CSES 1673

High Score

Prof. Edson Alves

Faculdade UnB Gama

You play a game consisting of n rooms and m tunnels. Your initial score is 0, and each tunnel increases your score by x where x may be both positive or negative. You may go through a tunnel several times.

Your task is to walk from room 1 to room n. What is the maximum score you can get?

Você vai jogar um jogo composto por n salas e m túneis. Sua pontuação inicial é igual a 0, e cada túnel aumenta sua pontuação em x unidades, onde x pode ser tanto positivo quanto negativo. Você pode passar por um mesmo túnel várias vezes.

Sua tarefa é ir da sala 1 para a sala n. Qual é a maior pontuação que você pode obter?

Input

The first input line has two integers n and m: the number of rooms and tunnels. The rooms are numbered $1, 2, \ldots, n$.

Then, there are m lines describing the tunnels. Each line has three integers a, b and x: the tunnel starts at room a, ends at room b, and it increases your score by x. All tunnels are one-way tunnels.

You can assume that it is possible to get from room 1 to room n.

Output

Print one integer: the maximum score you can get. However, if you can get an arbitrarily large score, print -1.

Entrada

A primeira linha da entrada contém dois inteiros n e m: o número de salas e de túneis. As salas são numeradas $1, 2, \ldots, n$.

As m linhas seguintes descrevem os túneis. Cada linha tem três inteiros a, b e x: o túnel começa na sala a, termina na sala b e aumenta sua pontuação em x unidades. Todos os túneis são de mão única.

Assuma que é possível ir da sala 1 para a sala n.

Saída

Imprima um inteiro: a pontuação máxima que você pode obter. Contudo, se você pode obter uma pontuação arbitrariamente grande, imprima -1.

Constraints

- $1 \leq 2500 \leq n$
- $1 \leq 5000 \leq m$
- $1 \leq a, b \leq n$
- $-10^9 \leq x \leq 10^9$

Restrições

- $1 \leq 2500 \leq n$
- $1 \leq 5000 \leq m$
- $1 \leq a, b \leq n$
- $-10^9 \leq x \leq 10^9$

