P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Trabalho Prático de

Matemática Discreta

2022/2023

Trabalho Elaborado por:

Grupo 20

8220169 César Ricardo Barbosa Castelo

8220337 Hugo Ricardo Almeida Guimarães

8220256 João Pedro Da Silva Santos

8220307 Pedro Marcelo Santos Pinho

Curso de:

Licenciatura em Engenharia Informática

Docentes:

Eliana Costa e Silva (eos@estg.ipp.pt)

Isabel Cristina Duarte (icd@estg.ipp.pt)

Felgueiras, 26 de maio de 2023

Conteúdo

Lista de Figuras

Lista de Tabelas

Pergunta 1

Para fazer as alíneas da pergunta 1, antes foi preciso criar um conjunto U, e dois sub-conjuntos A e B do conjunto U. O cálculo das mesmas foi feita a partir do programa *SciLab*[5]

$$U = \{x \in \mathbb{N} : x \leq 30\}, \quad A = \{x \in \mathbb{N} : x \leq 6\}, \quad B = \{x \in \mathbb{N} : 3 \leq x \leq 18\}$$
$$A \subseteq U \quad B \subseteq U$$

a.

$$\#A = 6 \text{ e } \#B = 16$$

O $\#A$ ou $\#B$. são o **cardinal** dos conjuntos, a medida que indica a quantidade de elementos que um conjunto tem.

Para realizar esta operação no SciLab, foi utilizado o comando *length*.

```
--> length(A)
 ans  =

   6.

--> length(B)
 ans  =

   16.
```

Figura 1.1: Cardinalidade dos conjuntos A e B em SciLab

b.

$\overline{A} = \{7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30\}$

O $\bar{A}$ é o **complementar absoluto**, ou simplesmente, **complementar** do conjunto A, que é o conjunto de elementos que pertencem ao conjunto U ,as não pertencem ao conjunto A.

Esta operação foi realizada no SciLab utilizando o comando *setdiff*.

```
--> setdiff(U, A)
 ans  =

         column 1 to 13

   7.   8.   9.   10.   11.   12.   13.   14.   15.   16.   17.   18.   19.

         column 14 to 24

   20.   21.   22.   23.   24.   25.   26.   27.   28.   29.   30.
```

Figura 1.2: Complementar absoluto do conjunto A

c.

$$A \cup B = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18\}$$

$A \cup B$ é a **união** ou **reunião** dos conjunto A e B, ele é o conjunto de todos os elementos que pertencem a A ou B.

Esta operação foi realizada no SciLab utilizando o comando *union*.

```
--> union(A, B)
 ans  =

         column 1 to 14

   1.    2.    3.    4.    5.    6.    7.    8.    9.    10.   11.   12.   13.   14.

         column 15 to 18

   15.   16.   17.   18.
```

Figura 1.3: União do conjunto A com B

d.

$$A \cap B = \{3, 4, 5, 6\}$$

$A \cap B$ é a **interseção** dos conjuntos A e b, que é o conjunto de todos os elementos que pertencem a A e B. Esta operação foi realizada no SciLab utilizando o comando *intersect*.

```
--> intersect(A, B)
 ans  =

    3.    4.    5.    6.
```

Figura 1.4: Interseção do conjunto A com B

e.

$$A - B = \{1, 2\}$$

$A - B$ ou $A \backslash B$, é a **diferença** entre A e B, que é o conjunto que contém os elementos que estão em A mas não estão em B.

Esta operação foi realizada no SciLab utilizando o comando *setdiff*.

```
--> setdiff(A, B)
 ans  =

   1.    2.
```

Figura 1.5: Diferença entre os conjuntos A e B

f.

$$A \oplus B = \{1, 2, 3, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18\}$$

$A \oplus B$ ou $A \cup B - A \cap B$, é a **diferença simétrica** entre o conjunto e B, ela é o conjunto de todos os objetos que são membros de um dos conjuntos. Esta operação foi realizada no SciLab ao utilizar dois comandos já usados anteriormente, sendo eles: o *union* para fazer uma união dos conjuntos A e B; o *intersect* para fazer uma interseção entre os conjuntos A e B; e o *setdiff* para fazer a diferença entre a união e a interseção dos conjuntos A e B.

```
--> uni = union(A, B);

--> inter = intersect(A, B);

--> setdiff(uni, inter)
 ans  =

         column 1 to 13

   1.    2.    7.    8.    9.    10.   11.   12.   13.   14.   15.   16.   17.

         column 14

   18.
```

Figura 1.6: diferença simétrica entre os conjuntos A e B

g.

$$\overline{A \oplus B} \cup (A - B) = \{1, 2, 3, 4, 5, 6, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30\}$$

Esta operação foi realizada no SciLab da seguinte forma: foram realizadas as operações da alínea anterior, e foram guardadas na variável *diferencaSimetrica*, a seguir, fez-se a união, utilizando o comando *union*, entre o complementar absoluto da diferença simétrica de A e B, e a diferença de A e B. As duas operações intermediárias foram realizadas com o comando *setdiff*.

```
--> diferencaSimetrica = setdiff(uni, inter);

--> union(setdiff(U, diferencaSimetrica), setdiff(A, B))
 ans  =

         column 1 to 14

   1.    2.    3.    4.    5.    6.    19.   20.   21.   22.   23.   24.   25.   26.

         column 15 to 18

   27.   28.   29.   30.
```

Figura 1.7: União do complementar absoluto da diferença simétrica de A com B, com a diferença de A com b

h.

$A \times B = \{(1,3),(1,4),(1,5),(1,6),(1,7),(1,8),(1,9),(1,10),(1,11),(1,12),$
$(1,13),(1,14),(1,15),(1,16),(1,17),(1,18),(2,3),(2,4),(2,5),(2,6),(2,7),$
$(2,8),(2,9),(2,10),(2,11),(2,12),(2,13),(2,14),(2,15),(2,16),(2,17),(2,18),$
$(3,3),(3,4),(3,5),(3,6),(3,7),(3,8),(3,9),(3,10),(3,11),(3,12),(3,13),(3,14),$
$(3,15),(3,16),(3,17),(3,18),(4,3),(4,4),(4,5),(4,6),(4,7),(4,8),(4,9),(4,10),$
$(4,11),(4,12),(4,13),(4,14),(4,15),(4,16),(4,17),(4,18),(5,3),(5,4),(5,5),$
$(5,6),(5,7),(5,8),(5,9),(5,10),(5,11),(5,12),(5,13),(5,14),(5,15),(5,16),$
$(5,17),(5,18),(6,3),(6,4),(6,5),(6,6),(6,7),(6,8),(6,9),(6,10),(6,11),(6,12),$
$(6,13),(6,14),(6,15),(6,16),(6,17),(6,18)\}$

$A \times B$ é o **produto cartesiano** entre o conjunto A e o conjunto B, que é o conjunto dos pares ordenados (a, b), onde $a \in A$ e $b \in B$.

Para fazer esta Operação no SciLab foi preciso criar uma função que juntaria cada elemento do conjunto A com cada elemento do conjunto B, a função criada foi nomeada como *produtoCarteseano*.

```
function [pares] = produtoCarteseano(conjuntoA, conjuntoB)
    tamA = length(conjuntoA);
    tamB = length(conjuntoB);

    par = [];
    pares = [];

    for i = 1:tamA
        for j = 1:tamB
            par = [conjuntoA(i), conjuntoB(j)];
            pares = [pares; par];
        end
    end

endfunction
```

Figura 1.8: Função do produto cartesiano

i.

$$A^3 = A \times A \times A = \{(1,1,1),(1,1,2),(1,1,3),...,(6,6,4),(6,6,5),(6,6,6)\}$$

Para fazer A^3 ou o *produto cartesiano* de $A \times A \times A$, foi preciso alterar a função *"produtoCartesiano"* criada na alínea anterior, para que esta suporta-se mais que dois conjuntos.

```
function [pares] = produtoCarteseano(conjuntoA, conjuntoB, conjuntoC)
    tamA = length(conjuntoA);
    tamB = length(conjuntoB);
    tamC = length(conjuntoC);

    par = [];
    pares = [];

    for i = 1:tamA
        for j = 1:tamB
            for k =1:tamC
                par = [conjuntoA(i), conjuntoB(j), conjuntoC(k)];
                pares = [pares; par];
            end
        end
    end

endfunction
```

Figura 1.9: Função do produto cartesiano para A^3

Pergunta 2

Na segunda pergunta, é dito que devemos substituir o β por o último algarismo do número do elemento de algum dos elementos do grupo, e o n por um número natural, então decidiu-se usar o número 7, e o n pelo número 30. Ficando assim:

$$50 + 7 < 2 * 30 < 100 - 7$$

$$\Leftrightarrow 57 < 60 < 93$$

a.

