OBI2018

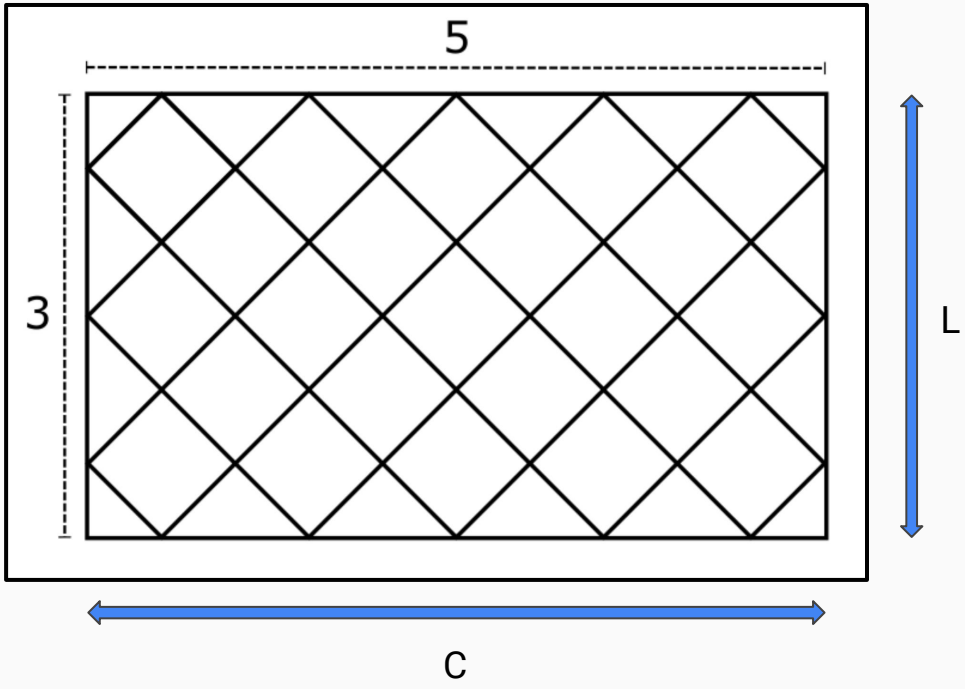
XX Olimpíada Brasileira de Informática

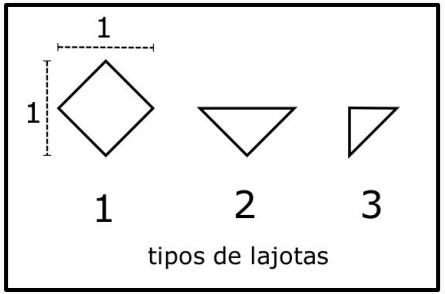
Piso da escola

- Fácil

- Entrada e saída
- Matemática

Solução

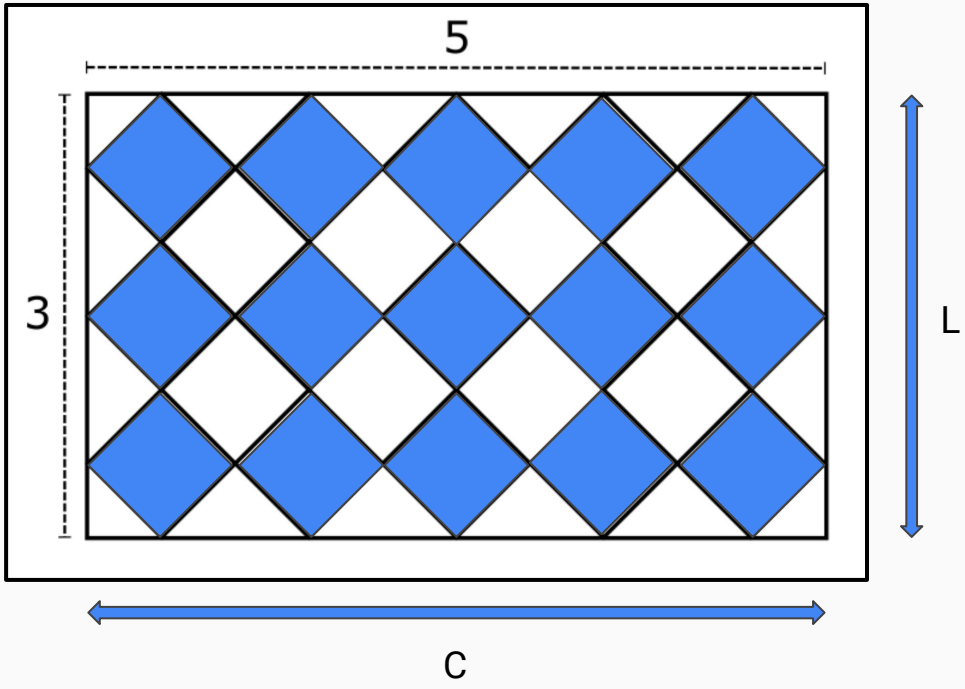


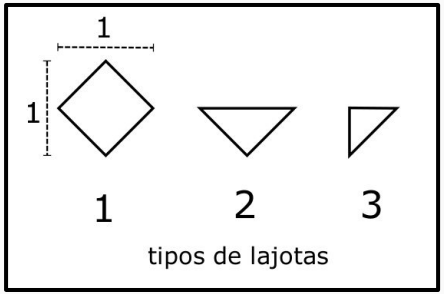


t1 = 

t2 = 

Solução

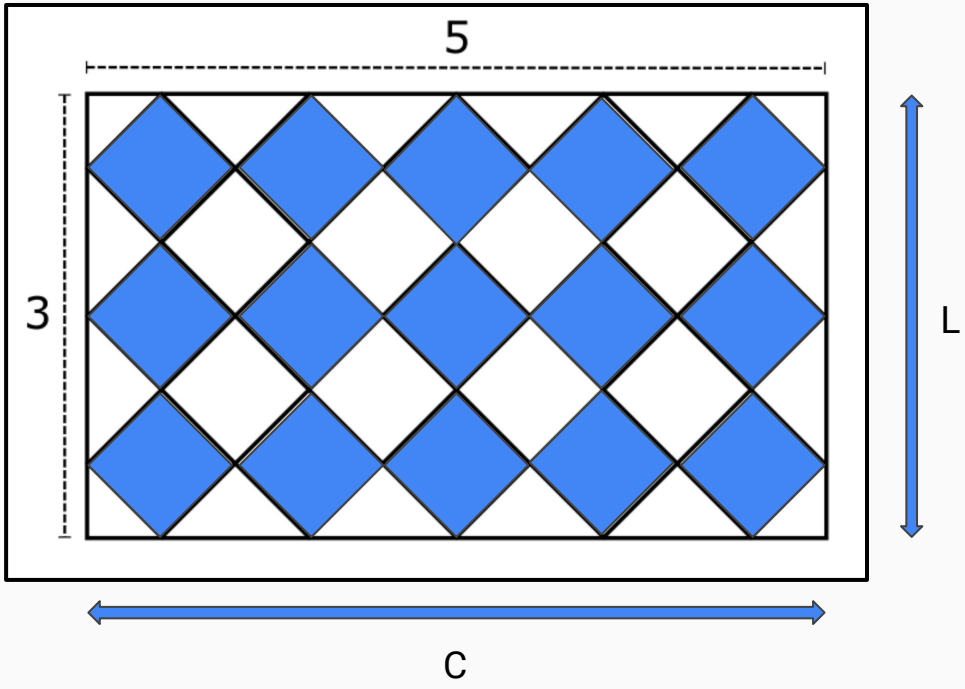


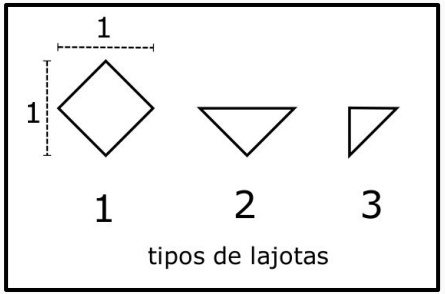


t1 =

t2 =

Solução

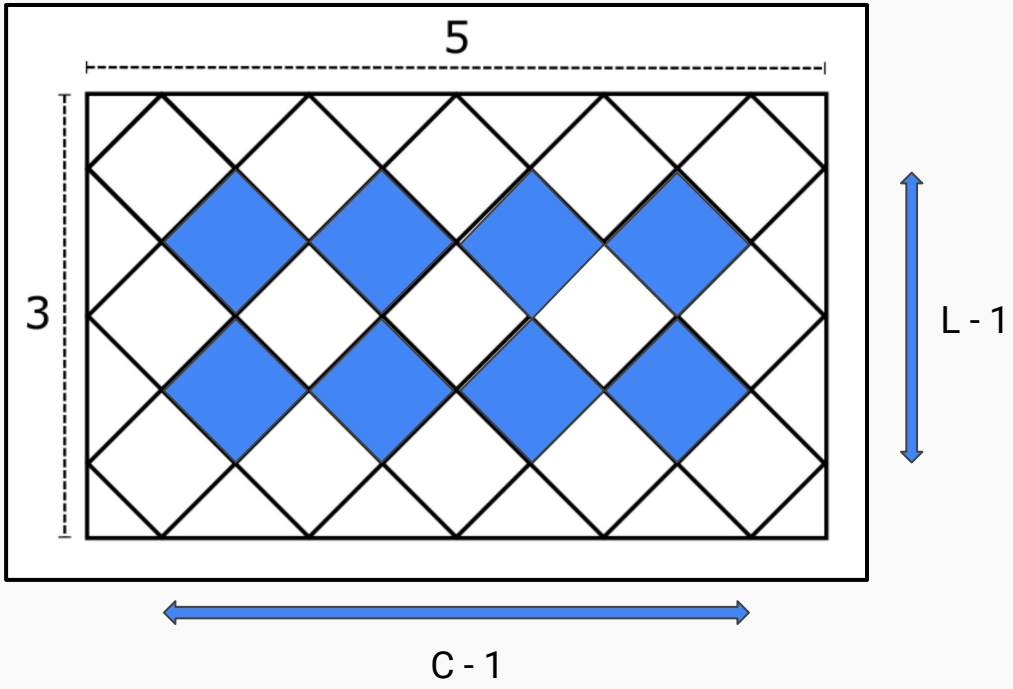


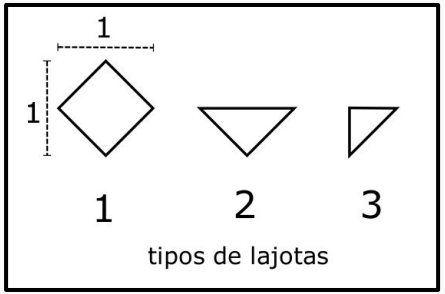


t1 = L*C + ?

t2 =

Solução

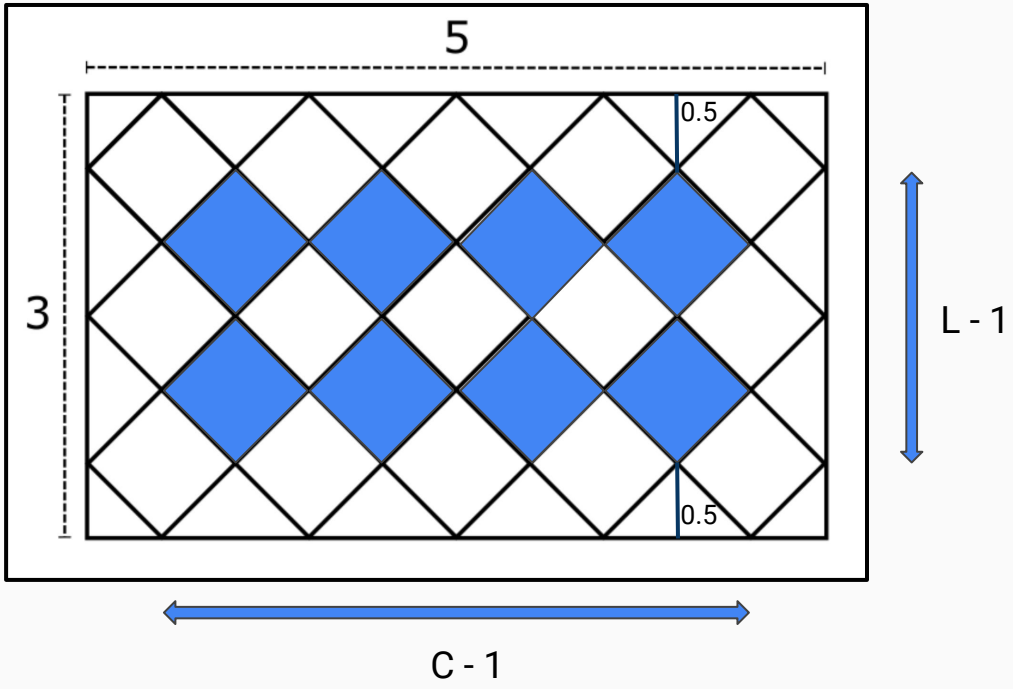


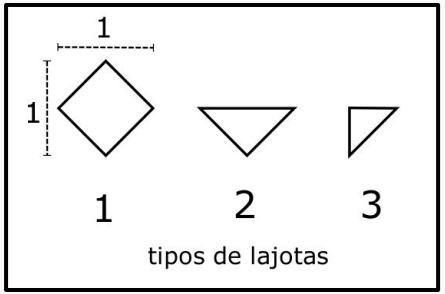


$t1 = L*C + ?$

$t2 =$

Solução

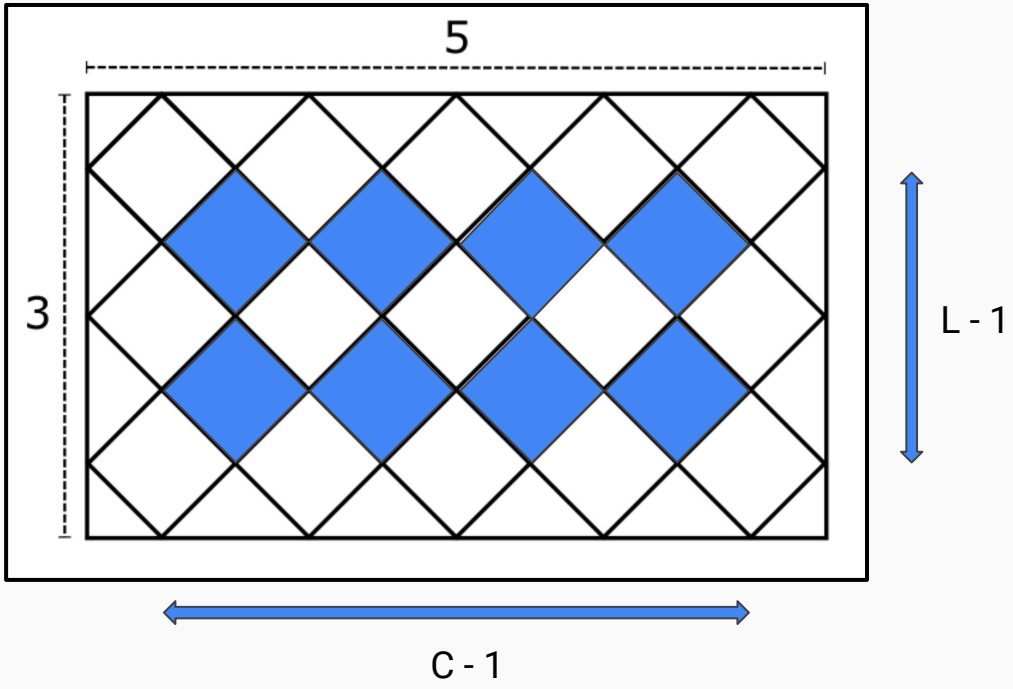


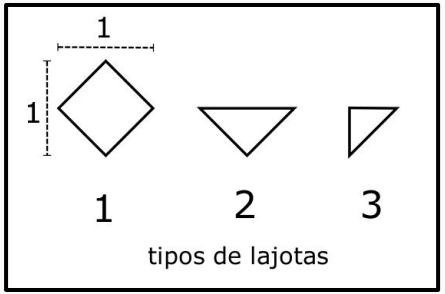


t1 = L*C + ?