Exemplo de entrada e saída

Exemplo de entrada e saída

```
4 5
```

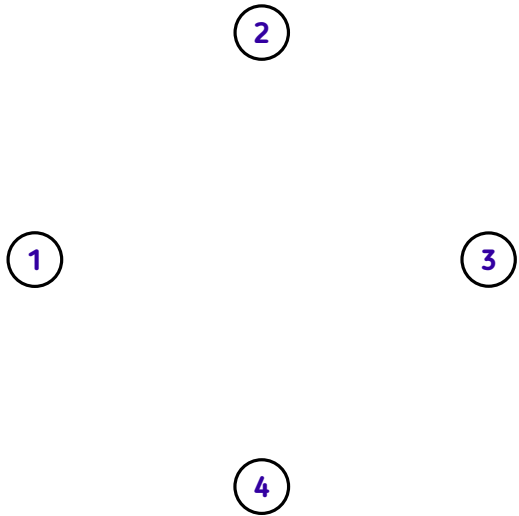
Exemplo de entrada e saída

4 5

Exemplo de entrada e saída

```
4 5
```

 # de salas

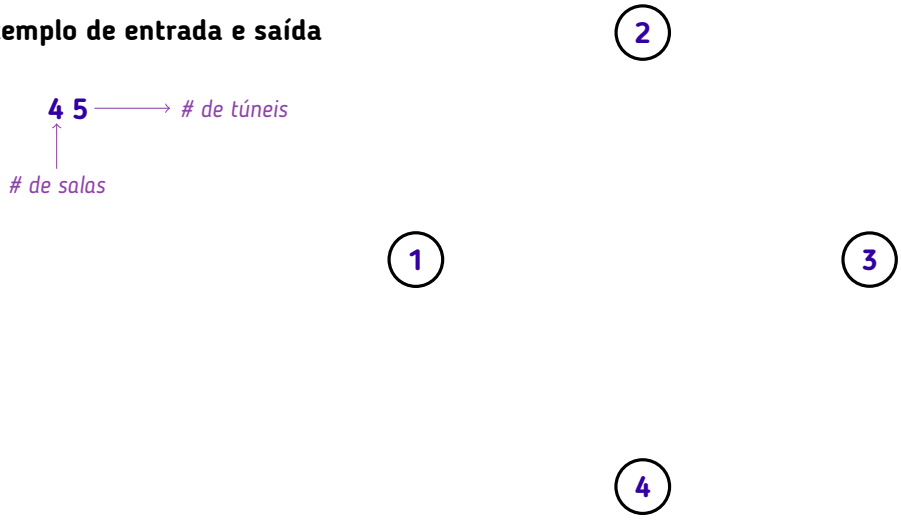


Exemplo de entrada e saída

4 5 ⟶ *# de túneis*

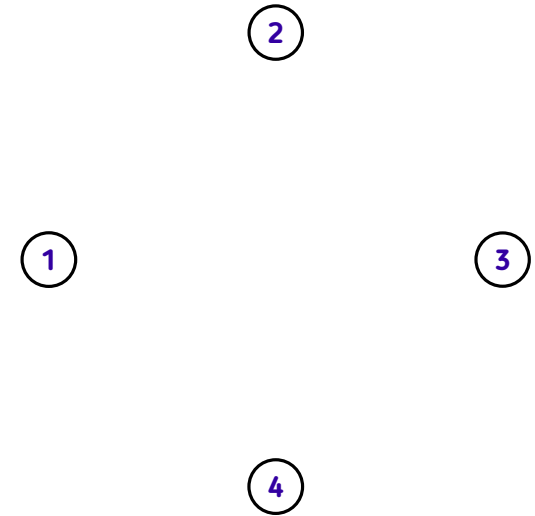
↑

 # de salas



Exemplo de entrada e saída

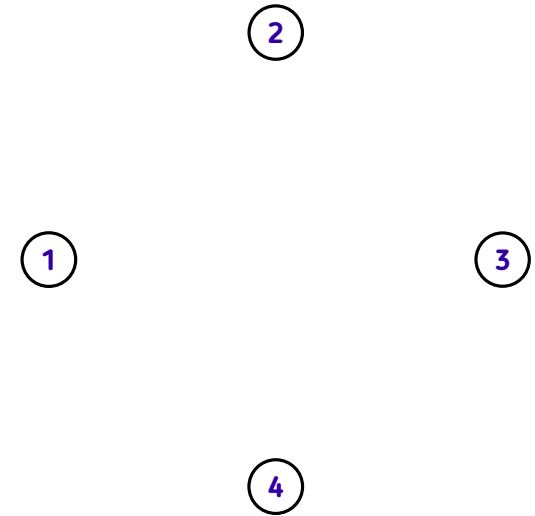
4 5



Exemplo de entrada e saída

4 5

1 2 3



Exemplo de entrada e saída

4 5

1 2 3

2

1

3

