

# CSES 1202

*Investigation*

**Prof. Edson Alves**

***Faculdade UnB Gama***

*You are going to travel from Syrjälä to Lehmälä by plane. You would like to find answers to the following questions:*

- ▶ *what is the minimum price of such a route?*
- ▶ *how many minimum-price routes are there? (modulo  $10^9 + 7$ )*
- ▶ *what is the minimum number of flights in a minimum-price route?*
- ▶ *what is the maximum number of flights in a minimum-price route?*

Você irá viajar de Syrjälä para Lehmälä de avião. Você gostaria de encontrar respostas para as seguintes questões:

- ▶ qual é o preço mínimo de tal rota?
- ▶ existem quantas rotas de preço mínimo? (módulo  $10^9 + 7$ )
- ▶ qual é o número mínimo de vôos em uma rota de preço mínimo?
- ▶ qual é o número máximo de vôos em uma rota de preço mínimo?

## Input

*The first input line contains two integers  $n$  and  $m$ : the number of cities and the number of flights. The cities are numbered  $1, 2, \dots, n$ . City 1 is Syrjälä, and city  $n$  is Lehmälä.*

*After this, there are  $m$  lines describing the flights. Each line has three integers  $a, b$ , and  $c$ : there is a flight from city  $a$  to city  $b$  with price  $c$ . All flights are one-way flights.*

*You may assume that there is a route from Syrjälä to Lehmälä.*

## Entrada

A primeira linha da entrada contém dois inteiros  $n$  e  $m$ : o número de cidades e o número de vôos. As cidades são numeradas  $1, 2, \dots, n$ . A cidade  $1$  é Syrjälä e a cidade  $n$  é Lehmälä.

Após isto, há  $m$  linhas descrevendo os vôos. Cada linha tem três inteiros  $a, b$ , e  $c$ : há um vôo da cidade  $a$  para a cidade  $b$  com preço  $c$ . Todos vôos são dados em sentido único.

Você pode assumir que existe uma rota de Syrjälä para Lehmälä.

## Output

*Print four integers according to the problem statement.*

## Constraints

- ▶  $1 \leq n \leq 10^5$
- ▶  $1 \leq m \leq 2 \times 10^5$
- ▶  $1 \leq a, b \leq n$
- ▶  $1 \leq c \leq 10^9$

## Saída

Imprima quatro inteiros, de acordo com o texto do problema.

## Restrições

- ▶  $1 \leq n \leq 10^5$
- ▶  $1 \leq m \leq 2 \times 10^5$
- ▶  $1 \leq a, b \leq n$
- ▶  $1 \leq c \leq 10^9$