Nesta alínea é pedido para fazer o somatório o seguinte somatório:

$$\prod_{i \in C} \left(\frac{3}{i} - 1 \right)^4 = 0.0001020$$

$$C = \{5m \in \mathbb{Z} : m = 1, ..., M\} \text{ e } M = min\left\{5 + \beta, \left\lceil \frac{100}{\beta+1} \right\rceil \right\}$$

Como $\beta = 7$, então $M = min\left\{5 + 7, \left\lceil \frac{100}{7+1} \right\rceil \right\} \Leftrightarrow M = min\left\{12, \left\lceil \frac{100}{8} \right\rceil \right\}$.

Como $i \in C$ e $5m \in \mathbb{Z}$, então $\prod_{i \in C} \left(\frac{3}{5m} - 1 \right)^4$.

Para simplificar o produtório, usou-se a seguinte regra $\prod(x_i)^k = \left(\prod(x_i)\right)^k$, então a expressão fica $\left(\prod_{i \in C} \left(\frac{3}{i} - 1\right)\right)^4$.

Progressão Aritmética:

$$\begin{aligned} C_{n+1} - C_n &= \frac{2-i}{i+1} - \frac{3-i}{i} = \frac{2i-i^2}{i(i+1)} - \frac{(3-i)(i+1)}{i(i+1)} \\ &= \frac{2i-i^2-(3i+3-i^2-i)}{i(i+1)} = \frac{2i-i^2-2i-3+i^2}{i(i+1)} \\ &= \frac{-3}{i(i+1)} \end{aligned}$$

Progressão Geométrica:

$$\frac{C_{n+1}}{C_n} = \frac{\left(\frac{3-(i+1)}{i+1}\right)}{\left(\frac{3-i}{i}\right)} = \frac{\left(\frac{3-i-1)}{i+1}\right)}{\left(\frac{3-i}{i}\right)} = \frac{\left(\frac{2-i)}{i+1}\right)}{\left(\frac{3-i}{i}\right)} = \left(\frac{i(2-i)}{(3-i)(i+1)}\right)$$

Como ambas as formas não apresentam uma razão constante então **não é uma progressão aritmética nem geométrica**.

Para calcular o valor desta expressão no SciLab, foi criado dois scripts que calculam o valor mínimo do conjunto M com a função *min* e iteram até calcular o valor final. A primeira dessas expressões usa um ciclo for até chegar ao resultado final, enquanto a outra, usa a função *prod* para fazer o mesmo.

```
clear
clc

b = 7; //beta
produtorio = 1;


for m = 1 : min((5 + b), ceil(100/(b+1)))
    produtorio = produtorio * ((3/(5*m)) - 1)^4;
end

disp(produtorio)


// ou


[minimo] = min((5 + b), ceil(100/(b+1)));
produtorio = prod(((3./(5.*(1:minimo))) - 1).^4);
disp(produtorio)
```

Figura 2.1: Funções que calculam o produtório da pergunta 2a

```
0.0001020

0.0001020
```

Figura 2.2: Output função do script da alínea 2a

b.

$$\sum_{j=\beta+3}^{n} \left(\frac{-\beta-1}{5}\right)^{j} = 818028.75$$

Como $\beta = 7$, então $\sum_{j=10}^{n} \left(-\frac{8}{5}\right)^{j}$

$$\left(\frac{Cn+1}{Cn}\right) = \frac{\left(\frac{-8}{5}\right)^{j+1}}{\left(\frac{-8}{5}\right)^{j}} = \frac{\left(\frac{-8}{5}\right)^{j} \times \left(\frac{-8}{5}\right)}{\left(\frac{-8}{5}\right)^{j}} = \left(-\frac{8}{5}\right)$$

Como r (razão) é constante para todos os valores de j, então esta expressão é uma **sucessão geométrica**.

Para calcular o valor desta expressão no SciLab, foi criado dois scripts que iteram até calcular o valor final. A primeira dessas expressões usa um ciclo for

até chegar ao resultado final, enquanto a outra, usa a função *sum* para fazer o mesmo.

```
clear
clc

b = 7; // beta
n = 30;
soma = 0;

for j = (7 + 3) : n
    soma = soma + ((-b-1)/5)^j;
end

disp(soma)

// ou

//n = 30;
soma = sum(((-7-1)/5)^(10:n));
disp(soma);
```

Figura 2.3: Funções que calculam o somatório da pergunta 2a

```
818028.75

818028.75
```

Figura 2.4: Output função do script da alínea 2c

c.

$$\prod_{k=1}^{n-15}\left(3\times\sum_{j=n-5}^{n}\left(\left\lfloor 1+\frac{j+k}{200}\right\rfloor\left\lceil\frac{6!}{\beta}\right\rceil\right)\right)=-1.187\times10^{61}$$

Como $\beta=7$ e $n=30$, então: $\prod_{k=1}^{n-15}\left(3\times\sum_{j=25}^{n}\left(\left\lfloor 1+\frac{j+k}{200}\right\rfloor\left\lceil\frac{6!}{7}\right\rceil\right)\right)$

Progressão Aritmética:

$$C_{n+1}-C_n=\left(3\times\left(\left\lfloor 1+\frac{(j+1)+(k+1)}{200}\right\rfloor\left\lceil\frac{6!}{7}\right\rceil\right)\right)-\left(3\times\left(\left\lfloor 1+\frac{j+k}{200}\right\rfloor\left\lceil\frac{6!}{7}\right\rceil\right)\right)$$

Progressão Geométrica:

$$\left(\frac{Cn+1}{Cn}\right)=\frac{\left(3\times\left(\left\lfloor 1+\frac{(j+1)+(k+1)}{200}\right\rfloor\left\lceil\frac{6!}{7}\right\rceil\right)\right)}{\left(3\times\left(\left\lfloor 1+\frac{j+k}{200}\right\rfloor\left\lceil\frac{6!}{7}\right\rceil\right)\right)}$$

Para calcular o valor desta expressão no SciLab, foi criado um script que itera com dois ciclos for até chegar ao valor final. O primeiro ciclo representa o produtório inicial, já o segundo ciclo representa o somatório que fica dentro do produtório. Dentro do seundo ciclo foi preciso usar o comando *floor*, que significa *maior inteiro que*, e o comando *ceiling*, que significa *menor inteiro que*

```
clear
clc

b = 7; // beta
n = 30;

soma = 0;
prod = 1;

for k = 1 : (n-15)
    for j = (n - 5) : n
        soma = soma + (floor(1+((j+k)/(200)))-ceil(factorial(6)/b));
    end
    prod = prod * (3 * soma);
end

disp(prod)
```

Figura 2.5: Função que calcula a expressão da pergunta 2c

```
-1.187D+61
```

Figura 2.6: Output função do script da alínea 2c

Pergunta 3

Neste exercício é pedido para considerar σ sendo os dois últimos algarismos do número de estudante de um dos elementos do grupo, então escolheram-se os números 07. Ficando Assim:

$$\sigma = 7$$

a.

Como o número $\sigma = 7$, então, neste exercício vai se usar o algoritmo de Dijkstra para calcular o caminho mínimo de Évora a todas as outras capitais do distrito.

