

# OJ 10171

*Meeting Prof. Miguel...*

**Prof. Edson Alves**

**Faculdade UnB Gama**

*I have always thought that someday I will meet Professor Miguel, who has allowed me to arrange so many contests. But I have managed to miss all the opportunities in reality. At last with the help of a magician I have managed to meet him in the magical **City of Hope**. The city of hope has many roads. Some of them are bi-directional and others are unidirectional. Another important property of these streets are that some of the streets are for people whose age is less than thirty and rest are for the others. This is to give the minors freedom in their activities. Each street has a certain length. Given the description of such a city and our initial positions, you will have to find the most suitable place where we can meet. The most suitable place is the place where our combined effort of reaching is minimum. You can assume that I am 25 years old and Prof. Miguel is 40+.*

Eu sempre pensei que algum dia eu iria me encontrar com o Professor Miguel, que tem me permitido preparar muitos eventos. Mas, de fato, eu perdi todas as oportunidades. Enfim, com a ajuda de um mágico eu consegui encontrá-lo na **mágica Cidade da Esperança**. A cidade da esperança tem muitas ruas. Algumas delas são bidirecionais, outras unidirecionais. Outra propriedade importante destas ruas é que algumas delas são para pessoas com menos do que 30 anos e as outras são para os demais cidadãos. Isto é feito para dar aos menores liberdade em suas atividades. Cada rua tem um certo comprimento. Dada a descrição de cada cidade e as nossas posições iniciais, determine o local mais apropriado para o encontro. O lugar mais apropriado é o lugar cujo esforço combinado de ambos para chegar lá é mínimo. Você pode assumir que eu tenho 25 anos e o Prof. Miguel tem 40 ou mais.



Figura: *Shahriar Manzoor and Miguel A. Revilla (Shanghai, 2005). First meeting after five years of collaboration*



Figura: **Shahriar Manzoor e Miguel A. Revilla (Shanghai, 2005). Primeiro encontro após cinco anos de colaboração**

## Input

The input contains several descriptions of cities. Each description of city is started by a integer  $N$ , which indicates how many streets are there. The next  $N$  lines contain the description of  $N$  streets.

The description of each street consists of four uppercase alphabets and an integer. The first alphabet is either 'Y' (indicates that the street is for young) or 'M' (indicates that the road is for people aged 30 or more) and the second character is either 'U' (indicates that the street is unidirectional) or 'B' (indicates that the street is bi-directional). The third and fourth characters,  $X$  and  $Y$  can be any uppercase alphabet and they indicate that place named  $X$  and  $Y$  of the city are connected (in case of unidirectional it means that there is a one-way street from  $X$  to  $Y$ ) and the last non-negative integer  $C$  indicates the energy required to walk through the street. If we are in the same place we can meet each other in zero cost anyhow. Every energy value is less than 500.

## Input

A entrada contém várias descrições de cidades. Cada descrição de cidade começa com um inteiro  $N$ , o qual indica quantas ruas há na cidade. As próximas  $N$  linhas contém as descrições das  $N$  ruas.

A descrição de cada rua consiste em quatro letras maiúsculas do alfabeto e um inteiro. A primeira letra é ou 'Y' (indicando que a rua é para os jovens) ou 'M' (indicando que a rua é para pessoas com 30 anos ou mais) e o segundo caractere ou é 'U' (indicando que a rua é unidirecional) ou 'B' (indicando que a rua é bidirecional). O terceiro e o quarto caracteres  $X$  e  $Y$  podem ser qualquer letra maiúscula do alfabeto e indicam que os lugares  $X$  e  $Y$  da cidade estão conectados (no caso unidirecional significa que há uma rua de mão única com sentido de  $X$  a  $Y$ ) e o inteiro não-negativo  $C$  indica que a energia necessária para caminhar pela rua. Se ambos estamos no mesmo lugar nós podemos nos encontrar com custo zero, de algum modo. Os valores da energia são menores do que 500.