## **Exemplo de entrada e saída**



## Exemplo de entrada e saída

4 5

## Exemplo de entrada e saída

4 5



*# de cidades*

## Exemplo de entrada e saída

4 5  
↑  
# de vôos

## Exemplo de entrada e saída

4 5

1

2

3

4

## Exemplo de entrada e saída

4 5

1 4 5

1

2

3

4

## Exemplo de entrada e saída

4 5  
1 4 5  
↑  
*a*

1

2

3

4

## Exemplo de entrada e saída

4 5  
1 4 5  
↑  
*b*

1

2

3

4

## Exemplo de entrada e saída

4 5  
1 4 5  
↑  
*c*

1

2

3

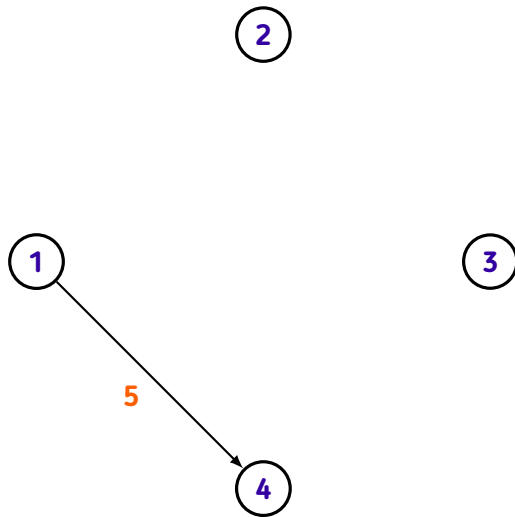
4



## Exemplo de entrada e saída

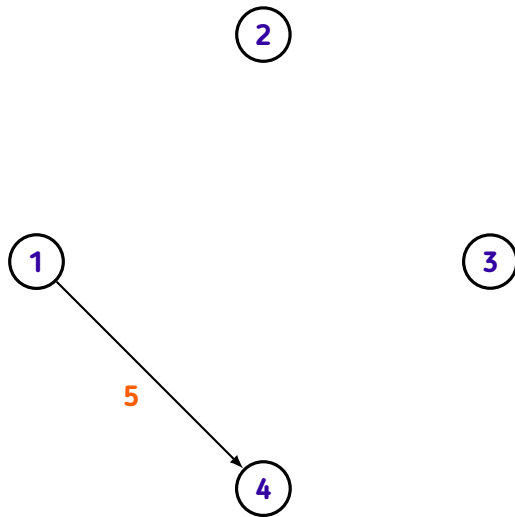
4 5

1 4 5



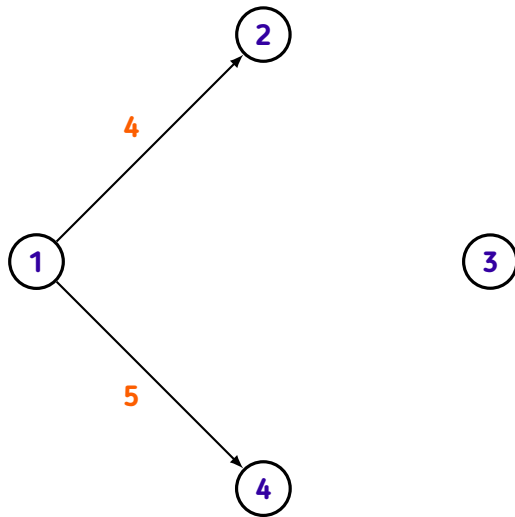
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4



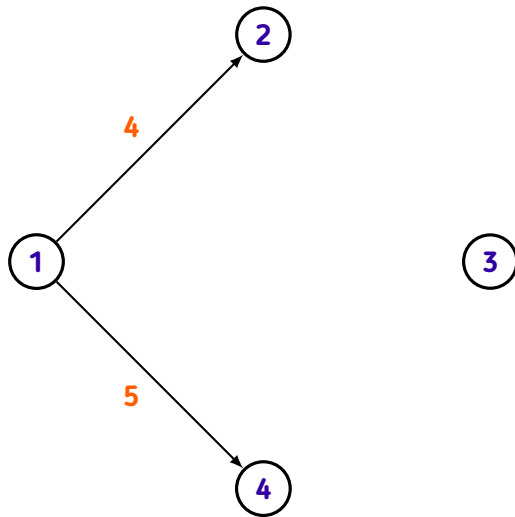
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4



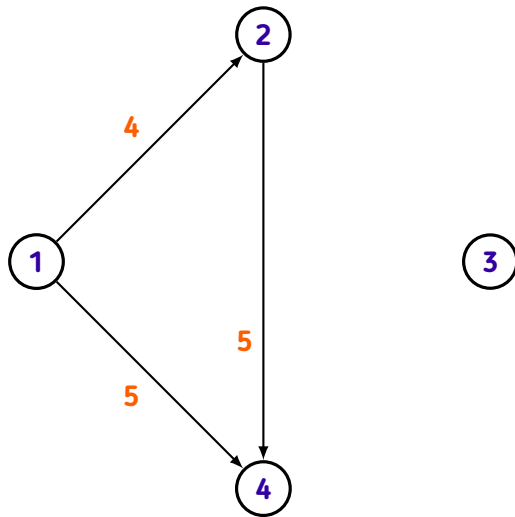
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4  
2 4 5



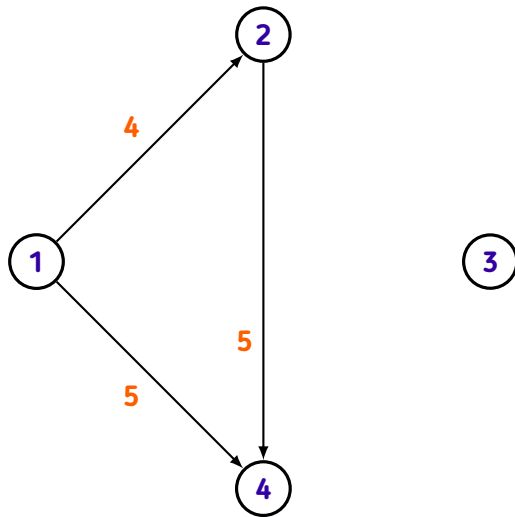
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4  
2 4 5