Tabela 3.1: Tabela Algoritmo de Dijkstra

It	Vd	Mc	A	vi, ..., vj e L(vj)	X e Xd	R: Caminho mínimo
0		7	{2,11,12,14}	$L(2) = 80$ $L(11) = 150$ $L(12) = 100$ $L(14) = 120$	{2, 12, 14, 11} {80, 100, 120, 150}	**7,2**; 7,12; 7,14; 7,11
1	2	7,2	{8,15}	$L(8) = 250$ $L(15) = 215$	{12, 14, 11, 15, 8} {100, 120, 150, 215, 250}	**7,12**; 7,14; 7,11; 7,2,15; 7,2,8
2	12	7,12	{5,14}	$L(5) = 180$ $L(14) = 250$	{14, 11, 5, 15, 8} {120, 150, 180, 215, 250}	**7,14**; 7,11; 7,12,5; 7,2,15; 7,2,8
3	14	7,14	{5,11,12}	$L(5) = 320$ $L(11) = 190$ $L(12) = 270$	{11, 5, 15, 8, 12} {150, 180, 215, 250, 270}	**7,11**; 7,12,5; 7,2,15; 7,2,8; 7,14,12
4	11	7,11	{10,14,15}	$L(10) = 280$ $L(14) = 220$ $L(15) = 200$	{5, 15, 8, 12, 10} {180, 200, 250, 270, 280}	**7,12,5**; 7,11,15; 7,2,8; 7,14,12; 7,11,10
5	5	7,12,5	{6,9,14}	$L(6) = 340$ $L(9) = 280$ $L(14) = 380$	{15, 8, 12, 10, 9, 6} {200, 250, 270, 280, 280, 340}	**7,11,15**; 7,2,8; 7,14,12; 7,11,10; 7,12,5,9; 7,12,5,6
6	15	7,11,15	{8,11}	$L(8) = 475$ $L(11) = 265$	{8, 12, 10, 9, 6} {250, 270, 280, 280, 340}	**7,2,8**; 7,14,12; 7,11,10; 7,12,5,9; 7,12,5,6
7	8	7,2,8	{15}	$L(15) = 510$	{12, 10, 9, 6} {270, 280, 280, 340}	**7,14,12**; 7,11,10; 7,12,5,9; 7,12,5,6
8	12	7,14,12	{5}	$L(5) = 350$	{10, 9, 6} {280, 280, 340}	**7,11,10**; 7,12,5,9; 7,12,5,6
9	10	7,11,10	{6}	$L(6) = 350$	{9, 6} {280, 340}	**7,12,5,9**; 7,12,5,6
10	9	7,12,5,9	{4,6,17,18}	$L(4) = 480$ $L(6) = 440$ $L(17) = 430$ $L(18) = 360$	{6, 18, 17, 4} {340, 360, 430, 480}	**7,12,5,6**; 7,12,5,9,18; 7,12,5,9,17; 7,12,5,9,4
11	6	7,12,5,6	{1,5,9,18}	$L(1) = 420$ $L(5) = 500$ $L(9) = 500$ $L(18) = 420$	{18, 1, 17, 4} {360, 420, 430, 480}	**7,12,5,9,18**; 7,12,5,6,1; 7,12,5,9,17; 7,12,5,9,4
12	18	7,12,5,9,18	{1,6,17}	$L(1) = 460$ $L(6) = 440$ $L(17) = 470$	{1, 17, 4} {420, 430, 480}	**7,12,5,6,1**; 7,12,5,9,17; 7,12,5,9,4
13	1	7,12,5,6,1	{6,13}	$L(6) = 500$ $L(13) = 490$	{17, 4, 13} {430, 480, 490}	**7,12,5,9,17**; 7,12,5,9,4; 7,12,5,6,1,13
14	17	7,12,5,9,17	{3,4,13,18}	$L(3) = 530$ $L(4) = 570$ $L(13) = 550$ $L(18) = 540$	{4, 13, 3} {480, 490, 530}	**7,12,5,9,4**; 7,12,5,6,1,13; 7,12,5,9,17,3
15	4	7,12,5,9,4	{17}	$L(17) = 620$	{13, 3} {490, 530}	**7,12,5,6,1,13**; 7,12,5,9,17,3
16	13	7,12,5,6,1,13	{3,16,17}	$L(3) = 540$ $L(16) = 570$ $L(17) = 610$	{3, 16} {530, 570}	**7,12,5,9,17,3**; 7,12,5,6,1,13,16
17	3	7,12,5,9,17,3	{13,16}	$L(13) = 580$ $L(16) = 580$	{16} {570}	**7,12,5,6,1,13,16**

Évora a Aveiro: na iteração 11

Évora → Portalegre → Castelo Branco → Coimbra → Aveiro **Custo:** 420

Évora a Beja: na iteração 0

Évora → Beja **Custo:** 80

Évora a Braga: iteração 14

Évora → Portalegre → Castelo Branco → Guarda → Vila Real → Braga

Custo: 530

Évora a Bragança: na iteração 10

Évora → Portalegre → Castelo Branco → Guarda → Bragança **Custo:** 480

Évora a Castelo Branco: na iteração 2

Évora → Portalegre → Castelo Branco **Custo:** 180

Évora a Coimbra: na iteração 5

Évora → Portalegre → Castelo Branco → Coimbra **Custo:** 340

Évora a faro: na iteração 1

Évora → Beja → Faro **Custo:** 250

Évora a Guarda: na iteração 5

Évora → Portalegre → Castelo Branco → Guarda **Custo:** 280

Évora a Leiria: na iteração 4

Évora → Lisboa → Leiria **Custo:** 280

Évora a Lisboa: na iteração 0

Évora → Lisboa **Custo:** 150

Évora a Portalegre: na iteração 0

Évora → Portalegre **Custo:** 100

Évora a Porto: iteração 13

Évora → Portalegre→ Castelo Branco → Coimbra → Aveiro → Porto

Custo: 490

Évora a Santarém: na iteração 0

Évora → Santarém **Custo:** 120

Évora a Setúbal: na iteração 1

Évora → Lisboa → Setúbal **Custo:** 200

Évora a Viana do Castelo: iteração 16

Évora → Portalegre → Castelo Branco → Coimbra → Aveiro → Porto → Viana do Castelo **Custo:** 570

Évora a Vila Real: na iteração 10

Évora → Portalegre → Castelo Branco → Guarda → Vila Real **Custo:** 430

Évora a Viseu: na iteração 10

Évora → Portalegre → Castelo Branco → Guarda → Viseu **Custo:** 360

b.

Para a resolução do exercício foi usado um código em Java encontrado no *osettacode.org*[1], para que o programa consiga fazer o algoritmo de Dijkstra é necessário fazer a conversão do mapa do enunciado para uma matriz de pesos, ficando da seguinte forma:

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 80 & 0 & 0 & 0 & 0 & 0 & 0 & 70 & 0 & 0 & 0 & 0 & 100 \\
0 & 0 & 0 & 0 & 0 & 0 & 80 & 170 & 0 & 0 & 0 & 0 & 0 & 0 & 135 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 & 0 & 0 & 50 & 100 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 200 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 140 & 0 \\
0 & 0 & 0 & 0 & 0 & 160 & 0 & 0 & 100 & 0 & 0 & 80 & 0 & 200 & 0 & 0 & 0 & 0 \\
80 & 0 & 0 & 0 & 160 & 0 & 0 & 0 & 160 & 70 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 80 \\
0 & 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 150 & 100 & 0 & 120 & 0 & 0 & 0 & 0 \\
0 & 170 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 260 & 0 & 0 & 0 \\
0 & 0 & 0 & 200 & 100 & 160 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 150 & 80 \\
0 & 0 & 0 & 0 & 0 & 70 & 0 & 0 & 0 & 0 & 130 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 150 & 0 & 0 & 130 & 0 & 0 & 0 & 70 & 50 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 80 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 150 & 0 & 0 & 0 & 0 \\
70 & 0 & 50 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 80 & 120 & 0 \\
0 & 0 & 0 & 0 & 200 & 0 & 120 & 0 & 0 & 0 & 70 & 150 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 135 & 0 & 0 & 0 & 0 & 0 & 260 & 0 & 0 & 50 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 50 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 80 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 100 & 140 & 0 & 0 & 0 & 0 & 150 & 0 & 0 & 0 & 120 & 0 & 0 & 0 & 0 & 110 \\
100 & 0 & 0 & 0 & 0 & 80 & 0 & 0 & 80 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 110 & 0
\end{bmatrix}$$

Após saber a matriz de pesos, foi usado o código em Java para calcular o caminho mais curto entre todos os vértices. Para o código funcionar como pedido no exercício, foi preciso primeiro converter a matriz de pesos para código, e depois fazer algumas alterações para ele iterar por todos os vértices e mostrar o caminho de menor custo de uma maneira mais organizada. O resultado obtido foi o seguinte:

Évora a Aveiro:

Évora → Portalegre → Castelo Branco → Coimbra → Aveiro **Custo:** 420

Évora a Beja:

Évora → Beja **Custo:** 80

Évora a Braga:

Évora → Portalegre → Castelo Branco → Guarda → Vila Real → Braga

Custo: 530

Évora a Bragança:

Évora → Portalegre → Castelo Branco → Guarda → Bragança **Custo:** 480

Évora a Castelo Branco:

Évora → Portalegre → Castelo Branco **Custo:** 180

Évora a Coimbra:

Évora → Portalegre → Castelo Branco → Coimbra **Custo:** 340

Évora a faro:

Évora → Beja → Faro **Custo:** 250

Évora a Guarda:

Évora → Portalegre → Castelo Branco → Guarda **Custo:** 280

Évora a Leiria:

Évora → Lisboa → Leiria **Custo:** 280

Évora a Lisboa:

Évora → Lisboa **Custo:** 150

Évora a Portalegre:

Évora → Portalegre **Custo:** 100

Évora a Porto:

Évora → Portalegre→ Castelo Branco → Coimbra → Aveiro → Porto

Custo: 490

Évora a Santarém:

Évora → Santarém **Custo:** 120

Évora a Setúbal:

Évora → Lisboa → Setúbal **Custo:** 200

Évora a Viana do Castelo:

Évora → Portalegre → Castelo Branco → Coimbra → Aveiro → Porto → Viana do Castelo **Custo:** 570

Évora a Vila Real:

Évora → Portalegre → Castelo Branco → Guarda → Vila Real **Custo:** 430

Évora a Viseu:

Évora → Portalegre → Castelo Branco → Guarda → Viseu **Custo: 360**

```
Caminho de Evora a Aveiro:
Evora -> Portalegre(100) -> Castelo Branco(180) -> Coimbra(340) -> Aveiro(420)

Caminho de Evora a Beja:
Evora -> Beja(80)

Caminho de Evora a Coimbra:
Evora -> Portalegre(100) -> Castelo Branco(180) -> Coimbra(340)

Caminho de Evora a Santarem:
Evora -> Santarem(120)

Caminho de Evora a Braganca:
Evora -> Portalegre(100) -> Castelo Branco(180) -> Guarda(280) -> Braganca(480)

Caminho de Evora a Lisboa:
Evora -> Lisboa(150)

Caminho de Evora a Castelo Branco:
Evora -> Portalegre(100) -> Castelo Branco(180)

Caminho de Evora a Braga:
Evora -> Portalegre(100) -> Castelo Branco(180) -> Guarda(280) -> Vila Real(430) -> Braga(530)

Caminho de Evora a Viseu:
Evora -> Portalegre(100) -> Castelo Branco(180) -> Guarda(280) -> Viseu(360)

Caminho de Evora a Leiria:
Evora -> Lisboa(150) -> Leiria(280)

Caminho de Evora a Setubal:
Evora -> Lisboa(150) -> Setubal(200)

Caminho de Evora a Aveiro:
Evora -> Portalegre(100) -> Castelo Branco(180) -> Coimbra(340) -> Aveiro(420)
```

Figura 3.1: Fragmento do Output do algoritmo de Dijkstra

```
private static final int TAMANHO_GRAFO = 18;
public static final String START = "Evora";
private static final String[] ARRAY_CAPITAIS = { "Aveiro", "Beja", "Braga", "Braganca",

Run | Debug
public static void main(String[] args) {
    Graph g = new Graph(GRAPH);

    System.out.println("\nCaminho de " + Dijkstra.START + " a " + "Aveiro" + ":");
    g.dijkstra(START);
    g.printPath(endName:"Aveiro"); // primeira iteração
    g.printAllPaths(); //restantes iterações

}
```

Figura 3.2: Função main do algoritmo de Dijkstra

c.

Para a resolução deste exercício é pedido a realização do algoritmo de Kruskal. O algoritmo de Kruskal é um algoritmo na teoria dos grafos que procura árvore geradora de custo mínimo do grafo, que é uma árvore constituída por todos os vértices desse grafo e pelas arestas para as quais se obtém um peso total[1] menor.

[1] soma dos pesos dessas arestas

Tabela 3.2: Tabela Algoritmo de Kruskal com os nomes das cidades

It	(u,v,c)	u	v	T	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19
1	Viana do Castelo, Braga	16	2	Viana do Castelo , 16, Braga , 2	Aveiro	Beja	Braga	Bragança	Castelo Branco	Coimbra	Évora	Faro	Guarda	Leiria	Lisboa	Portalegre	Porto	Santarém	Setúbal	Viana do Castelo, Braga	Vila Real	Viseu	
2	Braga, Porto	61	13	Braga , 3, Porto , 13	Aveiro	Beja	Bragança	Castelo Branco	Coimbra	Évora	Faro	Guarda	Leiria	Lisboa	Portalegre	Santarém	Setúbal	Viana do Castelo, Braga , Porto	Vila Real	Viseu			
3	Lisboa, Setúbal	32	15	Lisboa , 15, Setúbal , 15	Aveiro	Beja	Bragança	Castelo Branco	Coimbra	Évora	Faro	Guarda	Leiria	Lisboa, Setúbal	Portalegre	Santarém	Viana do Castelo, Braga , Porto	Vila Real	Viseu				
4	Porto, Aveiro	13	1	Porto , 13, Aveiro , 1	Beja	Bragança	Castelo Branco	Coimbra	Évora	Faro	Guarda	Leiria	Lisboa, Setúbal	Portalegre	Santarém	Viana do Castelo , Braga , Porto , Aveiro	Vila Real	Viseu					
5	Coimbra, Leiria	9	8	Coimbra , 9, Leiria , 10	Beja	Bragança	Castelo Branco	Coimbra, Leiria	Évora	Faro	Guarda	Lisboa, Setúbal	Portalegre	Santarém	Viana do Castelo , Braga , Porto , Aveiro ,	Vila Real	Viseu						
6	Lisboa, Santarém	8	16	Lisboa , 15, Santarém , 14	Beja	Bragança	Castelo Branco	Coimbra, Leiria	Évora	Faro	Guarda	Lisboa, Setúbal, Santarém	Portalegre	Viana do Castelo, Braga , Porto, Aveiro	Vila Real	Viseu							
7	Viana do Castelo, Porto	16	13	Viana do Castelo , 16, Porto , 13																			
8	Aveiro, Coimbra	30	9	Aveiro , 1, Coimbra , 6	Beja	Bragança	Castelo Branco	Évora	Faro	Guarda	Lisboa, Setúbal, Santarém	Portalegre	Viana do Castelo, Braga , Porto, Aveiro, Coimbra, Leiria ,	Vila Real	Viseu								
9	Coimbra, Viseu	9	31	Coimbra , 6, Viseu , 18	Beja	Bragança	Castelo Branco	Évora	Faro	Guarda	Lisboa, Setúbal, Santarém	Portalegre	Viana do Castelo, Braga , Porto, Aveiro, Coimbra, Leiria,Viseu	Vila Real									
10	Guarda, Viseu	5	9	Guarda , 9, Viseu , 16	Beja	Bragança	Castelo Branco	Évora	Faro	Guarda, Viseu, de Castelo, Braga , Porto, Aveiro, Coimbra, Leiria,Viseu	Lisboa, Setúbal, Santarém	Portalegre	Vila Real										
11	Castelo Branco, Portalegre	5	8	Castelo Branco , 5, Portalegre , 12	Beja	Bragança	Castelo Branco, Portalegre	Évora	Faro	Guarda, Viseu, de Castelo, Braga , Porto, Aveiro, Coimbra, Leiria,Viseu	Lisboa, Setúbal, Santarém	Vila Real											
12	Beja, Évora	7	4	Beja , 3, Évora , 7	Beja, Évora	Bragança	Castelo Branco, Portalegre	Faro	Guarda, Viseu, de Castelo, Braga , Porto, Aveiro, Coimbra, Leiria,Viseu	Lisboa, Setúbal, Santarém	Vila Real												
13	Braga, Vila Real	3	7	Braga , 3, Vila Real , 17	Beja, Évora	Bragança	Castelo Branco, Portalegre	Faro	Guarda, Viseu, de Castelo, Braga , Porto, Aveiro, Coimbra, Leiria, Viseu, Vila Real	Lisboa, Setúbal, Santarém													
14	Aveiro, Viseu	1	5	Aveiro , 5, Viseu , 5																			
15	Castelo Branco, Guarda	5	9	Castelo Branco , 5, Guarda , 9	Beja, Évora	Bragança	Castelo Branco, Portalegre, Guarda, Viseu, de Castelo, Braga , Porto, Aveiro, Coimbra, Leiria, Viseu, Vila Real	Faro	Lisboa, Setúbal, Santarém														
16	Évora, Portalegre	7	5	Évora , 7, Portalegre , 12	Beja, Évora, Castelo Branco, Faro, Portalegre, Guarda, Viseu de Castelo, Braga , Porto, Aveiro, Coimbra, Leiria, Viseu, Vila Real	Bragança	Faro	Lisboa, Setúbal, Santarém															
17	Vila Real, Viseu	1	1	Vila Real , 17, Viseu , 18																			
18	Porto, Vila Real	1	1	Porto , 13, Vila Real , 17																			
19	Évora, Santarém	7	3	Évora , 7, Santarém , 14	Beja, Évora, Castelo Branco, Faro, Portalegre, Guarda, Viseu de Castelo, Braga , Porto, Aveiro, Coimbra, Leiria, Viseu, Vila Real,Lisboa, Setúbal, Santarém	Bragança	Faro																
20	Leiria, Lisboa	1	1	Leiria , 10, Lisboa , 15																			
21	Setúbal, Beja	1	1	Setúbal , 15, Beja , 3																			
22	Vila Real, Bragança	1	2	Vila Real , 17, Bragança , 4	Beja, Évora, Castelo Branco, Faro, Portalegre, Guarda, Viseu de Castelo, Braga , Porto, Aveiro, Coimbra, Leiria, Viseu, Vila Real,Lisboa, Setúbal, Santarém, Bragança	Faro																	
23	Vila Real, Guarda	1	1	Vila Real , 17, Guarda , 9																			
24	Santarém, Portalegre	1	1	Santarém , 14, Portalegre , 12																			
25	Lisboa, Évora	1	1	Lisboa , 15, Évora , 7																			
26	Coimbra, Guarda	1	1	Coimbra , 6, Guarda , 9																			
27	Coimbra, Castelo Branco	1	1	Coimbra , 6, Castelo Branco , 5																			
28	Beja, Bragança	1	2	Beja , 3, Faro , 8	Beja, Évora, Castelo Branco, Faro, Portalegre, Guarda, Viseu de Castelo, Braga , Porto, Aveiro, Coimbra, Leiria, Viseu, Vila Real,Lisboa, Setúbal, Santarém, Bragança,Faro																		