*After the description of the city the last line of each input contains two place names, which are the initial position of me and Prof. Miguel respectively.*

*A value zero for  $N$  indicates end of input.*

## **Output**

*For each set of input, print the minimum energy cost and the place, which is most suitable for us to meet. If there is more than one place to meet print all of them in lexicographical order in the same line, separated by a single space. If there is no such places where we can meet then print the line 'You will never meet.'*



Após a descrição de uma cidade a última linha de cada entrada contém o nome de dois lugares, os quais são as posições iniciais onde estão eu e o prof. Miguel, respectivamente.

O valor  $N$  igual a zero indica o fim da entrada.

## Saída

Para cada conjunto da entrada, imprima o custo mínimo de energia necessário e o lugar mais apropriado para o encontro. Se existe mais de um lugar imprima todos eles, em ordem lexicográfica, na mesma linha, separados por um único espaço em branco. Se não há um lugar apropriado para o encontro imprima a mensagem 'You will never meet.'

## **Exemplo de entrada e saída**

## Exemplo de entrada e saída

4

## Exemplo de entrada e saída

4  
↑  
*# de estradas*

## Exemplo de entrada e saída

4

A

B

C

D

## Exemplo de entrada e saída

4  
Y U A B 4  
↑  
*público da rua*

A

B

C

D

## Exemplo de entrada e saída

4  
Y U A B 4  
↑  
*sentido do tráfego*

A

B

C

D

## Exemplo de entrada e saída

4  
Y U A B 4  
↑  
*ponto de partida*

A

B

C

D



## Exemplo de entrada e saída

4

Y U A B 4



*ponto de chegada*

A

B

C

D

## Exemplo de entrada e saída

4

Y U A B 4



*custo de energia*

A

B

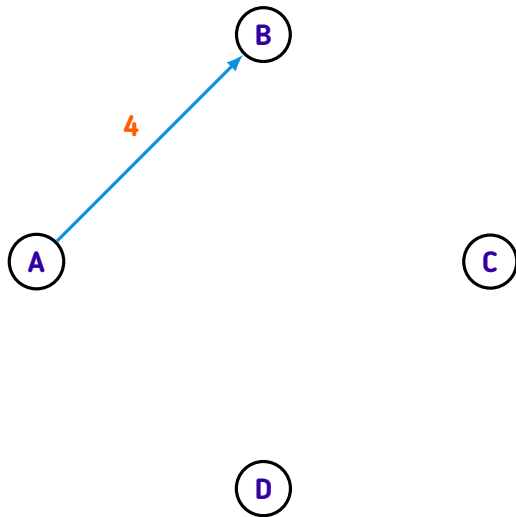
C

D

## Exemplo de entrada e saída

4

Y U A B 4

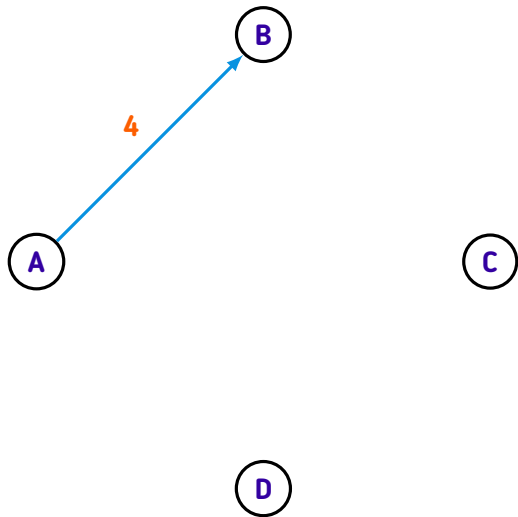


## Exemplo de entrada e saída

4

Y U A B 4

Y U C A 1

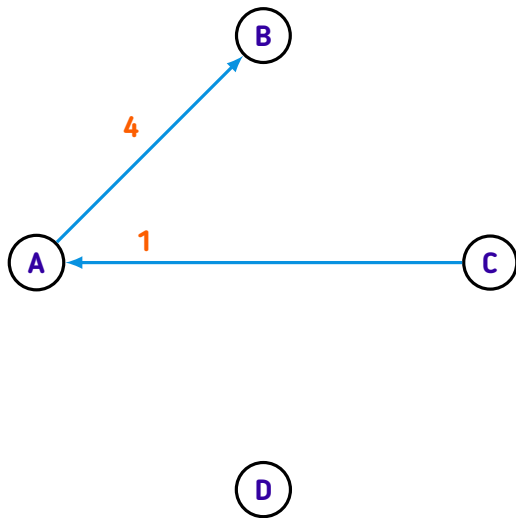


## Exemplo de entrada e saída

4

Y U A B 4

Y U C A 1



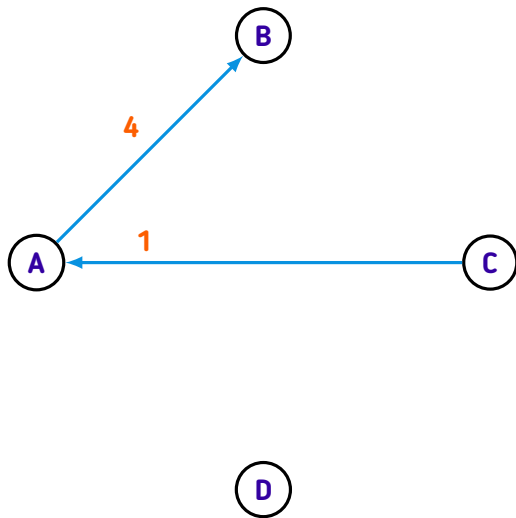
## Exemplo de entrada e saída

4

Y U A B 4

Y U C A 1

M U D B 6



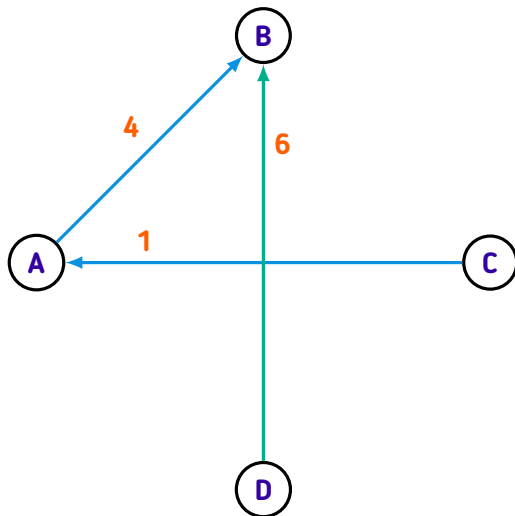
## Exemplo de entrada e saída

4

Y U A B 4

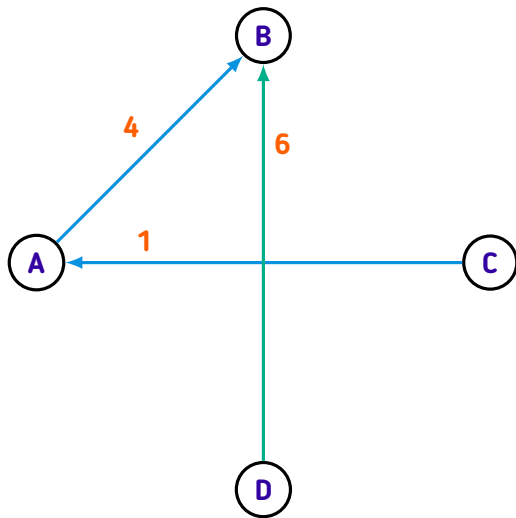
Y U C A 1

M U D B 6



## Exemplo de entrada e saída

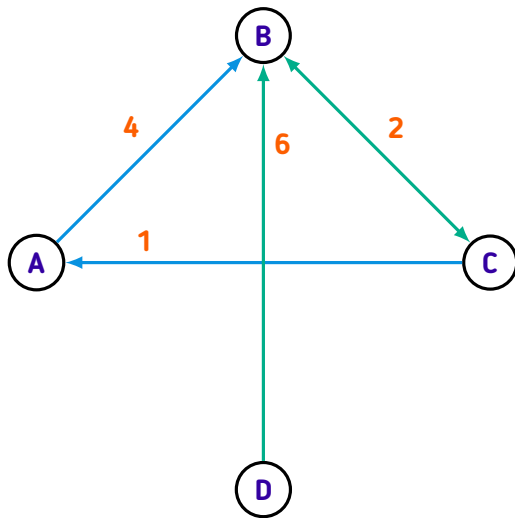
4  
Y U A B 4  
Y U C A 1  
M U D B 6  
M B C D 2





