# TEORIA DE GRAFOS E COMPUTABILIDADE

## CAMINHAMENTOS
## ALGORITMO DE DIJKSTRA

Prof. Alexei Machado

# Edsger Wybe Dijkstra (1930 – 2002)



- Algoritmos, grafos, linguagens de programação, compiladores, sistemas operacionais e distribuídos, programação concorrente...
- *A pronúncia aproximada em português para* Edsger Dijkstra *é étsrrar déikstra.*

*https://pt.wikipedia.org/wiki/Edsger_Dijkstra*

# Algoritmo de Dijkstra



- Baseado na busca em largura
- Encontra a menor distância entre dois vértices de um grafo ponderado
  - *encontra o menor caminho entre um vértice $v_i$ e todos os demais vértices do grafo*

# Algoritmo de Dijkstra



- Definir um vértice de origem $v_o$
- Utiliza um vetor de distâncias a partir de $v_o$
- Utiliza um vetor de caminhos a partir de $v_o$
- Utiliza um vetor de vértices não visitados do grafo

É um algoritmo guloso que, gera uma solução ótima em tempo polinomial

# Algoritmo de Dijkstra

TEMPO COMPUTACIONAL = O(E+VLOG(V))

V = número de vértices E = número de arestas



- Inicializações do algoritmo
  - $d[v_o] = 0$
  - Se existe $a[v_o, v_i]$, $d[v_i] = peso(v_o, v_i)$
    - Inserir $v_o$- $v_i$ no vetor de caminhos
  - Se não existe $a[v_o, v_i]$, $d[v_i] = max\_value$

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

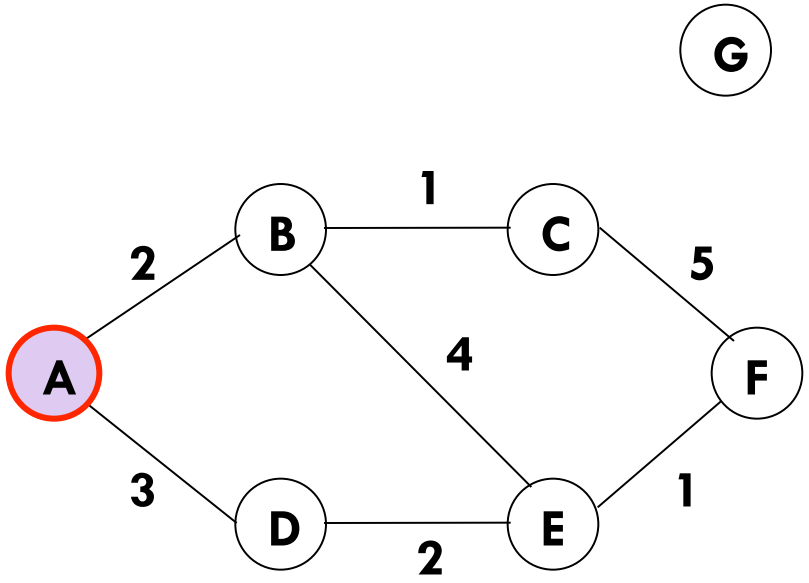
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| | | | | | | |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| | | | | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

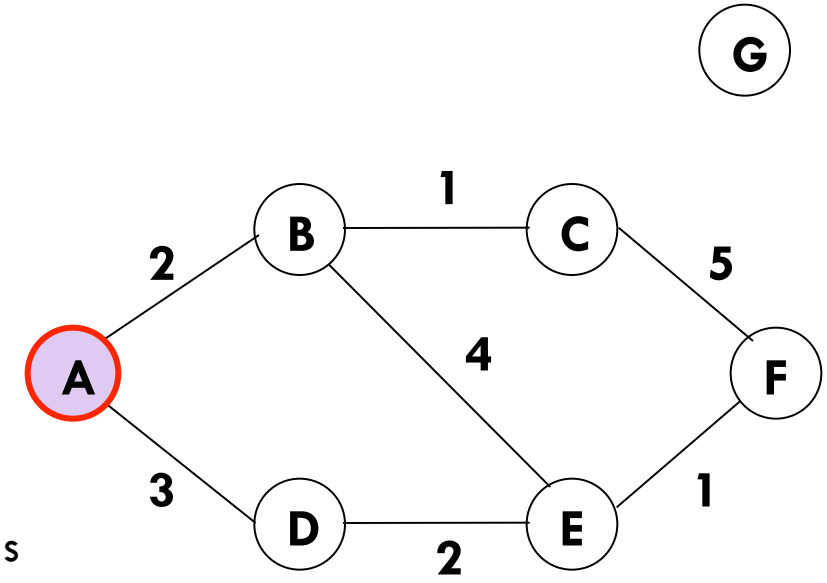
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | - | 3 | - | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | | AD | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