Tabela 3.3: Tabela Algoritmo de Kruskal com os números das cidades

It	(u,v,c)	u	v	T	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19
1	16, 2	16	2	16 - 16, 2 - 2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16, 2	17	18	19	1
2	2, 13	16	13	2 - 15, 13 - 13	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16, 2, 13	17	18		2
3	13, 15	16	15	13 - 16, 15 - 15	1	2	3	4	5	6	7	8	9	10	11, 15	12	13	14	15, 3, 15	17	18			3
4	15, 1	13	1	13 - 13, 1 - 1	2	1	3	4	5	6	7	8	9	10	11, 15	12	13	14	15, 3, 15, 1	17	18			4
5	6, 10	2	5	6 - 1, 10 - 8	2	1	3	4	5	6, 10	7	8	9	10	11, 15	12	13	14	15, 3, 15, 1	17	18			5
6	11, 14	8	10	11 - 8, 14 - 10	2	1	3	4	5	6, 10	7	8	9	10	11, 15, 14	12	13	14	15, 3, 15, 1	17	18			6
7	10, 12	10	12	10 - 10, 12 - 12																			
8	1, 6	10	4	1 - 10, 6 - 4	2	1	3	7	4	8	11, 15, 14	12	16, 3, 12, 1, 6, 10	17	18									7
9	6, 10	9	11	6 - 9, 10 - 11	2	1	3	7	4	8	11, 15, 14	12	16, 3, 12, 1, 6, 10,9	17										8
10	9, 10	8	9	9 - 8, 10 - 9	2	1	3	7	4	8, 16, 3, 12, 1, 6, 10,9,10	11, 15, 14	12	17											9
11	3, 12	3	6	3 - 3, 12 - 8	2	1	3, 12	7	4	8, 16, 3, 12, 1, 6, 10,9	11, 15, 14	17												10
12	2, 7	1	1	2 - 1, 7 - 4	2, 7	1	3, 12	4	5, 16, 3, 12, 1, 6, 10,10	11, 15, 14	17													11
13	3, 17	3	7	3 - 3, 17 - 7	2, 7	1	3, 12	5	5, 16, 3, 12, 1, 6, 10, 10, 17	11, 15, 12														12
14	1, 16	1	3	1 - 1, 16 - 5																				
15	5, 9	3	5	5 - 3, 9 - 5	2, 7	1	3, 12, 9, 16, 3, 12, 1, 6, 10, 16, 17	8	11, 15, 14															13
16	7, 12	1	8	7 - 1, 12 - 3	2, 7, 5, 12, 9, 16, 3, 12, 1, 6, 10, 16, 17,	4	11, 15, 14																	14
17	17, 16	4	1	17 - 4, 16 - 1																				
18	15, 17	1	4	15 - 1, 17 - 4																				
19	7, 14	1	2	7 - 1, 14 - 4	2, 7, 5, 12, 9, 16, 3, 12, 1, 6, 10, 16, 17,21, 15, 14,	4	8																	15
20	10, 11	2	4	10 - 1, 11 - 1																				
21	15, 2	4	1	15 - 1, 2 - 1																				
22	17, 4	4	2	17 - 1, 4 - 2	2, 7, 5, 12, 9, 16, 3, 12, 1, 6, 10, 16, 17,21, 15, 14, 4	8																		16
23	17, 9	4	1	17 - 1, 9 - 1																				
24	14, 12	4	3	14 - 1, 12 - 3																				
25	11, 7	2	4	11 - 1, 7 - 1																				
26	6, 9	2	4	6 - 1, 9 - 1																				
27	6, 5	2	3	6 - 1, 5 - 1																				
28	2, 4	1	2	2 - 1, 4 - 2	2, 7, 5, 12, 9, 16, 3, 12, 1, 6, 10, 16, 17,21, 15, 14, 4,8																			17

Custo Mínimo = 50 + 50 + 50 + 70 + 70 + 70 + 80 + 80 + 80 + 80 + 80 + 100 + 100 + 100 + 120 + 140 + 170 = 1490;

Árvore = {(16, 3) , (3, 13), (11, 15), (13, 1), (6, 10), (11, 14), (1, 6), (6, 18), (9, 18), (5, 12), (2, 7), (3, 17), (5, 9), (7, 12), (7, 14), (17, 4), (2, 8)}

ou seja:

Árvore = {(Viana do Castelo, Braga) , (Braga, Porto), (Lisboa, Setúbal), (Porto, Aveiro), (Coimbra, Leiria), (Lisboa, Santarém), (Aveiro, Coimbra), (Coimbra, Viseu), (Guarda, Viseu), (Castelo Branco, Portalegre), (Beja, Évora), (Braga, Vila Real), (Castelo Branco, Guarda), (Évora, Portalegre), (Évora, Santarém), (Vila Real, Bragança), (Beja, Faro)}

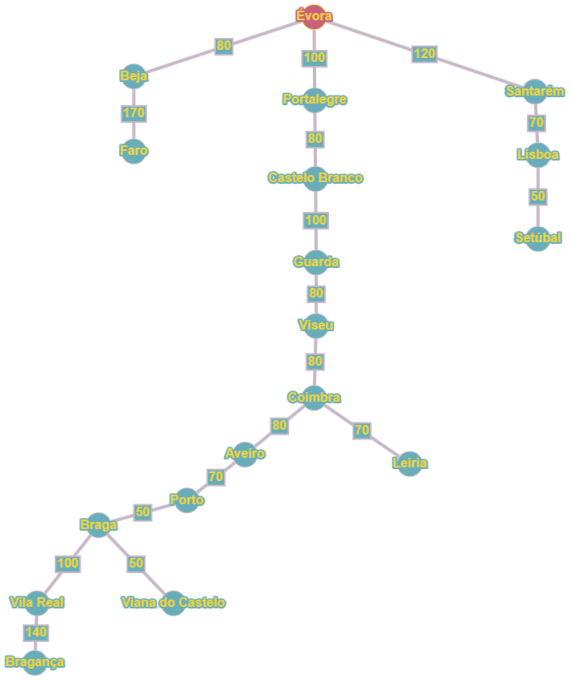

Figura 3.3: Árvore de Custo Mínimo

d.