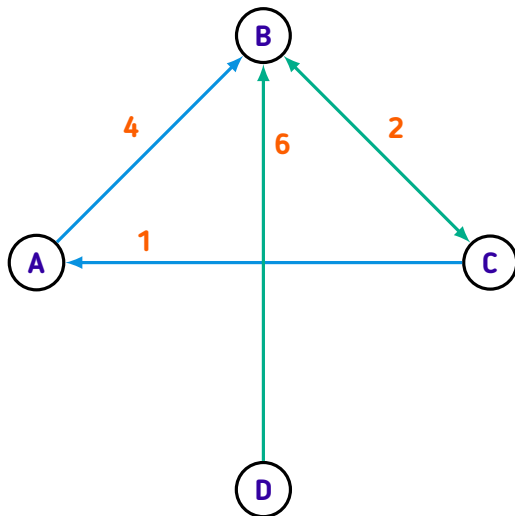
## Exemplo de entrada e saída

4  
Y U A B 4  
Y U C A 1  
M U D B 6  
M B C D 2



## Exemplo de entrada e saída

4  
Y U A B 4  
Y U C A 1  
M U D B 6  
M B C D 2  
A D



## Exemplo de entrada e saída

4

Y U A B 4

Y U C A 1

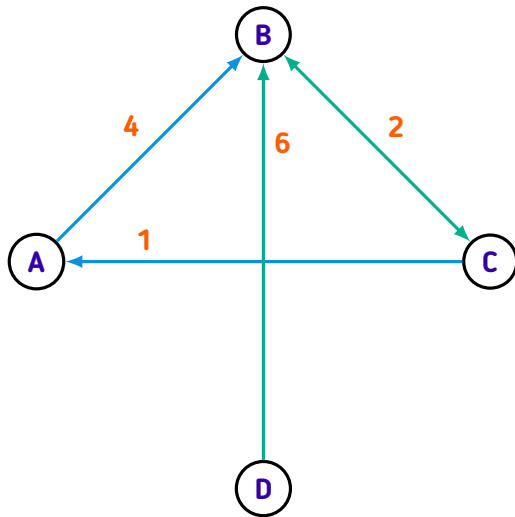
M U D B 6

M B C D 2

A D

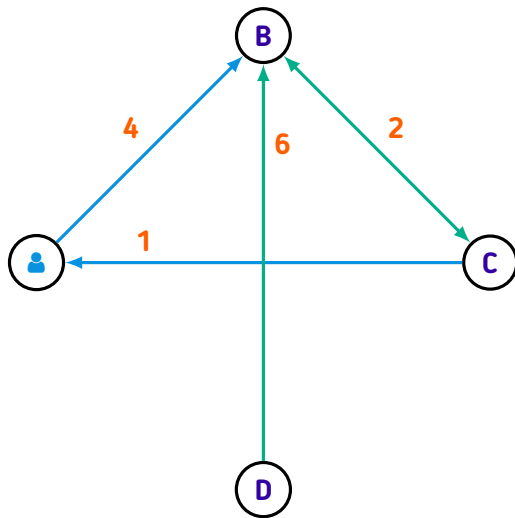


Prof. Shahriar



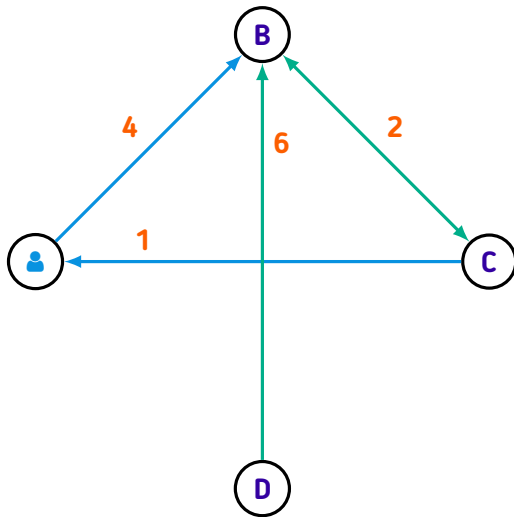
## Exemplo de entrada e saída

4  
Y U A B 4  
Y U C A 1  
M U D B 6  
M B C D 2  
A D



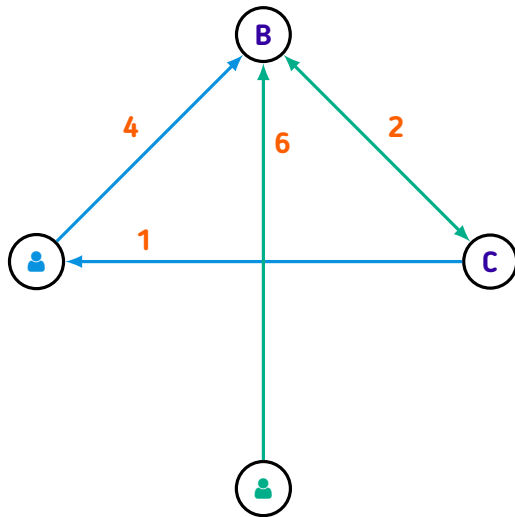
## Exemplo de entrada e saída

4  
Y U A B 4  
Y U C A 1  
M U D B 6  
M B C D 2  
A D  
↑  
Prof. Miguel



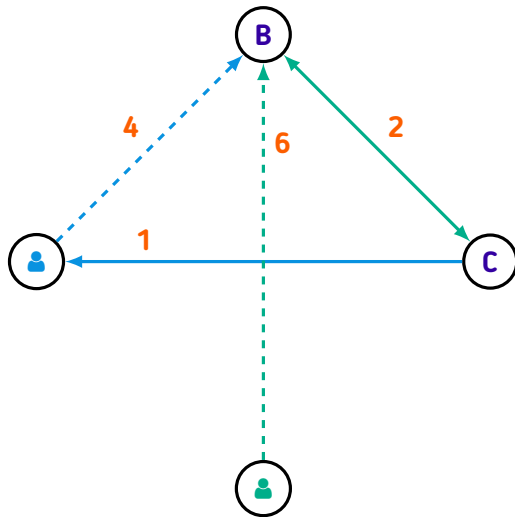
## Exemplo de entrada e saída

4  
Y U A B 4  
Y U C A 1  
M U D B 6  
M B C D 2  
A D



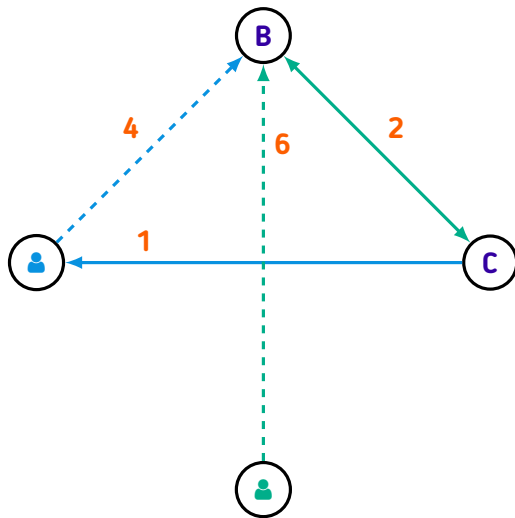
## Exemplo de entrada e saída

4  
Y U A B 4  
Y U C A 1  
M U D B 6  
M B C D 2  
A D



## Exemplo de entrada e saída

4  