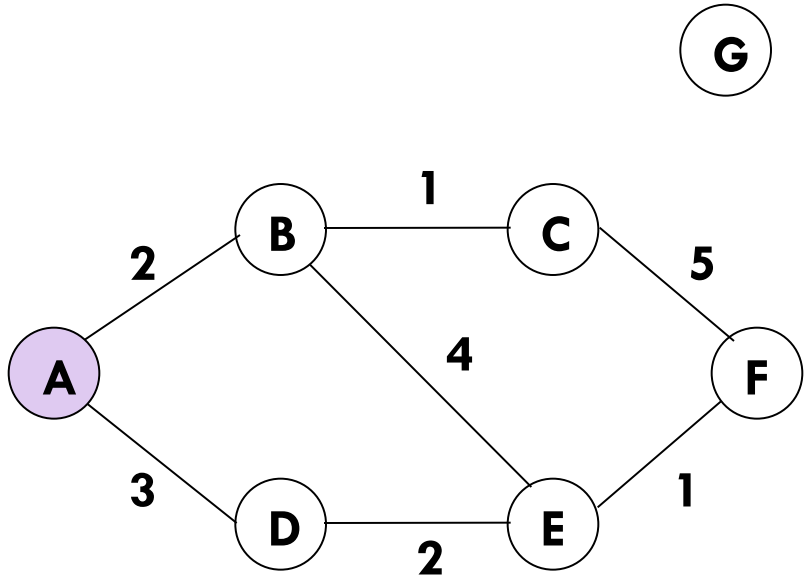
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | - | 3 | - | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | | AD | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

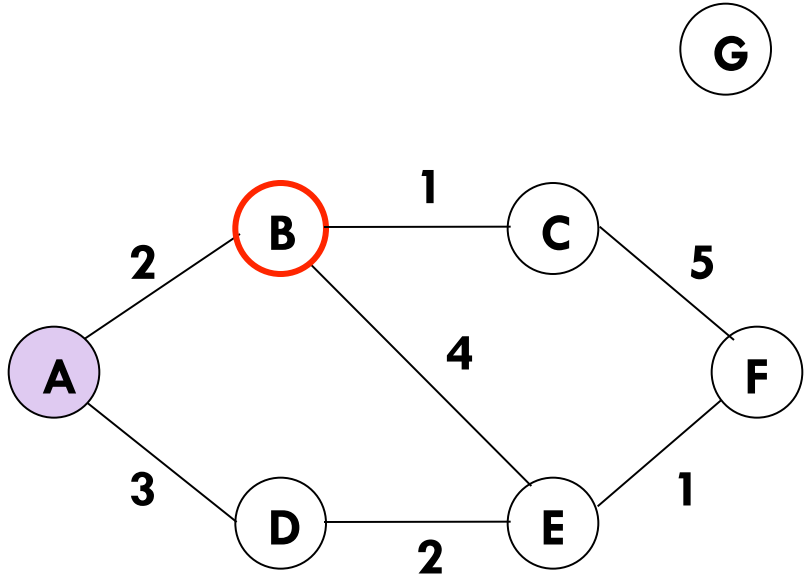
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | - | 3 | - | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | | AD | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

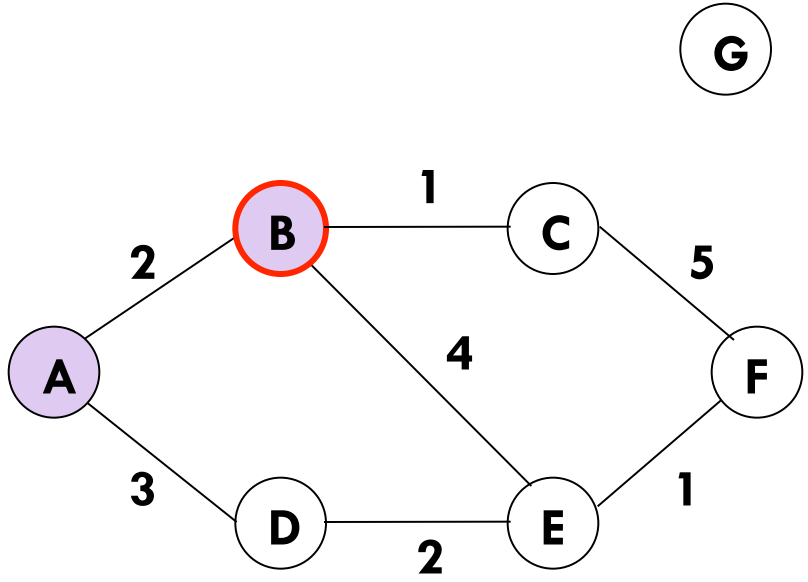
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | - | 3 | - | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | | AD | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

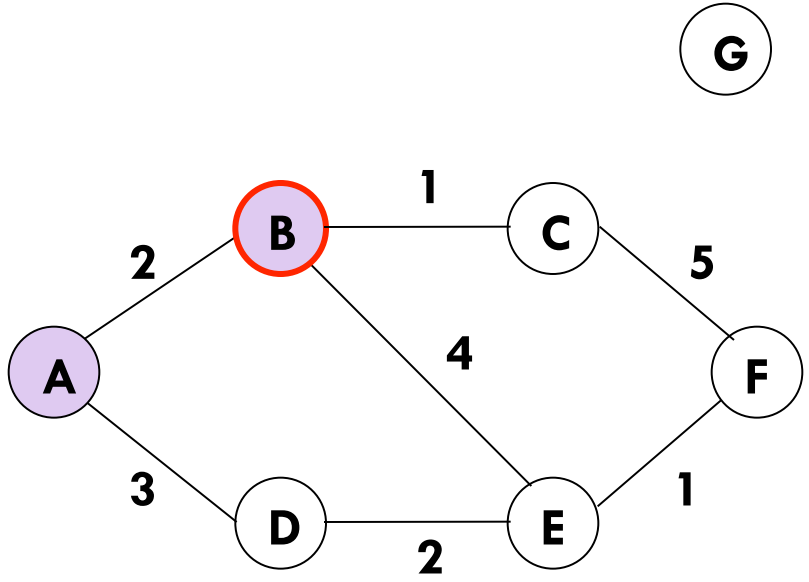
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | - | 3 | - | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | | AD | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