t2 =

Solução

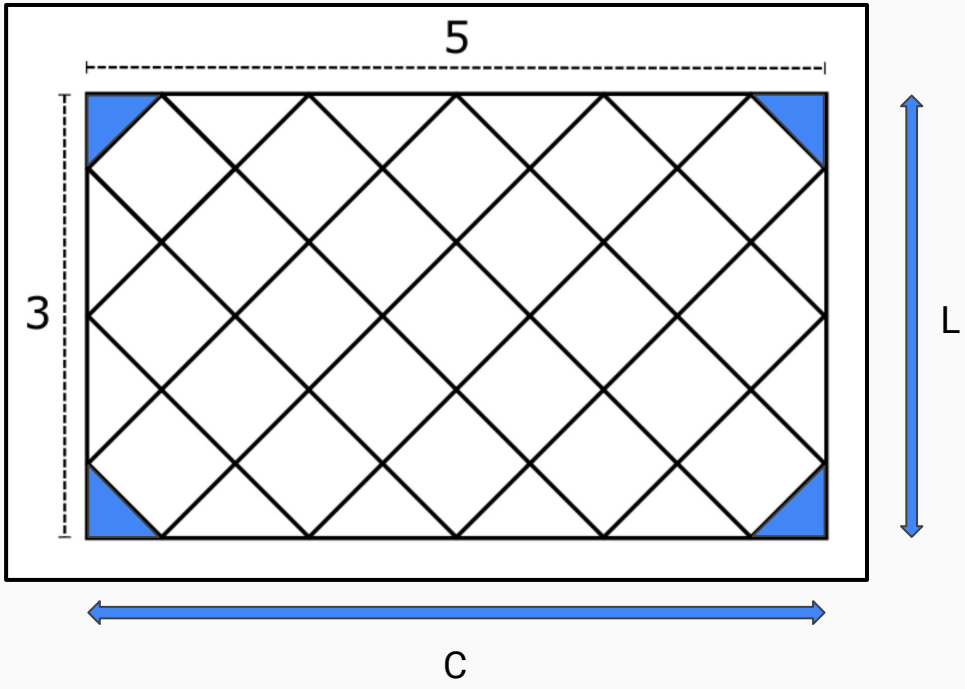


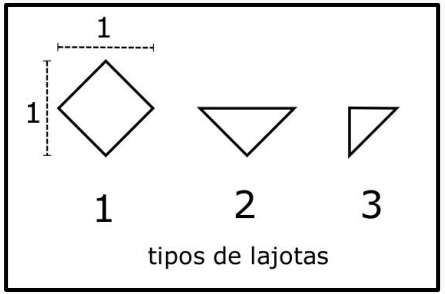


$$t1 = L*C + (L - 1)(C - 1)$$

$$t2 =$$

Solução

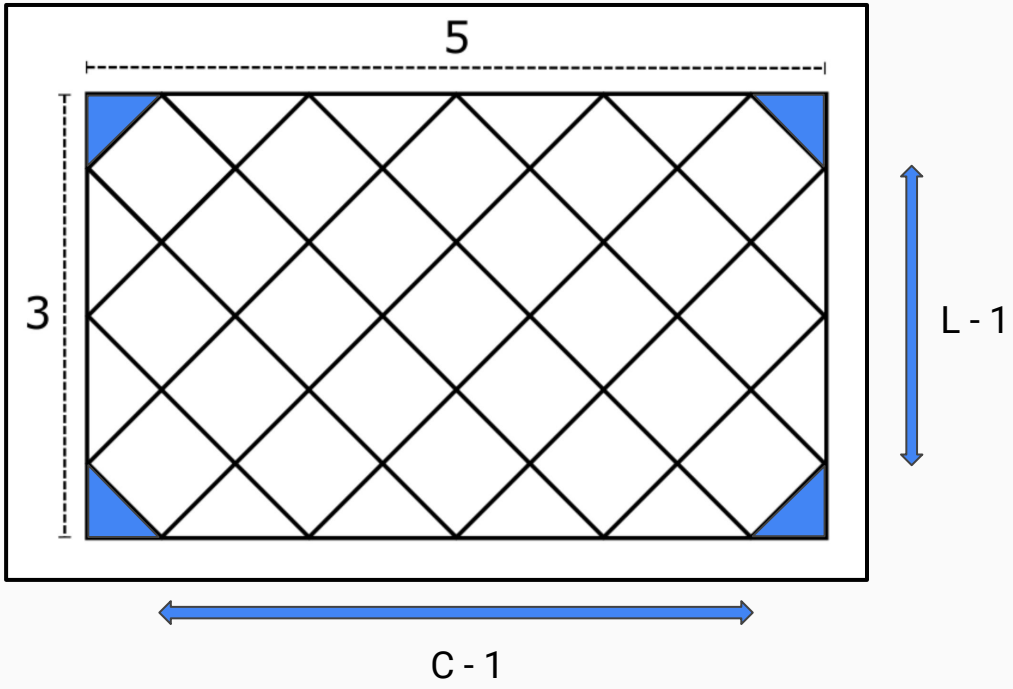


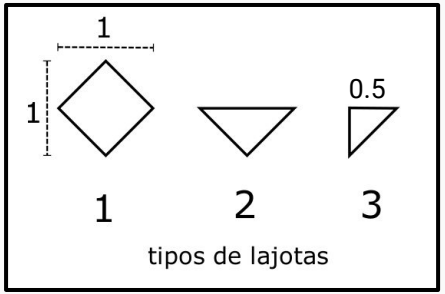


$$t1 = L*C + (L - 1)(C - 1)$$

$$t2 =$$

Solução

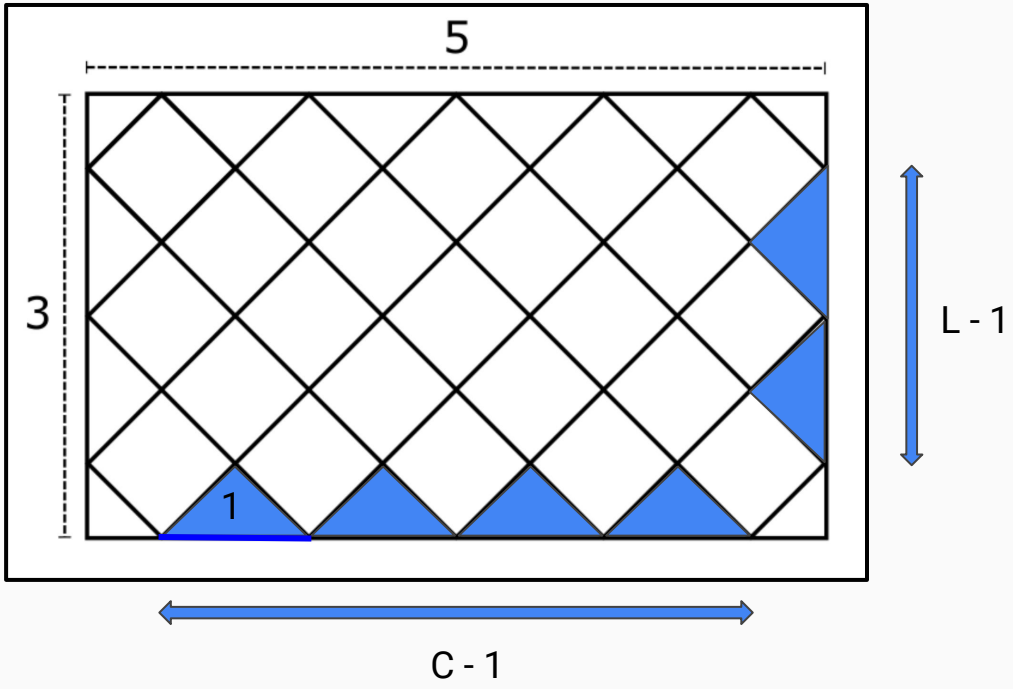


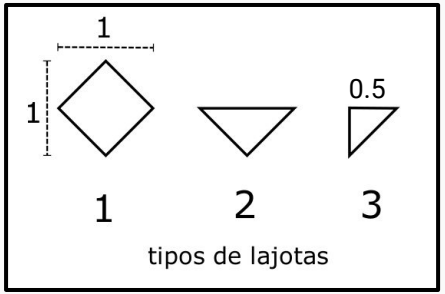


$$t1 = L*C + (L - 1)(C - 1)$$

$$t2 =$$

Solução

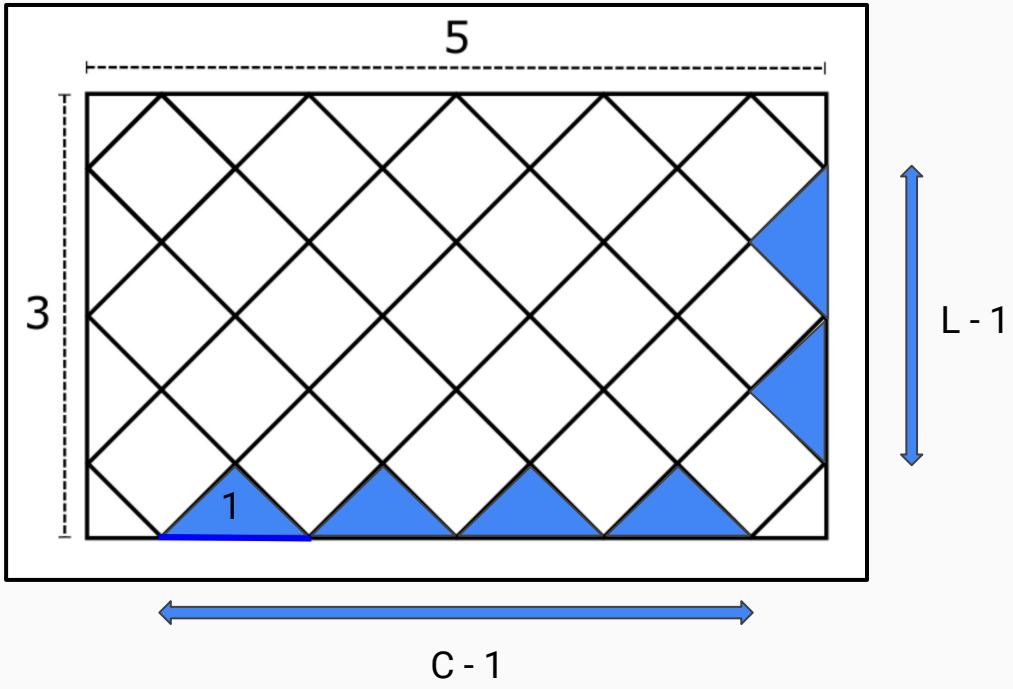


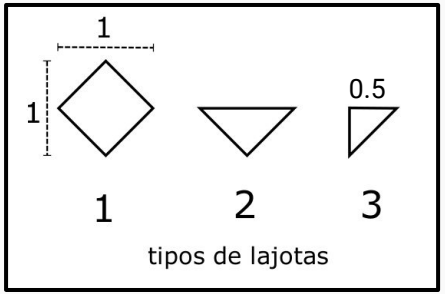


$$t1 = L*C + (L - 1)(C - 1)$$

$$t2 = (L - 1) + (C - 1)$$

Solução

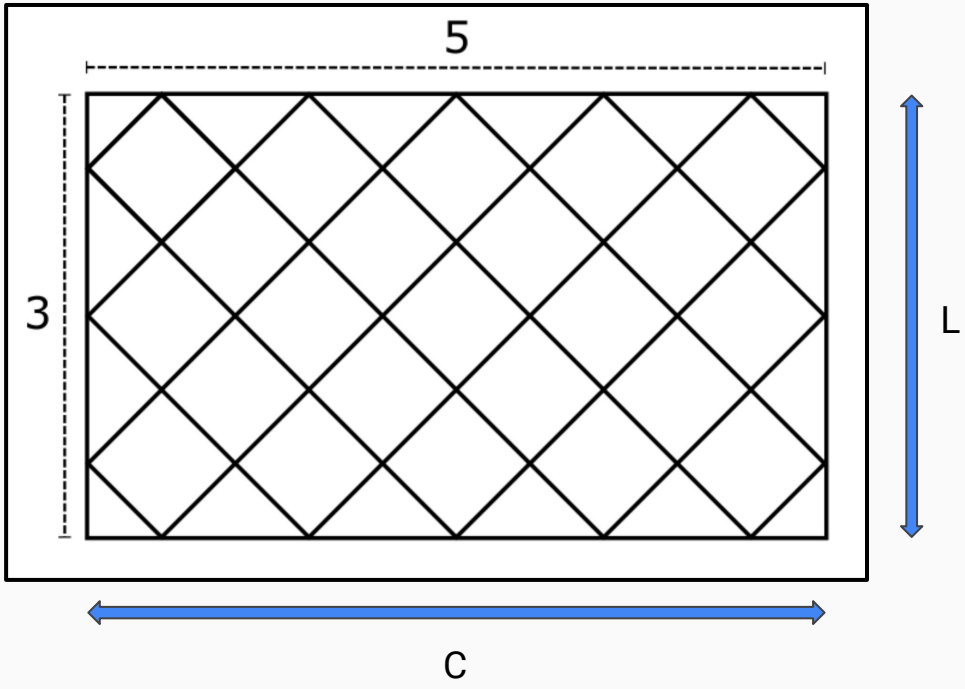


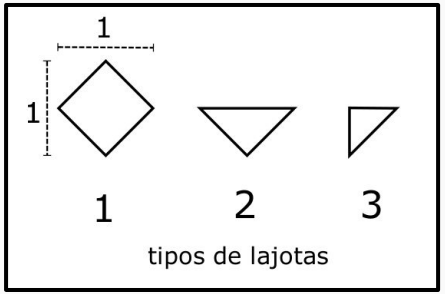


$$t1 = L*C + (L - 1)(C - 1)$$

$$t2 = 2(L - 1) + 2(C - 1)$$

Solução





$$t1 = L*C + (L - 1)(C - 1)$$

$$t2 = 2(L - 1) + 2(C - 1)$$

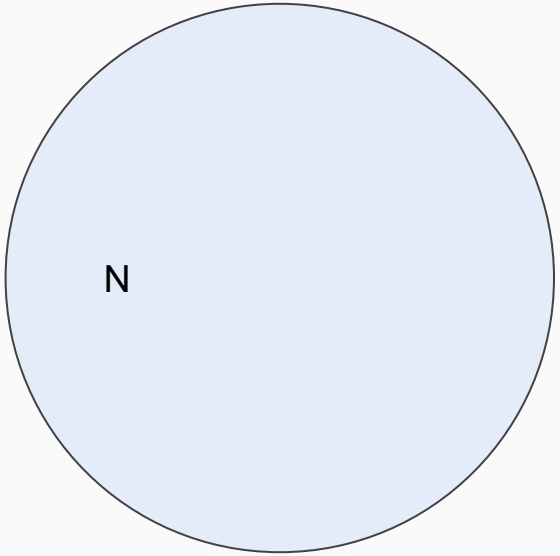
Implementação

Figurinhas da copa

- Médio

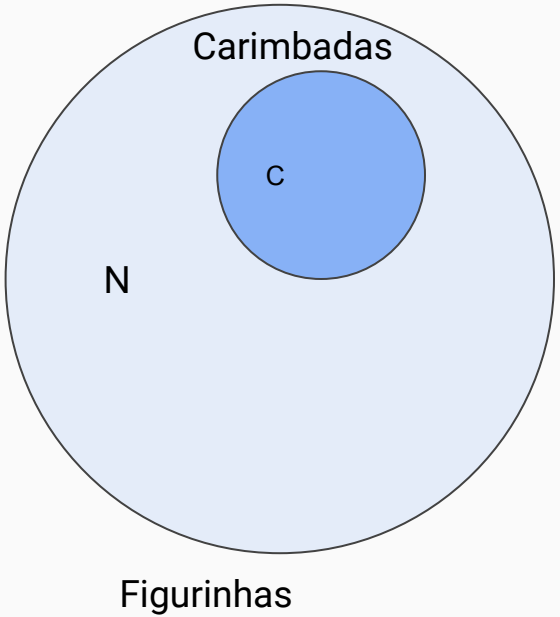
- Estrutura de dado simples
- Laço de repetição

Questão

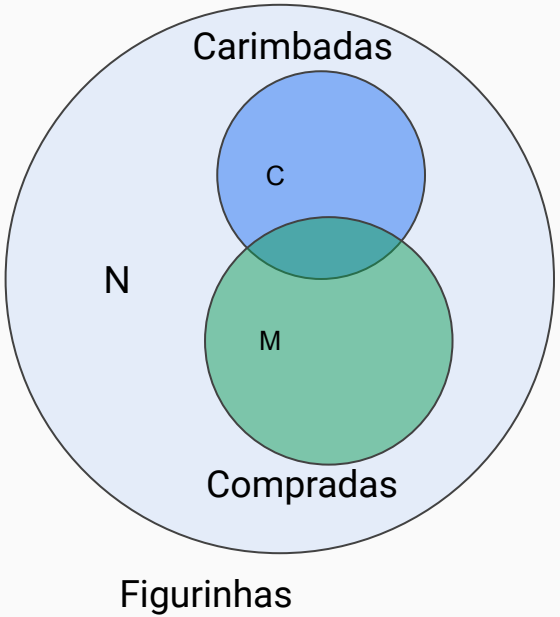


Figurinhas

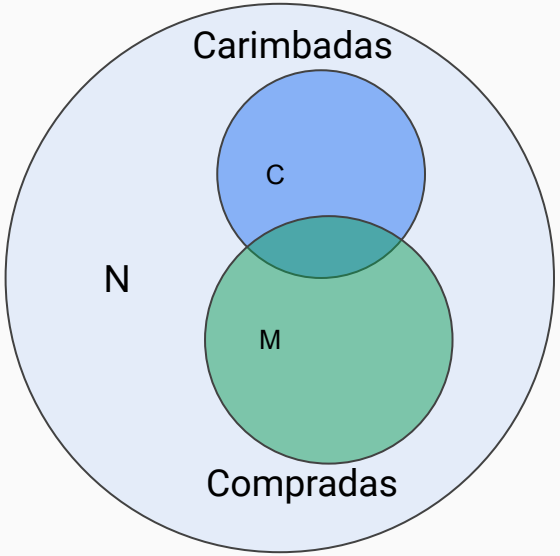
Questão



Questão



Questão



Figurinhas

Figurinhas carimbadas que não foram compradas

$R = C - M$

$\{x \mid x \in C \text{ e } x \notin M\}$

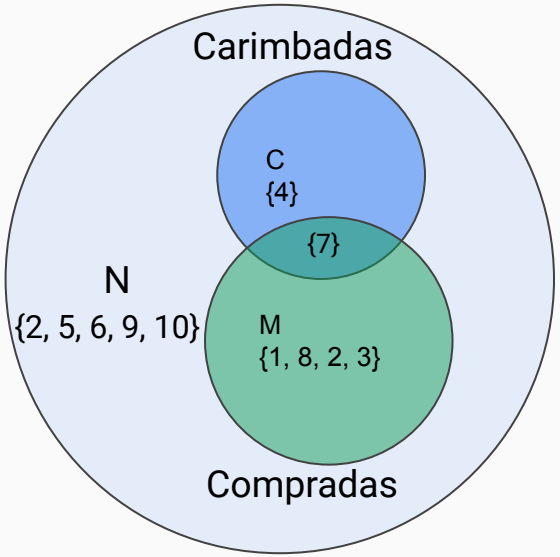
