Teoria dos Grafos

O Problema do Caminho Mais Curto

Prof. Tiago Eugenio de Melo
tmelo@uea.edu.br

www.tiagodemelo.info

Busca em Largura

Observações

- As definições e teorias apresentadas aqui foram baseadas no livro *Fundamentos da Teoria dos Grafos para Computação* (LTC, 2018).
- A numeração das definições e teoremas seguem as mesmas referências adotadas no livro para facilitar a localização.

Caminho Mais Curto

Caminho Mais Curto

- Algoritmo da Busca em Largura

Caminho Mais Curto

- Algoritmo da Busca em Largura
  - Seja G um grafo e sejam v1 e v2 dois vértices específicos de G.

Caminho Mais Curto

- Algoritmo da Busca em Largura
  - Seja G um grafo e sejam v1 e v2 dois vértices específicos de G.
  - O objetivo do algoritmo é encontrar o comprimento de um caminho (se existir) entre os vértices v1 e v2, que usa o **menor número de arestas**.

Caminho Mais Curto

- Algoritmo da Busca em Largura
  - Seja G um grafo e sejam v1 e v2 dois vértices específicos de G.
  - O objetivo do algoritmo é encontrar o comprimento de um caminho (se existir) entre os vértices v1 e v2, que usa o **menor número de arestas**.
  - Se existir, esse caminho é chamado de caminho mais curto.

Algoritmo de Busca em Largura

Algoritmo de Busca em Largura

- Entrada: Grafo G e dois vértices, s (início) e t (fim).

Algoritmo de Busca em Largura

- Entrada: Grafo G e dois vértices, s (início) e t (fim).
- Saída: Comprimento do caminho mais curto (em número de arestas) entre s e t.

Algoritmo de Busca em Largura

- Entrada: Grafo G e dois vértices, s (início) e t (fim).
- Saída: Comprimento do caminho mais curto (em número de arestas) entre s e t.
- Passo 1. $i \leftarrow 0$; $\lambda(s) \leftarrow 0$. {λ(vértice) representa um rótulo associado ao vértice}

Algoritmo de Busca em Largura

- Entrada: Grafo G e dois vértices, s (início) e t (fim).
- Saída: Comprimento do caminho mais curto (em número de arestas) entre s e t.
- Passo 1. $i \leftarrow 0$; $\lambda(s) \leftarrow 0$. {λ(vértice) representa um rótulo associado ao vértice}
- Passo 2. Encontre todos os vértices não rotulados em G que são adjacentes a vértices rotulados i. Se não existirem tais vértices, então t não é conectado a s (por um caminho). Se existirem tais vértices, rotule-os i + 1.

Algoritmo de Busca em Largura

- Entrada: Grafo G e dois vértices, s (início) e t (fim).
- Saída: Comprimento do caminho mais curto (em número de arestas) entre s e t.
- Passo 1. $i \leftarrow 0$; $\lambda(s) \leftarrow 0$. {λ(vértice) representa um rótulo associado ao vértice}
- Passo 2. Encontre todos os vértices não rotulados em G que são adjacentes a vértices rotulados i. Se não existirem tais vértices, então t não é conectado a s (por um caminho). Se existirem tais vértices, rotule-os i + 1.
- Passo 3. Se t for rotulado, vá para o Passo 4, caso contrário $i \leftarrow i+1$ e vá para o Passo 2.

Algoritmo de Busca em Largura

- Entrada: Grafo G e dois vértices, s (início) e t (fim).
- Saída: Comprimento do caminho mais curto (em número de arestas) entre s e t.
- Passo 1. $i \leftarrow 0$; $\lambda(s) \leftarrow 0$. {λ(vértice) representa um rótulo associado ao vértice}
- Passo 2. Encontre todos os vértices não rotulados em G que são adjacentes a vértices rotulados i. Se não existirem tais vértices, então t não é conectado a s (por um caminho). Se existirem tais vértices, rotule-os i + 1.
- Passo 3. Se t for rotulado, vá para o Passo 4, caso contrário $i \leftarrow i+1$ e vá para o Passo 2.
- Passo 4. O comprimento do caminho mais curto de s a t é i+1. Pare e retorne i+1.