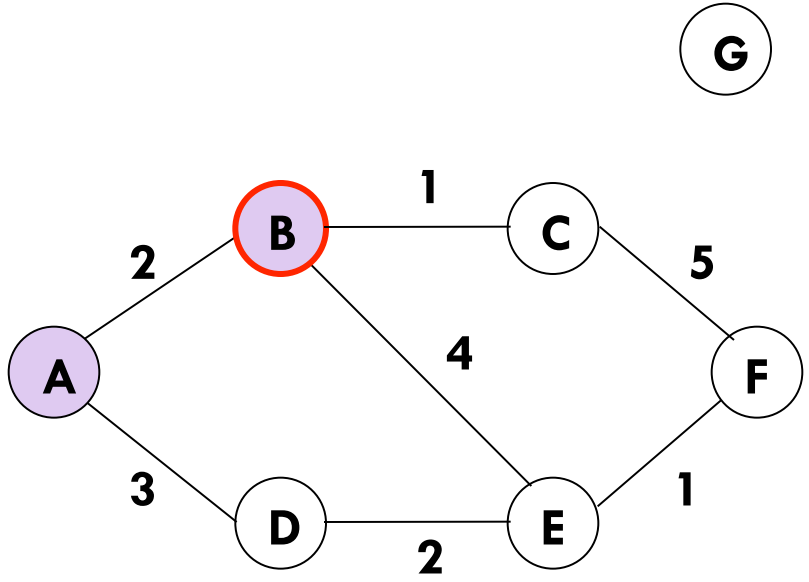
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | | | | | | |
| C | | | | | | |
| A | B | C | D | E | F | G |
| A | AB | | AD | | | |

São os vértices C e E

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

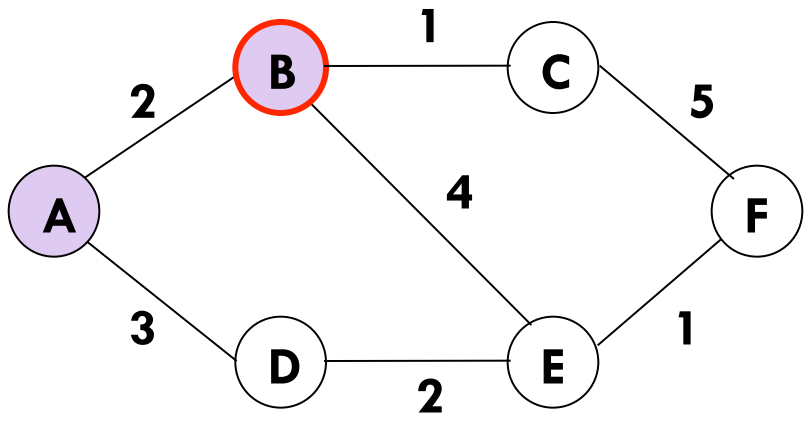
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | | | | | | |
| C | | | | | | |
| A | B | C | D | E | F | G |
| A | AB | | AD | | | |

Começando por C

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

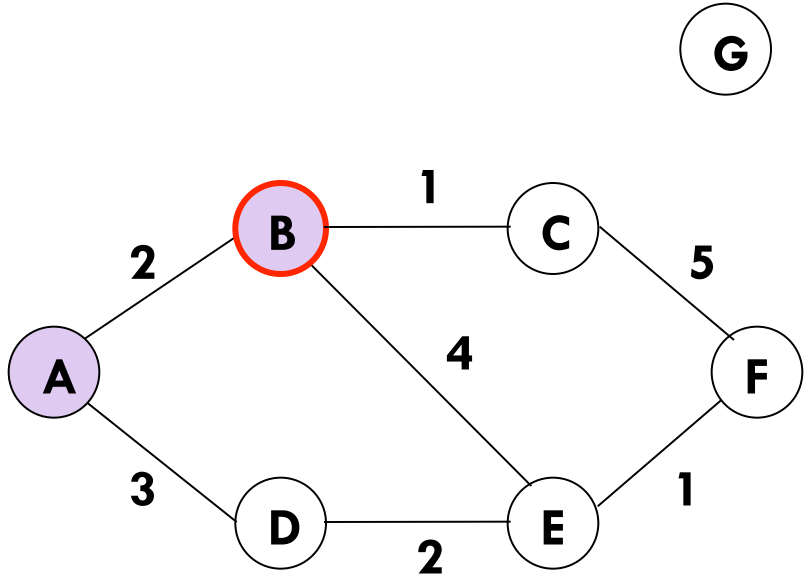
**(4)** **Para** cada vizinho i não visitado de x se d(A,B) + aresta(B,C) < d(A,C)

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | - | 3 | - | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | | AD | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

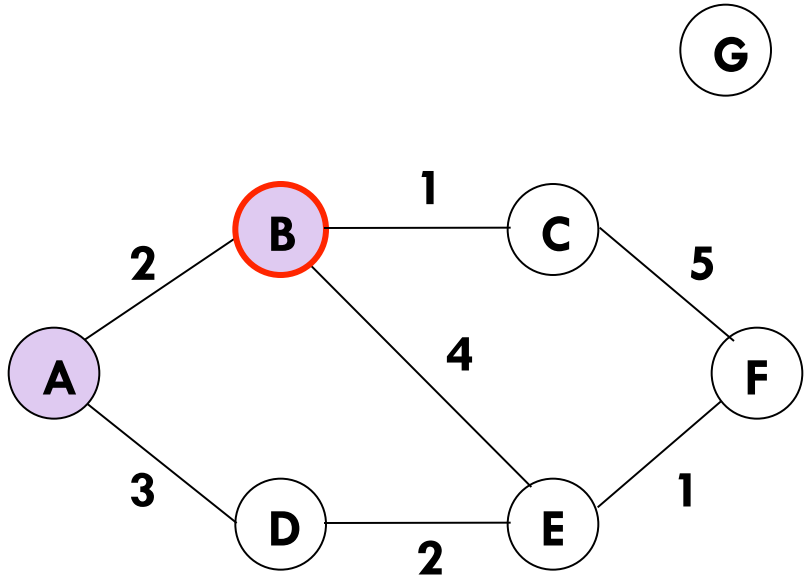
**(4)** **Para** cada vizinho i não visitado de x se d(A,B) + aresta(B,C) < d(A,C)