Questão

Entrada

```
10 2 5
4 7
7 1 2 8 3
```

Saída

```
1
```

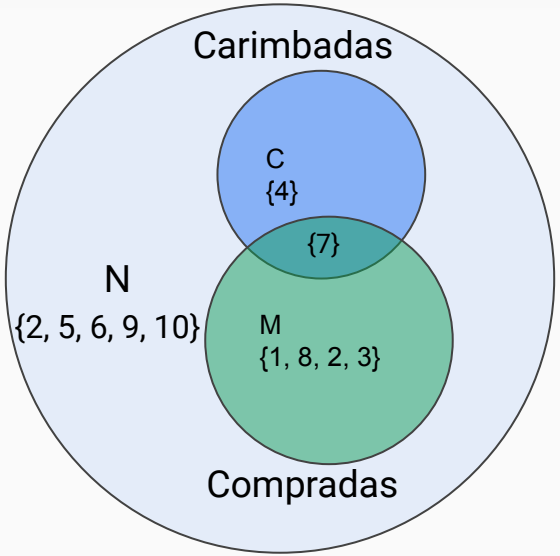


Figurinhas carimbadas que não foram compradas

R = C - M

{x | x ∈ C e x ∉ M}

Solução

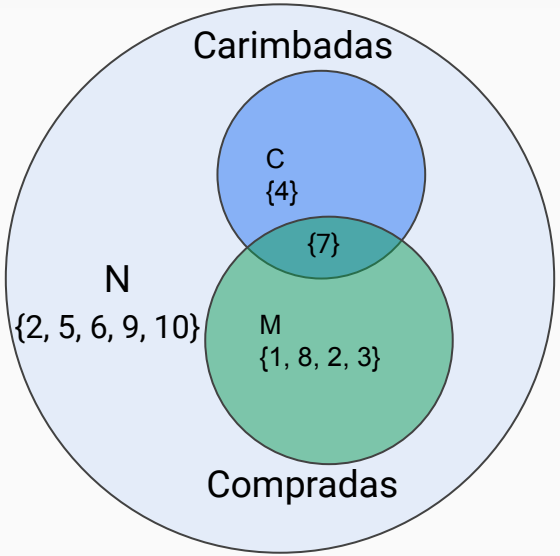


Figurinhas

0	1	2	3	4	5	6	7	8	9	10

N + 1 posições

Solução



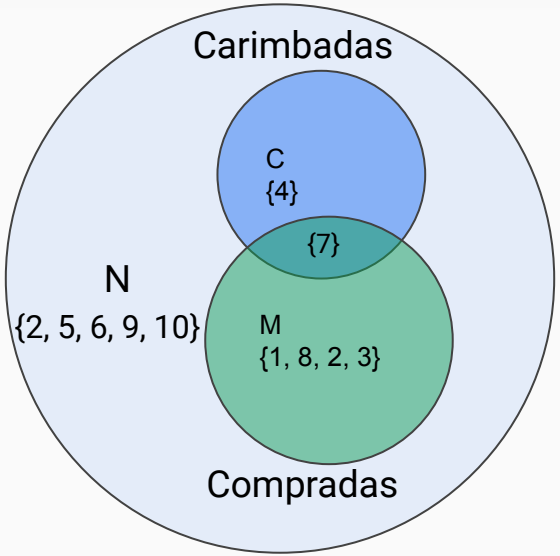
0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0

N + 1 posições

Carimbadas = {4, 7}

Compradas = {7, 1, 8, 2, 3}

Solução



Figurinhas

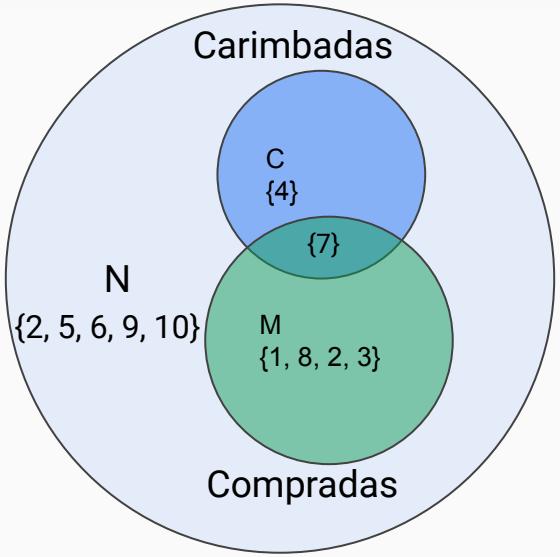
0	1	2	3	4	5	6	7	8	9	10
0	1	1	1	0	0	0	1	1	0	0

N + 1 posições

Carimbadas = {4, 7}

Compradas = {**7**, **1**, **8**, **2**, **3**}

Solução



Figurinhas

0	1	2	3	4	5	6	7	8	9	10
0	1	1	1	0	0	0	1	1	0	0

N + 1 posições

Carimbadas = {**4**, **7**}

Compradas = {7, 1, 8, 2, 3}

Resposta = Quais figurinhas do conjunto de carimbadas não foram compradas

Índices que possuem valor 0 no vetor

Implementação

Ilhas

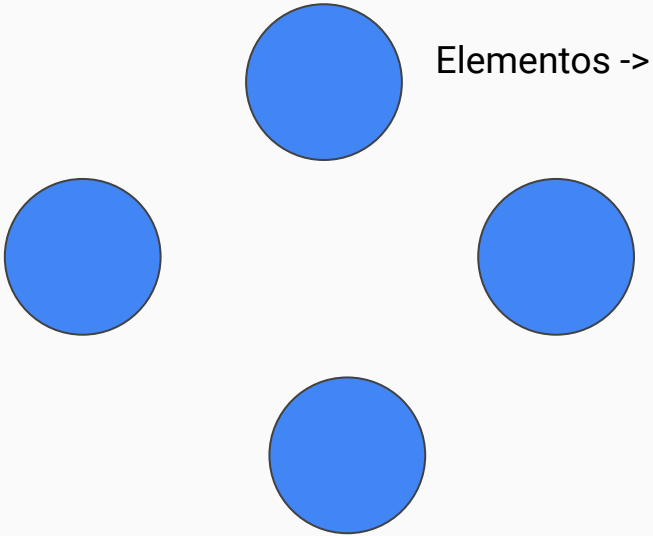
- Difícil

- Grafos
- Melhor caminho entre vértices
- Algoritmo de Dijkstra

O que são grafos?

Grafos

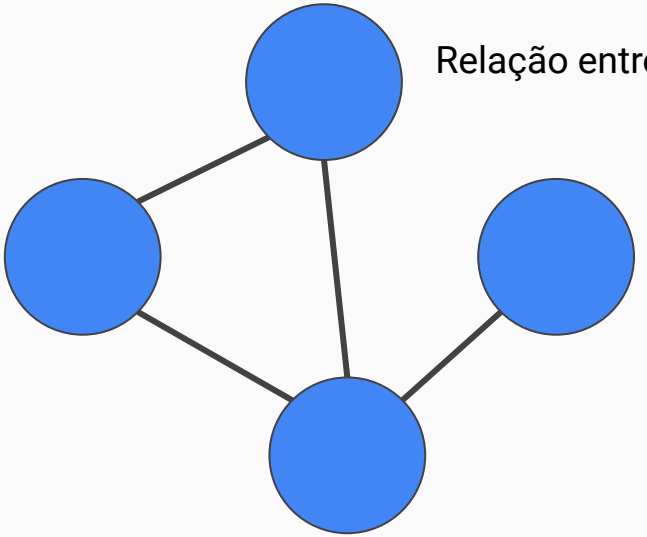
O que são grafos?



Elementos -> Vértices

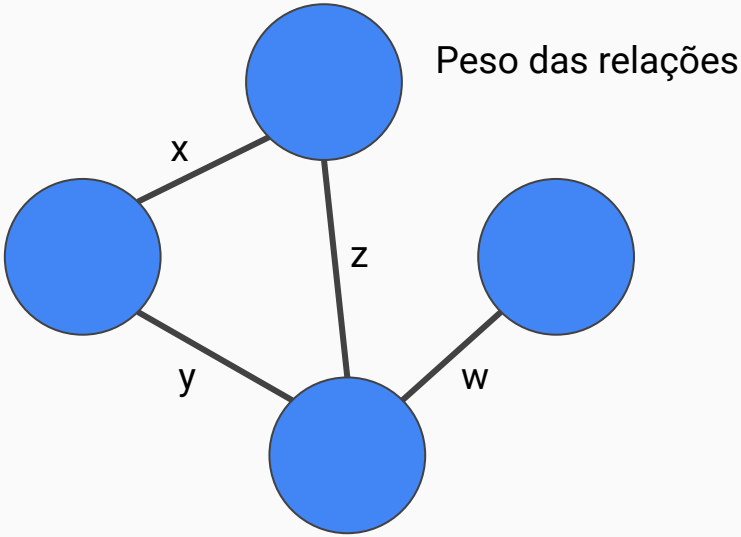
Grafos

O que são grafos?