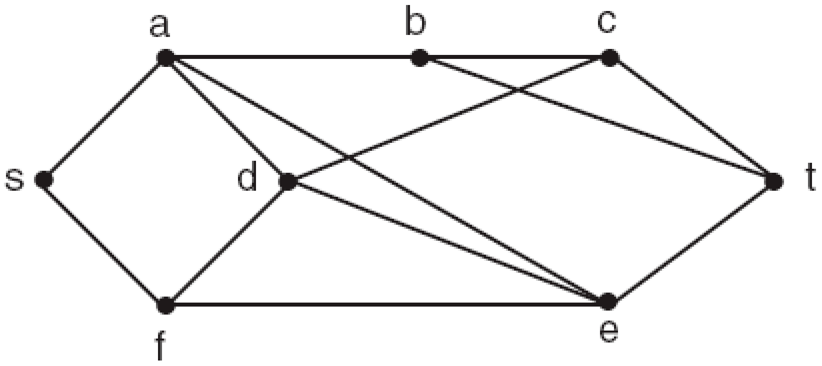
Exemplo 1 - Busca em Largura

Exemplo 1 - Busca em Largura

- Caminho s → t

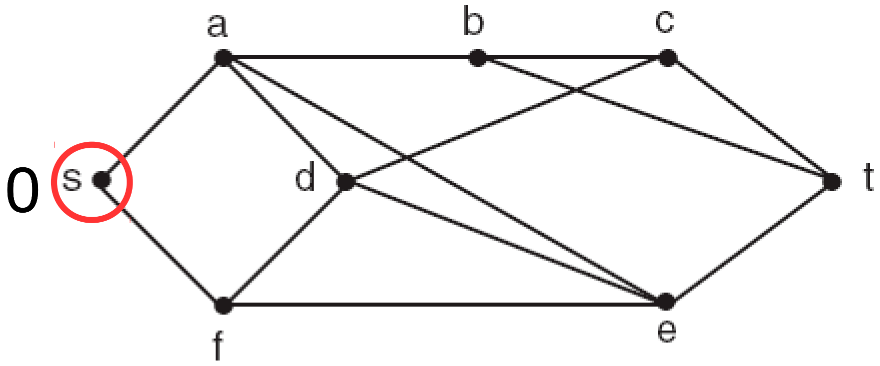
Exemplo 1 - Busca em Largura

- Caminho s → t



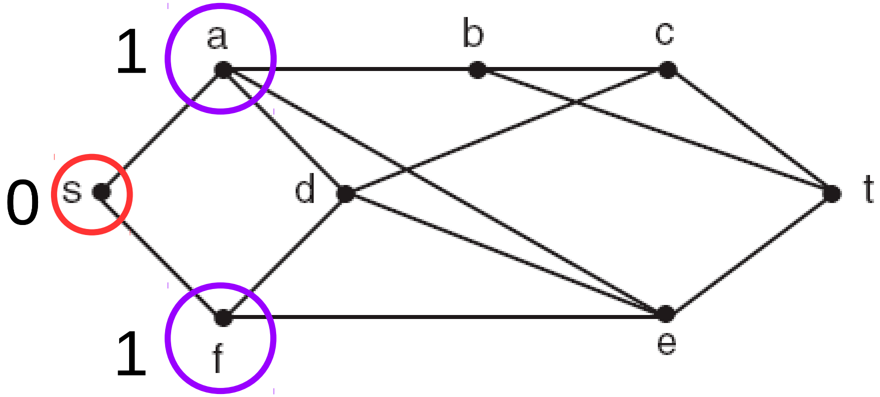
Exemplo 1 - Busca em Largura

- Caminho s → t



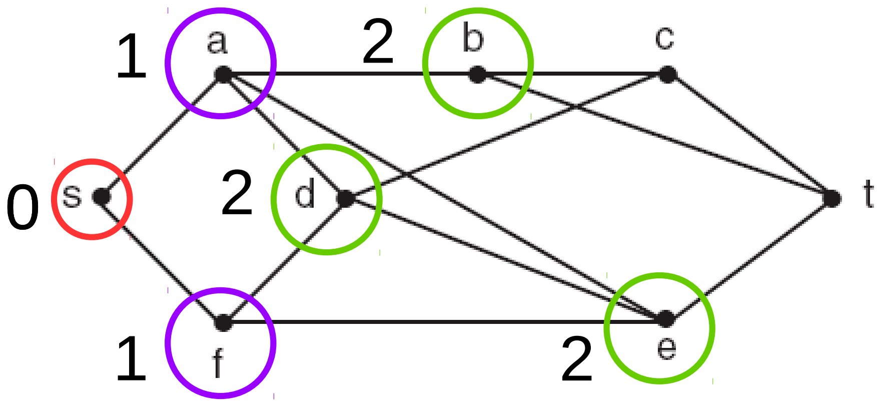
Exemplo 1 - Busca em Largura

- Caminho s → t



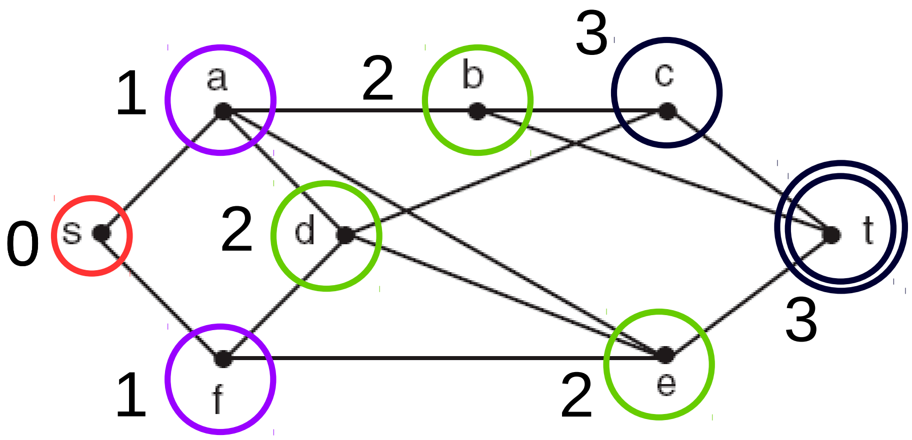
Exemplo 1 - Busca em Largura

- Caminho s → t



Exemplo 1 - Busca em Largura

- Caminho s → t



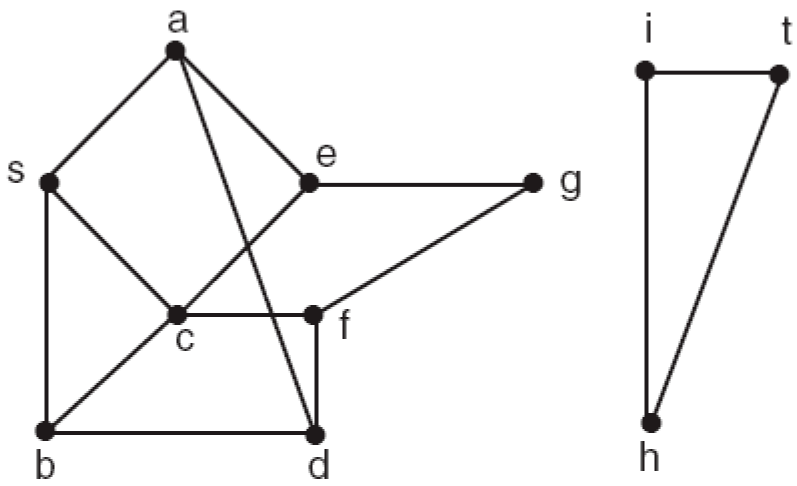
Exemplo 2 - Busca em Largura

Exemplo 2 - Busca em Largura

- Caminho s → t

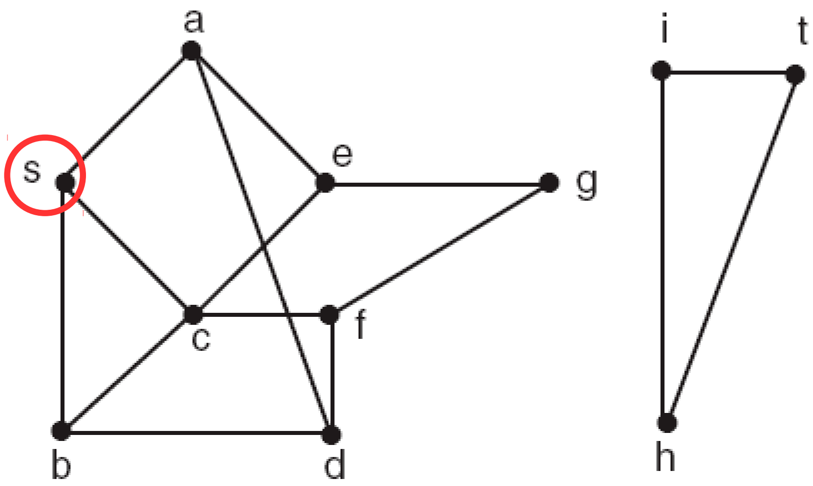
Exemplo 2 - Busca em Largura

- Caminho s → t



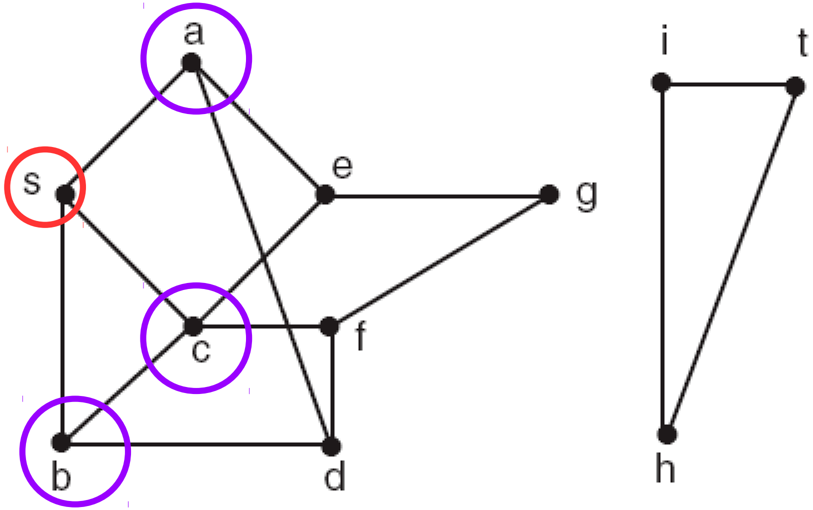
Exemplo 2 - Busca em Largura

- Caminho s → t



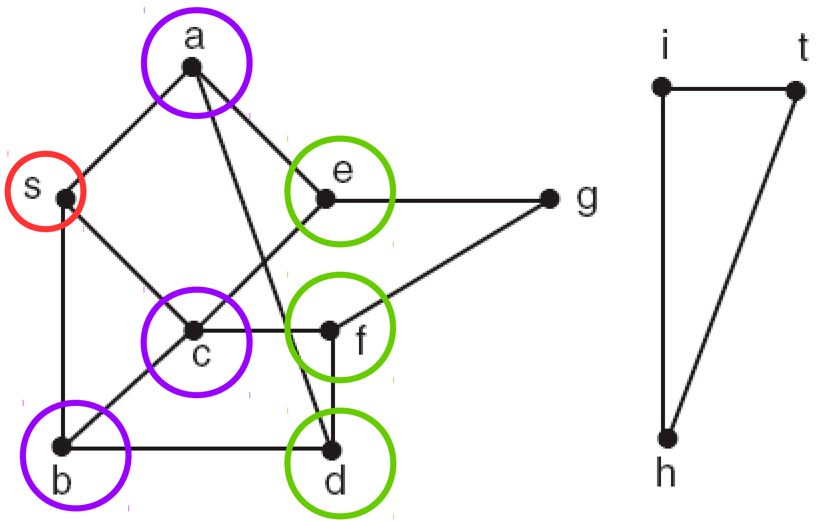
Exemplo 2 - Busca em Largura

- Caminho s → t



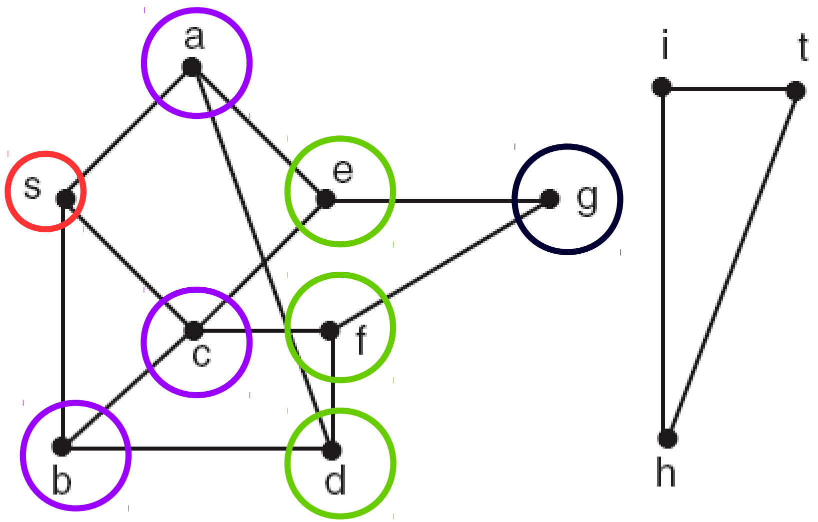
Exemplo 2 - Busca em Largura

- Caminho s → t



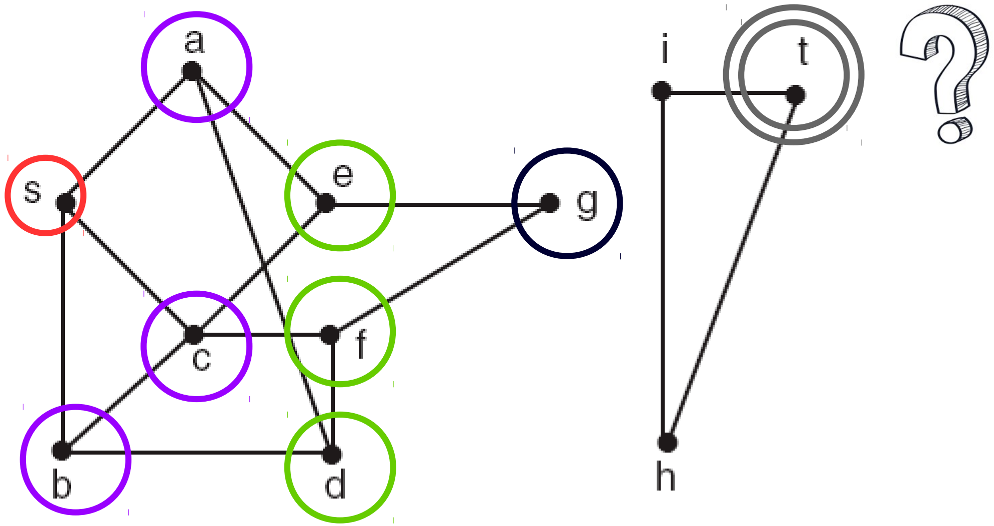
Exemplo 2 - Busca em Largura

- Caminho s → t



Exemplo 2 - Busca em Largura

- Caminho s → t



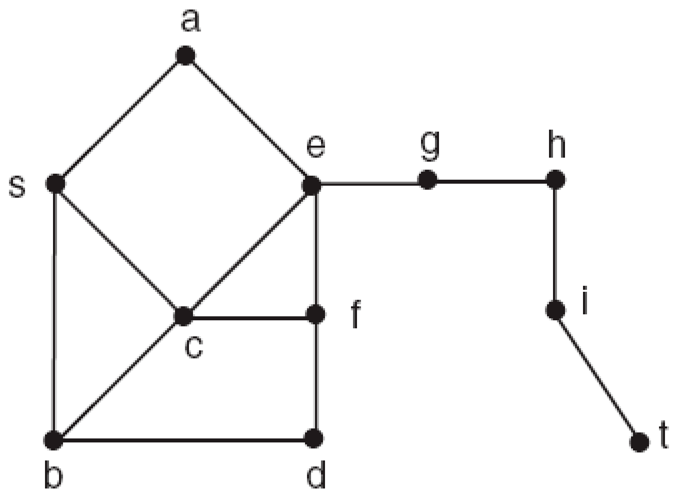
Exemplo 3 - Busca em Largura

Exemplo 3 - Busca em Largura

- Caminho s → t

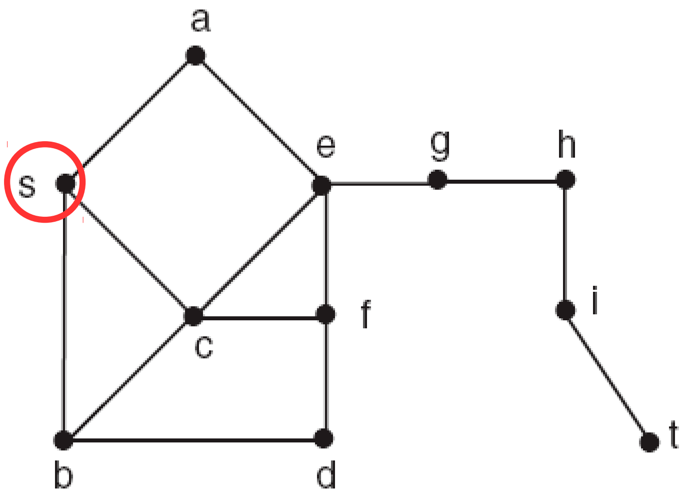
Exemplo 3 - Busca em Largura

- Caminho s → t



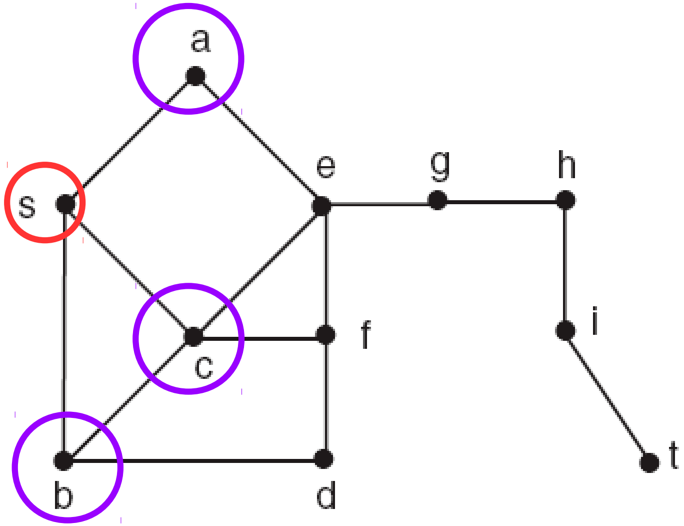
Exemplo 3 - Busca em Largura

- Caminho s → t



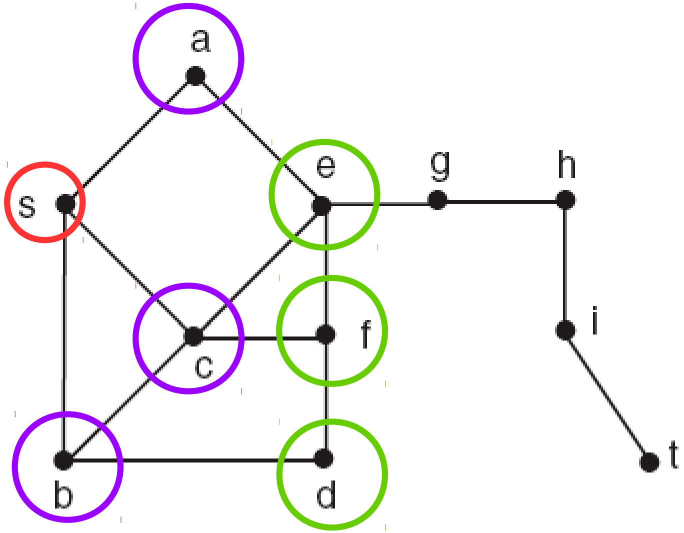
Exemplo 3 - Busca em Largura

- Caminho s → t



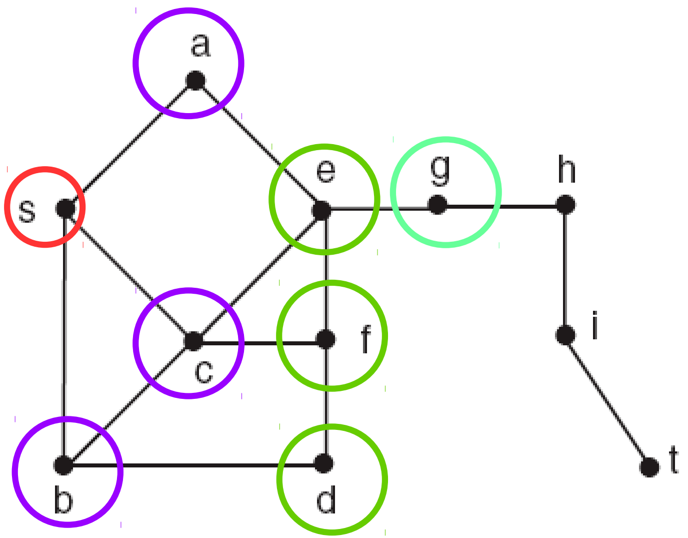
Exemplo 3 - Busca em Largura

- Caminho s → t



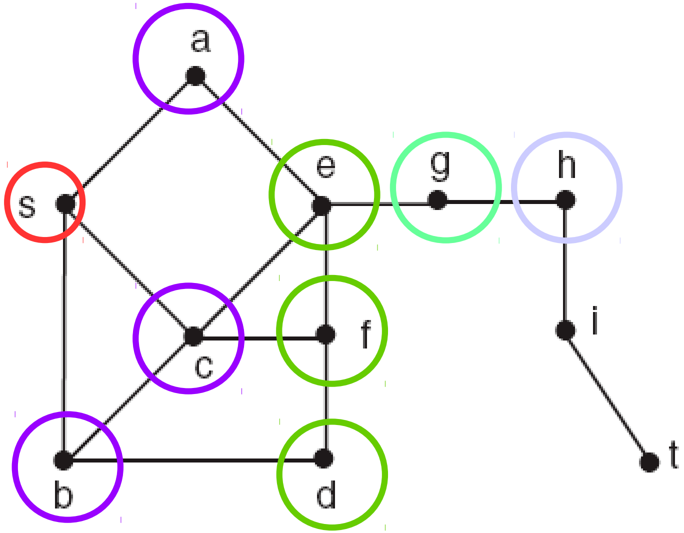
Exemplo 3 - Busca em Largura

- Caminho s → t



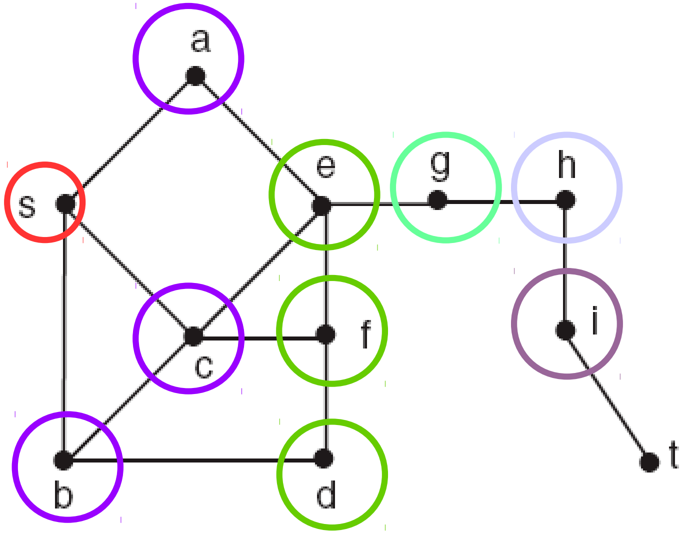
Exemplo 3 - Busca em Largura

- Caminho s → t



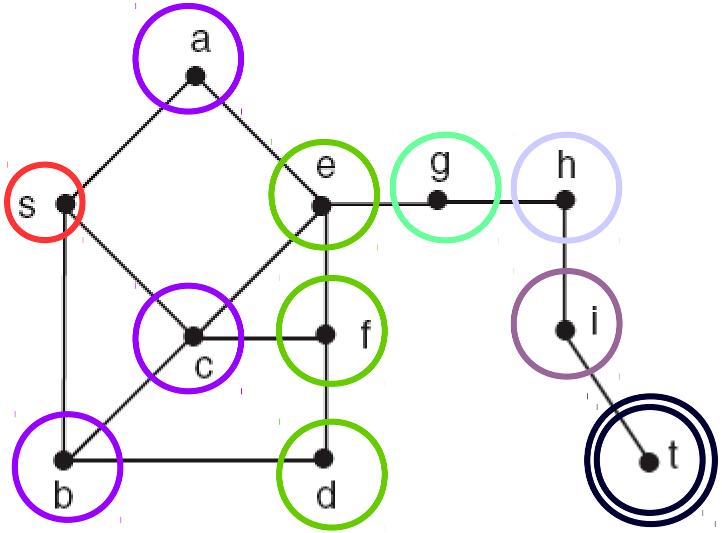
Exemplo 3 - Busca em Largura

- Caminho s → t



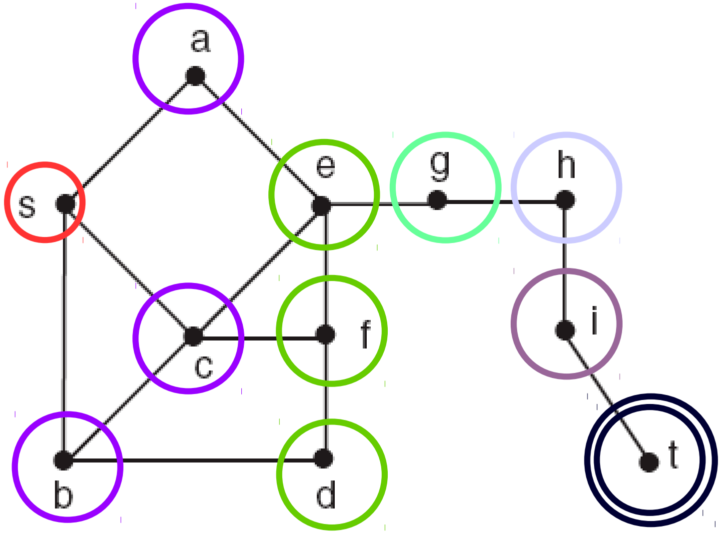
Exemplo 3 - Busca em Largura

- Caminho s → t



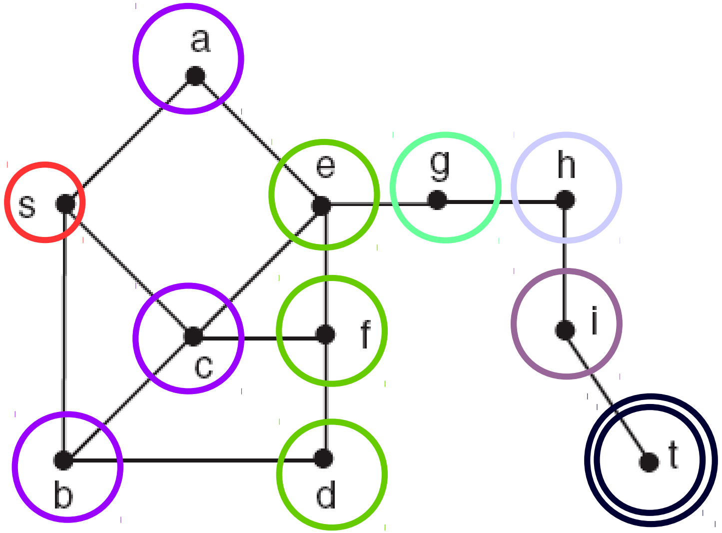
Exemplo 3 - Busca em Largura

- Caminho s → t



Exemplo 3 - Busca em Largura

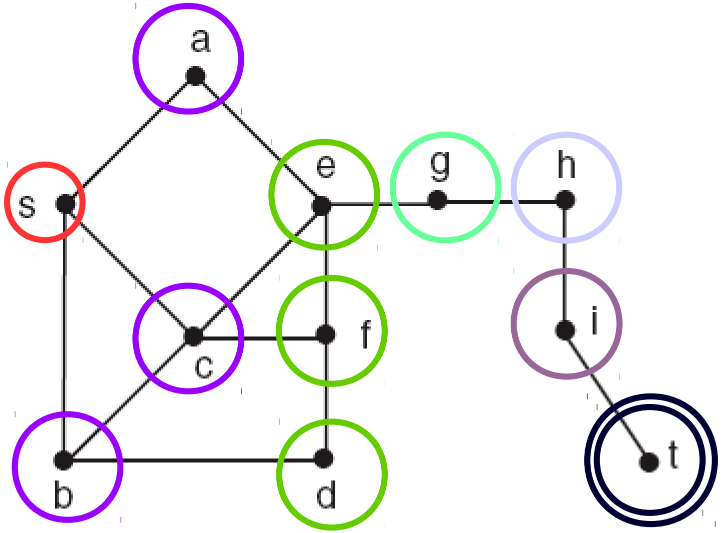
- Caminho s → t



Como saber o caminho de volta?