**faça**

(i) d(A, C) = d(A, B) + aresta(B,C)

(ii) c(C) = c(B) + C

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | - | 3 | - | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | | AD | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

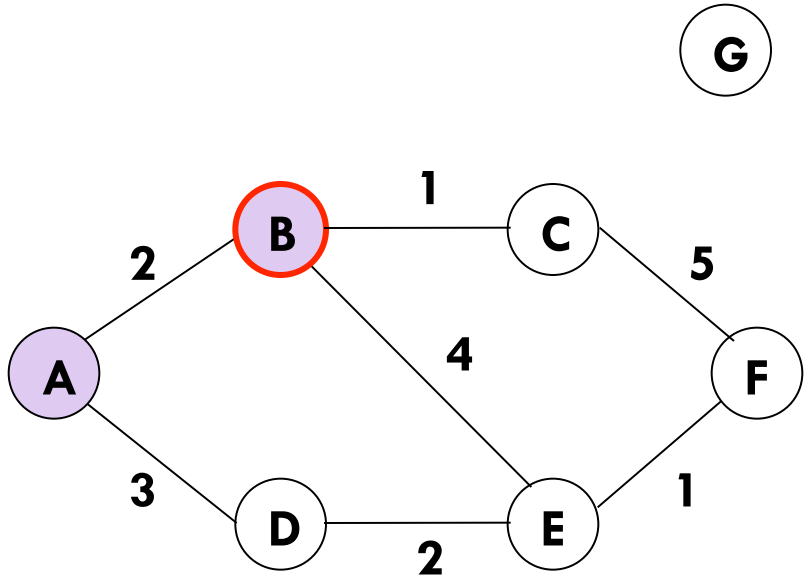
**(4)** **Para** cada vizinho i não visitado de x se d(A,B) + aresta(B,C) < d(A,C)

**faça**

(i) d(A, C) = d(A, B) + aresta(B,C)

(ii) c(C) = c(B) + C

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | - | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

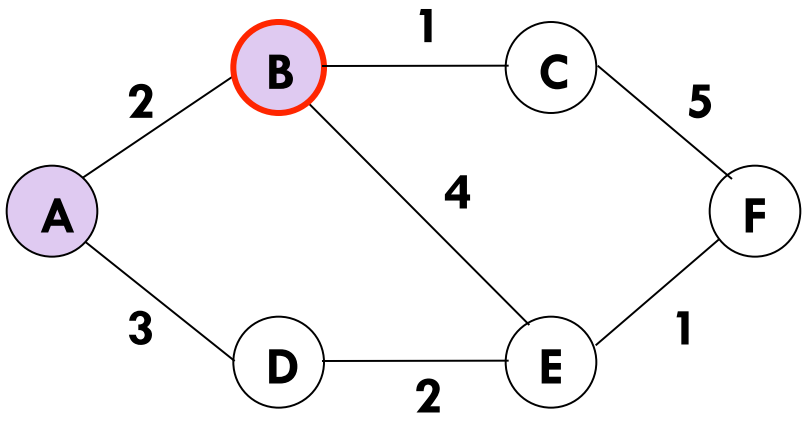
**(4)** **Para** cada vizinho i não visitado de x se d(A,B) + aresta(B,C) < d(A,C)

**faça**

(i) d(A, C) = d(A, B) + aresta(B,C)

(ii) c(C) = c(B) + C

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | | | | | | |

Agora, para E

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

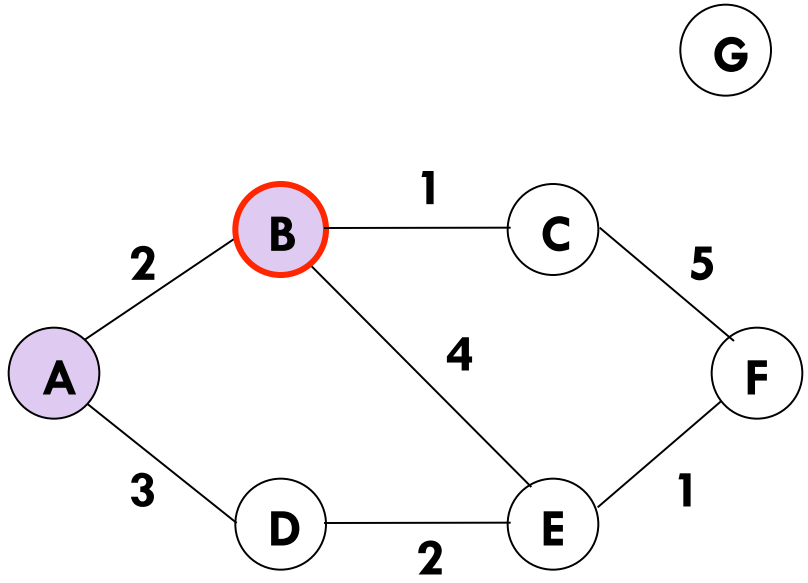
**(4)** **Para** cada vizinho i não visitado de x se d(A,B) + aresta(B,E) < d(A,E)