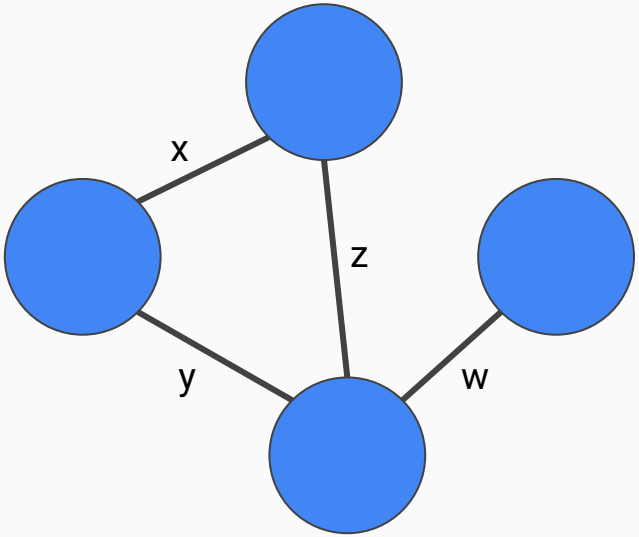
Relação entre elementos -> Aresta

O que são grafos?



Grafos

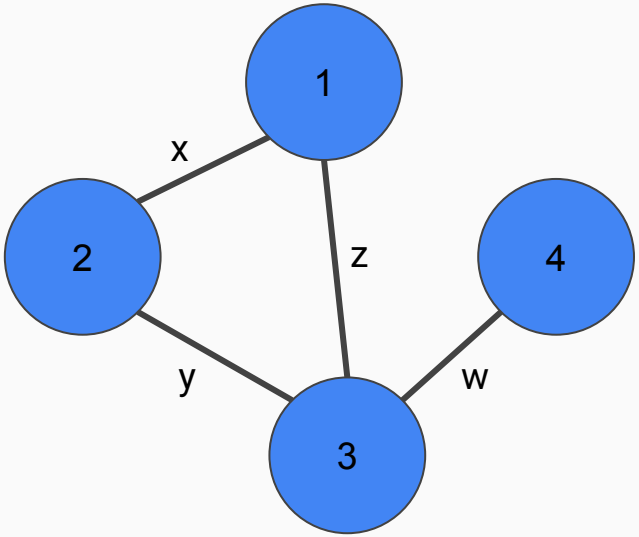
O que são grafos?



Como representar um grafo?

Grafos

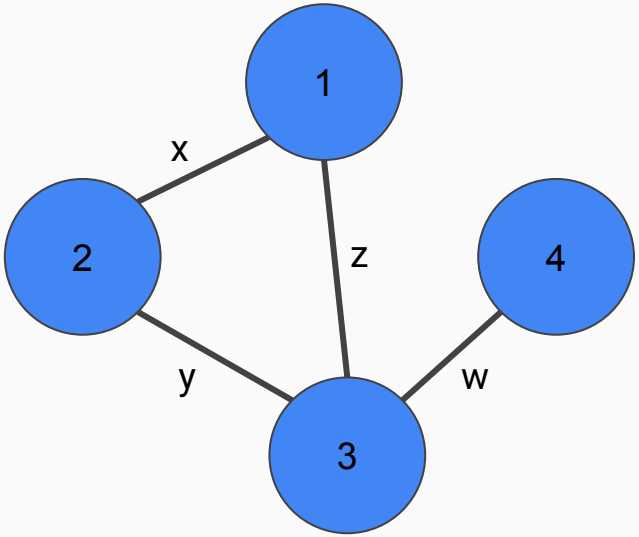
O que são grafos?



Como representar um grafo?

Grafos

O que são grafos?

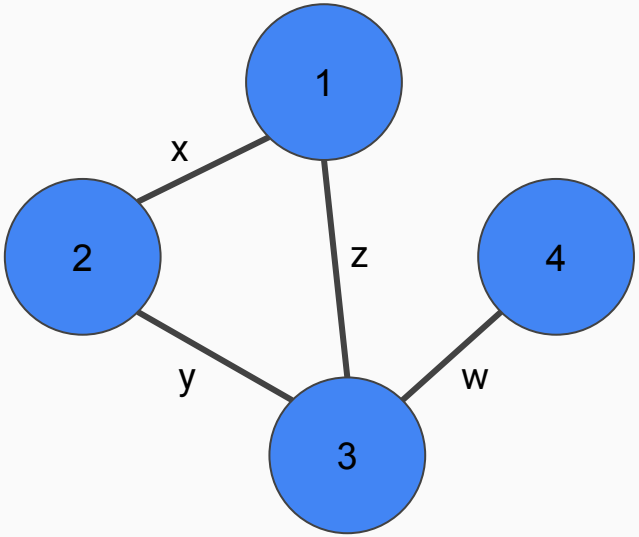


Como representar um grafo?

1	2 x	3 z

Grafos

O que são grafos?

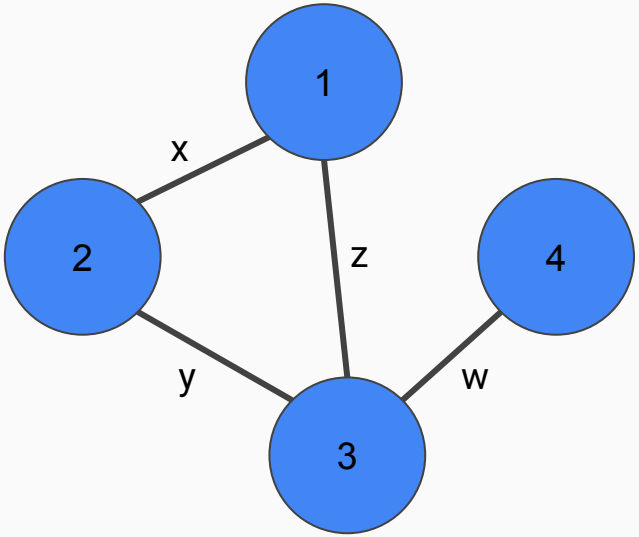


Como representar um grafo?

1	2 x	3 z	
2	1 x	3 y	
3	1 z	2 y	4 w
4	3 w		

Grafos

O que são grafos?

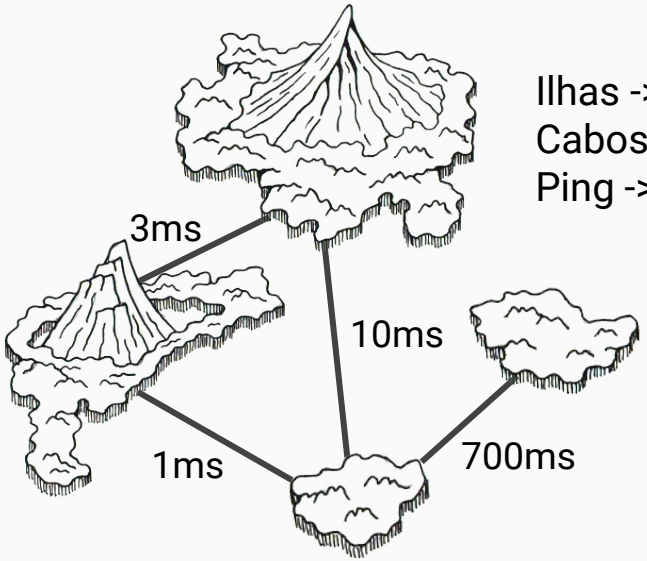


Como representar um grafo?

1	2 x	3 z	
2	1 x	3 y	
3	1 z	2 y	4 w
4	3 w		

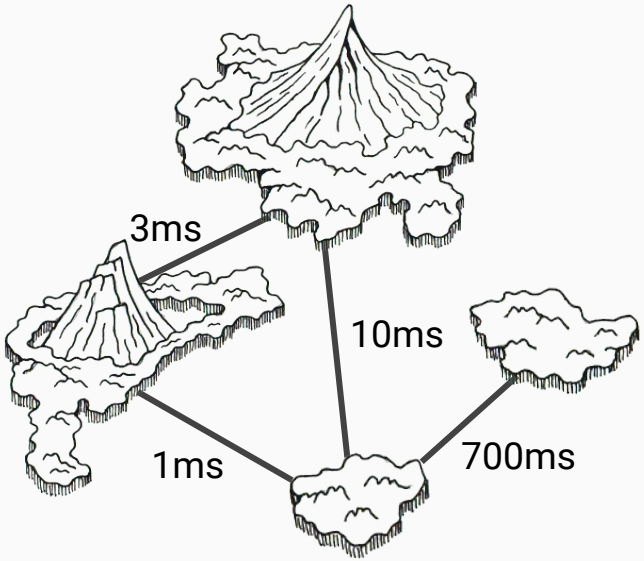
Lista de Adjacência

Questão



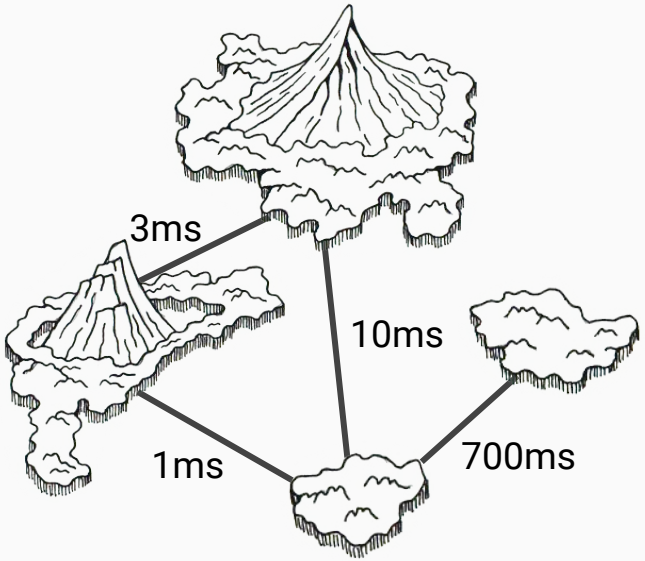
Ilhas -> Vértices
Cabos -> Arestas
Ping -> Pesos

Questão



Ilha principal -> Servidor

Questão

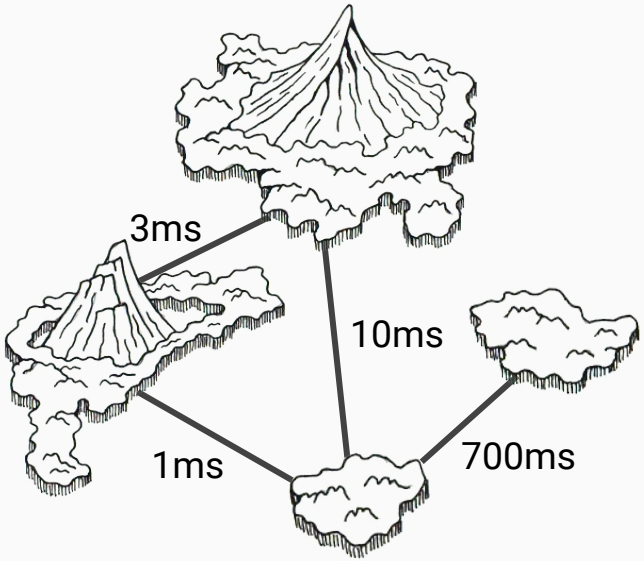


Ilha principal -> Servidor

Quando duas ilhas se comunicam através de uma série de cabos, o ping entre elas é a soma dos pings de cada cabo no caminho

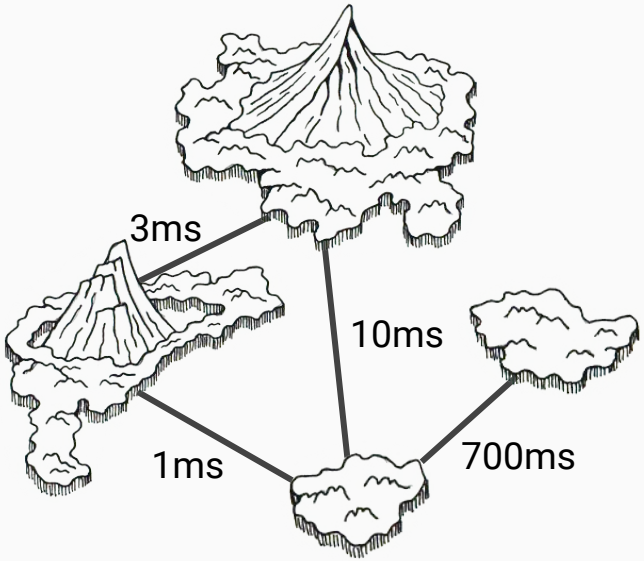
Um par de ilhas sempre se comunica através do caminho com menor ping possível

Questão



Diferença dos pings entre a ilha com o maior ping e a ilha com o menor ping até o servidor.

Questão

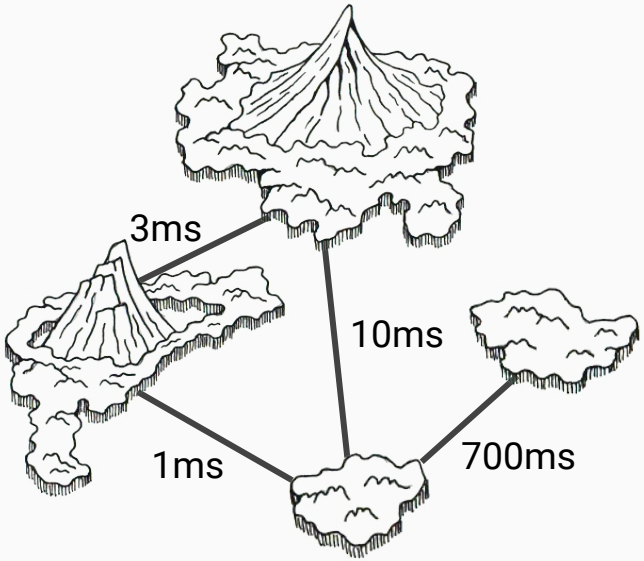


Diferença dos pings entre a ilha com o maior ping e a ilha com o menor ping até o servidor.