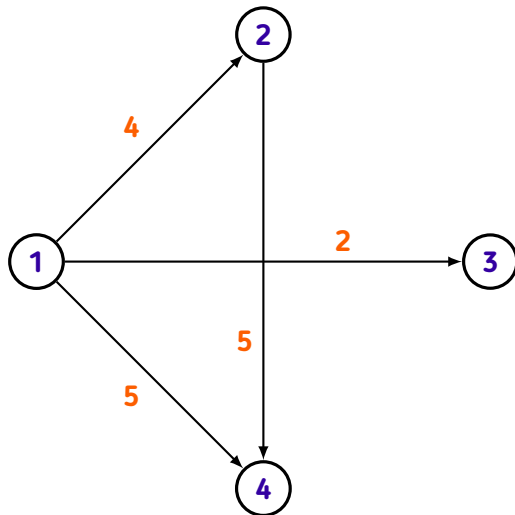
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4  
2 4 5  
1 3 2



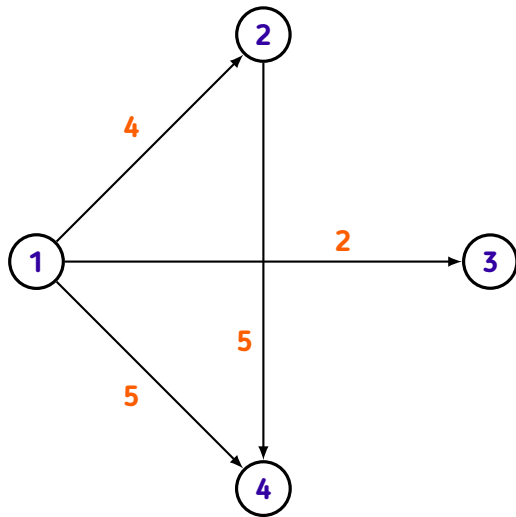
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4  
2 4 5  
1 3 2



## Exemplo de entrada e saída

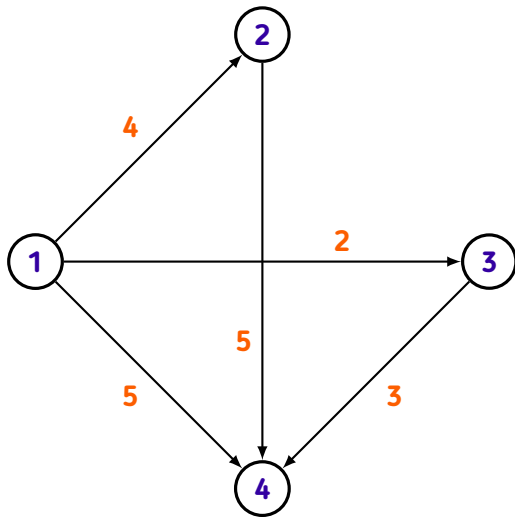
4 5  
1 4 5  
1 2 4  
2 4 5  
1 3 2  
3 4 3





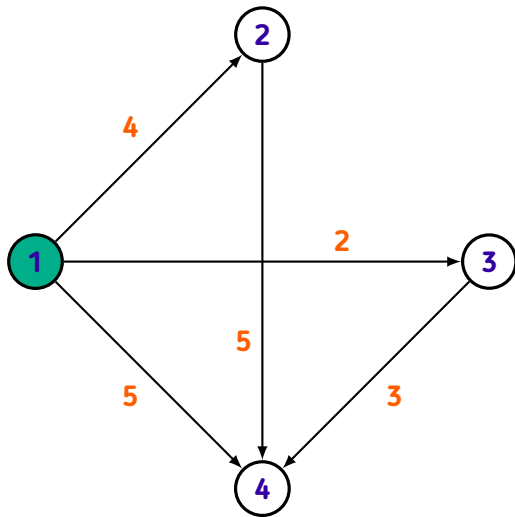
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4  
2 4 5  
1 3 2  
3 4 3