Exemplo 3 - Busca em Largura

- Caminho s → t

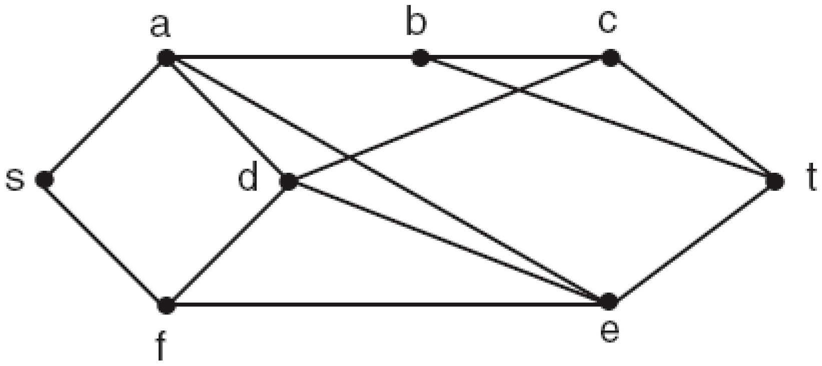


Como saber o caminho de volta?

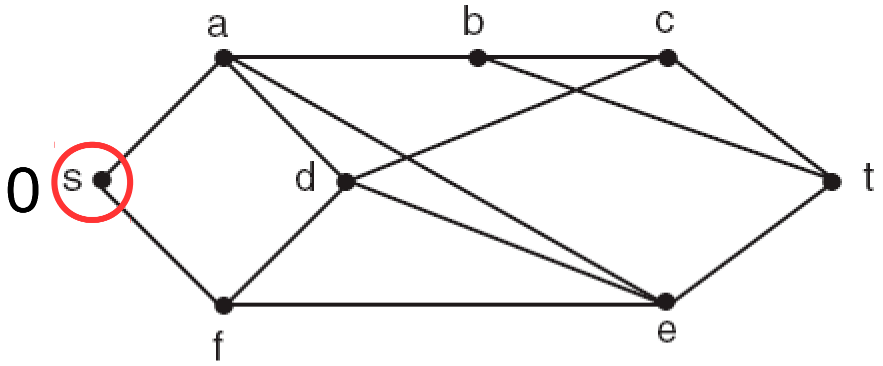
BACKTRACKING!