Y U A B 4  
Y U C A 1  
M U D B 6  
M B C D 2  
A D  
↑  
10





## **Exemplo de entrada e saída**

## Exemplo de entrada e saída

2

## Exemplo de entrada e saída

2

A

B

C

D

## Exemplo de entrada e saída

2

Y U A B 10

A

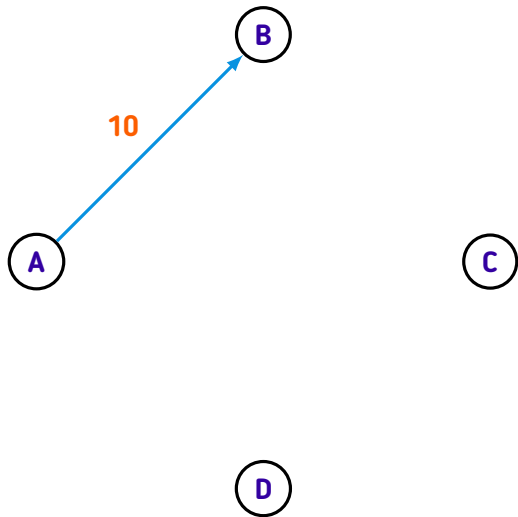
B

C

D

## Exemplo de entrada e saída

2  
Y U A B 10

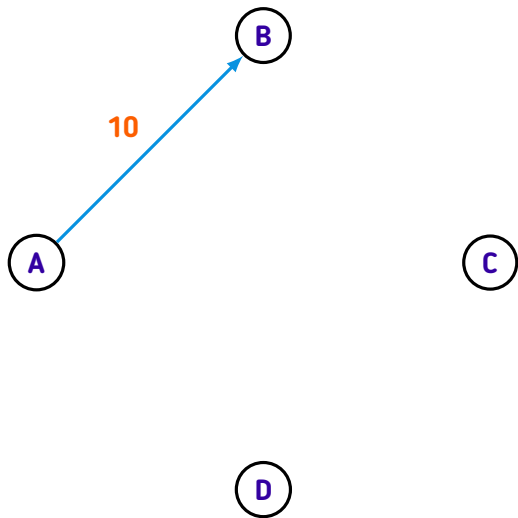


## Exemplo de entrada e saída

2

Y U A B 10

M U C D 20

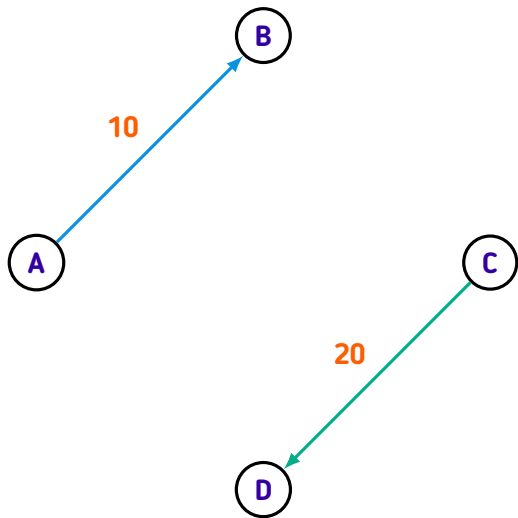


## Exemplo de entrada e saída

2

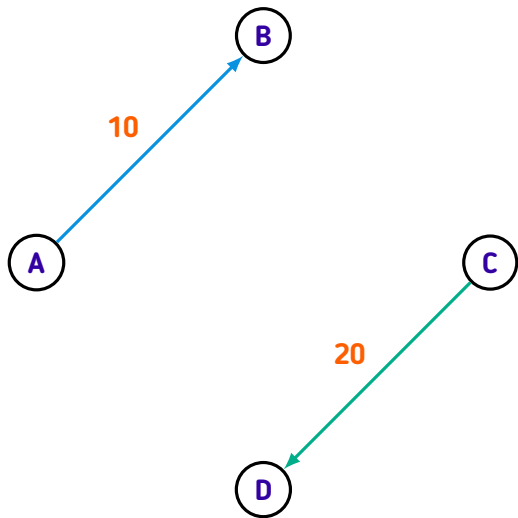
Y U A B 10

M U C D 20



## Exemplo de entrada e saída

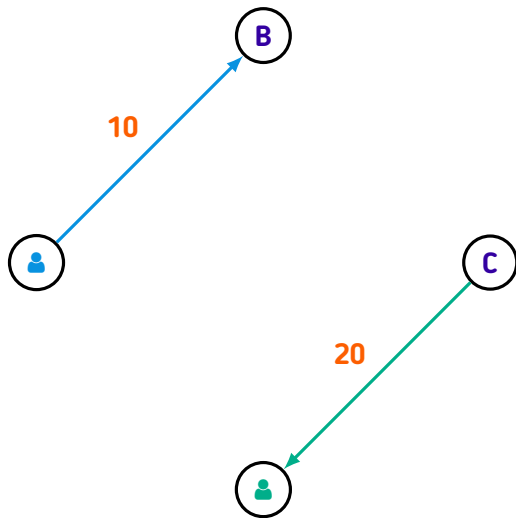
2  
Y U A B 10  
M U C D 20  
A D





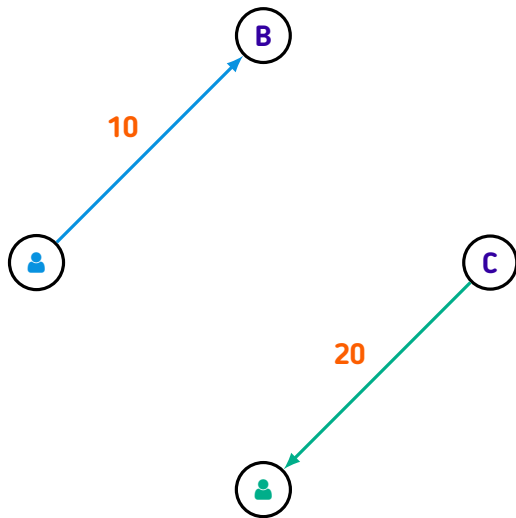
## Exemplo de entrada e saída

2  
Y U A B 10  
M U C D 20  
A D



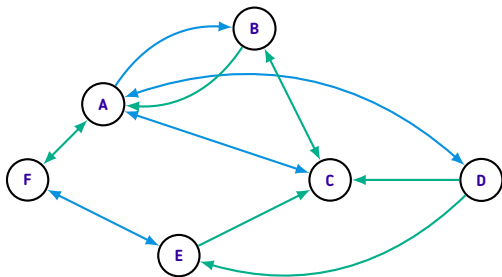
## Exemplo de entrada e saída

2  
Y U A B 10  
M U C D 20  
A D  
↑  
x

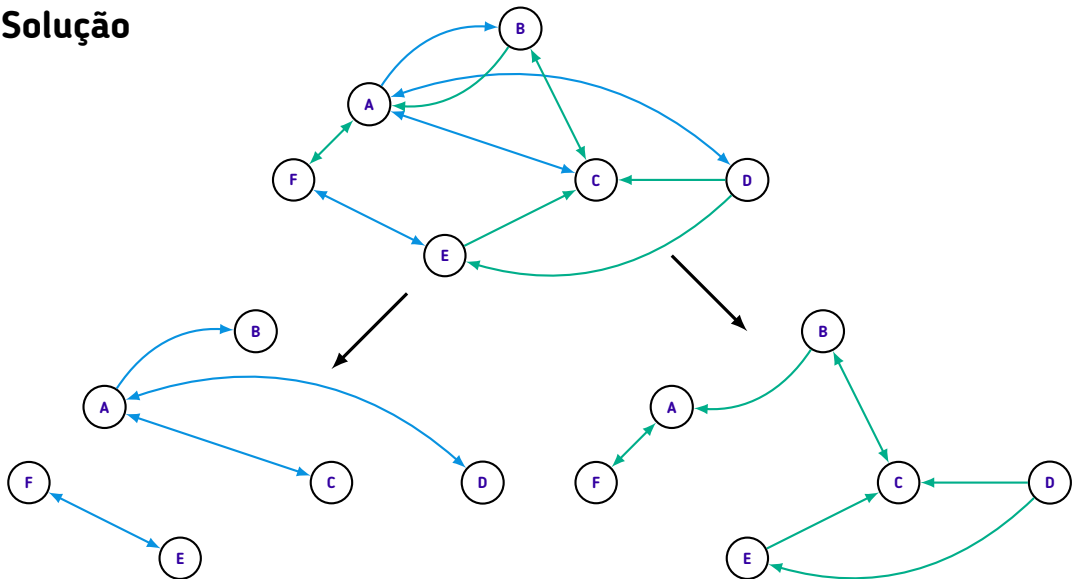