3
4
704

Maior - Menor = 704 - 3 = **701**

Questão



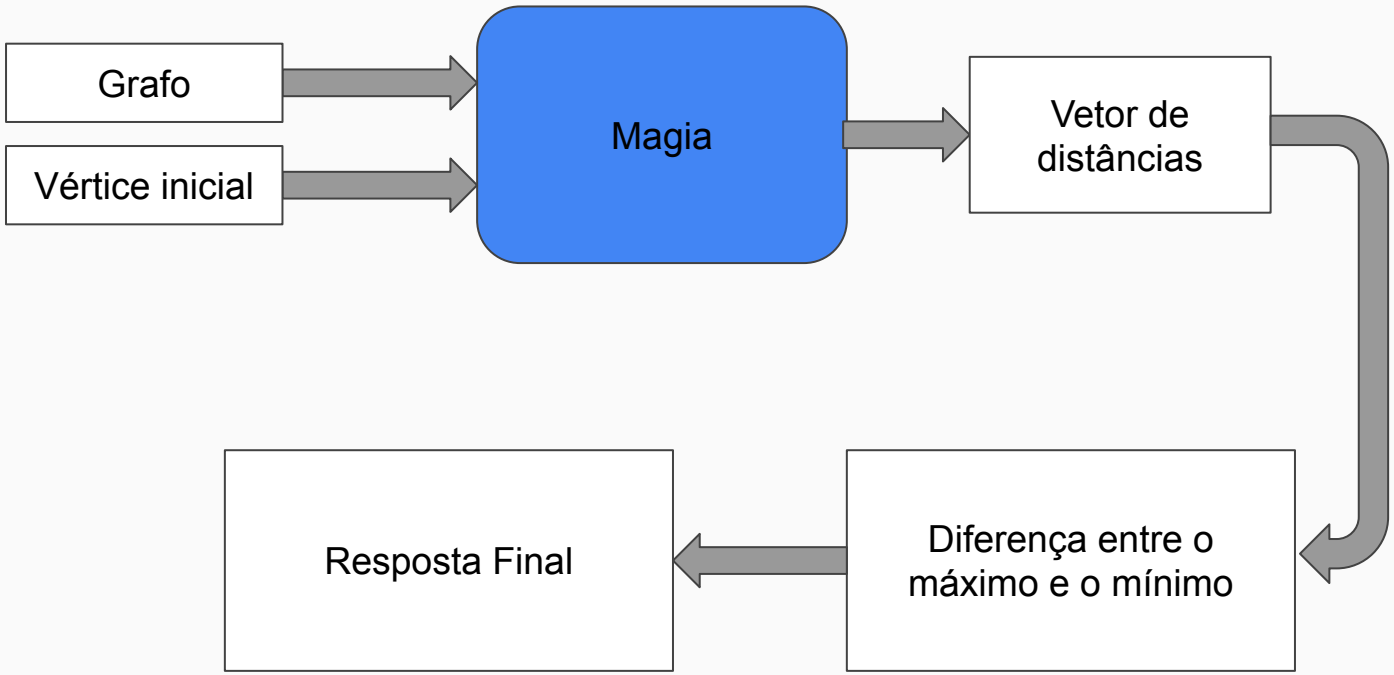
3
4
704

Vetor de distâncias

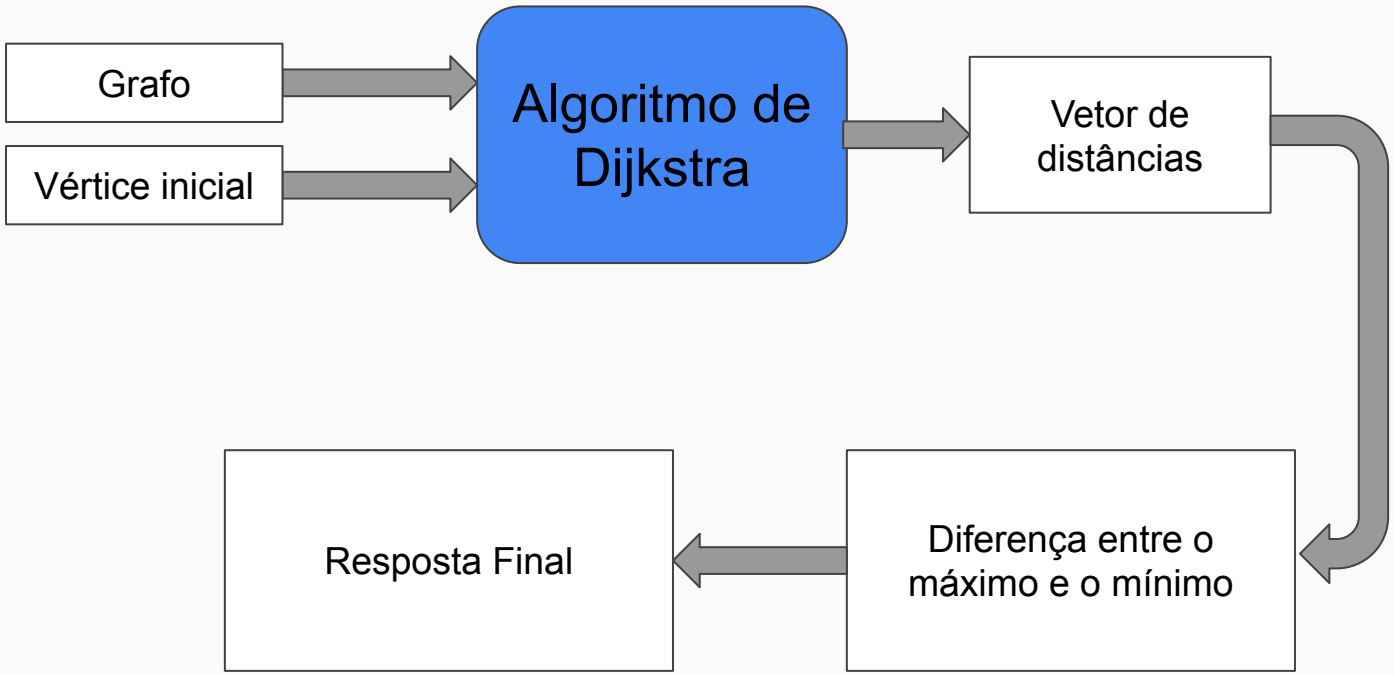
Pensar em distâncias é mais intuitivo
São problemas análogos

Solução

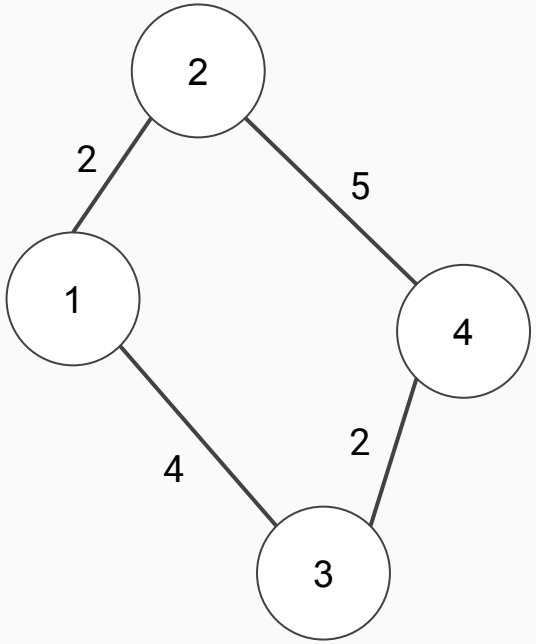
Solução


Grafo
Vértice inicial
Magia
Vetor de distâncias
Diferença entre o máximo e o mínimo
Resposta Final

Solução


Grafo
Vértice inicial
Algoritmo de Dijkstra
Vetor de distâncias
Diferença entre o máximo e o mínimo
Resposta Final

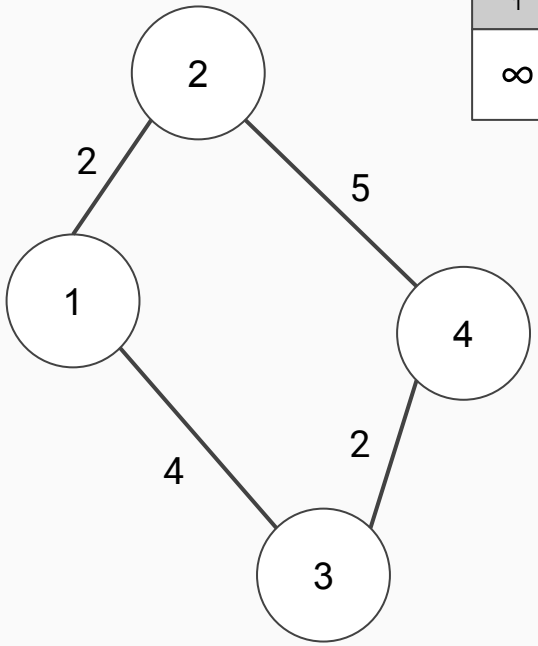
Dijkstra



Servidor -> Vértice 1

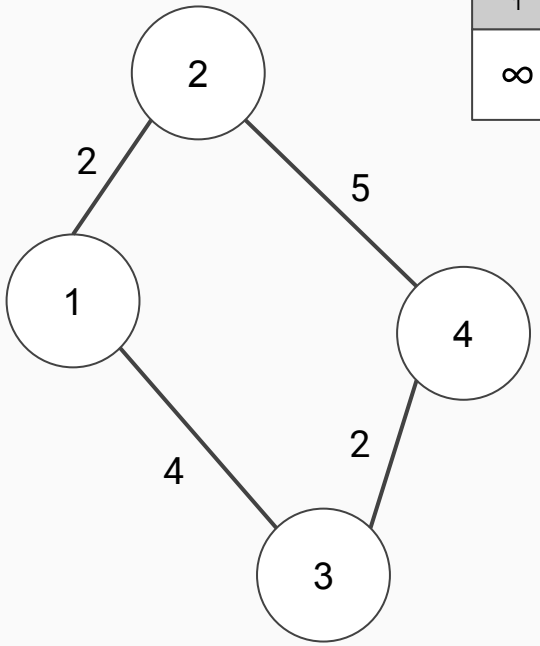
Calcular o vetor de distâncias

Dijkstra



1	2	3	4
∞	∞	∞	∞

Dijkstra

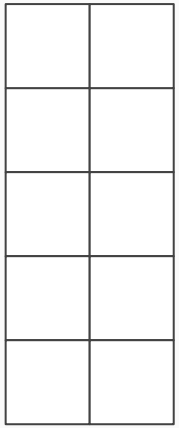


1	2	3	4
∞	∞	∞	∞

Fila de prioridade:

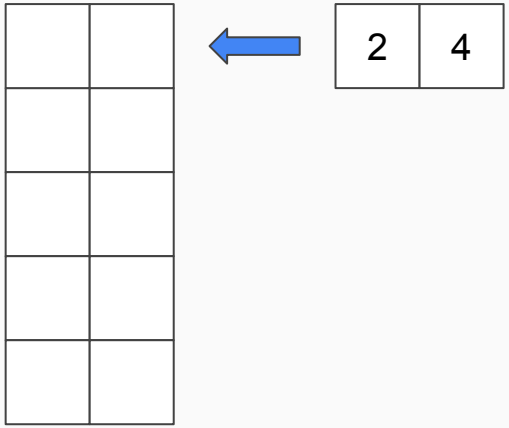
d	v

Fila de prioridade



Na fila, os menores elementos aparecem nas primeiras posições

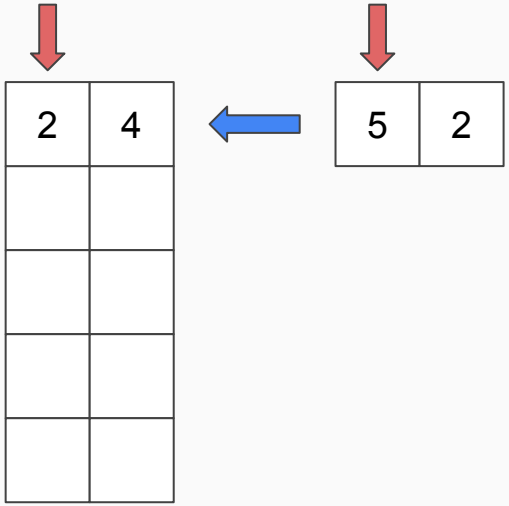
Fila de prioridade



Fila de prioridade

2	4

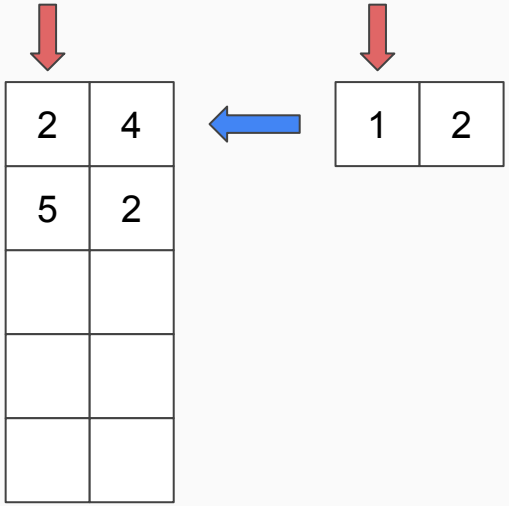
Fila de prioridade



Fila de prioridade

2	4
5	2

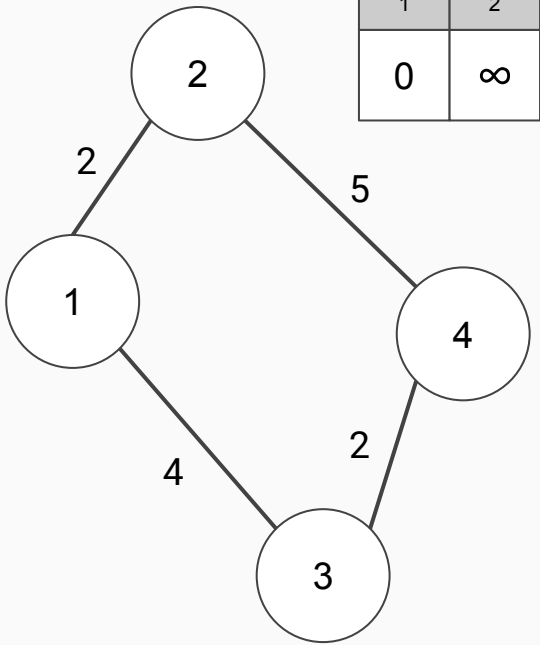
Fila de prioridade



Fila de prioridade

1	2
2	4
5	2

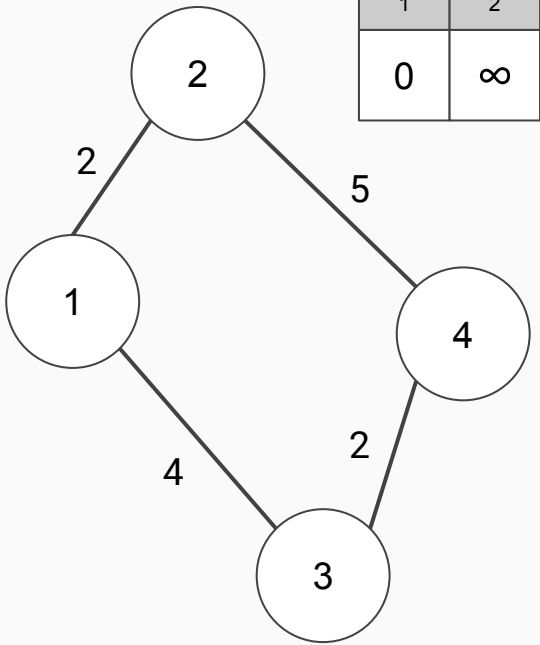
Dijkstra



1	2	3	4
0	∞	∞	∞

d	v
0	1

Dijkstra



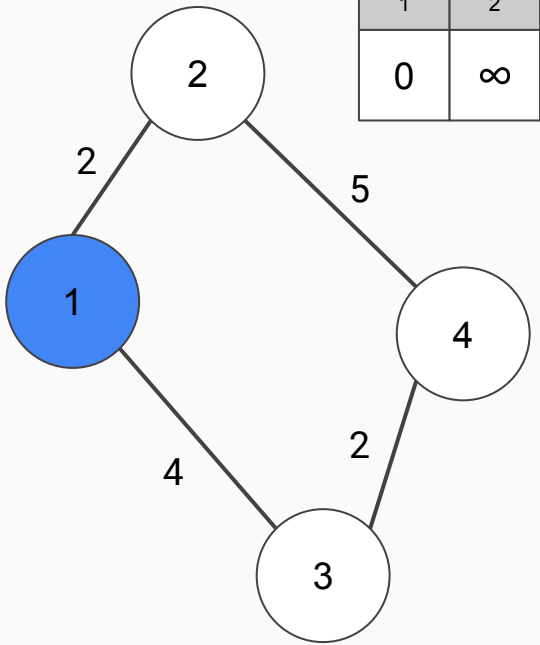
1	2	3	4
0	∞	∞	∞

0	1

1º Pegar o primeiro elemento da fila

d	v

Dijkstra



1	2	3	4
0	∞	∞	∞

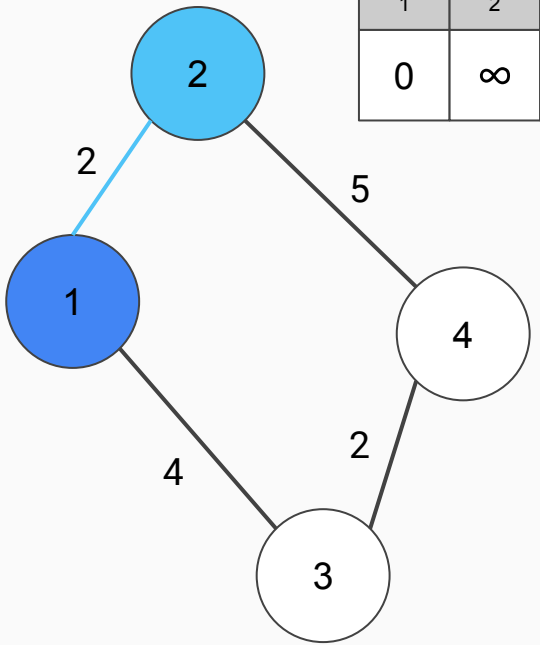
0	1

2º Verificar se ele já foi visitado.

Se sim, não faça nada.
Se não, marcá-lo como visitado

d	v

Dijkstra



1	2	3	4
0	∞	∞	∞

0	1

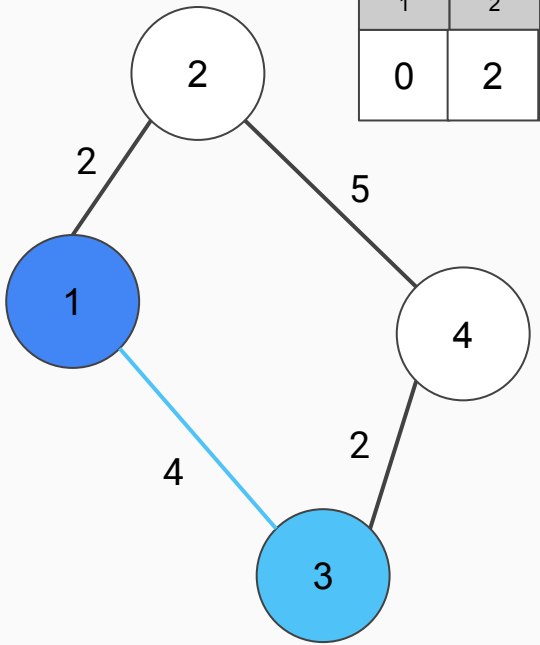
3º Processar esse vértice olhando para os vértices adjacentes a ele

2	2

Se d[1] + peso < d[2]:
d[2] = d[1] + peso
inclui o par {d[2], 2} na fila

d	v

Dijkstra



1	2	3	4
0	2	4	∞

0	1

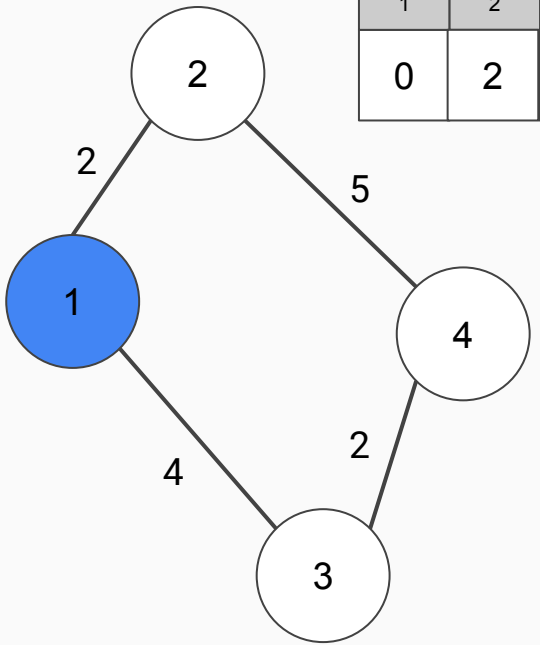
3º Processar esse vértice olhando para os vértices adjacentes a ele

4	3

Se d[1] + peso < d[3]:
d[3] = d[1] + peso
inclui o par {d[3], 3} na fila

d	v
2	2
4	3

Dijkstra

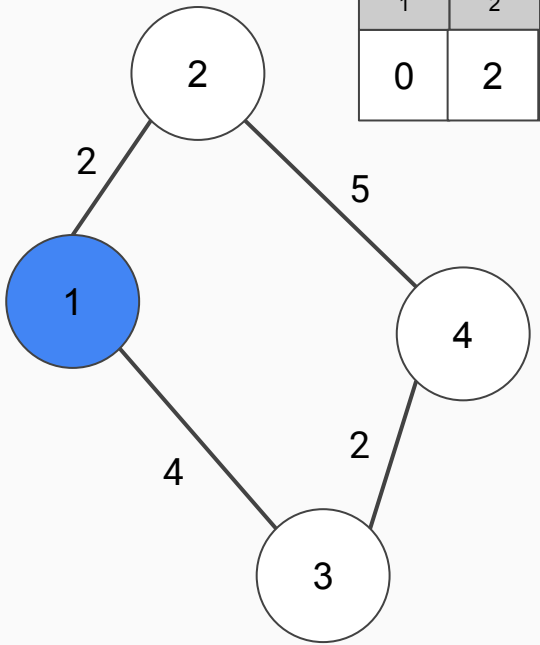


1	2	3	4
0	2	4	∞

4º Repita os passos 1 - 3 até a fila ficar vazia

d	v
2	2
4	3

Dijkstra



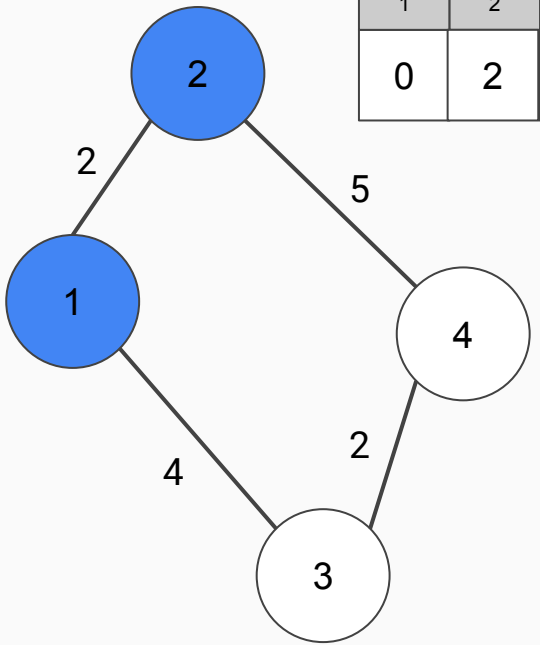
1	2	3	4
0	2	4	∞

2	2

1º Pegar o primeiro elemento da fila

d	v
4	3

Dijkstra



1	2	3	4
0	2	4	∞

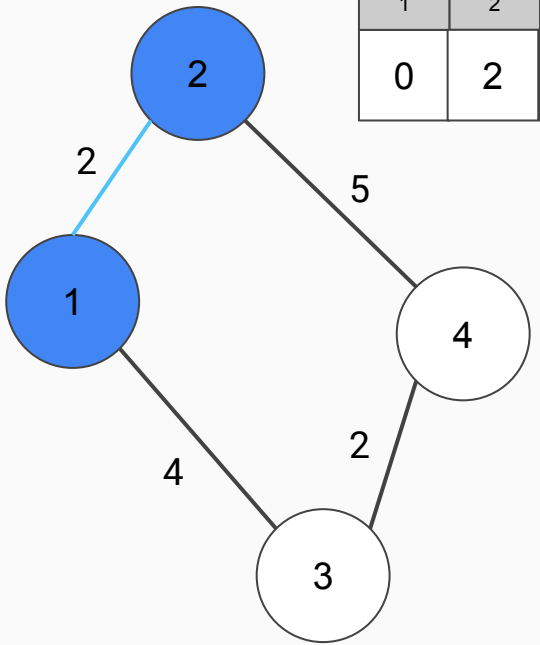
2	2

2º Verificar se ele já foi visitado.

Se sim, não faça nada.
Se não, marcá-lo como visitado

d	v
4	3

Dijkstra



1	2	3	4
0	2	4	∞

2	2

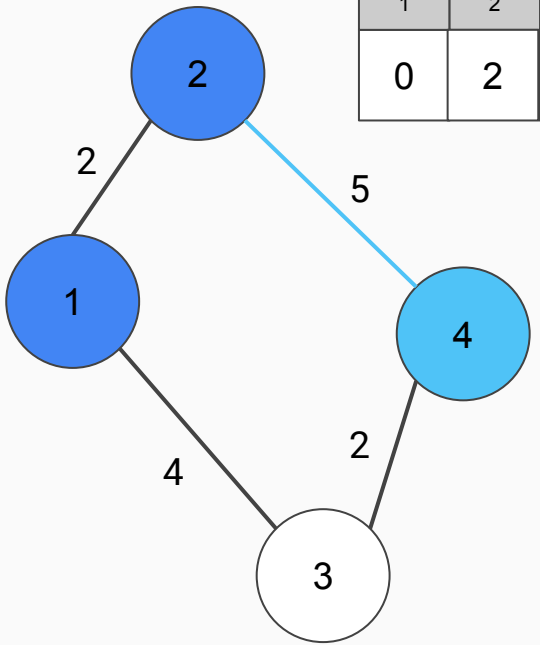
3º Processar esse vértice olhando para os vértices adjacentes a ele

2	1

Se d[2] + peso < d[1]:
d[1] = d[2] + peso
inclui o par {d[1], 1} na fila

d	v
4	3

Dijkstra



1	2	3	4
0	2	4	7

2	2

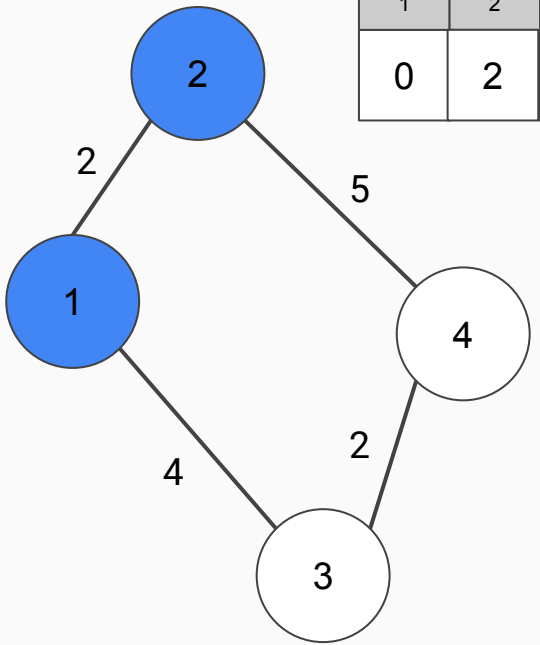
3º Processar esse vértice olhando para os vértices adjacentes a ele

5	4

Se d[2] + peso < d[4]:
d[4] = d[2] + peso
inclui o par {d[4], 4} na fila

d	v
4	3
7	4

Dijkstra



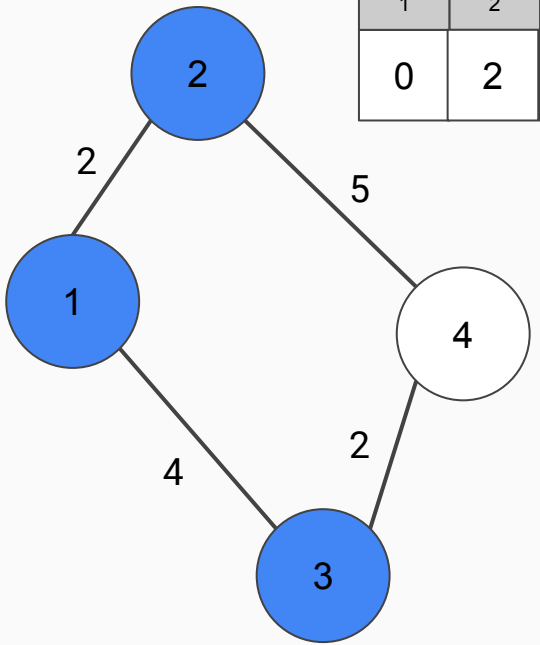
1	2	3	4
0	2	4	7

4	3

1º Pegar o primeiro elemento da fila

d	v
7	4

Dijkstra



1	2	3	4
0	2	4	7

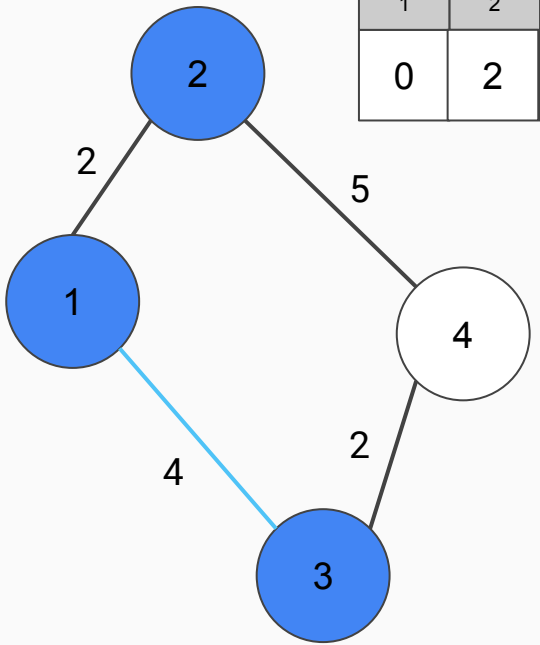
4	3

2º Verificar se ele já foi visitado.