4

Exemplo de entrada e saída

4 5

1 2 3

2

1

3

4

Exemplo de entrada e saída

4 5

1 2 3

2

1

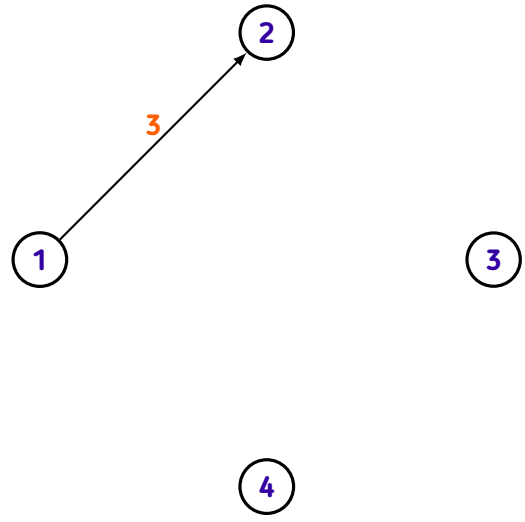
3

4

Exemplo de entrada e saída

4 5

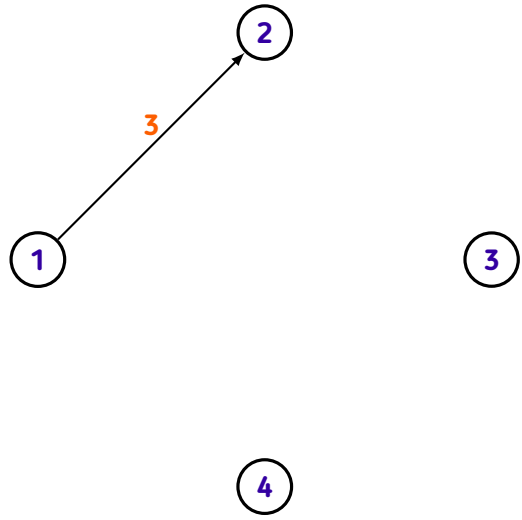
1 2 3



Exemplo de entrada e saída

4 5

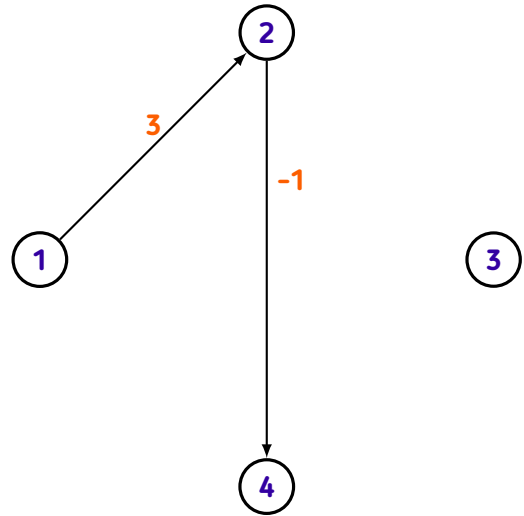
1 2 3
2 4 -1



Exemplo de entrada e saída

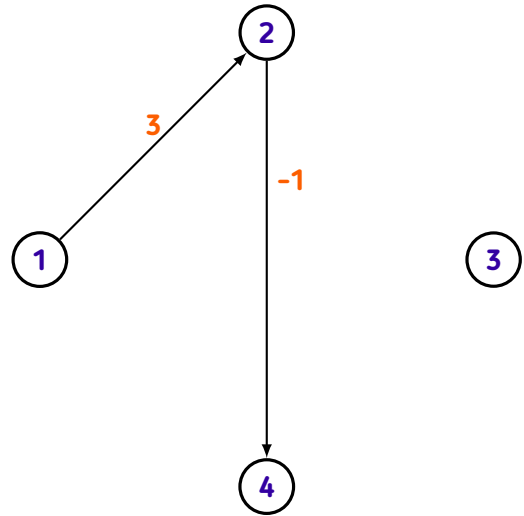
4 5

1 2 3
2 4 -1



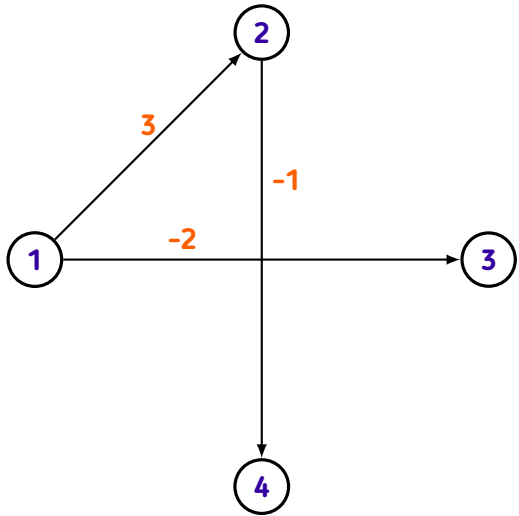
Exemplo de entrada e saída

```
4 5

1 2 3
2 4 -1
1 3 -2
```