**faça**

(i) d(A, E) = d(A, B) + aresta(B,E)

(ii) c(E) = c(B) + E

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | - | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

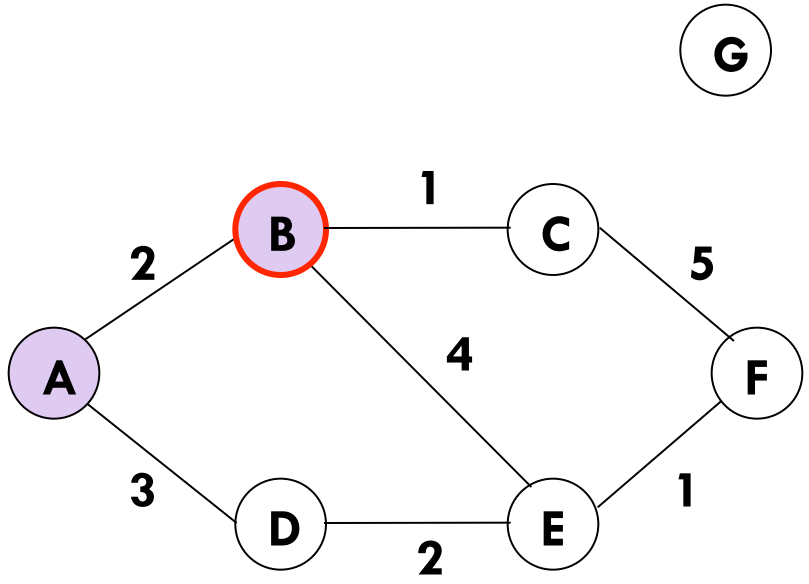
**(4)** **Para** cada vizinho i não visitado de x se d(A,B) + aresta(B,E) < d(A,E)

**faça**

(i) d(A, E) = d(A, B) + aresta(B,E)

(ii) c(E) = c(B) + E

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 6 | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ABE | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

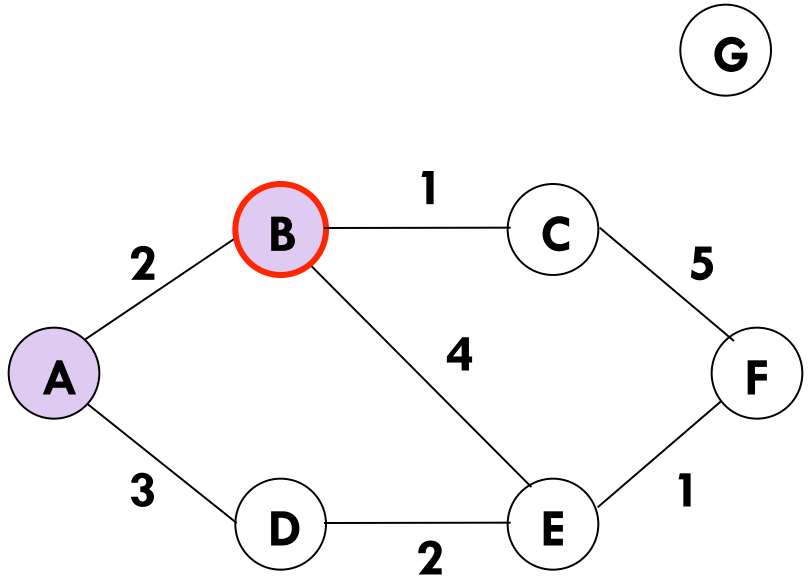
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 6 | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ABE | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

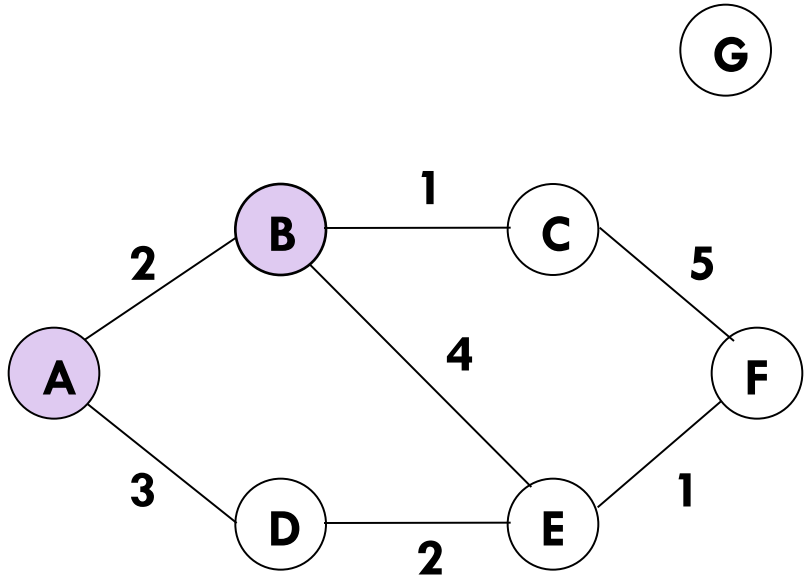
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 6 | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ABE | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

