> is;

    for (int i = 0; i < N; ++i)
        is[xs[i]] = i;

    for (int u = 1; u <= N; ++u)
        for (auto v : adj[u])
            if (is[u] > is[v])
                return "NO";

    return "YES";
}
```