Algoritmo de *Backtracking*

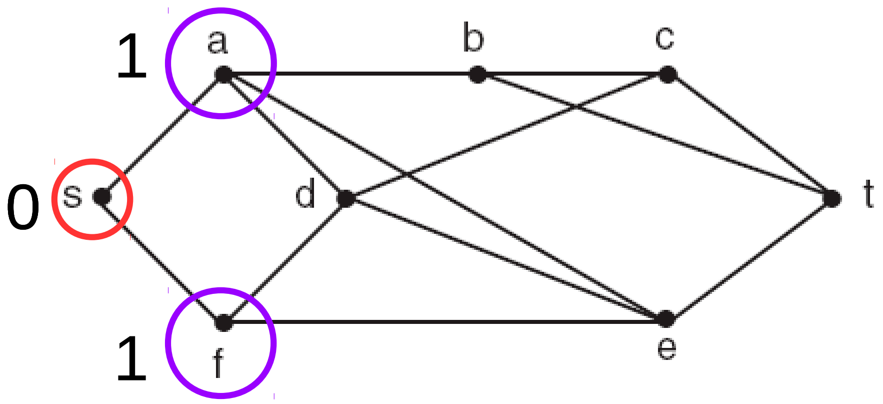
Algoritmo de *Backtracking*



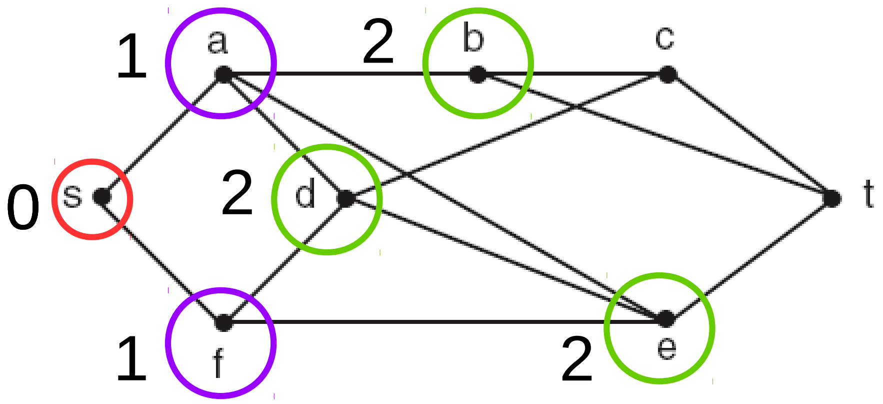
Algoritmo de *Backtracking*



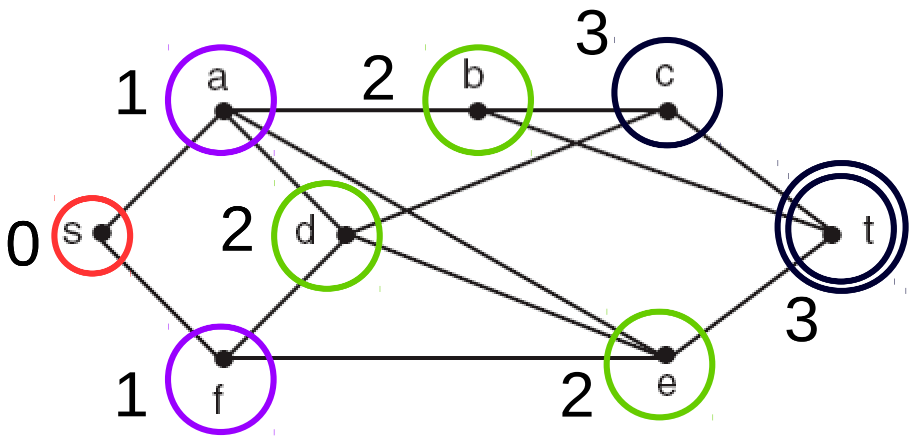
Algoritmo de *Backtracking*



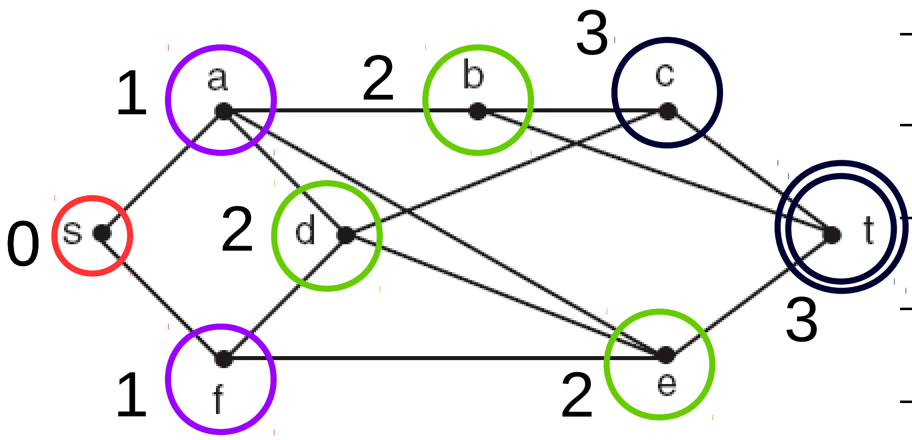
Algoritmo de *Backtracking*



Algoritmo de *Backtracking*



Algoritmo de *Backtracking*



vértice	λ(vértice)
t	3
c	3
e	2
b	2
d	2
f	1
a	1
s	0

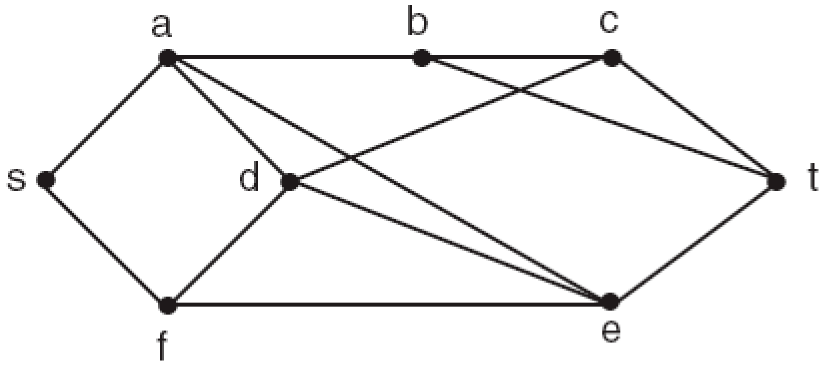
Trace Backtracking

Trace Backtracking

- **Passo 1**

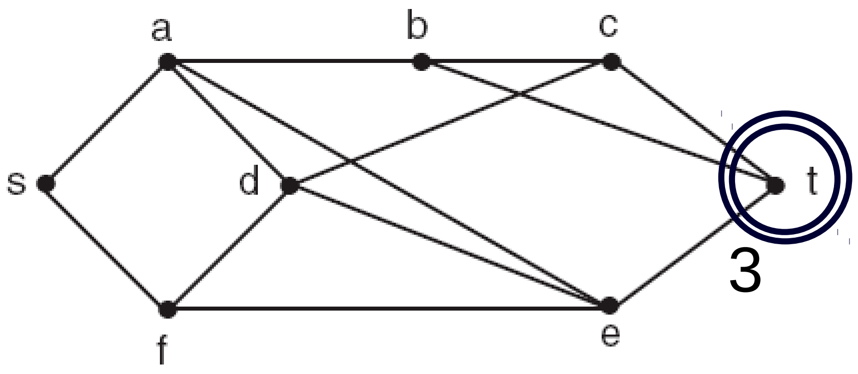
Trace Backtracking

- **Passo 1**



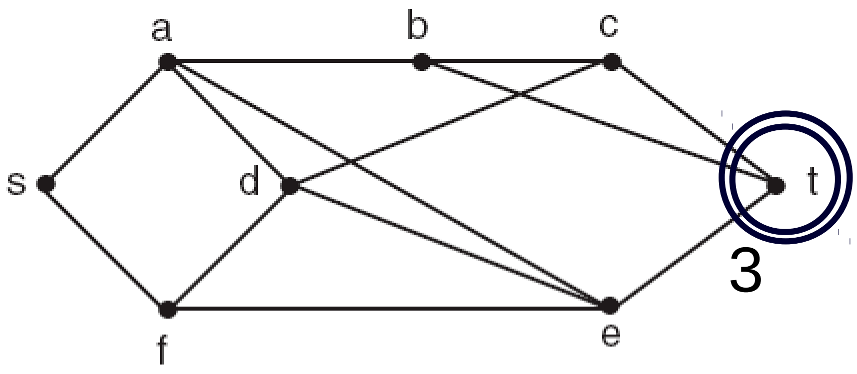
Trace Backtracking

- **Passo 1**



Trace Backtracking

- **Passo 1**



i	v_i
3	t

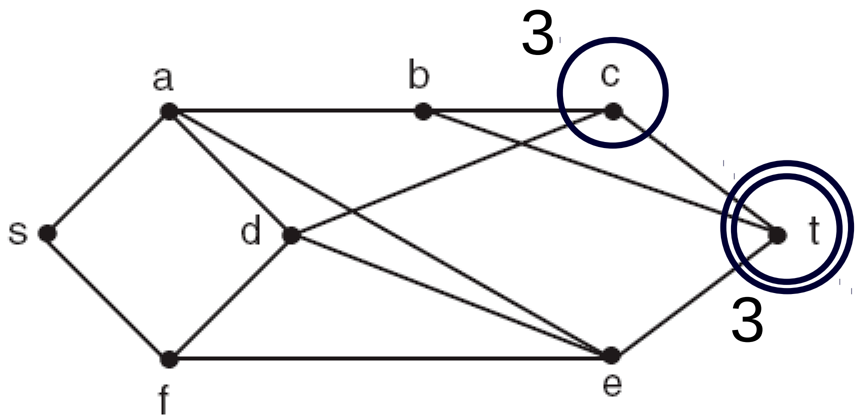
Trace Backtracking

Trace Backtracking

- **Passo 2**

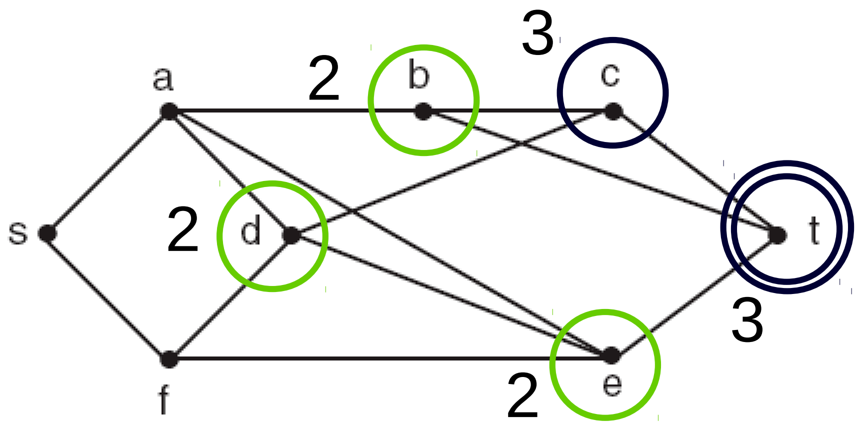
Trace Backtracking

- **Passo 2**



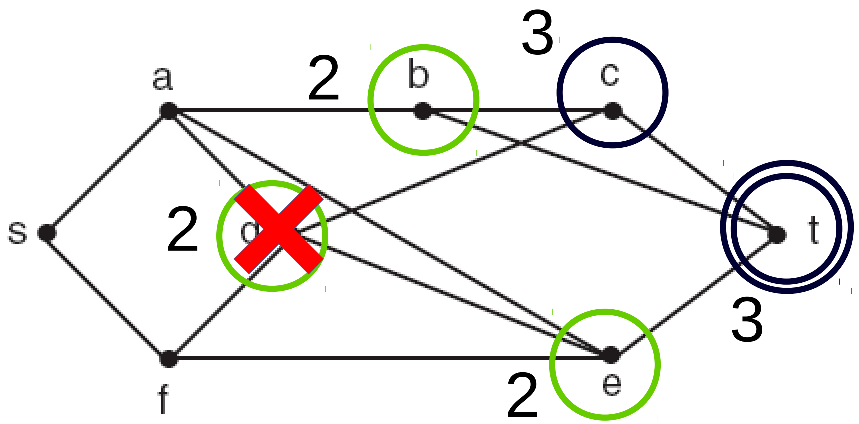
Trace Backtracking

- **Passo 2**



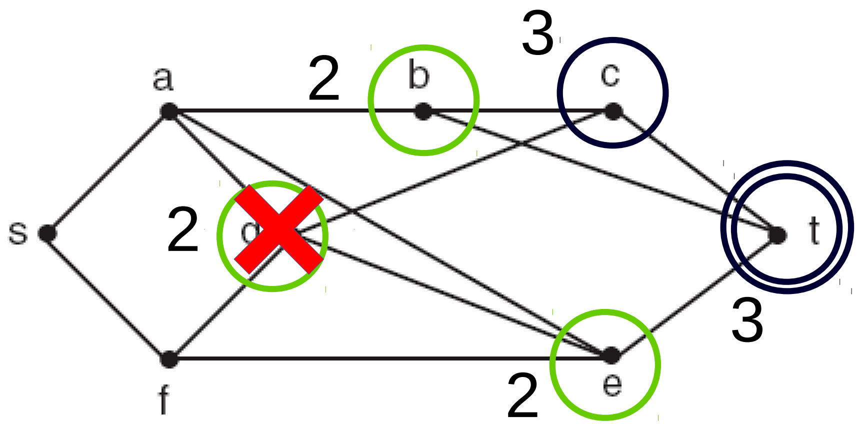
Trace Backtracking

- **Passo 2**



Trace Backtracking

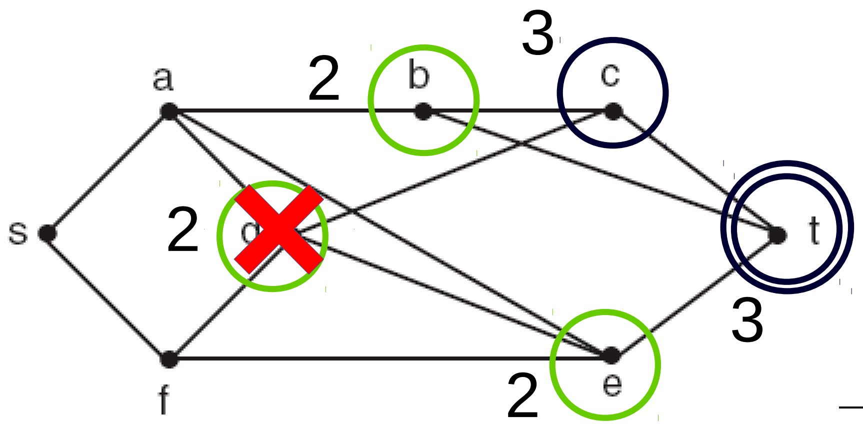
- **Passo 2**



d **não** é adjacente a t

Trace Backtracking

- **Passo 2**

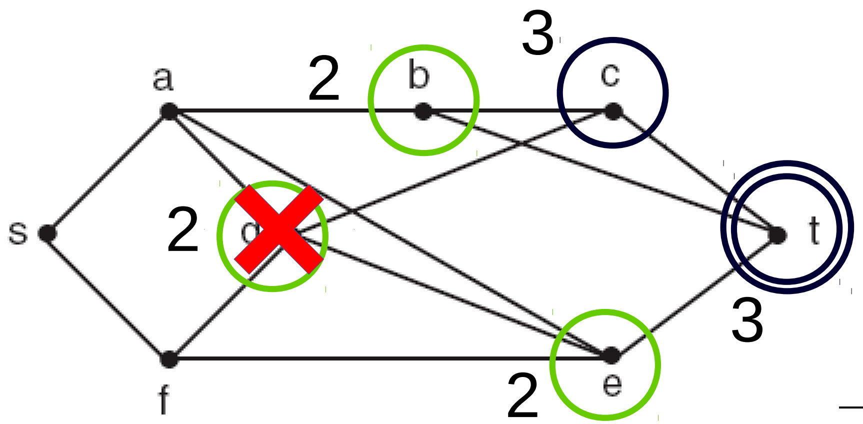


d **não** é adjacente a t

i	v_i
3	t
2	e

Trace Backtracking

- **Passo 2**



d **não** é adjacente a t

i	v_i
3	t
2	e

o **vértice b** também poderia ter sido escolhido

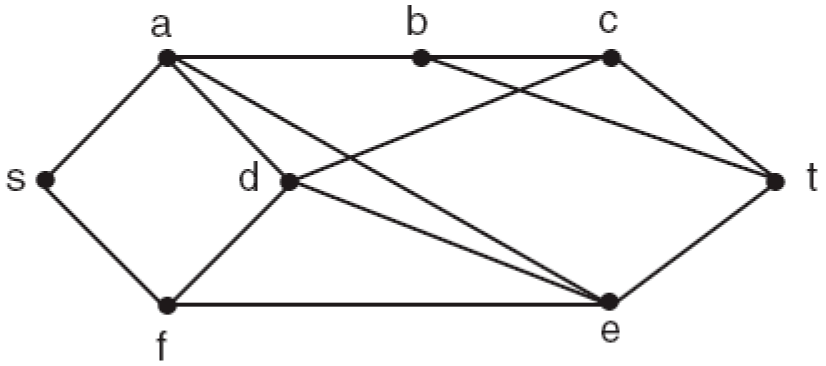
Trace Backtracking

Trace Backtracking

- **Passo 3**

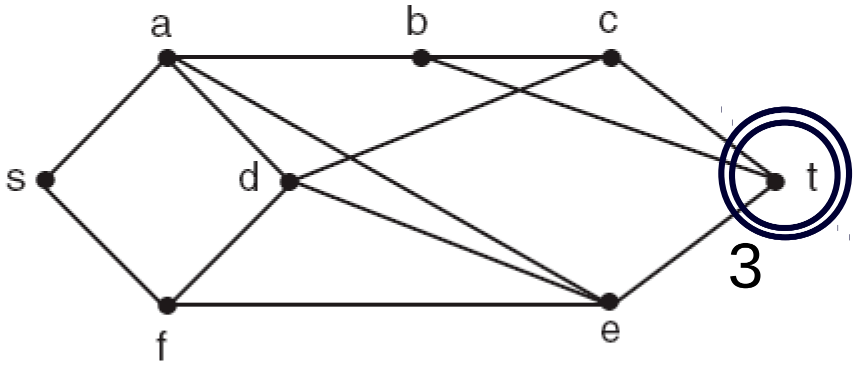
Trace Backtracking

- **Passo 3**



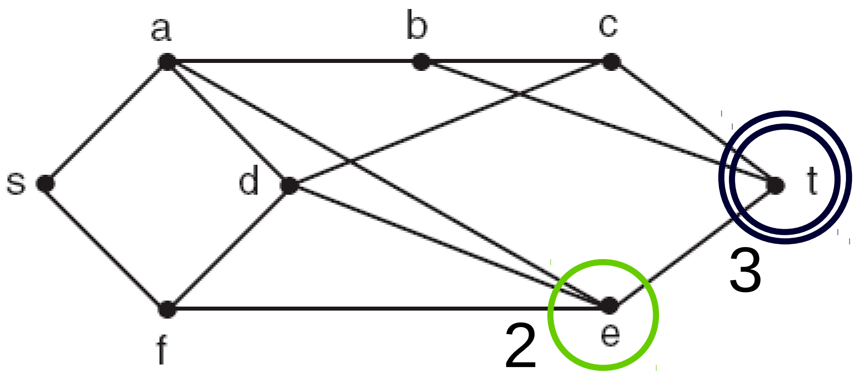
Trace Backtracking

- **Passo 3**



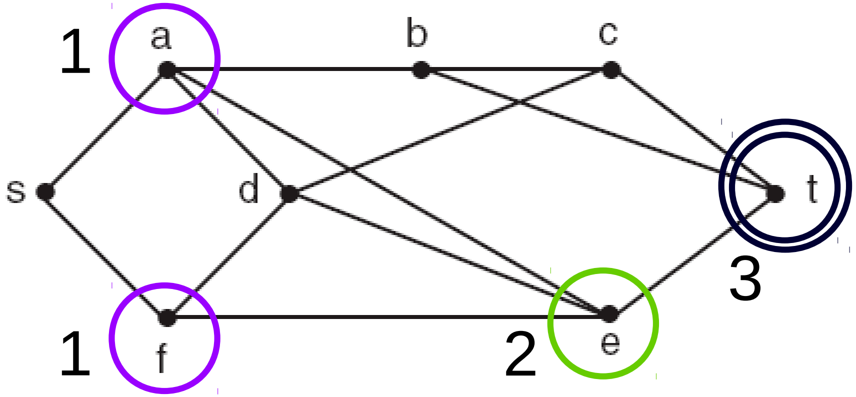
Trace Backtracking

- **Passo 3**



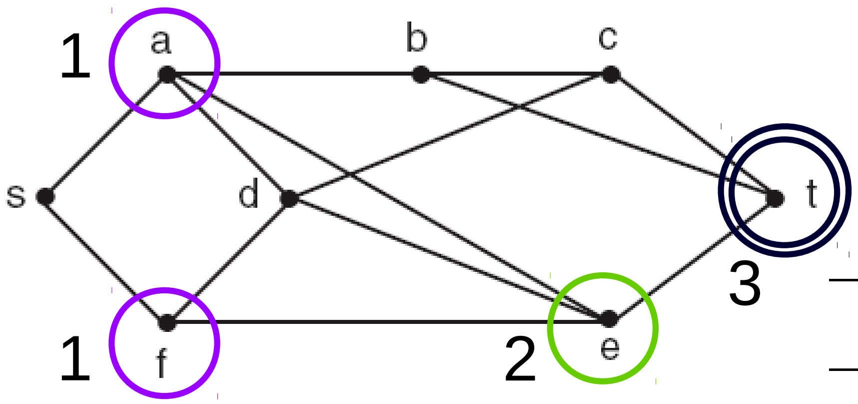
Trace Backtracking

- **Passo 3**



Trace Backtracking

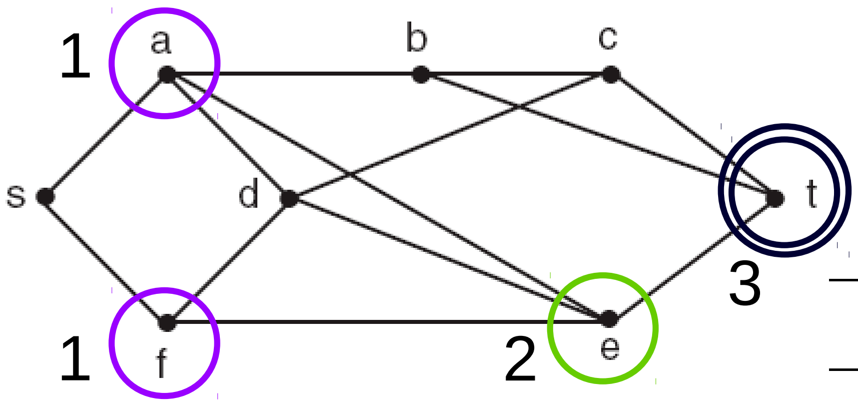
- **Passo 3**



i	v_i
3	t
2	e
1	f

Trace Backtracking

- **Passo 3**



i	v_i
3	t
2	e
1	f

o **vértice a** também poderia ter sido escolhido

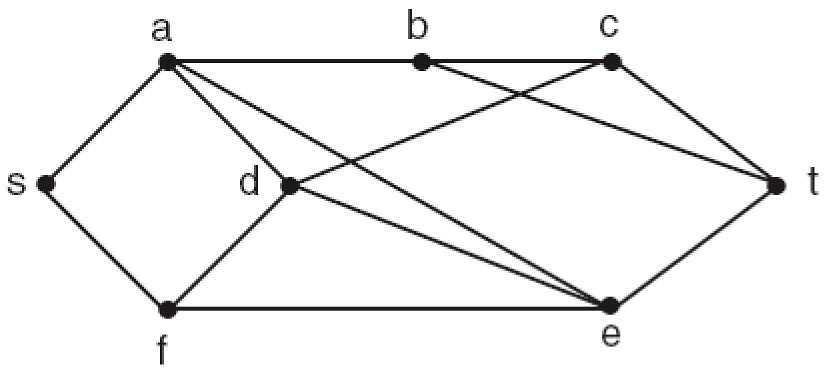
Trace Backtracking

Trace Backtracking

- **Passo 4**

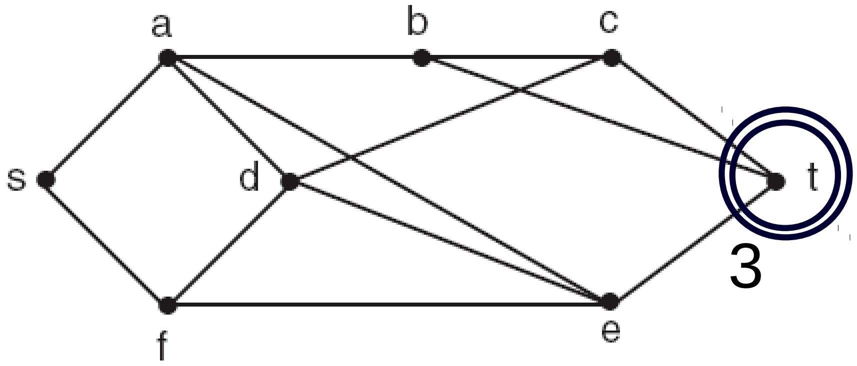
Trace Backtracking

- **Passo 4**



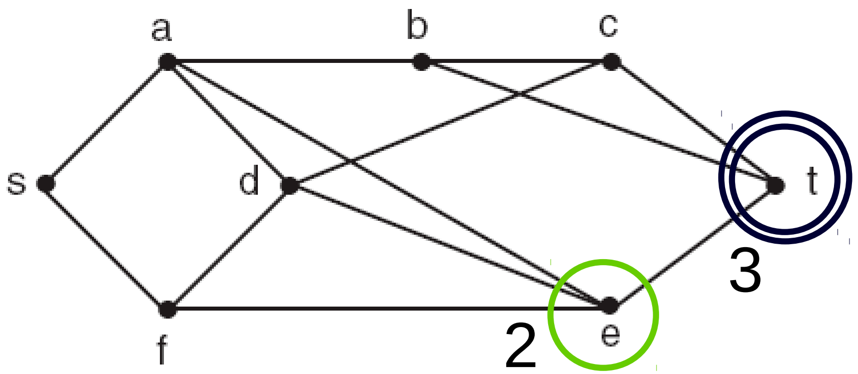
Trace Backtracking

- **Passo 4**



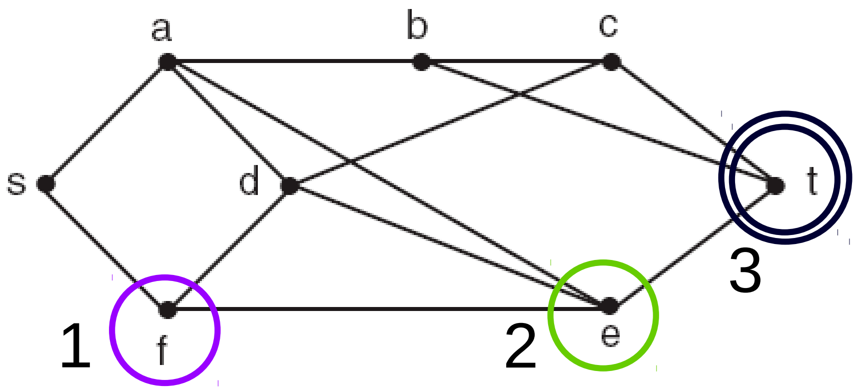
Trace Backtracking

- **Passo 4**



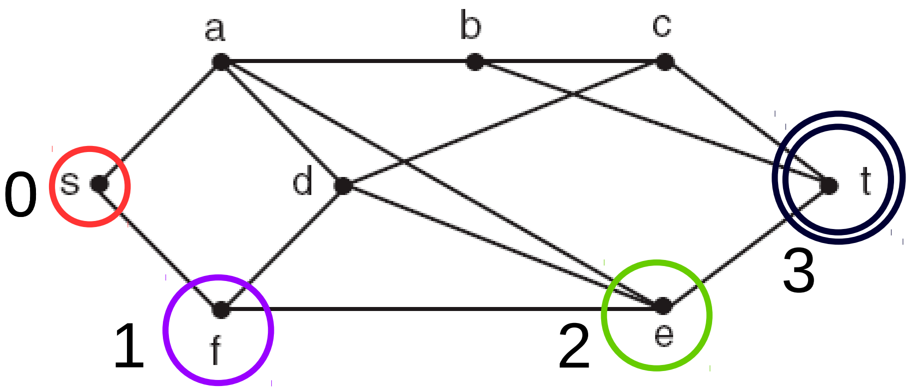
Trace Backtracking

- **Passo 4**



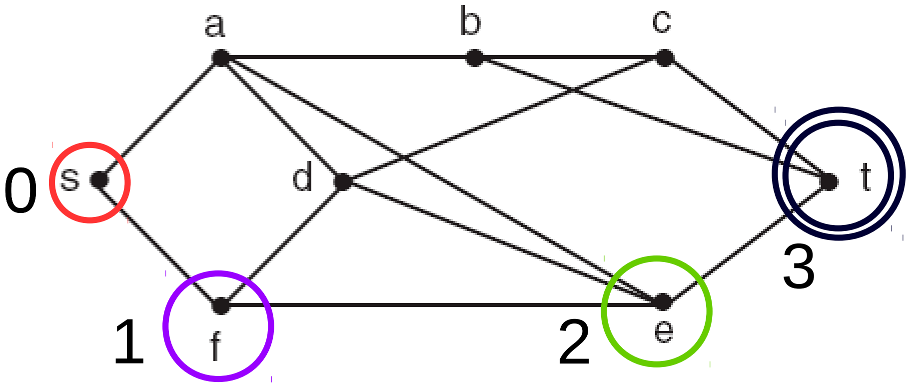
Trace Backtracking

- **Passo 4**



Trace Backtracking

- **Passo 4**



i	v_i
3	t
2	e
1	f
0	s

ALGORITMO DE DIJKSTRA

Algoritmo de Dijkstra

Algoritmo de Dijkstra

- Algoritmo que trata do problema do caminho mais curto de um vértice s a outro vértice t em **grafos ponderados**.

