OJ 1112

Mice and Maze

Prof. Edson Alves

Faculdade UnB Gama

A set of laboratory mice is being trained to escape a maze. The maze is made up of cells, and each cell is connected to some other cells. However, there are obstacles in the passage between cells and therefore there is a time penalty to overcome the passage. Also, some passages allow mice to go one-way, but not the other way round.

Suppose that all mice are now trained and, when placed in an arbitrary cell in the maze, take a path that leads them to the exit cell in minimum time.

Um conjunto de ratos de laboratório está sendo treinado para escapar de um labirinto. O labirinto é formado por celas, e cada cela está conectada a algumas outras celas. Contudo, há obstáculos nas passagens entre as celas e portanto há uma penalidade de tempo para atravessar a passagem. Além disso, algumas passagens permitem aos ratos seguir em uma direção, mas não na direção contrária.

Suponha que todos os ratos estejam treinados e, quando postos em uma cela arbitrária do labirinto, eles escolhem o caminho que os leva a saída no menor tempo possível.

We are going to conduct the following experiment: a mouse is placed in each cell of the maze and a count-down timer is started. When the timer stops we count the number of mice out of the maze.

Write a program that, given a description of the maze and the time limit, predicts the number of mice that will exit the maze. Assume that there are no bottlenecks in the maze, i.e. that all cells have room for an arbitrary number of mice.

Vamos conduzir o seguinte experimento: um rato será colocado em cada uma das celas do labirinto e um temporizador será iniciado. Quando o temporizador finalizar sua contagem decrescente contaremos o número de ratos que conseguiu escapar do labirinto.

Escreva um programa que, dada a descrição do labirinto e o limite de tempo, compute o número de ratos que irão escapar do labirinto. Assuma que não há gargalos no labirinto, isto é, que todas as celas tem espaço suficiente para um número arbitrário de ratos.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The maze cells are numbered $1, 2, \ldots, N$, where N is the total number of cells. You can assume that $N \leq 100$.

The first three input lines contain N, the number of cells in the maze, E, the number of the exit cell, and the starting value T for the count-down timer (in some arbitrary time unit).

Entrada

A entrada começa com uma linha contendo um único inteiro positivo, o qual indica o número de casos de teste que se seguem, cada um deles descritos como indicado abaixo. Esta linha é seguida por uma linha em branco, e há uma linha em branco entre dois casos de teste consecutivos.

As celas do labirinto são numeradas $1, 2, \ldots, N$, onde N é o número total de celas. Você pode assumir que $N \leq 100$.

As três primeiras linhas de um caso de teste contém o inteiro N, o número de celas no labirinto, E, o número da cela onde está a saída, e o valor inicial T do temporizador (em alguma unidade de tempo arbitrária).

Input

The fourth line contains the number M of connections in the maze, and is followed by M lines, each specifying a connection with three integer numbers: two cell numbers a and b (in the range $1, \ldots, N$) and the number of time units it takes to travel from a to b.

Notice that each connection is one-way, i.e., the mice can't travel from b to a unless there is another line specifying that passage. Notice also that the time required to travel in each direction might be different.

Entrada

A quarta linha contém o número M de passagens no labirinto, e é seguida por M linhas, cada uma especificando uma passagem por meio de três números inteiros: os números das celas a e b (no intervalo $1, \ldots, N$) e o tempo necessário para ir da cela a para a cela b.

Note que cada passagem é de mão única, isto é, os ratos não podem ir de b para a a menos que exista uma outra linha especificando esta passagem. Note também que o tempo necessário para atravessar a passagem em uma direção pode ser diferente do tempo necessário para atravessá-la na outra direção.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The output consists of a single line with the number of mice that reached the exit cell E in at most T time units.

Saída

Para cada caso de teste, a saída deve seguir a descrição abaixo. As saídas de dois casos de teste consecutivos devem ser separadas por uma linha em branco.

A saída consiste em uma única linha com o número de ratos que chegaram a saída E em, no máximo, T unidades de tempo.