Com a matriz de pesos calculada no exercício 3b, foi usado um código em python encontrado no *www.programiz.com*[4], para calcular a árvore geradora de custo mínimo do grafo. Para o código funcionar como foi pedido no exercício, foi preciso primeiro converter a matriz de pesos para código, e depois fez-se uma função que converte os números do resultado para a respetiva capital do distrito. O resultado obtido foi o seguinte:

```
Braga - Porto: 50
Braga - Viana do Castelo: 50
Lisboa - Setubal: 50
Aveiro - Porto: 70
Coimbra - Leiria: 70
Lisboa - Santarem: 70
Aveiro - Coimbra: 80
Beja - Evora: 80
Castelo Branco - Portalegre: 80
Coimbra - Viseu: 80
Guarda - Viseu: 80
Braga - Vila Real: 100
Castelo Branco - Guarda: 100
Evora - Portalegre: 100
Evora - Santarem: 120
Braganca - Vila Real: 140
Beja - Faro: 170
```

Figura 3.4: Output algoritmo de Kruskal

```python
# Troca os numeros pelos nomes das regioes
def switchRegioes(i):
    match i:
        case 0:
            return "Aveiro"
        case 1:
            return "Beja"
        case 2:
            return "Braga"
        case 3:
            return "Braganca"
        case 4:
            return "Castelo Branco"
        case 5:
            return "Coimbra"
        case 6:
            return "Evora"
        case 7:
            return "Faro"
        case 8:
            return "Guarda"
        case 9:
            return "Leiria"
        case 10:
            return "Lisboa"
        case 11:
            return "Portalegre"
        case 12:
            return "Porto"
        case 13:
            return "Santarem"
        case 14:
            return "Setubal"
        case 15:
            return "Viana do Castelo"
        case 16:
            return "Vila Real"
        case 17:
            return "Viseu"
        case _:
            return "Erro"
```

Figura 3.5: Função que troca os números pelas respetivas capitais

Pergunta 4

a.

Neste exercício foi pedido para que se fizesse uma pesquisa sobre um dos algoritmos que tínhamos à escolha, e o escolhido foi o algoritmo de Floyd-Warshall.

O algoritmo de Floyd-Warshall serve para calcular o caminho mais curto entre todos os pares de vértices num grafo orientado, ao comparar todos os caminhos possíveis através do gráfico entre cada par de vértices. Ele é bastante útil, pois e ele consegue encontrar p caminho mais curto entre todos os pares de vértices do grafo independentemente da presença de arestas negativos. Isso significa que ele é aplicável a uma ampla variedade de problemas. No entanto, embora o algoritmo de Floyd-Warshall consiga trabalhar com arestas negativos, ele não consegue trabalhar com ciclos negativos[1], e é mais lento que outros algoritmos que foram desenvolvidos para fazer a mesma tarefa, isso deve-se por ter uma complexidade de tempo $O(V^3)$ na notação *Big O*[2]

Na demonstração da utilização do algoritmo de Floyd-Warshall usaremos o seguinte grafo e a seguinte matriz de pesos desenhado no *graphonline.ru*[3]:

[1] soma de todos os vértices ser negativa

[2] notação usada na análise de algoritmos e complexidade computacional.

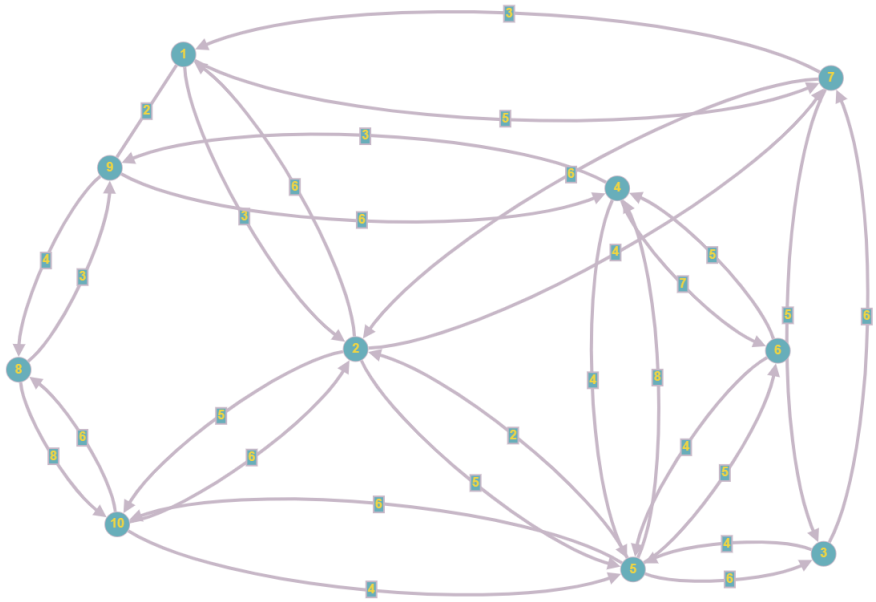

Figura 4.1: Grafo utilizado no algoritmo de Floyd-Warshall

$$\begin{bmatrix}
0 & 3 & 0 & 0 & 0 & 0 & 5 & 0 & 2 & 0 \\
6 & 0 & 0 & 0 & 5 & 0 & 4 & 0 & 0 & 5 \\
0 & 0 & 0 & 0 & 4 & 0 & 6 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 7 & 0 & 0 & 3 & 0 \\
0 & 2 & 6 & 8 & 0 & 5 & 0 & 0 & 0 & 6 \\
0 & 0 & 0 & 5 & 4 & 0 & 0 & 0 & 0 & 0 \\
3 & 6 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 8 \\
2 & 0 & 0 & 6 & 0 & 0 & 0 & 4 & 0 & 0 \\
0 & 6 & 0 & 0 & 4 & 0 & 0 & 6 & 0 & 0
\end{bmatrix}$$

Antes de começar a realizar o algoritmo, primeiro é preciso alterar a matriz de pesos, pois, tanto os vértices que não têm nenhum caminho, como os vértices que têm caminhos indiretos que os unem estão representados na matriz com o algarismo 0, então efetuar-se-à uma troca desses valores por ∞. Assim, os únicos caminhos que têm um 0, são aqueles em que não existem nenhum caminho que os une. Ficando assim uma matriz de distâncias.

$$A^0 = \begin{bmatrix} 0 & 3 & \infty & \infty & \infty & \infty & 5 & \infty & 2 & \infty \\ 6 & 0 & \infty & \infty & 5 & \infty & 4 & \infty & \infty & 5 \\ \infty & \infty & 0 & \infty & 4 & \infty & 6 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 4 & 7 & \infty & \infty & 3 & \infty \\ \infty & 2 & 6 & 8 & 0 & 5 & \infty & \infty & \infty & 6 \\ \infty & \infty & \infty & 5 & 4 & 0 & \infty & \infty & \infty & \infty \\ 3 & 6 & 5 & \infty & \infty & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 3 & 8 \\ 2 & \infty & \infty & 6 & \infty & \infty & \infty & 4 & 0 & \infty \\ \infty & 6 & \infty & \infty & 4 & \infty & \infty & 6 & \infty & 0 \end{bmatrix}$$

Para começar o algoritmo de Floyd, será necessário criar uma matriz A^1 com os valores da matriz A^0, onde apenas sobrarão os elementos da primeira linha, primeira coluna e diagonal principal. Depois será necessário preencher os espaços em branco da matriz, para isso, vai se somar cada elemento da coluna com cada elemento da linha que se separou, e fazer-se-à uma comparação dessa soma com o valor que está na posição da i, j dos valores que se somou, assim vai-se verificar se a distância entre um par de vértices é menor ao passar por um vértice intermediário.

Por exemplo, ao somar $A[2][1]$ com $A[1][2]$, será feita uma comparação dessa soma com a posição $A[2][2]$. Se o resultado da soma for **maior** ou **igual** ao valor contido em $A[2][2]$, então o valor que estava nessa posição na matriz $A0$ será mantido. Caso seja **menor**, o valor da posição $A[2][2]$ será substituído pelo valor da soma.