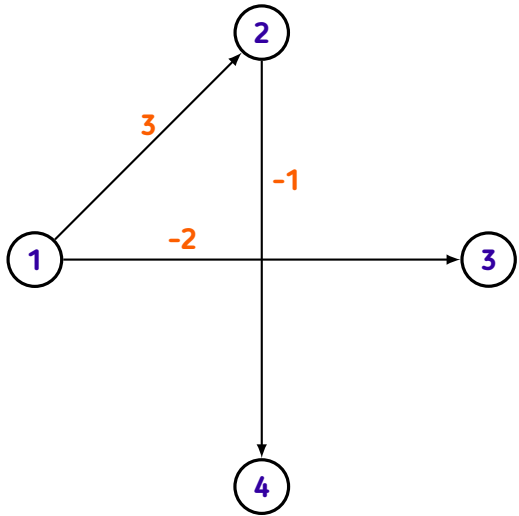
Exemplo de entrada e saída

```
4 5

1 2 3
2 4 -1
1 3 -2
```



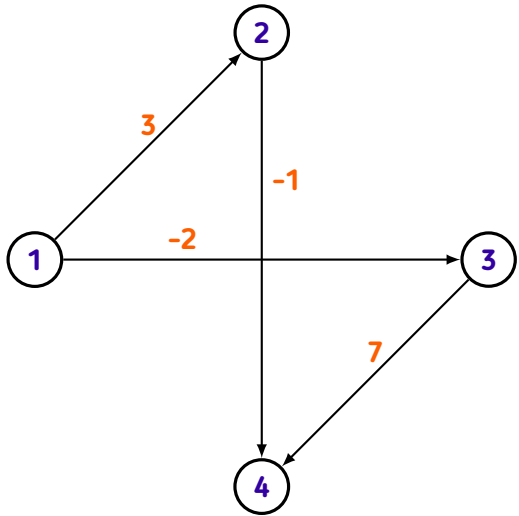
Exemplo de entrada e saída

```
4 5

1 2 3
2 4 -1
1 3 -2
3 4 7
```



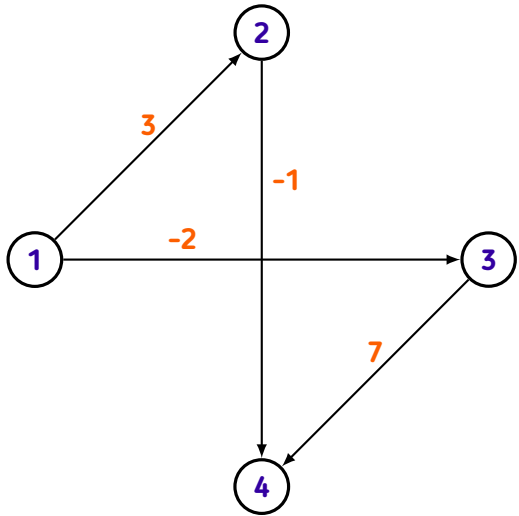
Exemplo de entrada e saída

```
4 5

1 2 3
2 4 -1
1 3 -2
3 4 7
```



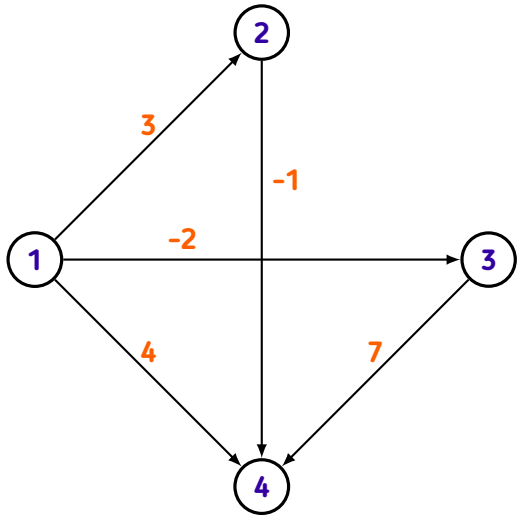
Exemplo de entrada e saída

```
4 5

1 2 3
2 4 -1
1 3 -2
3 4 7
1 4 4
```