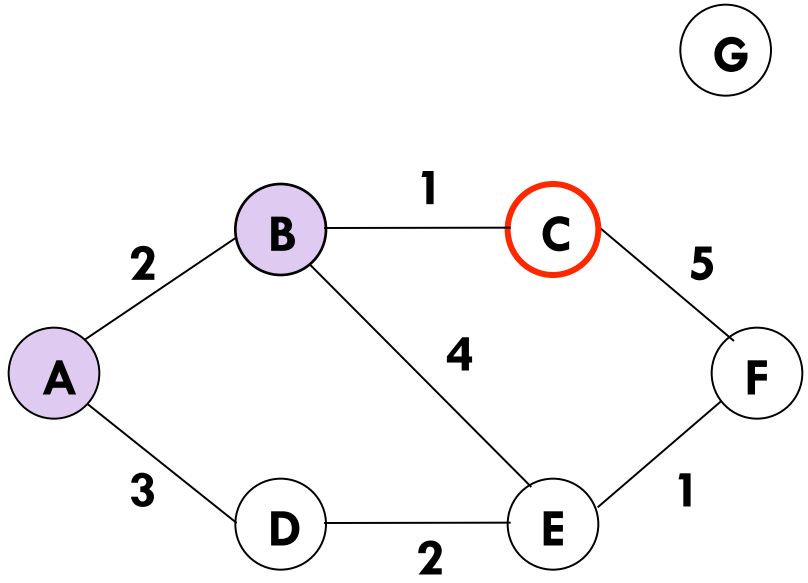
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 6 | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ABE | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

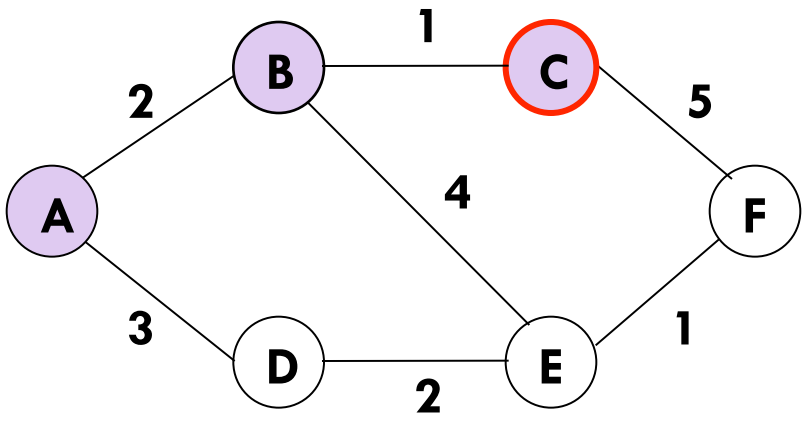
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 6 | - | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ABE | | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

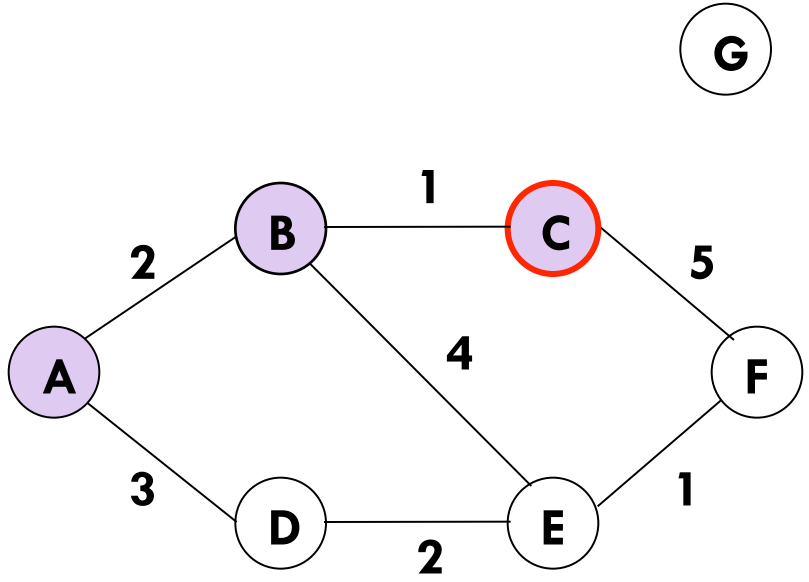
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 6 | 8 | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ABE | ABCF | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

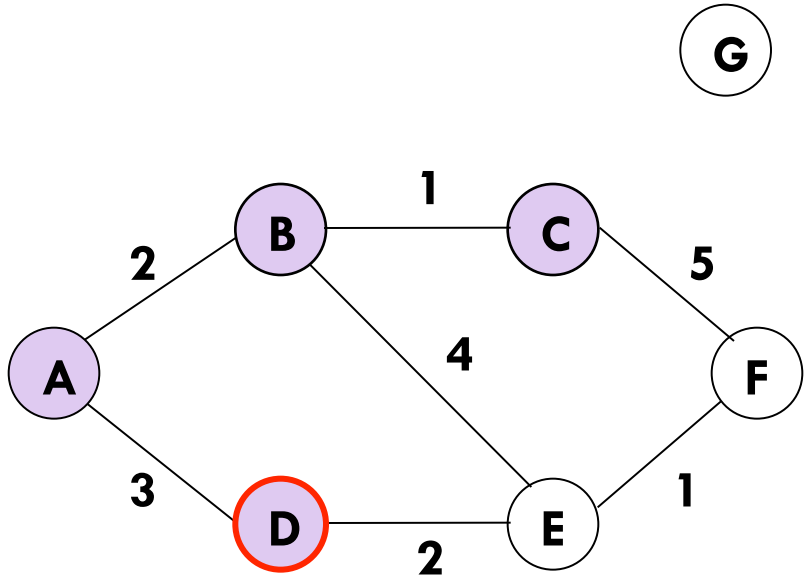
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 6 | 8 | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ABE | ABCF | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