Se sim, não faça nada.
Se não, marcá-lo como visitado

d	v
7	4

Dijkstra



1	2	3	4
0	2	4	7

4	3

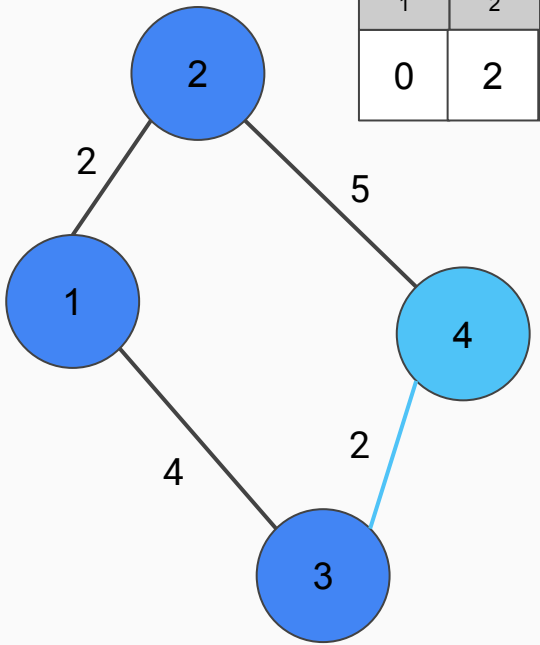
3º Processar esse vértice olhando para os vértices adjacentes a ele

4	1

Se d[1] + peso < d[3]:
d[1] = d[3] + peso
inclui o par {d[1], 1} na fila

d	v
7	4

Dijkstra



1	2	3	4
0	2	4	7

4	3

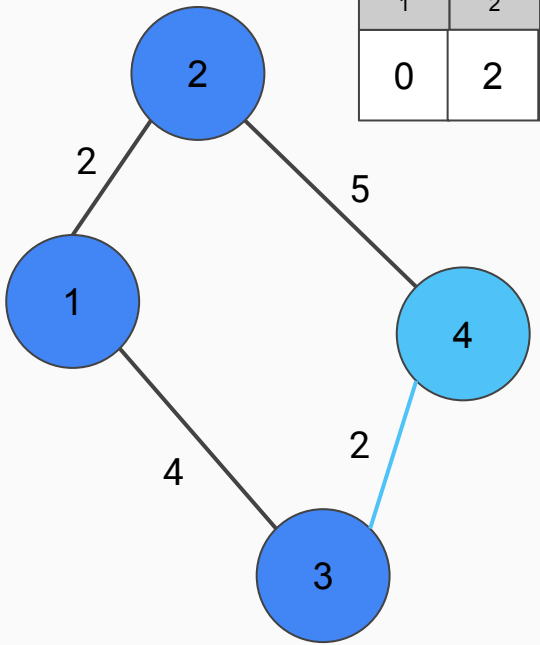
3º Processar esse vértice olhando para os vértices adjacentes a ele

2	4

Se d[4] + peso < d[3]:
d[4] = d[3] + peso
inclui o par {d[4], 4} na fila

d	v
7	4

Dijkstra



1	2	3	4
0	2	4	6

4	3

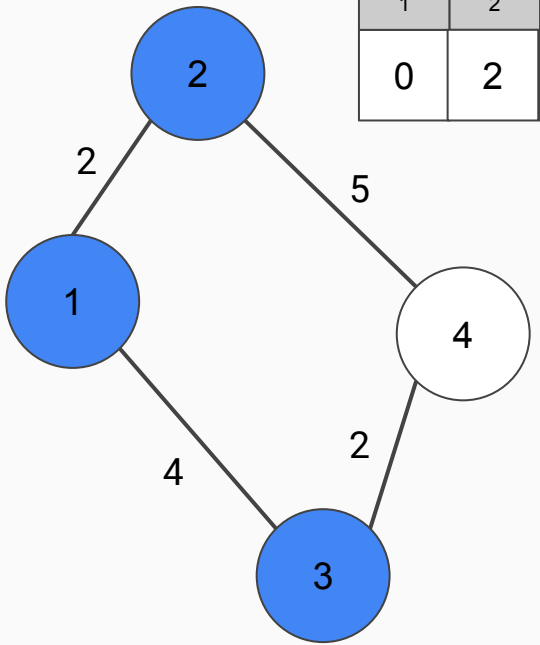
3º Processar esse vértice olhando para os vértices adjacentes a ele

2	4

Se d[4] + peso < d[3]:
d[4] = d[3] + peso
inclui o par {d[4], 4} na fila

d	v
6	4
7	4

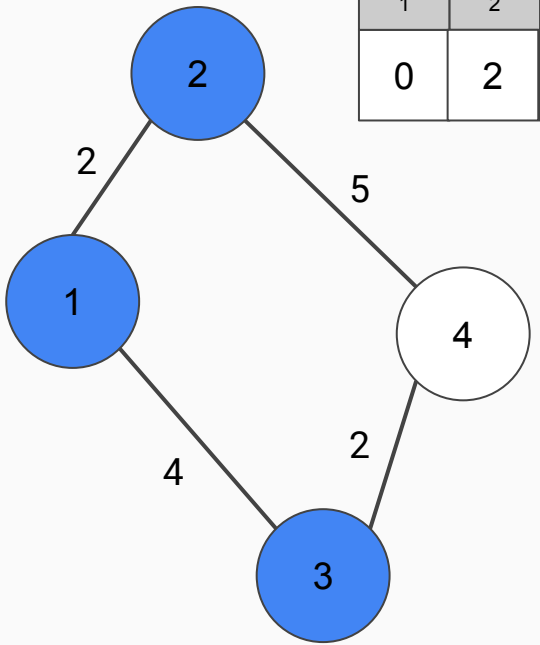
Dijkstra



1	2	3	4
0	2	4	6

d	v
6	4
7	4

Dijkstra

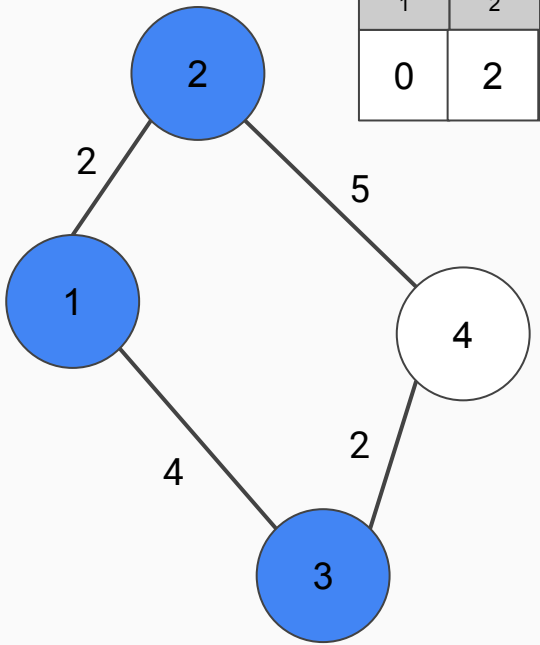


1	2	3	4
0	2	4	6

d	v
6	4
7	4

Existem dois caminhos do 1 até o 4, um que demora 7, outro que demora 6

Dijkstra



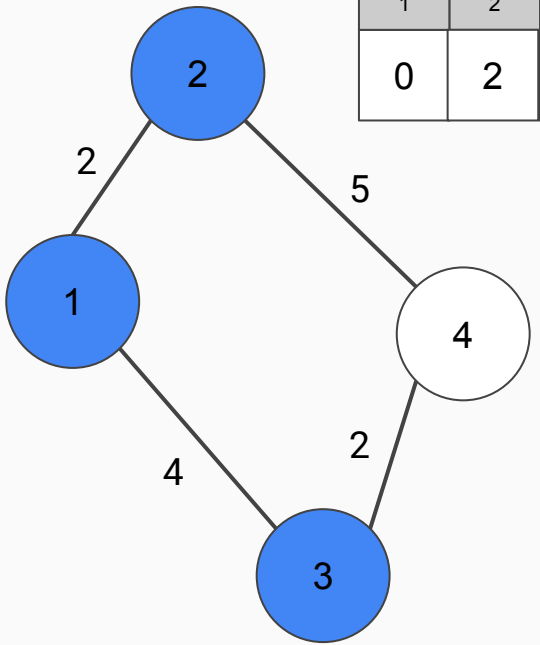
1	2	3	4
0	2	4	6

d	v
6	4
7	4

Existem dois caminhos do 1 até o 4, um que demora 7, outro que demora 6

A fila de prioridade faz com que eu sempre olhe as menores distâncias primeiro

Dijkstra



1	2	3	4
0	2	4	6

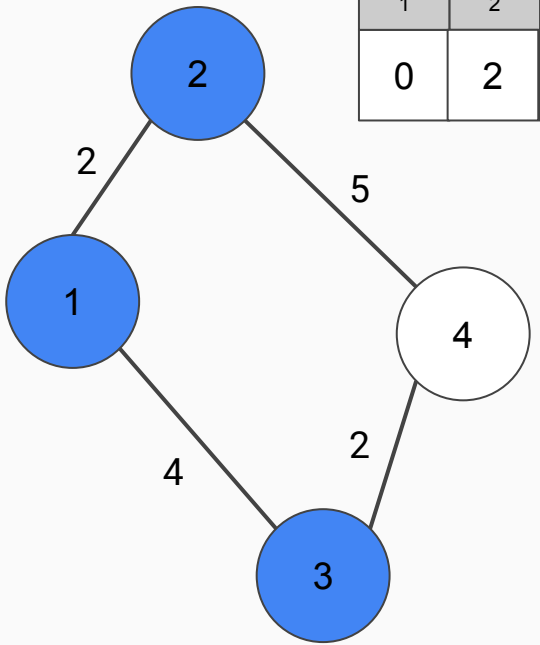
d	v
6	4
7	4

Existem dois caminhos do 1 até o 4, um que demora 7, outro que demora 6

A fila de prioridade faz com que eu sempre olhe as menores distâncias primeiro

Depois de processar o par {6, 4}, eu não irei olhar o par {7, 4}, pois eu já marquei que o 4 foi visitado

Dijkstra



1	2	3	4
0	2	4	6

d	v
6	4
7	4

Existem dois caminhos do 1 até o 4, um que demora 7, outro que demora 6

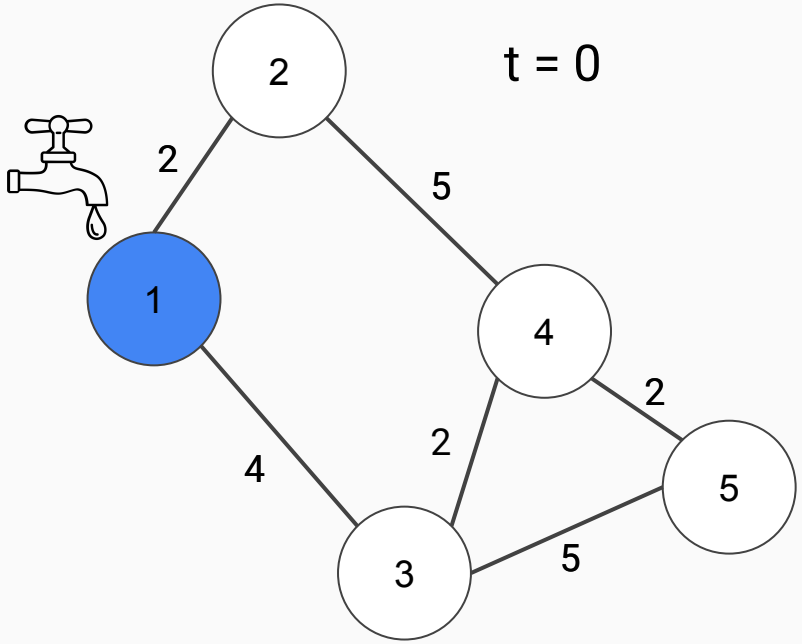
A fila de prioridade faz com que eu sempre olhe as menores distâncias primeiro

Depois de processar o par {6, 4}, eu não irei olhar o par {7, 4}, pois eu já marquei que o 4 foi visitado

Isso garante que eu sempre olhos os melhores caminhos até certo ponto

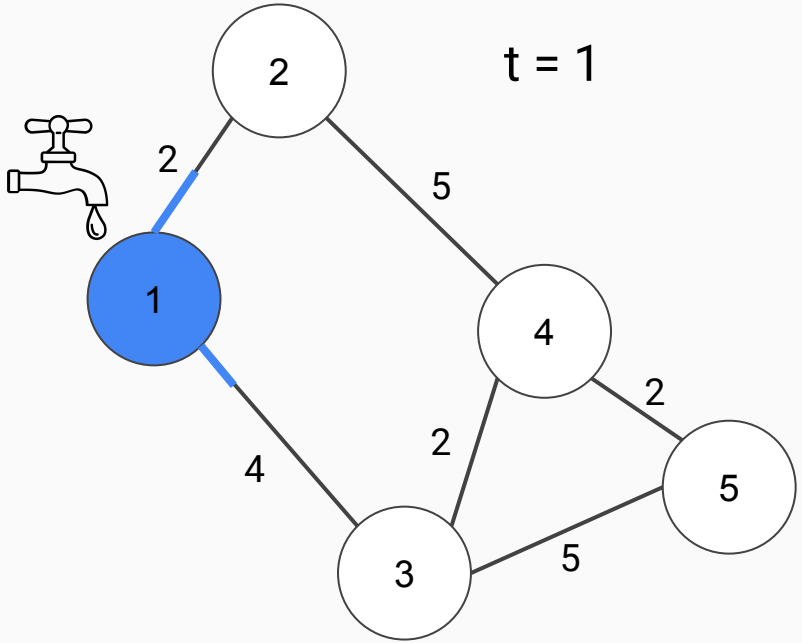
Porque funciona?