1ª Iteração:

$$A^1 = \begin{bmatrix} 0 & 3 & \infty & \infty & \infty & \infty & 5 & \infty & 2 & \infty \\ 6 & 0 & & & & & & & & \\ \infty & & 0 & & & & & & & \\ \infty & & & 0 & & & & & & \\ \infty & & & & 0 & & & & & \\ \infty & & & & & 0 & & & & \\ 3 & & & & & & 0 & & & \\ \infty & & & & & & & 0 & & \\ 2 & & & & & & & & 0 & \\ \infty & & & & & & & & & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 3 & \infty & \infty & \infty & \infty & 5 & \infty & 2 & \infty \\ 6 & 0 & \infty & \infty & 5 & \infty & 4 & \infty & 8 & 5 \\ \infty & \infty & 0 & \infty & 4 & \infty & 6 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 4 & 7 & \infty & \infty & 3 & \infty \\ \infty & 2 & 6 & 8 & 0 & 5 & \infty & \infty & \infty & 6 \\ \infty & \infty & \infty & 5 & 4 & 0 & \infty & \infty & \infty & \infty \\ 3 & 6 & 5 & \infty & \infty & \infty & 0 & \infty & 5 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 3 & 8 \\ 2 & 5 & \infty & 6 & \infty & \infty & 7 & 4 & 0 & \infty \\ \infty & 6 & \infty & \infty & 4 & \infty & \infty & 6 & \infty & 0 \end{bmatrix}$$

$(A[2][1] + A[1][2]) > A[2][2]$ $\qquad$ $(A[2][1] + A[1][3]) = A[2][3]$

$(A[2][1] + A[1][4]) = A[2][4]$ $\qquad$ $(A[2][1] + A[1][5]) > A[2][5]$

$(A[2][1] + A[1][6]) = A[2][6]$ $\qquad$ $(A[2][1] + A[1][7]) > A[2][7]$

$(A[2][1] + A[1][8]) = A[2][2]$ $\qquad$ $(A[2][1] + A[1][9]) < A[2][9]$

$(A[2][1] + A[1][10]) > A[2][10]$ $\qquad$ $(...)$

2ª Iteração:

$$A^2 = \begin{bmatrix} 0 & 3 & & & & & & & & \\ 6 & 0 & \infty & \infty & 5 & \infty & 4 & \infty & 8 & 5 \\ & \infty & 0 & & & & & & & \\ & \infty & & 0 & & & & & & \\ & 2 & & & 0 & & & & & \\ & \infty & & & & 0 & & & & \\ & 6 & & & & & 0 & & & \\ & \infty & & & & & & 0 & & \\ & 5 & & & & & & & 0 & \\ & 6 & & & & & & & & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 3 & \infty & \infty & 8 & \infty & 5 & \infty & 2 & 8 \\ 6 & 0 & \infty & \infty & 5 & \infty & 4 & \infty & 8 & 5 \\ \infty & \infty & 0 & \infty & 4 & \infty & 6 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 4 & 7 & \infty & \infty & 3 & \infty \\ 8 & 2 & 6 & 8 & 0 & 5 & 6 & \infty & 10 & 6 \\ \infty & \infty & \infty & 5 & 4 & 0 & \infty & \infty & \infty & \infty \\ 3 & 6 & 5 & \infty & 11 & \infty & 0 & \infty & 5 & 11 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 3 & 8 \\ 2 & 5 & \infty & 6 & 10 & \infty & 7 & 4 & 0 & 10 \\ 12 & 6 & \infty & \infty & 4 & \infty & 10 & 6 & 14 & 0 \end{bmatrix}$$

$(A[3][2] + A[2][1]) = A[3][1]$ $\qquad$ $(A[1][2] + A[2][3]) = A[1][3]$

$(A[3][2] + A[3][1]) > A[1][2]$ $\qquad$ $(...)$

3ª Iteração:

$$A^3 = \begin{bmatrix} 0 & & \infty & & & & & & & \\ & 0 & \infty & & & & & & & \\ \infty & \infty & 0 & \infty & 4 & \infty & 6 & \infty & \infty & \infty \\ & & \infty & 0 & & & & & & \\ & & 6 & & 0 & & & & & \\ & & \infty & & & 0 & & & & \\ & & 5 & & & & 0 & & & \\ & & \infty & & & & & 0 & & \\ & & \infty & & & & & & 0 & \\ & & \infty & & & & & & & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 3 & \infty & \infty & 8 & \infty & 5 & \infty & 2 & 8 \\ 6 & 0 & \infty & \infty & 5 & \infty & 4 & \infty & 8 & 5 \\ \infty & \infty & 0 & \infty & 4 & \infty & 6 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 4 & 7 & \infty & \infty & 3 & \infty \\ 8 & 2 & 6 & 8 & 0 & 5 & 6 & \infty & 10 & 6 \\ \infty & \infty & \infty & 5 & 4 & 0 & \infty & \infty & \infty & \infty \\ 3 & 6 & 5 & \infty & 9 & \infty & 0 & \infty & 5 & 11 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 3 & 8 \\ 2 & 5 & \infty & 6 & 10 & \infty & 7 & 4 & 0 & 10 \\ 12 & 6 & \infty & \infty & 4 & \infty & 10 & 6 & 14 & 0 \end{bmatrix}$$

$(A[1][3] + A[3][1]) > A[1][1]$ $\qquad$ $(A[3][2] + A[3][1]) > A[1][2]$

$(A[3][1] + A[2][3]) > A[2][1]$ $\qquad$ $(...)$

4ª Iteração:

$$A^4 = \begin{bmatrix} 0 & & & \infty & & & & & & \\ & 0 & & \infty & & & & & & \\ & & 0 & \infty & & & & & & \\ \infty & \infty & \infty & 0 & 4 & 7 & \infty & \infty & 3 & \infty \\ & & & 8 & 0 & & & & & \\ & & & 5 & & 0 & & & & \\ & & & \infty & & & 0 & & & \\ & & & \infty & & & & 0 & & \\ & & & 6 & & & & & 0 & \\ & & & \infty & & & & & & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 3 & \infty & \infty & 8 & \infty & 5 & \infty & 2 & 8 \\ 6 & 0 & \infty & \infty & 5 & \infty & 4 & \infty & 8 & 5 \\ \infty & \infty & 0 & \infty & 4 & \infty & 6 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 4 & 7 & \infty & \infty & 3 & \infty \\ 8 & 2 & 6 & 8 & 0 & 5 & 6 & \infty & 10 & 6 \\ \infty & \infty & \infty & 5 & 4 & 0 & \infty & \infty & 8 & \infty \\ 3 & 6 & 5 & \infty & 9 & \infty & 0 & \infty & 5 & 11 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 3 & 8 \\ 2 & 5 & \infty & 6 & 10 & 13 & 7 & 4 & 0 & 10 \\ 12 & 6 & \infty & \infty & 4 & \infty & 10 & 6 & 14 & 0 \end{bmatrix}$$

5ª Iteração:

$$A^5 = \begin{bmatrix} 0 & & & & 8 & & & & & \\ & 0 & & & 5 & & & & & \\ & & 0 & & 4 & & & & & \\ & & & 0 & 4 & & & & & \\ 8 & 2 & 6 & 8 & 0 & 5 & 6 & \infty & 10 & 6 \\ & & & & 4 & 0 & & & & \\ & & & & 9 & & 0 & & & \\ & & & & \infty & & & 0 & & \\ & & & & 10 & & & & 0 & \\ & & & & 4 & & & & & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 3 & 14 & 16 & 8 & 13 & 5 & \infty & 2 & 8 \\ 6 & 0 & 11 & 13 & 5 & 10 & 4 & \infty & 8 & 5 \\ 12 & 6 & 0 & 12 & 4 & 9 & 6 & \infty & 14 & 10 \\ 12 & 6 & 10 & 0 & 4 & 7 & 10 & \infty & 3 & 10 \\ 8 & 2 & 6 & 8 & 0 & 5 & 6 & \infty & 10 & 6 \\ 12 & 6 & 10 & 5 & 4 & 0 & 10 & \infty & 8 & 10 \\ 3 & 6 & 5 & 17 & 9 & 14 & 0 & \infty & 5 & 11 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 3 & 8 \\ 2 & 5 & 16 & 6 & 10 & 13 & 7 & 4 & 0 & 10 \\ 12 & 6 & 10 & 12 & 4 & 9 & 10 & 6 & 14 & 0 \end{bmatrix}$$

6ª Iteração:

$$A^6 = \begin{bmatrix} 0 & & & & & & 13 & & & \\ & 0 & & & & & 10 & & & \\ & & 0 & & & & 9 & & & \\ & & & 0 & & & 7 & & & \\ & & & & 0 & & 5 & & & \\ 12 & 6 & 10 & 5 & 4 & 0 & 10 & \infty & 8 & 10 \\ & & & & & 14 & 0 & & & \\ & & & & & \infty & & 0 & & \\ & & & & & 13 & & & 0 & \\ & & & & & 9 & & & & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 3 & 14 & 16 & 8 & 13 & 5 & \infty & 2 & 8 \\ 6 & 0 & 11 & 13 & 5 & 10 & 4 & \infty & 8 & 5 \\ 12 & 6 & 0 & 12 & 4 & 9 & 6 & \infty & 14 & 10 \\ 12 & 6 & 10 & 0 & 4 & 7 & 10 & \infty & 3 & 10 \\ 8 & 2 & 6 & 8 & 0 & 5 & 6 & \infty & 10 & 6 \\ 12 & 6 & 10 & 5 & 4 & 0 & 10 & \infty & 8 & 10 \\ 3 & 6 & 5 & 17 & 9 & 14 & 0 & \infty & 5 & 11 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 3 & 8 \\ 2 & 5 & 16 & 6 & 10 & 13 & 7 & 4 & 0 & 10 \\ 12 & 6 & 10 & 12 & 4 & 9 & 10 & 6 & 14 & 0 \end{bmatrix}$$