Algoritmo de Dijkstra

- Algoritmo que trata do problema do caminho mais curto de um vértice s a outro vértice t em **grafos ponderados**.
- Dado um caminho P de um vértice s a um vértice t em um grafo ponderado G, o comprimento de P é definido como a soma dos pesos das arestas que fazem parte de P.

Algoritmo de Dijkstra

- Algoritmo que trata do problema do caminho mais curto de um vértice s a outro vértice t em **grafos ponderados**.
- Dado um caminho P de um vértice s a um vértice t em um grafo ponderado G, o comprimento de P é definido como a soma dos pesos das arestas que fazem parte de P.
- Corresponde ao comprimento convencional de caminho em um grafo não ponderado, se um peso com valor 1 for associado a cada aresta.

Algoritmo de Dijkstra

- O algoritmo de Dijkstra é restrito a grafos ponderados nos quais o peso p(e) associado a cada aresta e é não negativo, ou seja, $p(e) \geq 0$.

Algoritmo de Dijkstra

Algoritmo de Dijkstra

- Entrada: Grafo G e dois vértices, s (início) e t (fim)

Algoritmo de Dijkstra

- Entrada: Grafo G e dois vértices, s (início) e t (fim)
- Saída: Sequência de vértices que compõem o caminho mais curto entre s e t.

Algoritmo de Dijkstra

- Entrada: Grafo G e dois vértices, s (início) e t (fim)
- Saída: Sequência de vértices que compõem o caminho mais curto entre s e t.
- Passo 1. Faça $\lambda(s) = 0$. Para todos os vértices v $\neq$ s faça $\lambda(v) = \infty$. Faça T = V (o conjunto de vértices de G, referido como conjunto de vértices não coloridos).

Algoritmo de Dijkstra

- Entrada: Grafo G e dois vértices, s (início) e t (fim)
- Saída: Sequência de vértices que compõem o caminho mais curto entre s e t.
- Passo 1. Faça $\lambda(s) = 0$. Para todos os vértices $v \neq s$ faça $\lambda(v) = \infty$. Faça $T = V$ (o conjunto de vértices de G, referido como conjunto de vértices não coloridos).
- Passo 2. Seja u um vértice de T para o qual $\lambda(u)$ é mínimo.

Algoritmo de Dijkstra

- Entrada: Grafo G e dois vértices, s (início) e t (fim)
- Saída: Sequência de vértices que compõem o caminho mais curto entre s e t.
- Passo 1. Faça $\lambda(s) = 0$. Para todos os vértices $v \neq s$ faça $\lambda(v) = \infty$. Faça $T = V$ (o conjunto de vértices de G, referido como conjunto de vértices não coloridos).
- Passo 2. Seja u um vértice de T para o qual $\lambda(u)$ é mínimo.
- Passo 3. Se $u = t$, pare.

Algoritmo de Dijkstra

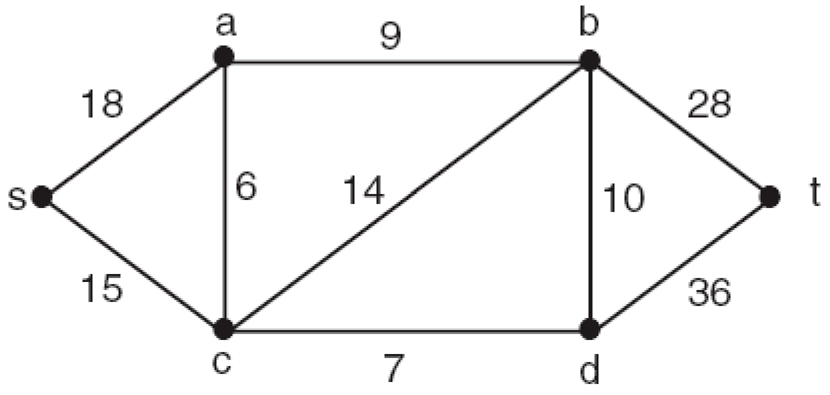
- Entrada: Grafo G e dois vértices, s (início) e t (fim)
- Saída: Sequência de vértices que compõem o caminho mais curto entre s e t.
- Passo 1. Faça $\lambda(s) = 0$. Para todos os vértices $v \neq s$ faça $\lambda(v) = \infty$. Faça $T = V$ (o conjunto de vértices de G, referido como conjunto de vértices não coloridos).
- Passo 2. Seja u um vértice de T para o qual $\lambda(u)$ é mínimo.
- Passo 3. Se $u = t$, pare.
- Passo 4. Para toda aresta $e = uv$ incidente com u, se $v \in T$ e $\lambda(v) > \lambda(u) + p(e)$, troque o valor de $\lambda(v)$ para $\lambda(u) + p(e)$ (ou seja, dada uma aresta e de um vértice não colorido v a u, mude $\lambda(v)$ para $\min\{\lambda(v), \lambda(u) + p(e)\}$).

Algoritmo de Dijkstra

- Entrada: Grafo G e dois vértices, s (início) e t (fim)
- Saída: Sequência de vértices que compõem o caminho mais curto entre s e t.
- Passo 1. Faça $\lambda(s) = 0$. Para todos os vértices $v \neq s$ faça $\lambda(v) = \infty$. Faça $T = V$ (o conjunto de vértices de G, referido como conjunto de vértices não coloridos).
- Passo 2. Seja u um vértice de T para o qual $\lambda(u)$ é mínimo.
- Passo 3. Se $u = t$, pare.
- Passo 4. Para toda aresta $e = uv$ incidente com u, se $v \in T$ e $\lambda(v) > \lambda(u) + p(e)$, troque o valor de $\lambda(v)$ para $\lambda(u) + p(e)$ (ou seja, dada uma aresta e de um vértice não colorido v a u, mude $\lambda(v)$ para $\min\{\lambda(v), \lambda(u) + p(e)\}$).
- Passo 5. $T \leftarrow T - \{u\}$ e vá para o Passo 2 (ou seja, colorir u e então retornar ao Passo 2 para encontrar outro vértice não colorido com rótulo mínimo).

Exemplo

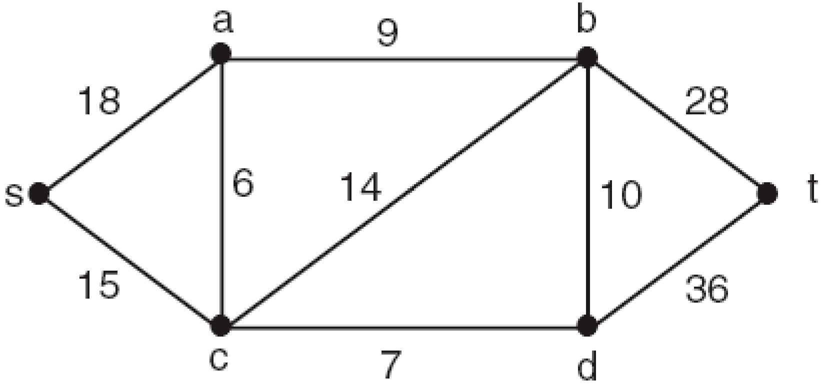
- Considere o grafo abaixo:



Exemplo

- **Passo 1**. Inicialmente os nós são rotulados como:

vértice v	s	a	b	c	d	t
λ(v)	0	∞	∞	∞	∞	∞
T	{s,	a,	b,	c,	d,	t}



Exemplo

- **Passo 2**. O vértice $u = s$ tem $\lambda(u)$ mínimo (com valor 0).

Exemplo

Exemplo

- **Passo 4**. Existem duas arestas incidentes a **u**, **sa** e **sc**, e tanto o vértice **a** quanto o vértice **c** estão em **T** (ou seja, não foram coloridos ainda).

Exemplo

- **Passo 4**. Existem duas arestas incidentes a **u**, **sa** e **sc**, e tanto o vértice **a** quanto o vértice **c** estão em **T** (ou seja, não foram coloridos ainda).
- Então, para $v \in \{a,c\}$, $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$. Tem-se:

Exemplo

- **Passo 4**. Existem duas arestas incidentes a **u**, **sa** e **sc**, e tanto o vértice **a** quanto o vértice **c** estão em **T** (ou seja, não foram coloridos ainda).
- Então, para $v \in \{a,c\}$, $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$. Tem-se:
  - $\lambda(a) \leftarrow \min\{\lambda(a), \lambda(s) + p(sa)\}$ ou seja, $\lambda(a) \leftarrow \min\{\infty, 0 + 18\}$ e, portanto, $\lambda(a) = 18$

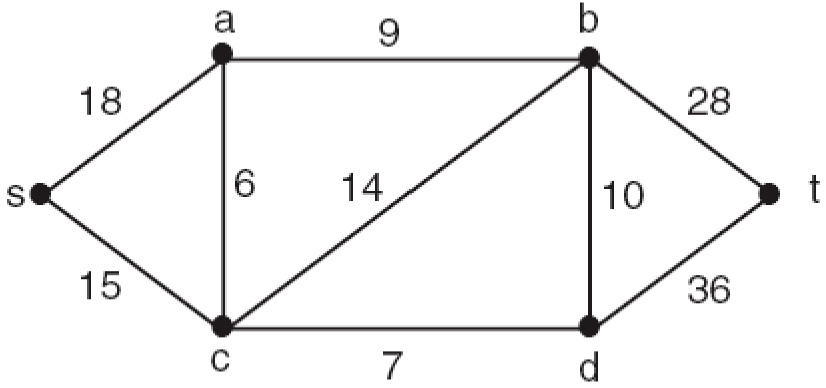
Exemplo

- **Passo 4**. Existem duas arestas incidentes a **u**, **sa** e **sc**, e tanto o vértice **a** quanto o vértice **c** estão em **T** (ou seja, não foram coloridos ainda).
- Então, para $v \in \{a,c\}$, $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$. Tem-se:
  - $\lambda(a) \leftarrow \min\{\lambda(a), \lambda(s) + p(sa)\}$ ou seja, $\lambda(a) \leftarrow \min\{\infty, 0 + 18\}$ e, portanto, $\lambda(a) = 18$
  - $\lambda(c) \leftarrow \min\{\lambda(c), \lambda(s) + p(sc)\}$ ou seja $\lambda(c) \leftarrow \min\{\infty, 0 + 15\}$ e, portanto, $\lambda(c) = 15$

Exemplo

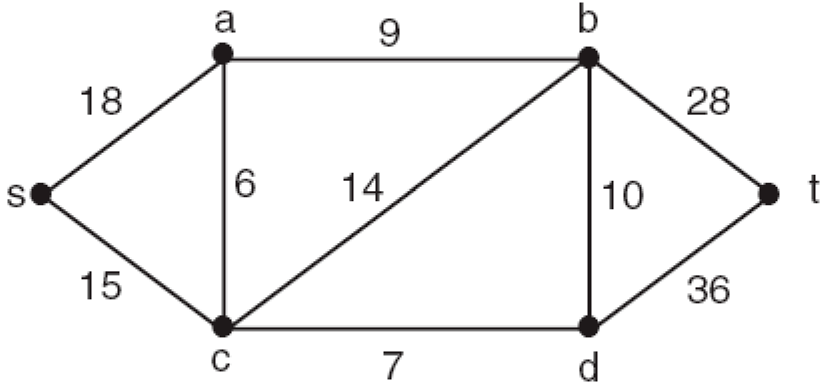
- **Passo 5**. T $\leftarrow$ T – {u} (ou seja, o vértice s é colorido}. Tem-se então:

vértice v	s	a	b	c	d	t
λ(v)	0	18	∞	15	∞	∞
T		{a,	b,	c,	d,	t}



Exemplo

- Passo 2. O vértice $u = c \in T$ e tem $\lambda(u)$ mínimo ($\lambda(u) = 15$).



Exemplo

Exemplo

- **Passo 4**. Existem quatro arestas incidentes a u = c, a saber, cs, ca, cb e cd.

Exemplo

- **Passo 4**. Existem quatro arestas incidentes a u = c, a saber, cs, ca, cb e cd.
- Dessas quatro, uma tem vértice v $\notin$ T (aresta cs).

Exemplo

- **Passo 4**. Existem quatro arestas incidentes a $u = c$, a saber, cs, ca, cb e cd.
- Dessas quatro, uma tem vértice $v \notin T$ (aresta cs).
- Então, para $v \in \{a,b,d\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.