Exemplo de entrada e saída

Exemplo de entrada e saída

1

Exemplo de entrada e saída

1 ⟶ *# de casos de teste*

Exemplo de entrada e saída

1

4

Exemplo de entrada e saída

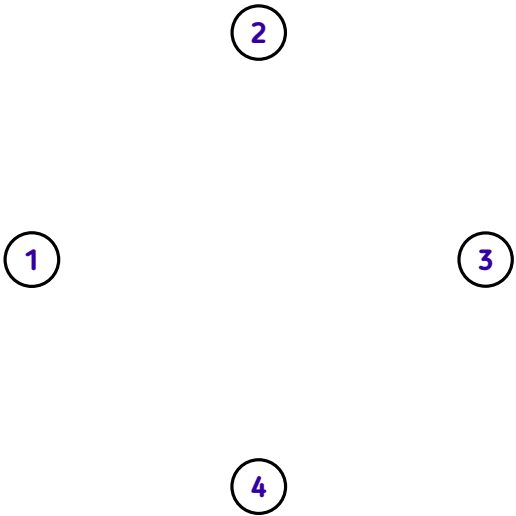
1

4 ⟶ *# de celas*

Exemplo de entrada e saída

1

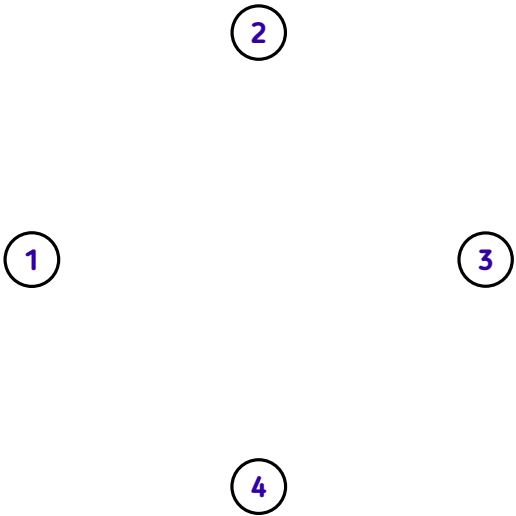
4 ⟶ *# de celas*



Exemplo de entrada e saída

1

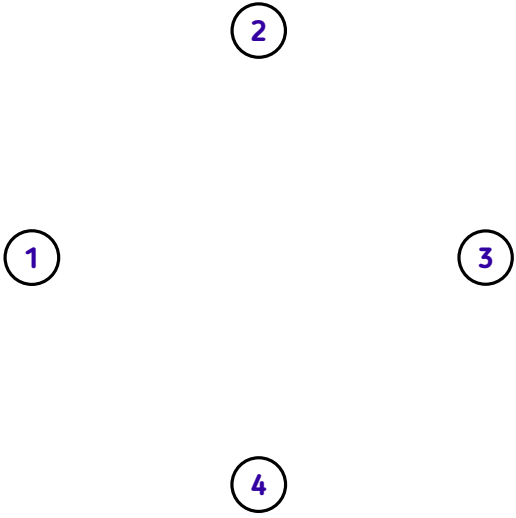
4
2



Exemplo de entrada e saída

```
1

4
2 ⟶ # da saída
```



Exemplo de entrada e saída

1

4
2 ⟶ *# da saída*

Exemplo de entrada e saída

1

4
2 ⟶ *# da saída*

Solução

Solução

Como computar o caminho máximo usando Bellman-Ford?

Solução

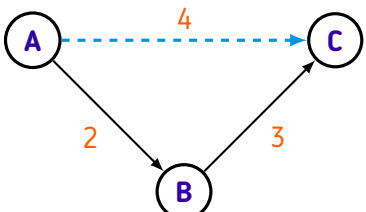
Como computar o caminho máximo usando Bellman-Ford?

Inverta o sinal das arestas e compute o caminho mínimo!

Solução

Como computar o caminho máximo usando Bellman-Ford?

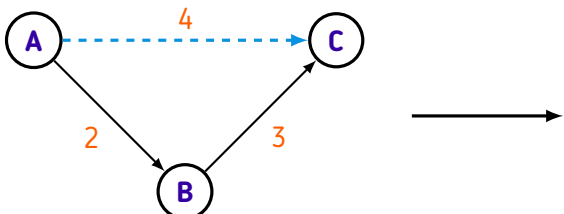
Inverta o sinal das arestas e compute o caminho mínimo!



Solução

Como computar o caminho máximo usando Bellman-Ford?

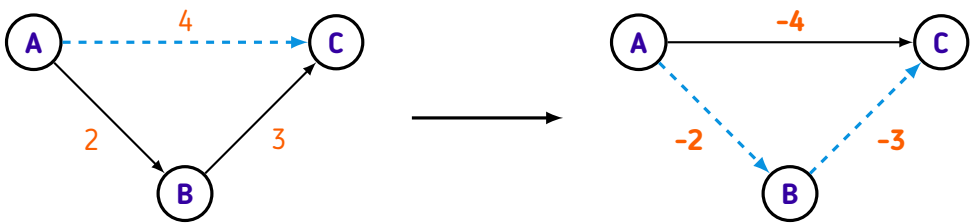
Inverta o sinal das arestas e compute o caminho mínimo!



Solução

Como computar o caminho máximo usando Bellman-Ford?

Inverta o sinal das arestas e compute o caminho mínimo!