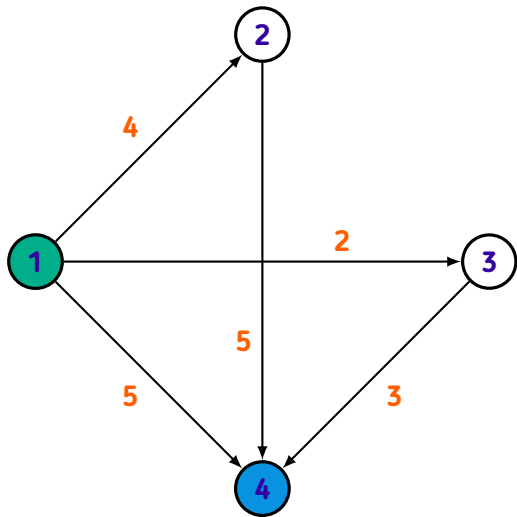
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4  
2 4 5  
1 3 2  
3 4 3



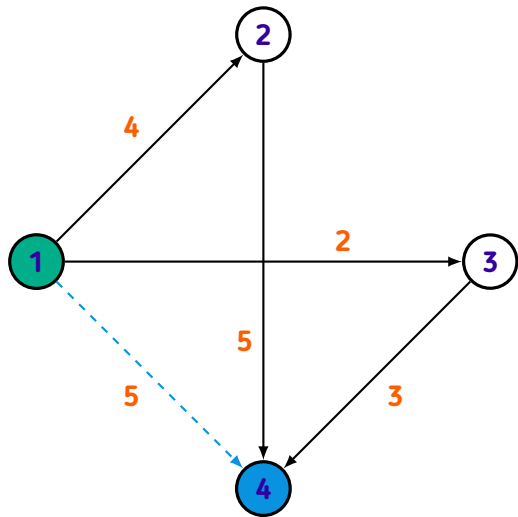
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4  
2 4 5  
1 3 2  
3 4 3



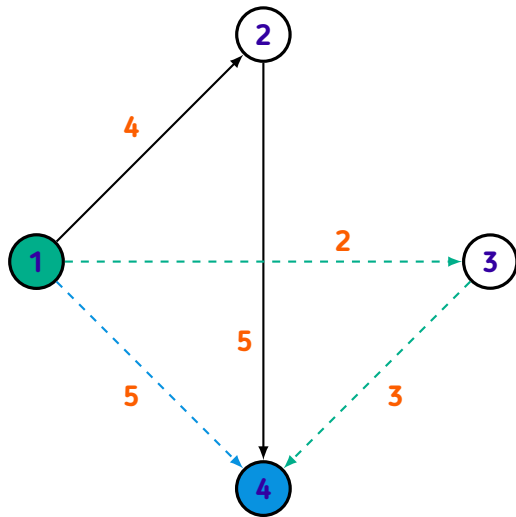
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4  
2 4 5  
1 3 2  
3 4 3



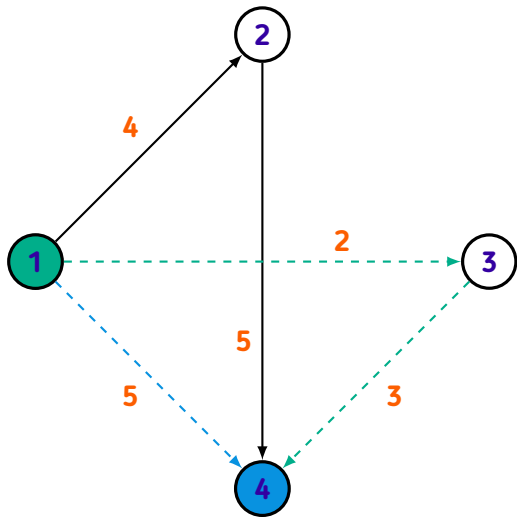
## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4  
2 4 5  
1 3 2  
3 4 3



## Exemplo de entrada e saída

4 5  
1 4 5  
1 2 4  
2 4 5  
1 3 2  
3 4 3  
↓  
5 2 1 2



## Solução

## Solução

- ★ Os quatro subproblemas apresentados podem ser divididos em dois grupos: o problema de distância mínima e os outros três



## Solução

- ★ Os quatro subproblemas apresentados podem ser divididos em dois grupos: o problema de distância mínima e os outros três
- ★ O algoritmo de Dijkstra resolve o problema das distâncias mínimas

## Solução

- ★ Os quatro subproblemas apresentados podem ser divididos em dois grupos: o problema de distância mínima e os outros três
- ★ O algoritmo de Dijkstra resolve o problema das distâncias mínimas
- ★ Além disso, ele gera dois subprodutos úteis para os demais problemas: uma ordenação de vértices  $O$  e um subgrafo  $G'(V, E')$  de  $G(V, E)$