## Solução

## Solução



## Solução



## Solução

## Solução

$$\begin{bmatrix} m_{11} & m_{12} & \dots & \infty \\ m_{21} & \infty & \dots & m_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N1} & m_{N2} & \dots & \infty \end{bmatrix}$$


$$\begin{bmatrix} s_{11} & \infty & \dots & s_{1N} \\ \infty & s_{22} & \dots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \dots & \infty \end{bmatrix}$$

## Solução

$$\begin{bmatrix} m_{11} & m_{12} & \dots & \infty \\ m_{21} & \infty & \dots & m_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N1} & m_{N2} & \dots & \infty \end{bmatrix}$$

$$\begin{bmatrix} s_{11} & \infty & \dots & s_{1N} \\ \infty & s_{22} & \dots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \dots & \infty \end{bmatrix}$$

*distâncias mínimas para  
o prof. Miguel*





## Solução

$$\begin{bmatrix} m_{11} & m_{12} & \dots & \infty \\ m_{21} & \infty & \dots & m_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N1} & m_{N2} & \dots & \infty \end{bmatrix}$$

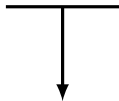
$$\begin{bmatrix} s_{11} & \infty & \dots & s_{1N} \\ \infty & s_{22} & \dots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \dots & \infty \end{bmatrix}$$



*distâncias mínimas para  
o prof. Shahriar*

## Solução

$$\begin{bmatrix} m_{11} & m_{12} & \dots & \infty \\ m_{21} & \infty & \dots & m_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N1} & m_{N2} & \dots & \infty \end{bmatrix}$$



$$\begin{bmatrix} s_{11} & \infty & \dots & s_{1N} \\ \infty & s_{22} & \dots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \dots & \infty \end{bmatrix}$$

$$\begin{bmatrix} d_{11} & \infty & \dots & \infty \\ \infty & \infty & \dots & d_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N1} & d_{N2} & \dots & \infty \end{bmatrix}$$

$$d_{ij} = m_{ij} + s_{ij}$$

```
vector<vector<int>> floyd_warshall(int N, const vector<edge>& edges)
{
    vector<vector<int>> dist(N + 1, vector<int>(N + 1, oo));

    for (const auto& [u, v, w] : edges)
        dist[u][v] = w;

    for (int u = 0; u < N; ++u)
        dist[u][u] = 0;

    for (int k = 0; k < N; ++k)
        for (int u = 0; u < N; ++u)
            for (int v = 0; v < N; ++v)
                dist[u][v] = min(dist[u][v], dist[u][k] + dist[k][v]);

    return dist;
}
```

```
vector<int>
solve(int m, int r, const vector<edge>& ys, const vector<edge>& ms)
{
    auto distM = floyd_warshall(MAX, ys);
    auto distS = floyd_warshall(MAX, ms);

    int min_cost = oo;
    vector<int> ans;

    for (int u = 0; u < MAX; ++u)
    {
        if (distM[r][u] == oo or distS[m][u] == oo)
            continue;

        auto cost = distM[r][u] + distS[m][u];

        if (cost > min_cost)
            continue;
    }
}
```

```
        if (cost == min_cost)
            ans.push_back(u);
        else
        {
            min_cost = cost;
            ans = vector<int> { min_cost, u };
        }
    }

    return ans;
}
```