Solução

Solução

Como identificar ao menos um elemento de um ciclo negativo?

Solução

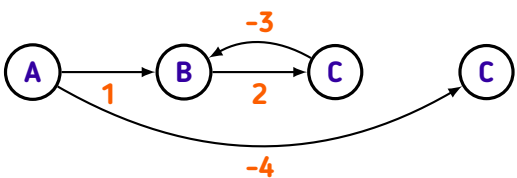
Como identificar ao menos um elemento de um ciclo negativo?

Se dist$[u]$ *é atualizado no Round #*$|V|$*, então* u *pertence a ao menos um ciclo negativo!*

Solução

Como identificar ao menos um elemento de um ciclo negativo?

Se dist$[u]$ *é atualizado no Round #*$|V|$*, então* u *pertence a ao menos um ciclo negativo!*



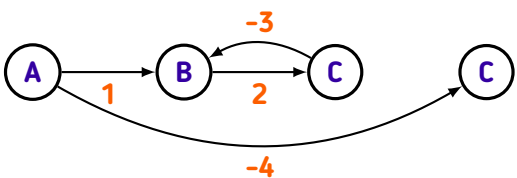
Round #1

A	B	C	D
0	0	3	-4

Solução

Como identificar ao menos um elemento de um ciclo negativo?

Se dist$[u]$ *é atualizado no Round #*$|V|$*, então* u *pertence a ao menos um ciclo negativo!*



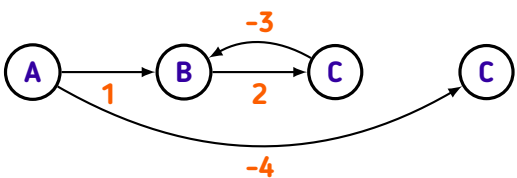
Round #2

A	B	C	D
0	-1	2	-4

Solução

Como identificar ao menos um elemento de um ciclo negativo?

Se dist$[u]$ *é atualizado no Round #*$|V|$*, então* u *pertence a ao menos um ciclo negativo!*



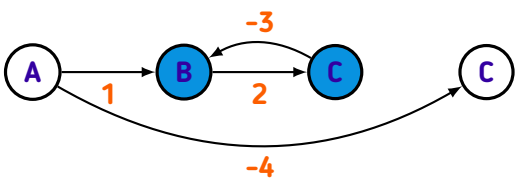
Round #3

A	B	C	D
0	-2	1	-4

Solução

Como identificar ao menos um elemento de um ciclo negativo?

Se $\text{dist}[u]$ *é atualizado no Round #*$|V|$*, então* u *pertence a ao menos um ciclo negativo!*



Round #4

A	B	C	D
0	-3	0	-4

Solução

Solução

Em que casos a pontuação pode ser arbitrariamente grande?

Solução

Em que casos a pontuação pode ser arbitrariamente grande?

Quando há um caminho de um nó de um ciclo negativo até N!

Solução

Em que casos a pontuação pode ser arbitrariamente grande?

Quando há um caminho de um nó de um ciclo negativo até N!

E como posso identificar a existência ou não de tais caminhos?

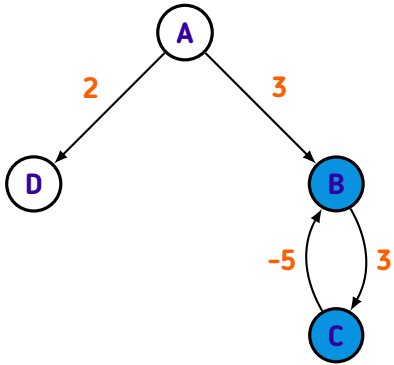
Solução

Em que casos a pontuação pode ser arbitrariamente grande?

Quando há um caminho de um nó de um ciclo negativo até N!

E como posso identificar a existência ou não de tais caminhos?

Inverta as arestas e use uma DFS!


A
2
3
D
B
-5
3
C

```
vector<int> dijkstra(int s, int N)
{
    vector<int> dist(N + 1, oo);
    dist[s] = 0;

    set<ii> U;
    U.emplace(0, s);

    while (not U.empty())
    {
        auto [d, u] = *U.begin();
        U.erase(U.begin());

        for (auto [v, w] : adj[u])
        {
            if (dist[v] > d + w)
            {
                if (U.count(ii(dist[v], v)))
                    U.erase(ii(dist[v], v));
```

```
                dist[v] = d + w;
                U.emplace(dist[v], v);
            }
        }
    }

    return dist;
}

int solve(int N, int E, int T)
{
    auto dist = dijkstra(E, N);

    auto ans = 0;

    for (int u = 1; u <= N; ++u)
        ans += (dist[u] <= T ? 1 : 0);

    return ans;
}
```