Dijkstra



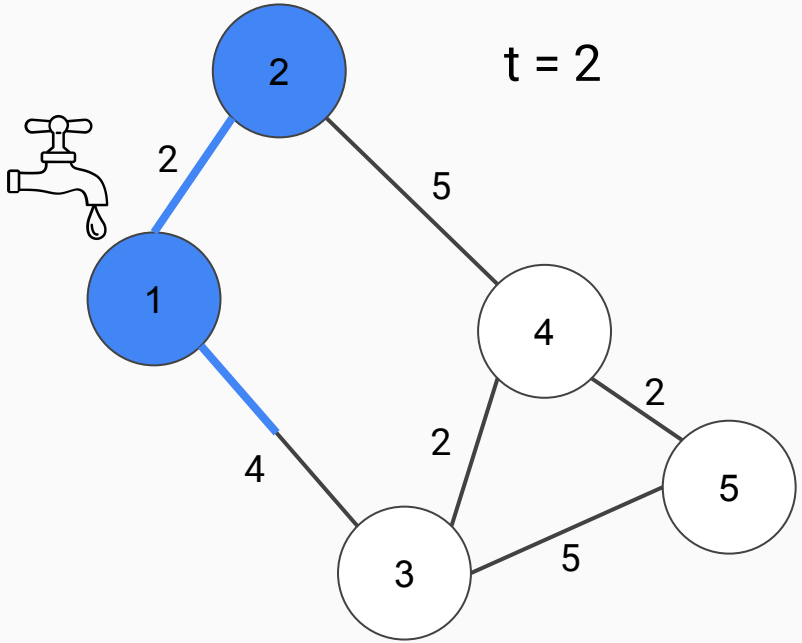
1	2	3	4	5
0				

Dijkstra



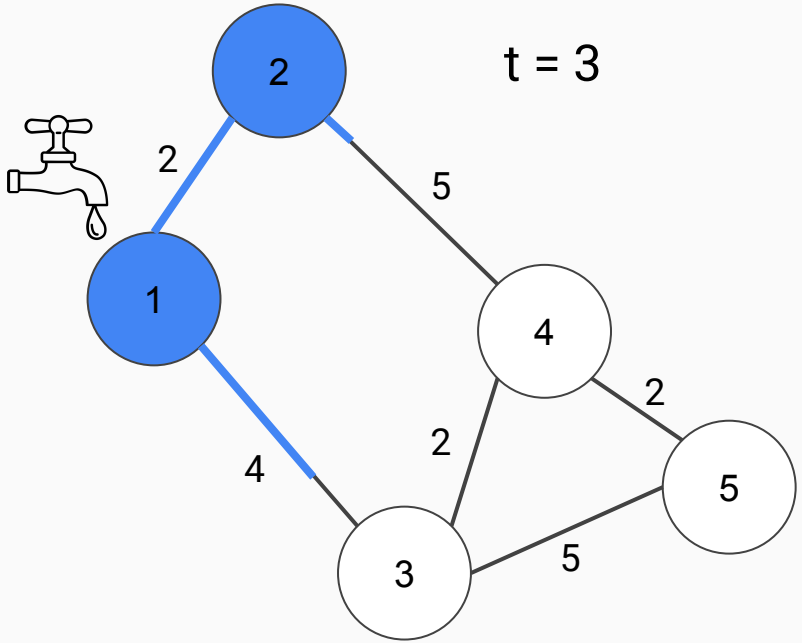
1	2	3	4	5
0				

Dijkstra



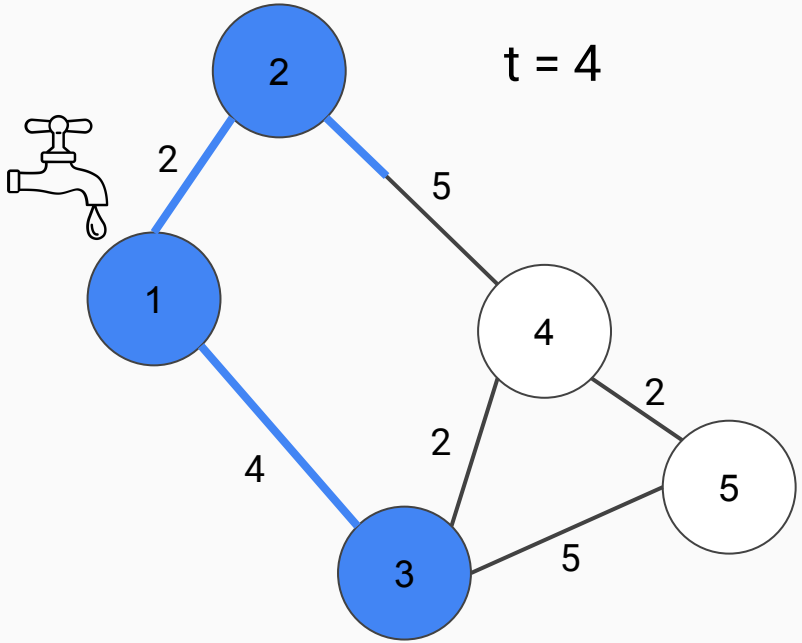
1	2	3	4	5
0	2			

Dijkstra



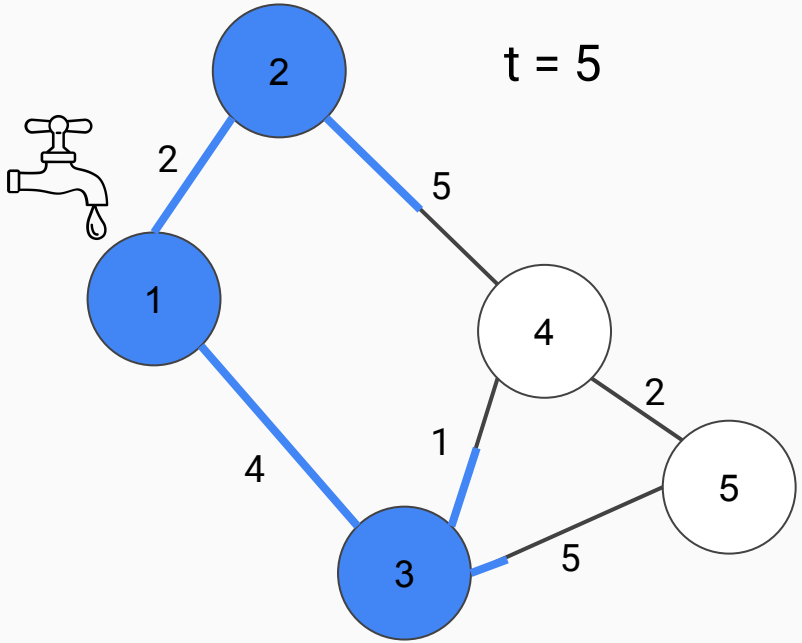
1	2	3	4	5
0	2			

Dijkstra



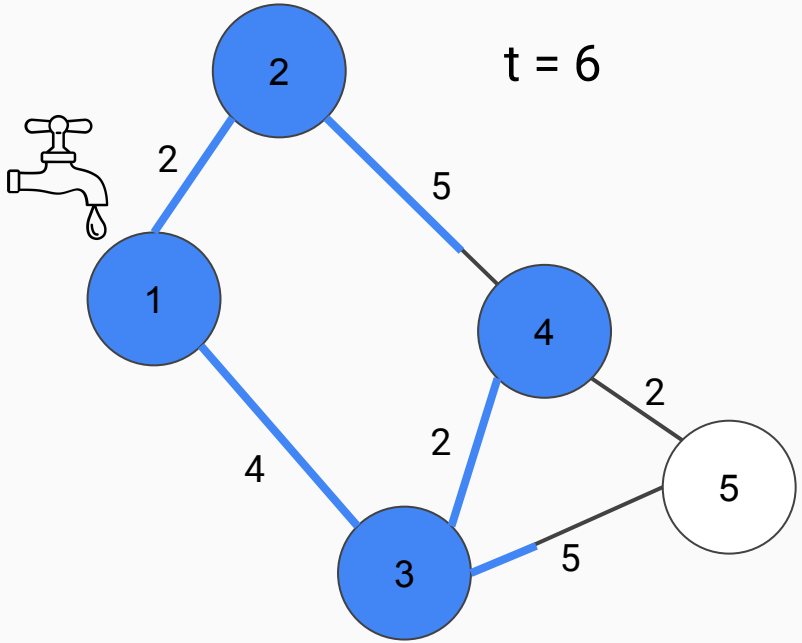
1	2	3	4	5
0	2	4		

Dijkstra



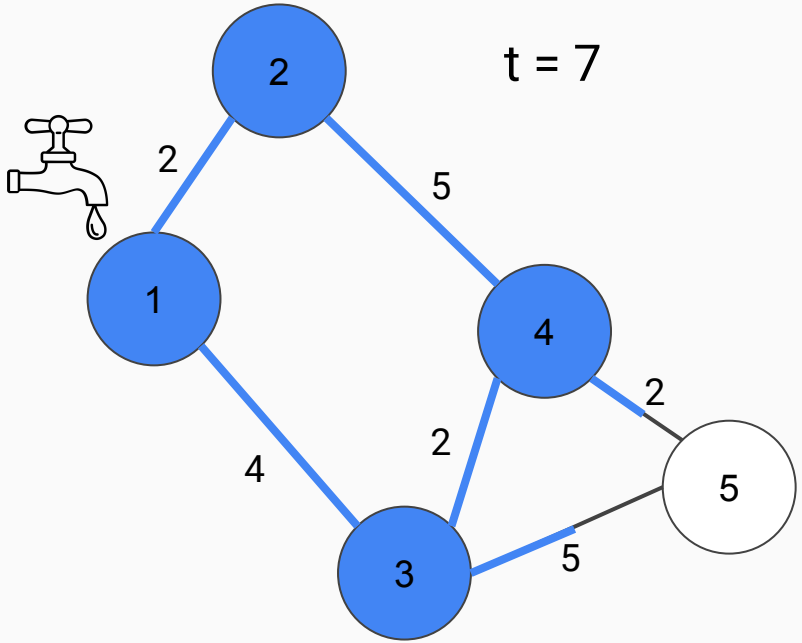
1	2	3	4	5
0	2	4		

Dijkstra



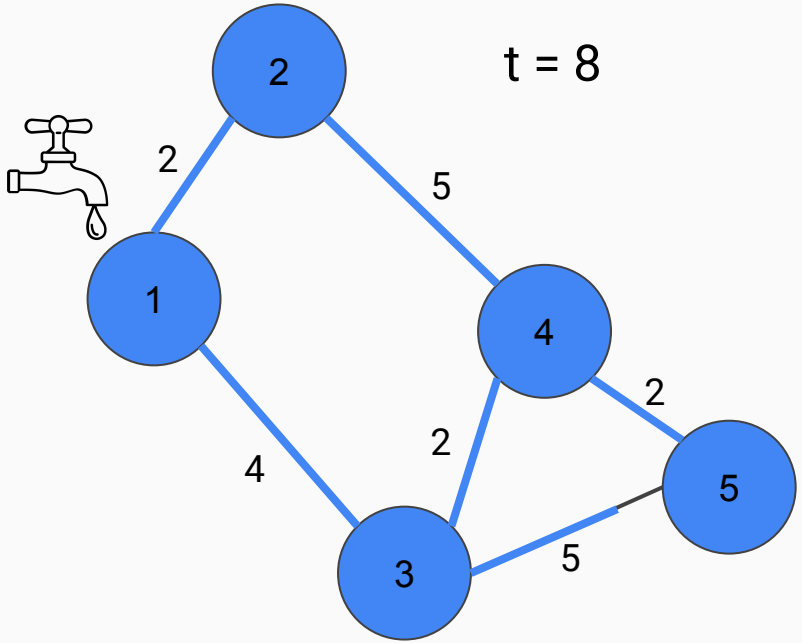
1	2	3	4	5
0	2	4	6	

Dijkstra



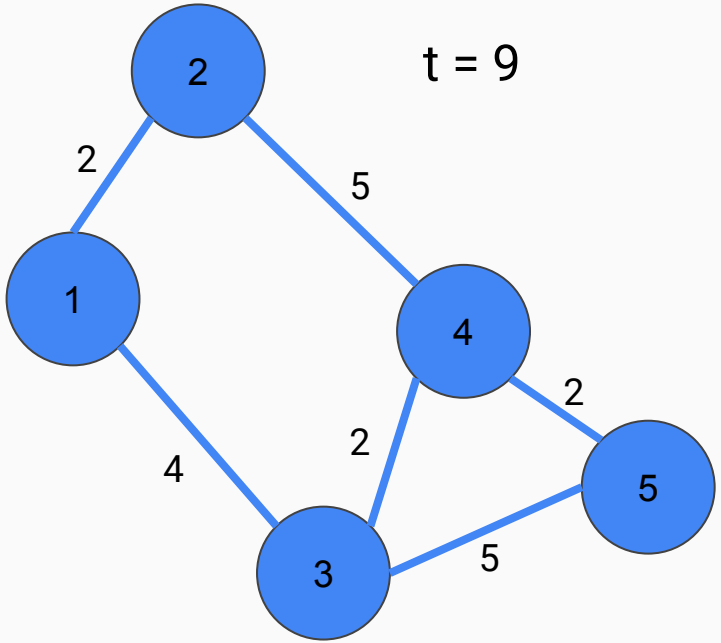
1	2	3	4	5
0	2	4	6	

Dijkstra



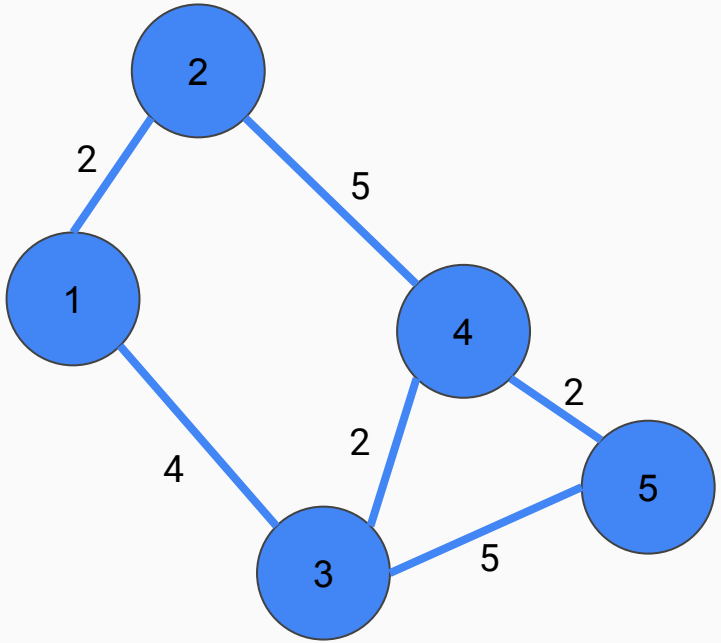
1	2	3	4	5
0	2	4	6	8

Dijkstra



1	2	3	4	5
0	2	4	6	8

Dijkstra



1	2	3	4	5
0	2	4	6	8

Vetor de distâncias

Implementação

Visualização do Algoritmo

Pathfinding Visualizer