## Solução

- ★ Os quatro subproblemas apresentados podem ser divididos em dois grupos: o problema de distância mínima e os outros três
- ★ O algoritmo de Dijkstra resolve o problema das distâncias mínimas
- ★ Além disso, ele gera dois subprodutos úteis para os demais problemas: uma ordenação de vértices  $O$  e um subgrafo  $G'(V, E')$  de  $G(V, E)$
- ★ A aresta  $(v, u) \in E'$  se  $(u, v) \in E$  finaliza um caminho mínimo de 1 a  $v$

# Solução

- ★ A partir de  $G'$  os três outros subproblemas podem ser resolvidos por DP

## Solução

- ★ A partir de  $G'$  os três outros subproblemas podem ser resolvidos por DP
- ★ Os casos base são:  $\text{minPaths}[1] = 1$  e  $\text{minEdges}[1] = \text{maxEdges}[1] = 0$

## Solução

- ★ A partir de  $G'$  os três outros subproblemas podem ser resolvidos por DP
- ★ Os casos base são:  $\text{minPaths}[1] = 1$  e  $\text{minEdges}[1] = \text{maxEdges}[1] = 0$
- ★ As transições são dadas por:

## Solução

- ★ A partir de  $G'$  os três outros subproblemas podem ser resolvidos por DP
- ★ Os casos base são:  $\text{minPaths}[1] = 1$  e  $\text{minEdges}[1] = \text{maxEdges}[1] = 0$
- ★ As transições são dadas por:

$$\text{minPaths}[u] = \sum_{(v,u) \in E'} \text{minPaths}[v]$$

## Solução

- ★ A partir de  $G'$  os três outros subproblemas podem ser resolvidos por DP
- ★ Os casos base são:  $\text{minPaths}[1] = 1$  e  $\text{minEdges}[1] = \text{maxEdges}[1] = 0$
- ★ As transições são dadas por:

$$\text{minPaths}[u] = \sum_{(v,u) \in E'} \text{minPaths}[v]$$

$$\text{minEdges}[u] = \min_{(v,u) \in E'} \{ \text{minEdges}[u], \text{minEdges}[v] + 1 \}$$



## Solução

- ★ A partir de  $G'$  os três outros subproblemas podem ser resolvidos por DP
- ★ Os casos base são:  $\text{minPaths}[1] = 1$  e  $\text{minEdges}[1] = \text{maxEdges}[1] = 0$
- ★ As transições são dadas por:

$$\text{minPaths}[u] = \sum_{(v,u) \in E'} \text{minPaths}[v]$$

$$\text{minEdges}[u] = \min_{(v,u) \in E'} \{ \text{minEdges}[u], \text{minEdges}[v] + 1 \}$$

$$\text{maxEdges}[u] = \max_{(v,u) \in E'} \{ \text{maxEdges}[u], \text{maxEdges}[v] + 1 \}$$

```
{  
    auto [dist, order] = dijkstra(1, N);  
    auto [ps, ms, Ms] = min_paths(1, N, order);  
  
    return { dist[N], ps[N], ms[N], Ms[N] };  
}
```

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

    processed.reset();

    priority_queue<ii, vector<ii>, greater<ii>> pq;
    pq.emplace(0, s);

    while (not pq.empty())
    {
        auto [d, u] = pq.top();
        pq.pop();

        if (processed[u])
            continue;

        order.emplace_back(u);
        processed[u] = true;
    }
}
```

```
    for (auto [v, w] : adj[u])
    {
        if (dist[v] > d + w)
        {
            dist[v] = d + w;
            pq.emplace(dist[v], v);
            in[v].clear();
            in[v].push_back(u);
        } else if (dist[v] == d + w)
        {
            in[v].push_back(u);
        }
    }

    return { dist, order };
}
```

```
tuple<vector<ll>, vector<ll>, vector<ll>>
min_paths(int s, int N, const vector<ll>& order)
{
    vector<ll> ps(N + 1, 0), ms(N + 1, oo), Ms(N + 1, 0);

    ps[s] = 1;
    ms[s] = 0;

    for (auto x : order)
        for (auto v : in[x])
        {
            ps[x] = (ps[x] + ps[v]) % MOD;
            ms[x] = min(ms[x], 1 + ms[v]);
            Ms[x] = max(Ms[x], 1 + Ms[v]);
        }

    return { ps, ms, Ms };
}
```