Exemplo de entrada e saída

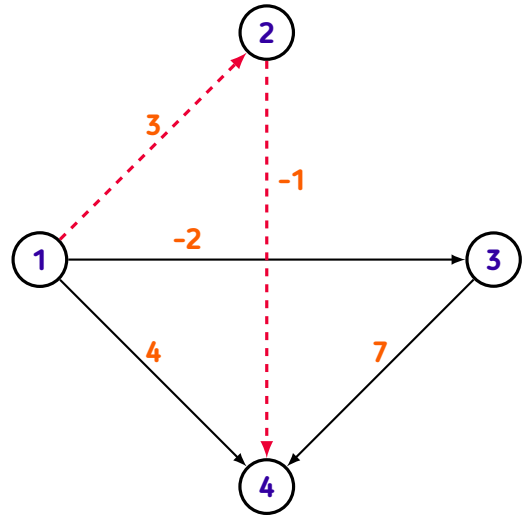
```
4 5

1 2 3
2 4 -1
1 3 -2
3 4 7
1 4 4
```



Exemplo de entrada e saída

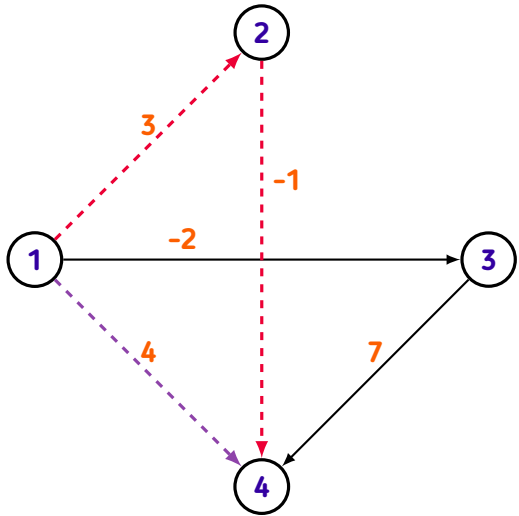
```
4 5

1 2 3
2 4 -1
1 3 -2
3 4 7
1 4 4
```

$3 - 1 = 2$



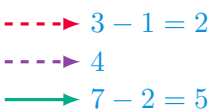
Exemplo de entrada e saída

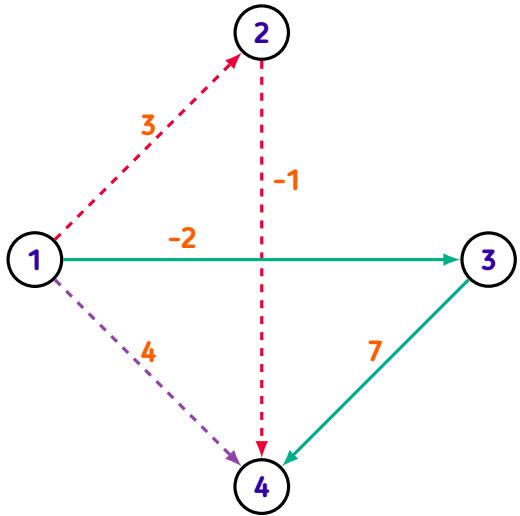
```
4 5

1 2 3
2 4 -1
1 3 -2
3 4 7
1 4 4
```



Exemplo de entrada e saída

```
4 5

1 2 3
2 4 -1
1 3 -2
3 4 7
1 4 4
```





Solução

```
ll solve(int N, const vector<edge>& es) {
    vector<ll> dist(N + 1, oo);
    dist[1] = 0;

    for (int i = 1; i <= N - 1; ++i)
        for (auto [a, b, x] : es)
            if (dist[a] < oo and dist[b] > dist[a] + x)
                dist[b] = dist[a] + x;

    set<int> us;

    for (auto [a, b, x] : es)
        if (dist[a] < oo and dist[b] > dist[a] + x) {
            us.insert(b); dist[b] = dist[a] + x;
        }

    if (dfs(N, us)) return -1;

    return -dist[N];
}
```

```
bool dfs(int u, const set<int>& us)
{
    if (visited[u])
        return false;

    if (us.count(u))
        return true;

    visited[u] = true;

    for (auto v : adj[u])
        if (dfs(v, us))
            return true;

    return false;
}
```