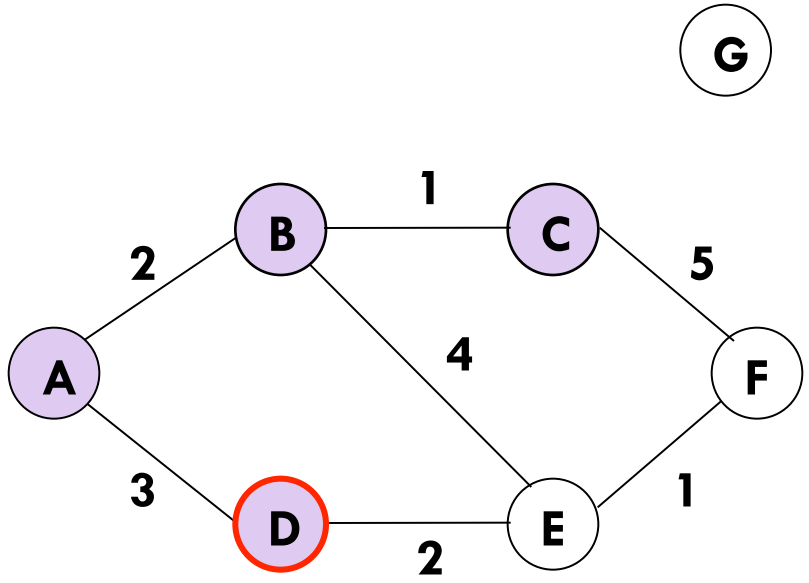
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 6 | 8 | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ABE | ABCF | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

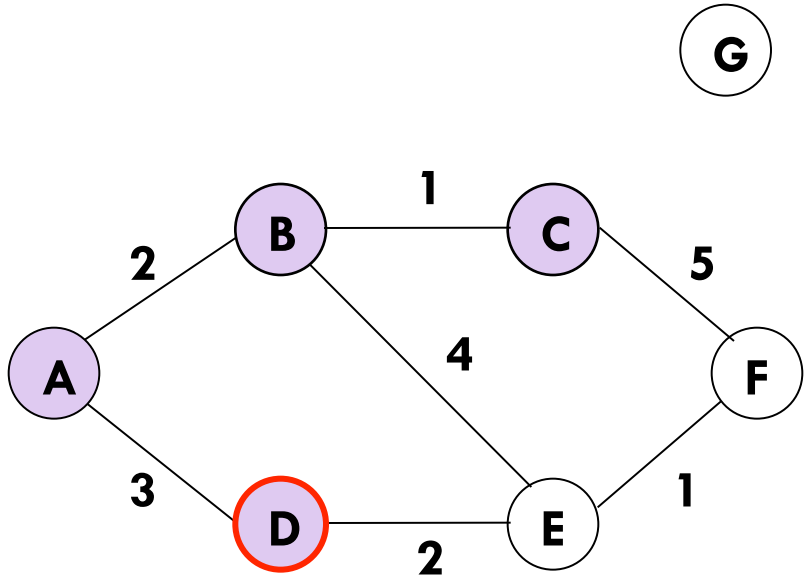
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 5 | 8 | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ADE | ABCF | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

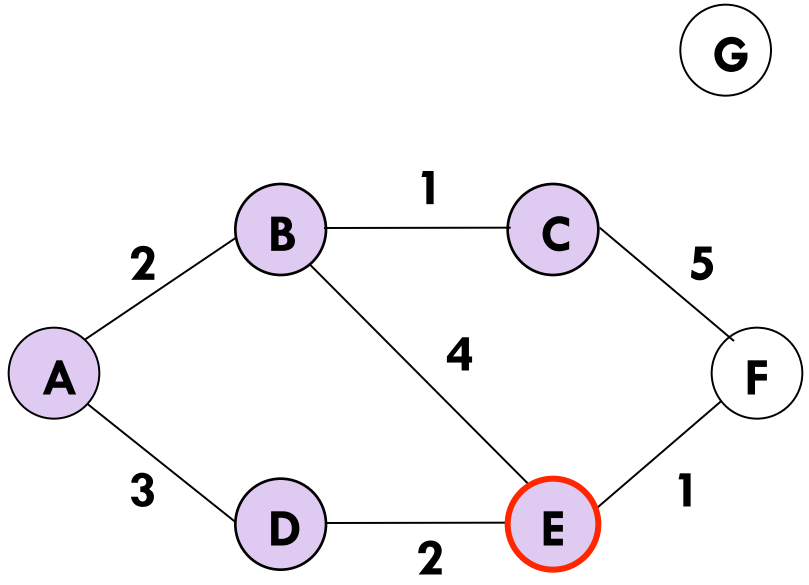
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 5 | 8 | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ADE | ABCF | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

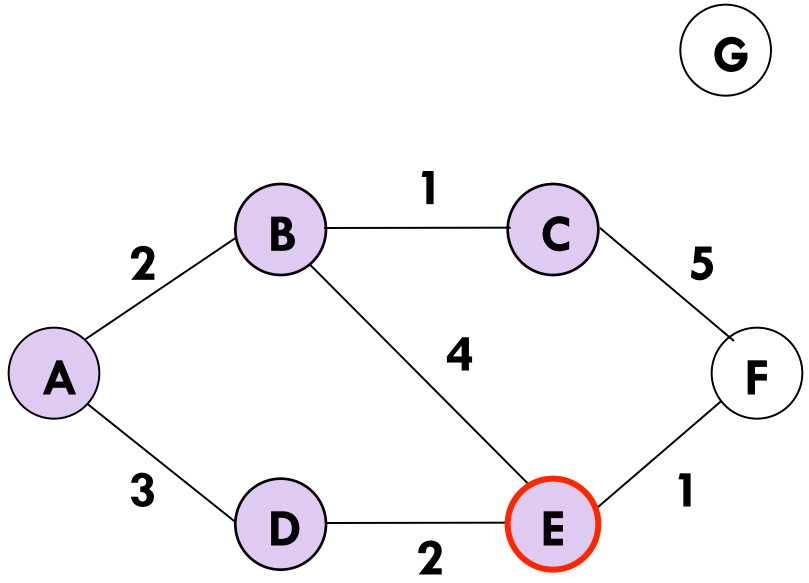
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 5 | 8 | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ADE | ABCF | |