7ª Iteração:

$$A^7 = \begin{bmatrix} 0 & & & & & & & 5 & & \\ & 0 & & & & & & 4 & & \\ & & 0 & & & & & 6 & & \\ & & & 0 & & & & 10 & & \\ & & & & 0 & & & 6 & & \\ & & & & & 0 & & 10 & & \\ 3 & 6 & 5 & 17 & 9 & 14 & 0 & \infty & 5 & 11 \\ & & & & & & \infty & 0 & & \\ & & & & & & 7 & & 0 & \\ & & & & & & 10 & & & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 3 & 10 & 16 & 8 & 13 & 5 & \infty & 2 & 8 \\ 6 & 0 & 9 & 13 & 5 & 10 & 4 & \infty & 8 & 5 \\ 9 & 6 & 0 & 12 & 4 & 9 & 6 & \infty & 11 & 10 \\ 12 & 6 & 10 & 0 & 4 & 7 & 10 & \infty & 3 & 10 \\ 8 & 2 & 6 & 8 & 0 & 5 & 6 & \infty & 10 & 6 \\ 12 & 6 & 10 & 5 & 4 & 0 & 10 & \infty & 8 & 10 \\ 3 & 6 & 5 & 17 & 9 & 14 & 0 & \infty & 5 & 11 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 3 & 8 \\ 2 & 5 & 16 & 6 & 10 & 13 & 7 & 4 & 0 & 10 \\ 12 & 6 & 10 & 12 & 4 & 9 & 10 & 6 & 14 & 0 \end{bmatrix}$$

8ª Iteração:

$$A^8 = \begin{bmatrix} 0 & & & & & & & \infty & & \\ & 0 & & & & & & \infty & & \\ & & 0 & & & & & \infty & & \\ & & & 0 & & & & \infty & & \\ & & & & 0 & & & \infty & & \\ & & & & & 0 & & \infty & & \\ & & & & & & 0 & \infty & & \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 3 & 8 \\ & & & & & & & 4 & 0 & \\ & & & & & & & 6 & & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 3 & 10 & 16 & 8 & 13 & 5 & \infty & 2 & 8 \\ 6 & 0 & 9 & 13 & 5 & 10 & 4 & \infty & 8 & 5 \\ 9 & 6 & 0 & 12 & 4 & 9 & 6 & \infty & 11 & 10 \\ 12 & 6 & 10 & 0 & 4 & 7 & 10 & \infty & 3 & 10 \\ 8 & 2 & 6 & 8 & 0 & 5 & 6 & \infty & 10 & 6 \\ 12 & 6 & 10 & 5 & 4 & 0 & 10 & \infty & 8 & 10 \\ 3 & 6 & 5 & 17 & 9 & 14 & 0 & \infty & 5 & 11 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 3 & 8 \\ 2 & 5 & 12 & 6 & 10 & 13 & 7 & 4 & 0 & 10 \\ 12 & 6 & 10 & 12 & 4 & 9 & 10 & 6 & 9 & 0 \end{bmatrix}$$

9ª Iteração:

$$A^9 = \begin{bmatrix} 0 & & & & & & & & 2 & \\ & 0 & & & & & & & 8 & \\ & & 0 & & & & & & 11 & \\ & & & 0 & & & & & 3 & \\ & & & & 0 & & & & 10 & \\ & & & & & 0 & & & 8 & \\ & & & & & & 0 & & 5 & \\ & & & & & & & 0 & 3 & \\ 2 & 5 & 12 & 6 & 10 & 13 & 7 & 4 & 0 & 10 \\ & & & & & & & & 9 & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 3 & 10 & 8 & 8 & 13 & 5 & 6 & 2 & 8 \\ 6 & 0 & 9 & 13 & 5 & 10 & 4 & 12 & 8 & 5 \\ 9 & 6 & 0 & 12 & 4 & 9 & 6 & 15 & 11 & 10 \\ 5 & 6 & 10 & 0 & 4 & 7 & 10 & 7 & 3 & 10 \\ 8 & 2 & 6 & 8 & 0 & 5 & 6 & 14 & 10 & 6 \\ 10 & 6 & 10 & 5 & 4 & 0 & 10 & 12 & 8 & 10 \\ 3 & 6 & 5 & 11 & 9 & 14 & 0 & 9 & 5 & 11 \\ 5 & 8 & 15 & 9 & 13 & 16 & 10 & 0 & 3 & 8 \\ 2 & 5 & 12 & 6 & 10 & 13 & 7 & 4 & 0 & 10 \\ 11 & 6 & 10 & 12 & 4 & 9 & 10 & 6 & 9 & 0 \end{bmatrix}$$

10ª Iteração:

$$A^{10} = \begin{bmatrix} 0 & & & & & & & & & 8 \\ & 0 & & & & & & & & 5 \\ & & 0 & & & & & & & 10 \\ & & & 0 & & & & & & 10 \\ & & & & 0 & & & & & 6 \\ & & & & & 0 & & & & 10 \\ & & & & & & 0 & & & 11 \\ & & & & & & & 0 & & 8 \\ & & & & & & & & 0 & 10 \\ 11 & 6 & 10 & 12 & 4 & 9 & 10 & 6 & 9 & 0 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & 3 & 10 & 8 & 8 & 13 & 5 & 6 & 2 & 8 \\ 6 & 0 & 9 & 13 & 5 & 10 & 4 & 11 & 8 & 5 \\ 9 & 6 & 0 & 12 & 4 & 9 & 6 & 15 & 11 & 10 \\ 5 & 6 & 10 & 0 & 4 & 7 & 10 & 7 & 3 & 10 \\ 8 & 2 & 6 & 8 & 0 & 5 & 6 & 12 & 10 & 6 \\ 10 & 6 & 10 & 5 & 4 & 0 & 10 & 12 & 8 & 10 \\ 3 & 6 & 5 & 11 & 9 & 14 & 0 & 9 & 5 & 11 \\ 5 & 8 & 15 & 9 & 12 & 16 & 10 & 0 & 3 & 8 \\ 2 & 5 & 12 & 6 & 10 & 13 & 7 & 4 & 0 & 10 \\ 11 & 6 & 10 & 12 & 4 & 9 & 10 & 6 & 9 & 0 \end{bmatrix}$$

Para validar se o resultado obtido estava correto, foi criado um código em C que fizesse o algoritmo de Floyd Warshall e que mostrasse a matriz com o resultado final. Esse código foi feito em cima do pseudocódigo do algoritmo que foi encontrado na Wikipédia[2].

```
  0   3  10   8   8  13   5   6   2   8
  6   0   9  13   5  10   4  11   8   5
  9   6   0  12   4   9   6  15  11  10
  5   6  10   0   4   7  10   7   3  10
  8   2   6   8   0   5   6  12  10   6
 10   6  10   5   4   0  10  12   8  10
  3   6   5  11   9  14   0   9   5  11
  5   8  15   9  12  16  10   0   3   8
  2   5  12   6  10  13   7   4   0  10
 11   6  10  12   4   9  10   6   9   0


--------------------------------
Process exited after 0.041 seconds with return value 0
Press any key to continue . . .
```

Figura 4.2: Output do resultado do algoritmo de Floyd Warshall em C

```c
// Algoritmo de Floyd-Warshall
void floydWarshall(int matriz[][nV]) {
    int i, j, k;

    for (k = 0; k < nV; k++) {
        for (i = 0; i < nV; i++) {
          for (j = 0; j < nV; j++) {
            if ( (matriz[i][k] + matriz[k][j]) < matriz[i][j] ) {
                matriz[i][j] = matriz[i][k] + matriz[k][j];
            }
            }
        }
    }

  printMatriz(matriz);
}
```

Figura 4.3: Função em C de que realiza o algoritmo de Floyd Warshall

Bibliografia

[1] Dijkstra's algorithm. https://rosettacode.org/wiki/Dijkstra%27s_algorithm#Java.

[2] Floyd–warshall algorithm. https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm.

[3] Graph online. https://graphonline.ru/en/.

[4] Kruskal's algorithm. https://www.programiz.com/dsa/kruskal-algorithm.

[5] Scilab. https://www.scilab.org.

[6] H. Martinez. algoritmo de floyd-warshall. https://www.youtube.com/watch?v=h-nmexY9gtA.