# Algoritmo de Dijkstra

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

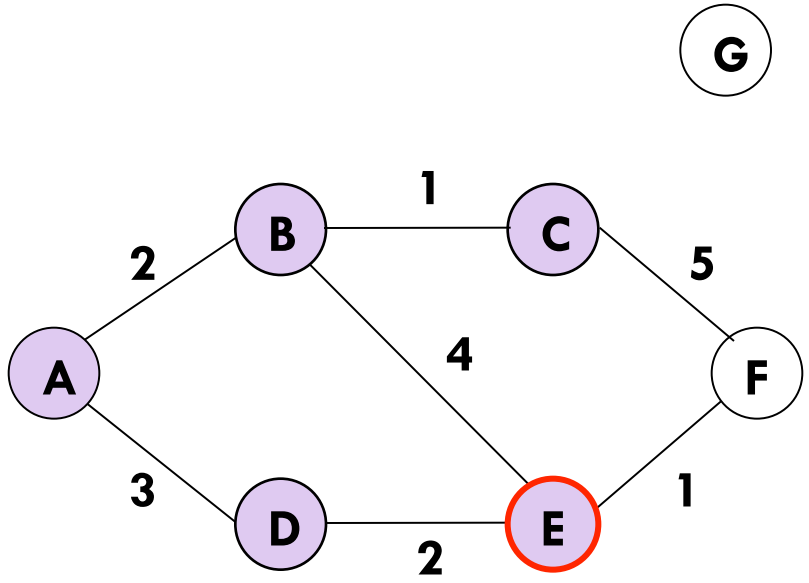
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 5 | 6 | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ADE | ADEF | |

# Algoritmo de Dijkstra

G

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

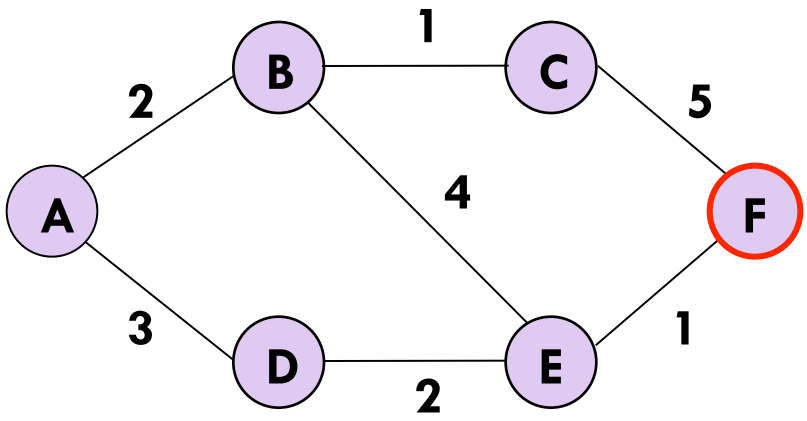
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 5 | 6 | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ADE | ADEF | |

# Algoritmo de Dijkstra

G

(1) Inicializações

(2) Escolher um vértice não visitado x cuja distância mínima para $V_0$ seja a menor conhecida. Se x for NULO, termine o algoritmo

(3) Marcar x como visitado

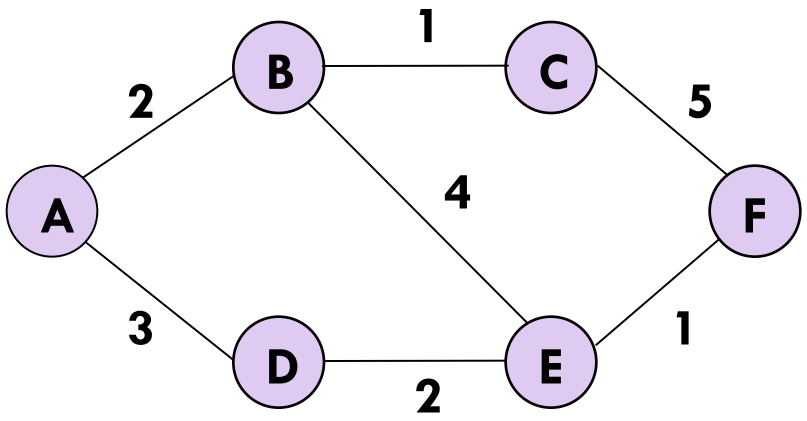
**(4)** **Para** cada vizinho i não visitado de x se $d(V_0,x) + aresta(x,i) < d(V_0, i)$

**faça**

(i) $d(V_0, i) = d(V_0, x) + aresta(x,i)$

(ii) $c(i) = c(x) + i$

(5) Se existirem vértices não visitados voltar para o passo (2)




Distâncias

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 0 | 2 | 3 | 3 | 5 | 6 | - |

Caminhos

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| A | AB | ABC | AD | ADE | ADEF | |