Exemplo

- **Passo 4**. Existem quatro arestas incidentes a $u = c$, a saber, cs, ca, cb e cd.
- Dessas quatro, uma tem vértice $v \notin T$ (aresta cs).
- Então, para $v \in \{a,b,d\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se:

Exemplo

- **Passo 4**. Existem quatro arestas incidentes a u = c, a saber, cs, ca, cb e cd.
- Dessas quatro, uma tem vértice $v \notin T$ (aresta cs).
- Então, para $v \in \{a,b,d\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se:
  - $\lambda(a) \leftarrow \min\{\lambda(a), \lambda(c) + p(ca)\}$ ou seja, $\lambda(a) \leftarrow \min\{18, 15 + 6\}$ e, portanto, $\lambda(a) = 18$

Exemplo

- **Passo 4**. Existem quatro arestas incidentes a $u = c$, a saber, cs, ca, cb e cd.
- Dessas quatro, uma tem vértice $v \notin T$ (aresta cs).
- Então, para $v \in \{a,b,d\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se:
  - $\lambda(a) \leftarrow \min\{\lambda(a), \lambda(c) + p(ca)\}$ ou seja, $\lambda(a) \leftarrow \min\{18, 15 + 6\}$ e, portanto, $\lambda(a) = 18$
  - $\lambda(b) \leftarrow \min\{\lambda(b), \lambda(c) + p(cb)\}$ ou seja $\lambda(b) \leftarrow \min\{\infty, 15 + 14\}$ e, portanto, $\lambda(b) = 29$

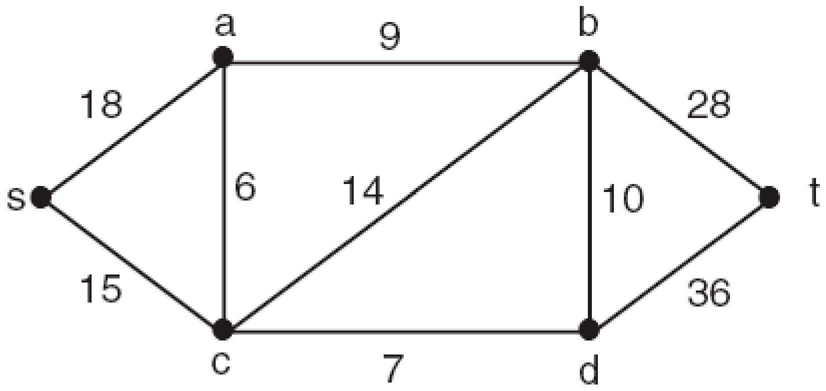
Exemplo

- **Passo 4**. Existem quatro arestas incidentes a $u = c$, a saber, cs, ca, cb e cd.
- Dessas quatro, uma tem vértice $v \notin T$ (aresta cs).
- Então, para $v \in \{a,b,d\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se:
  - $\lambda(a) \leftarrow \min\{\lambda(a), \lambda(c) + p(ca)\}$ ou seja, $\lambda(a) \leftarrow \min\{18, 15 + 6\}$ e, portanto, $\lambda(a) = 18$
  - $\lambda(b) \leftarrow \min\{\lambda(b), \lambda(c) + p(cb)\}$ ou seja $\lambda(b) \leftarrow \min\{\infty, 15 + 14\}$ e, portanto, $\lambda(b) = 29$
  - $\lambda(d) \leftarrow \min\{\lambda(d), \lambda(c) + p(cd)\}$ ou seja $\lambda(d) \leftarrow \min\{\infty, 15 + 7\}$ e, portanto, $\lambda(d) = 22$

Exemplo

- **Passo 5**. T $\leftarrow$ T – {u} (ou seja, o vértice c é colorido}. Então, tem-se:

vértice v	s	a	b	c	d	t
λ(v)	0	18	29	15	22	∞
T		{a,	b,		d,	t}



Exemplo

- **Passo 2**. $u = a \in T$ e tem $\lambda(u)$ mínimo ($\lambda(u) = 18$).

Exemplo

Exemplo

- **Passo 4**. Existem três arestas incidentes a u = a, a saber, as, ab e ac. Dessas três, apenas uma tem vértice $v \in T$ (aresta ab).

Exemplo

- **Passo 4**. Existem três arestas incidentes a $u = a$, a saber, as, ab e ac. Dessas três, apenas uma tem vértice $v \in T$ (aresta ab).
- Então, para $v \in \{b\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.

Exemplo

- **Passo 4**. Existem três arestas incidentes a u = a, a saber, as, ab e ac. Dessas três, apenas uma tem vértice $v \in T$ (aresta ab).
- Então, para $v \in \{b\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se:

Exemplo

- **Passo 4**. Existem três arestas incidentes a $u = a$, a saber, as, ab e ac. Dessas três, apenas uma tem vértice $v \in T$ (aresta ab).
- Então, para $v \in \{b\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se:
  - $\lambda(b) \leftarrow \min\{\lambda(b), \lambda(a) + p(ab)\}$ ou seja, $\lambda(b) \leftarrow \min\{29, 18 + 9\}$ e portanto, $\lambda(b) = 27$

Exemplo

- **Passo 5**. T ← T − {u} (ou seja, vértice a é colorido}. Então tem-se:

vértice v	s	a	b	c	d	t
λ(v)	0	18	27	15	22	∞
T			{b,		d,	t}

Exemplo

- **Passo 2**. O vértice $u = d \in T$ e tem $\lambda(u)$ mínimo ($\lambda(u) = 22$).

Exemplo

Exemplo

- **Passo 4**. Existem três arestas incidentes a u = d, a saber, dc, db e dt.

Exemplo

- **Passo 4**. Existem três arestas incidentes a u = d, a saber, dc, db e dt.
- Dessas três, duas têm vértices $v \in T$ (aresta db e dt).

Exemplo

- **Passo 4**. Existem três arestas incidentes a $u = d$, a saber, dc, db e dt.
- Dessas três, duas têm vértices $v \in T$ (aresta db e dt).
- Então, para $v \in \{b,t\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.

Exemplo

- **Passo 4**. Existem três arestas incidentes a $u = d$, a saber, dc, db e dt.
- Dessas três, duas têm vértices $v \in T$ (aresta db e dt).
- Então, para $v \in \{b,t\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se:

Exemplo

- **Passo 4**. Existem três arestas incidentes a u = d, a saber, dc, db e dt.
- Dessas três, duas têm vértices $v \in T$ (aresta db e dt).
- Então, para $v \in \{b,t\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se:
  - $\lambda(b) \leftarrow \min\{\lambda(b), \lambda(d) + p(db)\}$ ou seja, $\lambda(b) \leftarrow \min\{27, 22 + 10\}$ e, portanto, $\lambda(b) = 27$

Exemplo

- **Passo 4**. Existem três arestas incidentes a $u = d$, a saber, dc, db e dt.
- Dessas três, duas têm vértices $v \in T$ (aresta db e dt).
- Então, para $v \in \{b,t\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se:
  - $\lambda(b) \leftarrow \min\{\lambda(b), \lambda(d) + p(db)\}$ ou seja, $\lambda(b) \leftarrow \min\{27, 22 + 10\}$ e, portanto, $\lambda(b) = 27$
  - $\lambda(t) \leftarrow \min\{\lambda(t), \lambda(d) + p(dt)\}$ ou seja, $\lambda(t) \leftarrow \min\{\infty, 22 + 36\}$ e, portanto, $\lambda(t) = 58$

Exemplo

- Passo 5. T ← T − {u} (ou seja, vértice d é colorido}. Então, tem-se:

vértice v	s	a	b	c	d	t
λ(v)	0	18	27	15	22	58
T			{b,			t}

Exemplo

- **Passo 2**. O vértice $u = b \in T$ e tem $\lambda(u)$ mínimo ($\lambda(u) = 27$)

Exemplo

Exemplo

- **Passo 4**. Existe uma única aresta incidente a u = b, a saber, bt e t $\in$ T.

Exemplo

- **Passo 4**. Existe uma única aresta incidente a $u = b$, a saber, bt e $t \in T$.
- Então, para $v \in \{t\}$, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$. Tem-se então:

Exemplo

- **Passo 4**. Existe uma única aresta incidente a u = b, a saber, bt e t $\in$ T.
- Então, para v $\in$ {t}, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$. Tem-se então:
- $\lambda(t) \leftarrow \min\{\lambda(t), \lambda(b) + p(bt)\}$ ou seja, $\lambda(t) \leftarrow \min\{58, 27 + 28\}$ e, portanto, $\lambda(t) = 55$

Exemplo

- **Passo 2**. O vértice $u = b \in T$ e tem $\lambda(u)$ mínimo ($\lambda(u) = 27$)

Exemplo

Exemplo

- **Passo 4**. Existe uma única aresta incidente a u = b, a saber, bt e t $\in$ T. Então, para v $\in$ {t}, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.

Exemplo

- **Passo 4**. Existe uma única aresta incidente a u = b, a saber, bt e t $\in$ T. Então, para v $\in$ {t}, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se então:

Exemplo

- **Passo 4**. Existe uma única aresta incidente a u = b, a saber, bt e t $\in$ T. Então, para v $\in$ {t}, calcula-se $\lambda(v) \leftarrow \min\{\lambda(v), \lambda(u) + p(e)\}$.
- Tem-se então:
  - $\lambda(t) \leftarrow \min\{\lambda(t), \lambda(b) + p(bt)\}$ ou seja, $\lambda(t) \leftarrow \min\{58, 27 + 28\}$ e, portanto, $\lambda(t) = 55$

Exemplo

- Passo 5. $T \leftarrow T - \{u\}$ (ou seja, vértice b é colorido}. Tem-se então:

vértice v	s	a	b	c	d	t
λ(v)	0	18	27	15	22	55
T						{t}

Exemplo

Exemplo

- **Passo 2**. $u = t \in T$ (única escolha)

Exemplo

- **Passo 2**. u = t ∈ T (única escolha)
- **Passo 3**. u = t, pare.

Exemplo

Exemplo

- Quando o algoritmo termina, os valores $\lambda(v)$ fornecem os comprimentos dos caminhos mais curtos do vértice s a cada um dos vértices v.

Exemplo

- Quando o algoritmo termina, os valores λ(v) fornecem os comprimentos dos caminhos mais curtos do vértice s a cada um dos vértices v.
- Assim, os comprimentos desses caminhos do vértice s aos vértices σ, b, c, d, t são 18, 27, 15, 22 e 55, respectivamente.

Exemplo

- Quando o algoritmo termina, os valores λ(v) fornecem os comprimentos dos caminhos mais curtos do vértice s a cada um dos vértices v.
- Assim, os comprimentos desses caminhos do vértice s aos vértices σ, b, c, d, t são 18, 27, 15, 22 e 55, respectivamente.
- A identificação do caminho mais curto (ou seja, a identificação da sequência de vértices) pode ser feita usando *backtracking*.

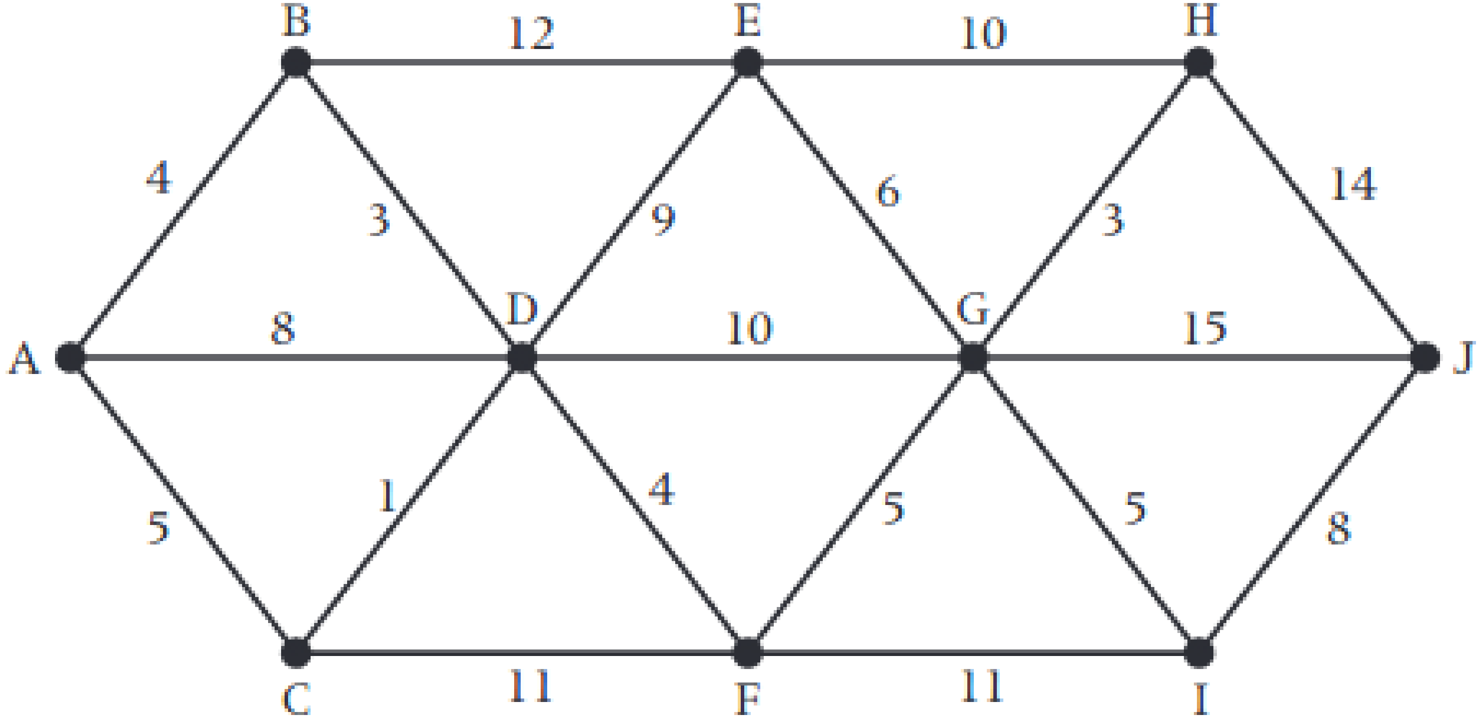
Exemplo

vértices						T
s	a	b	c	d	t	
0	∞	∞	∞	∞	∞	{ s, a, b, c, d, t}
↑	18	∞	**15**	∞	∞	{ a, b, c, d, t}
	18	29		22	∞	{ a, b, d, t}
	↑	27		**22**	∞	{ b, d, t}
		27			58	{ b, t}
		↑			**55**	{ t}
					↑	

NOVO EXEMPLO

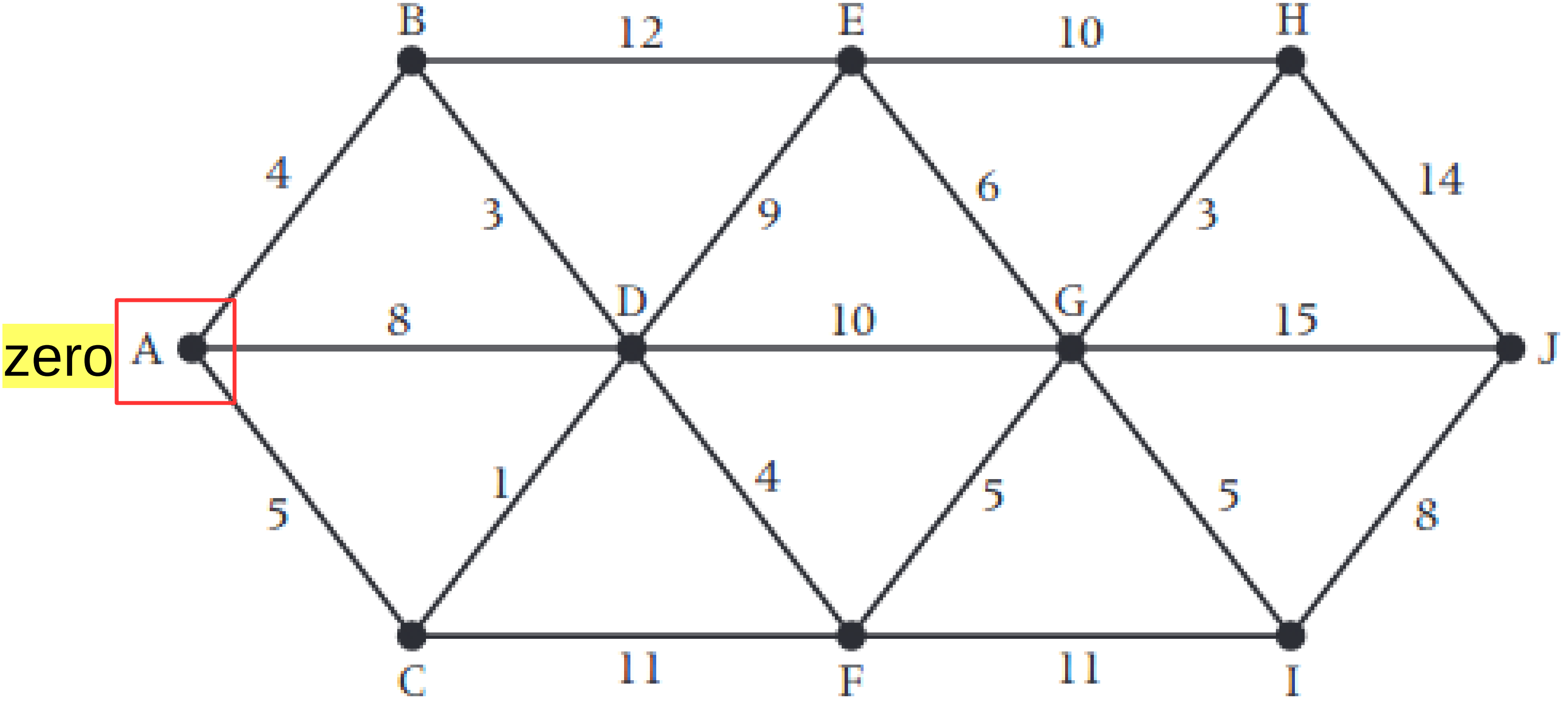
Exemplo 2

- Ache o caminho mais curto entre A e J:



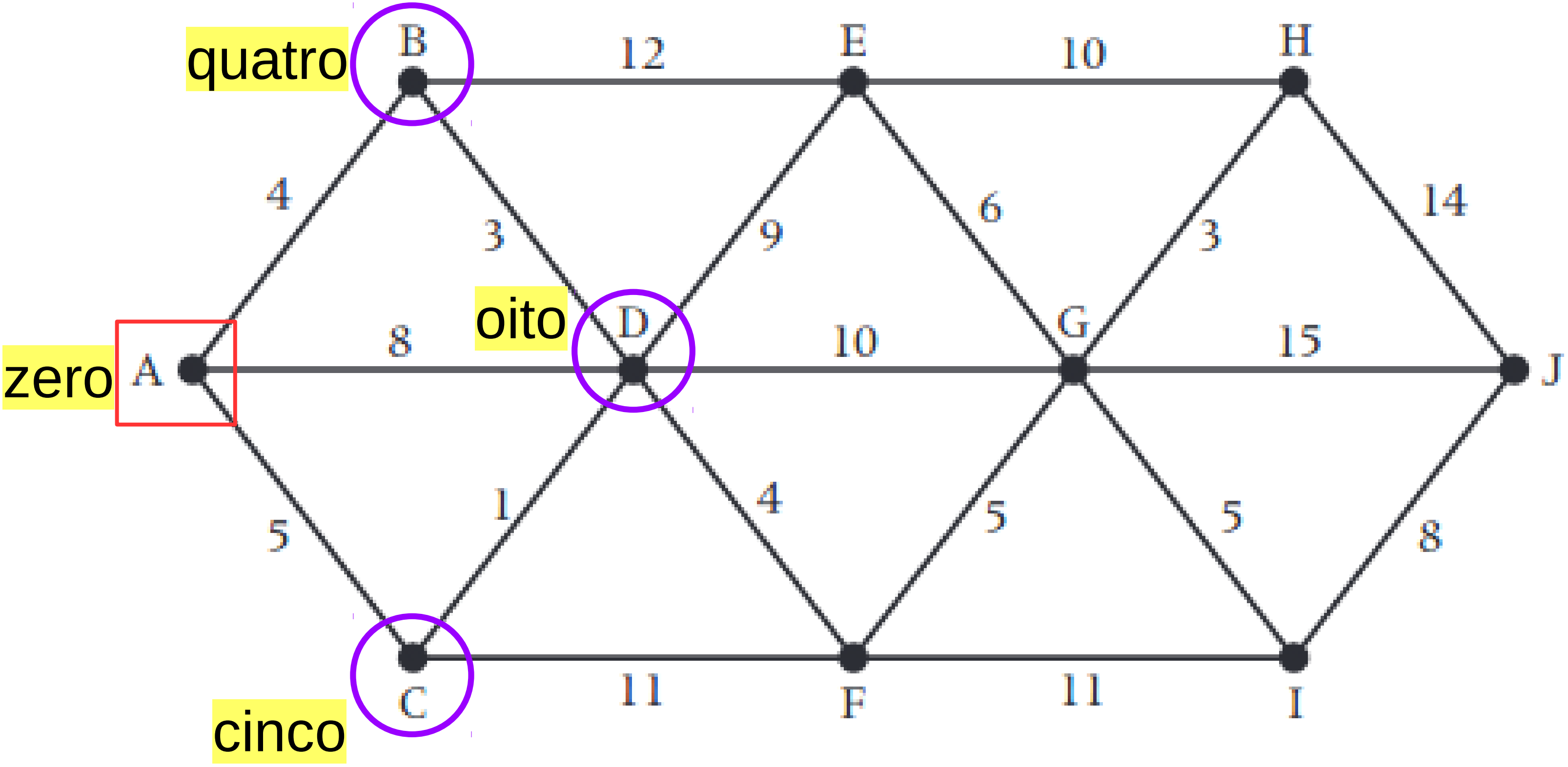
Exemplo 2

- Ache o caminho mais curto entre A e J:



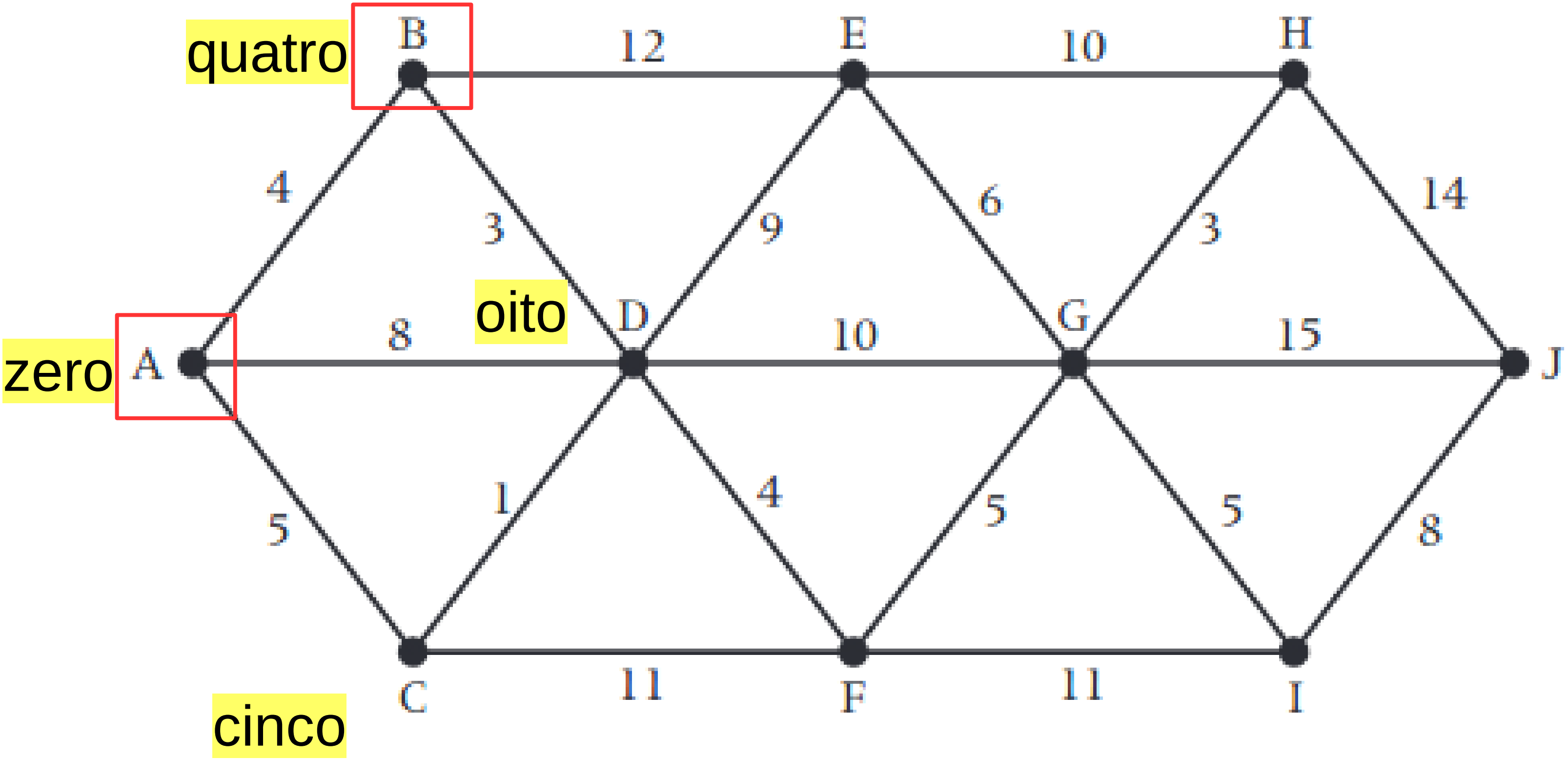
Exemplo 2

- Ache o caminho mais curto entre A e J:



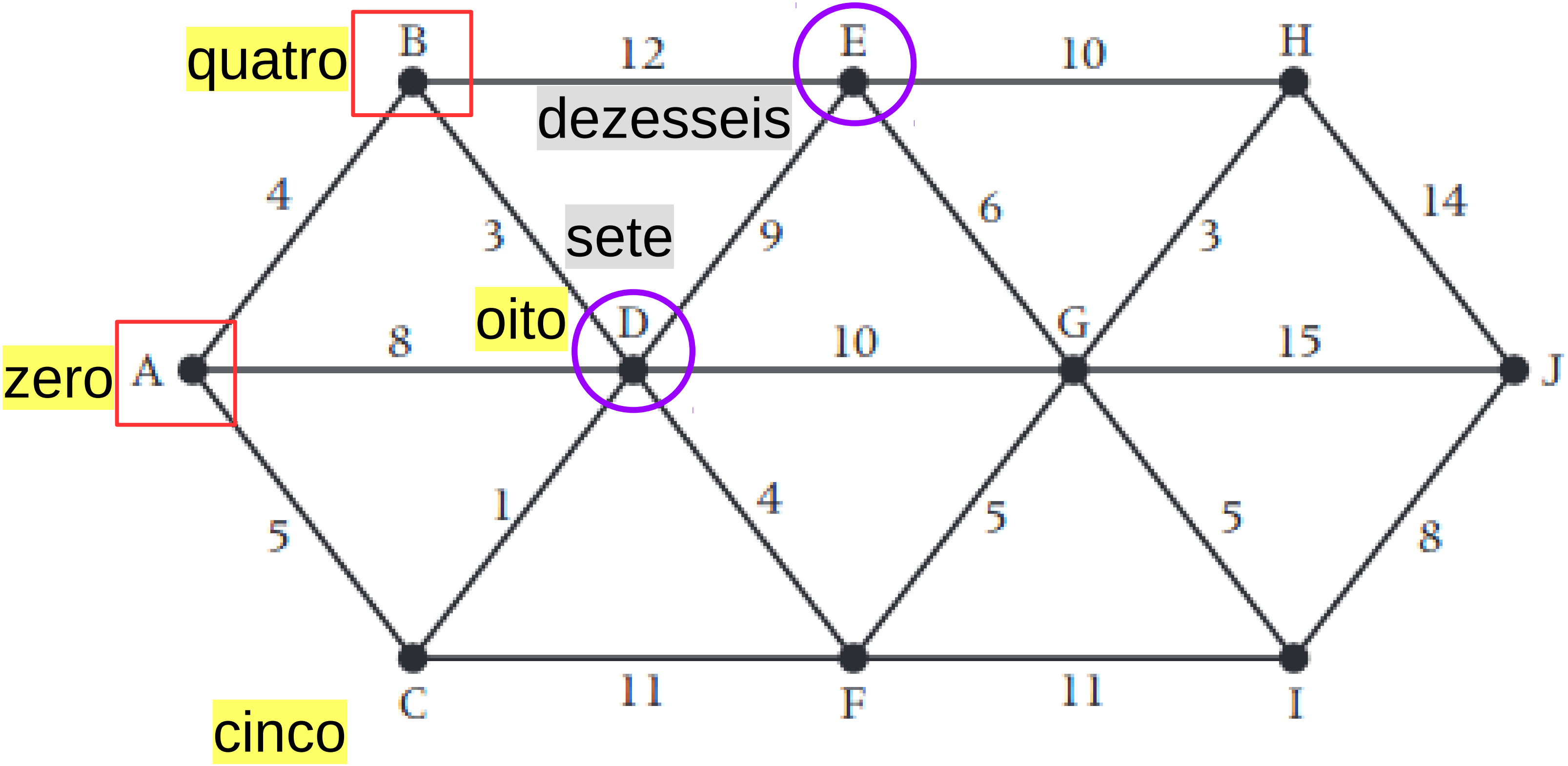
Exemplo 2

- Ache o caminho mais curto entre A e J:



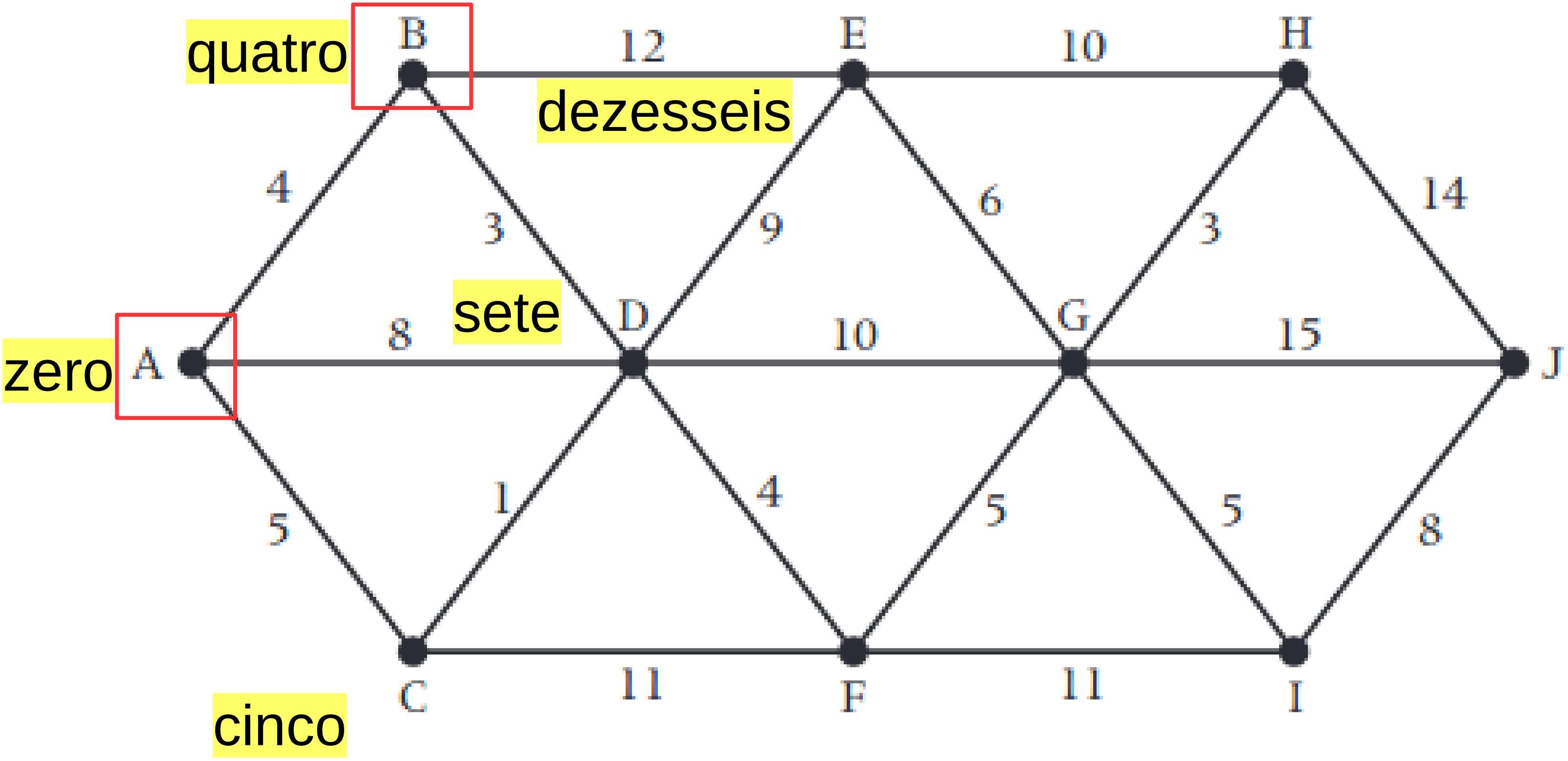
Exemplo 2

- Ache o caminho mais curto entre A e J:



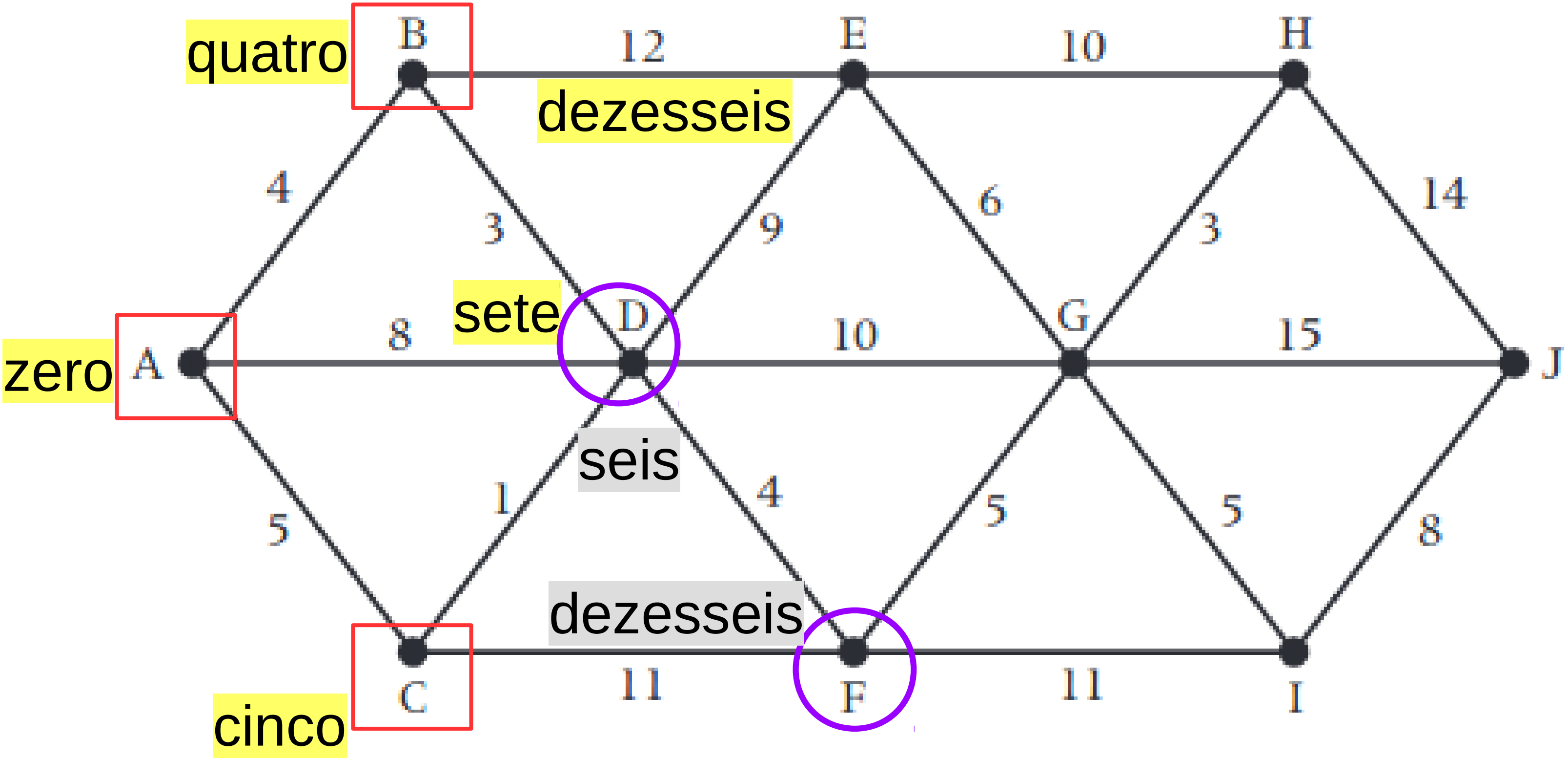
Exemplo 2

- Ache o caminho mais curto entre A e J:



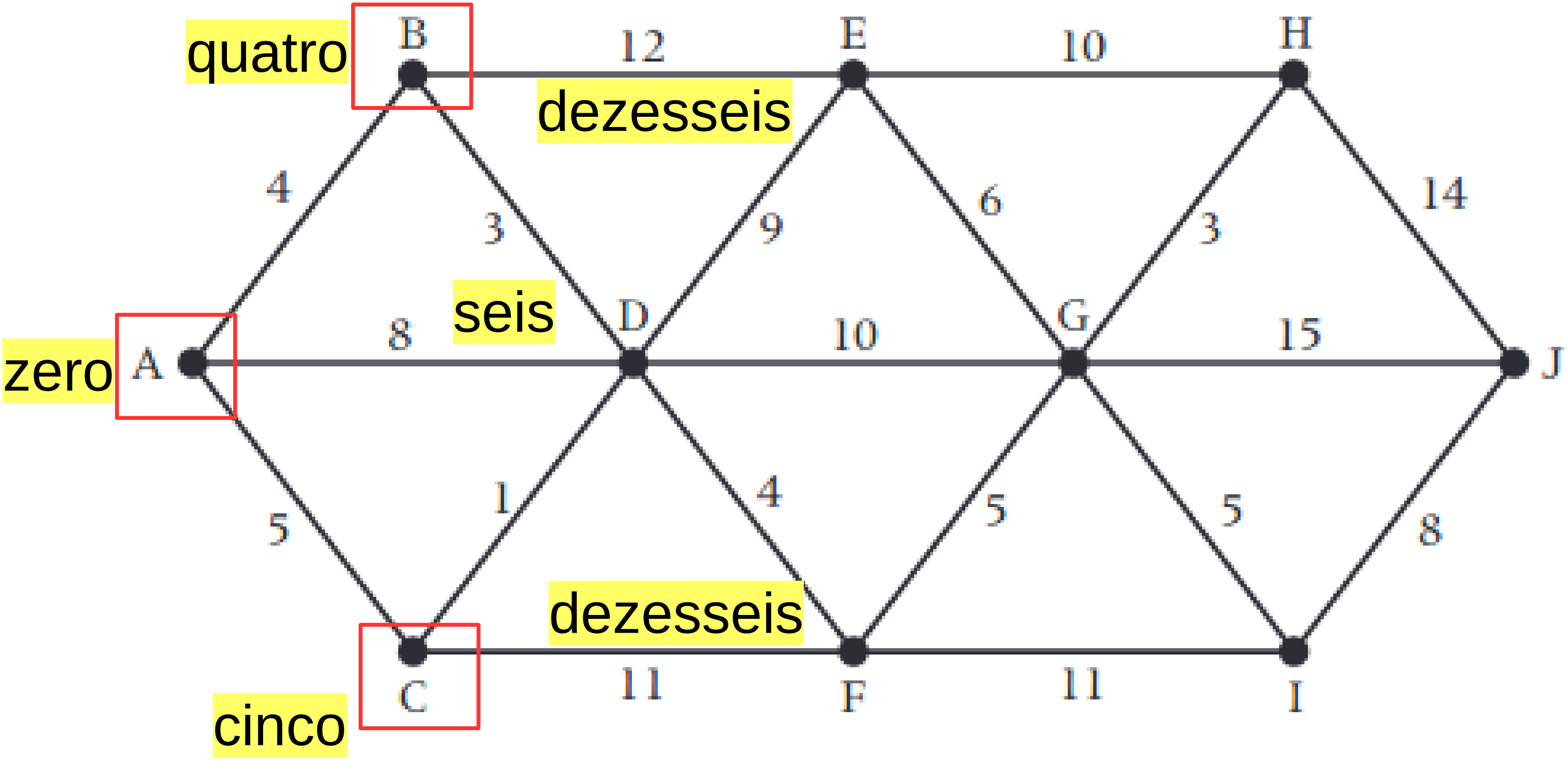
Exemplo 2

- Ache o caminho mais curto entre A e J:



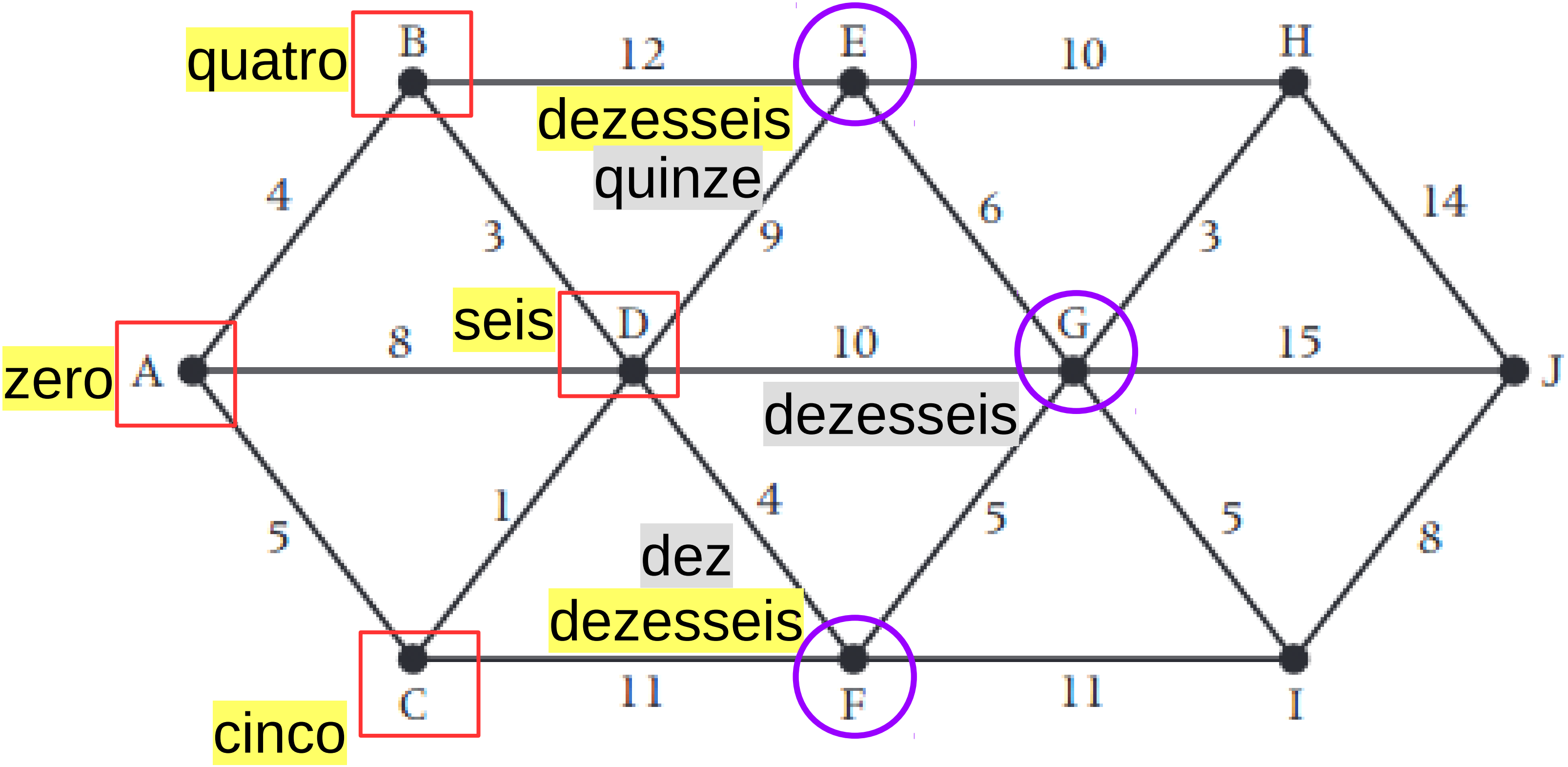
Exemplo 2

- Ache o caminho mais curto entre A e J:



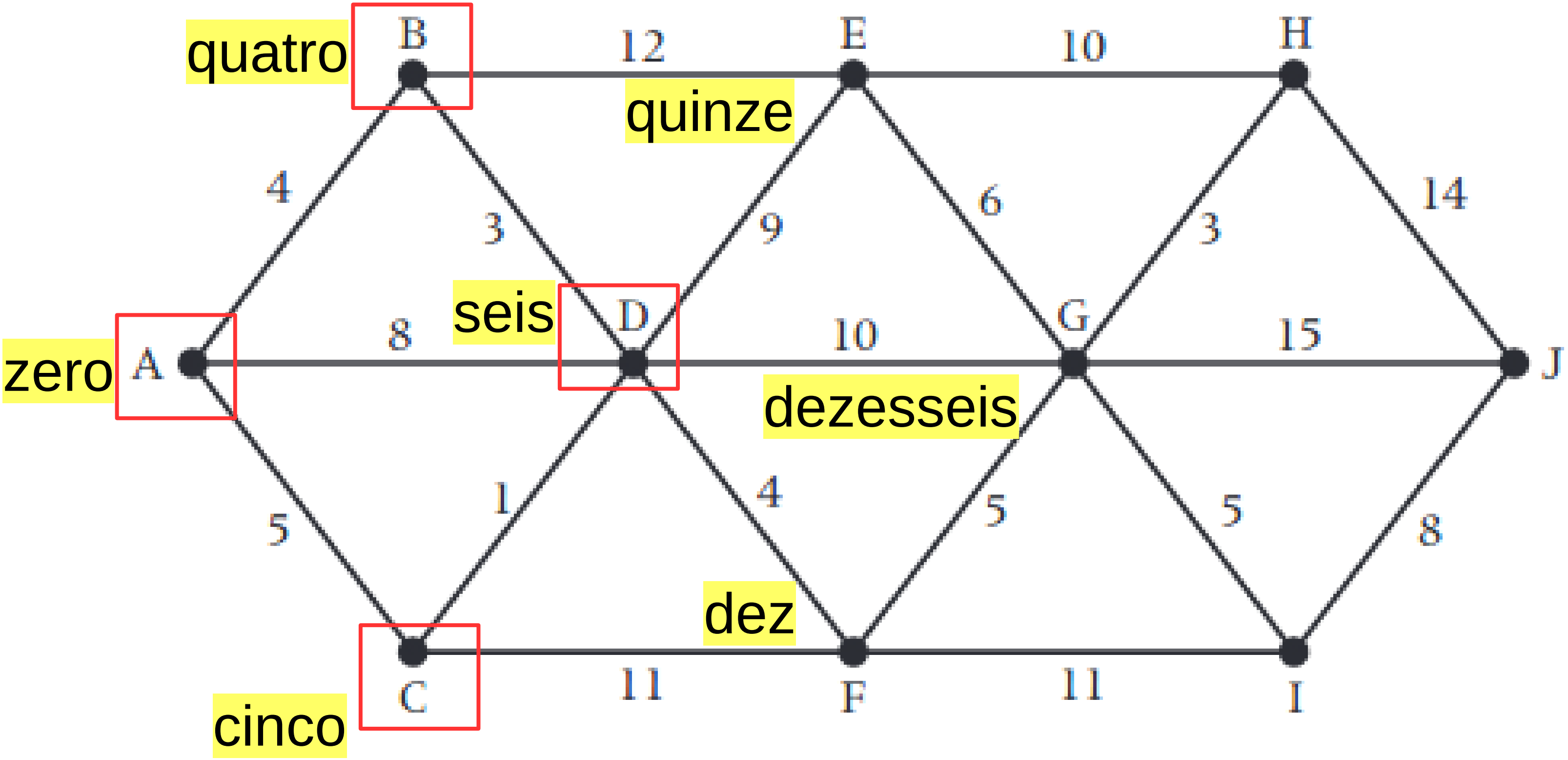
Exemplo 2

- Ache o caminho mais curto entre A e J:



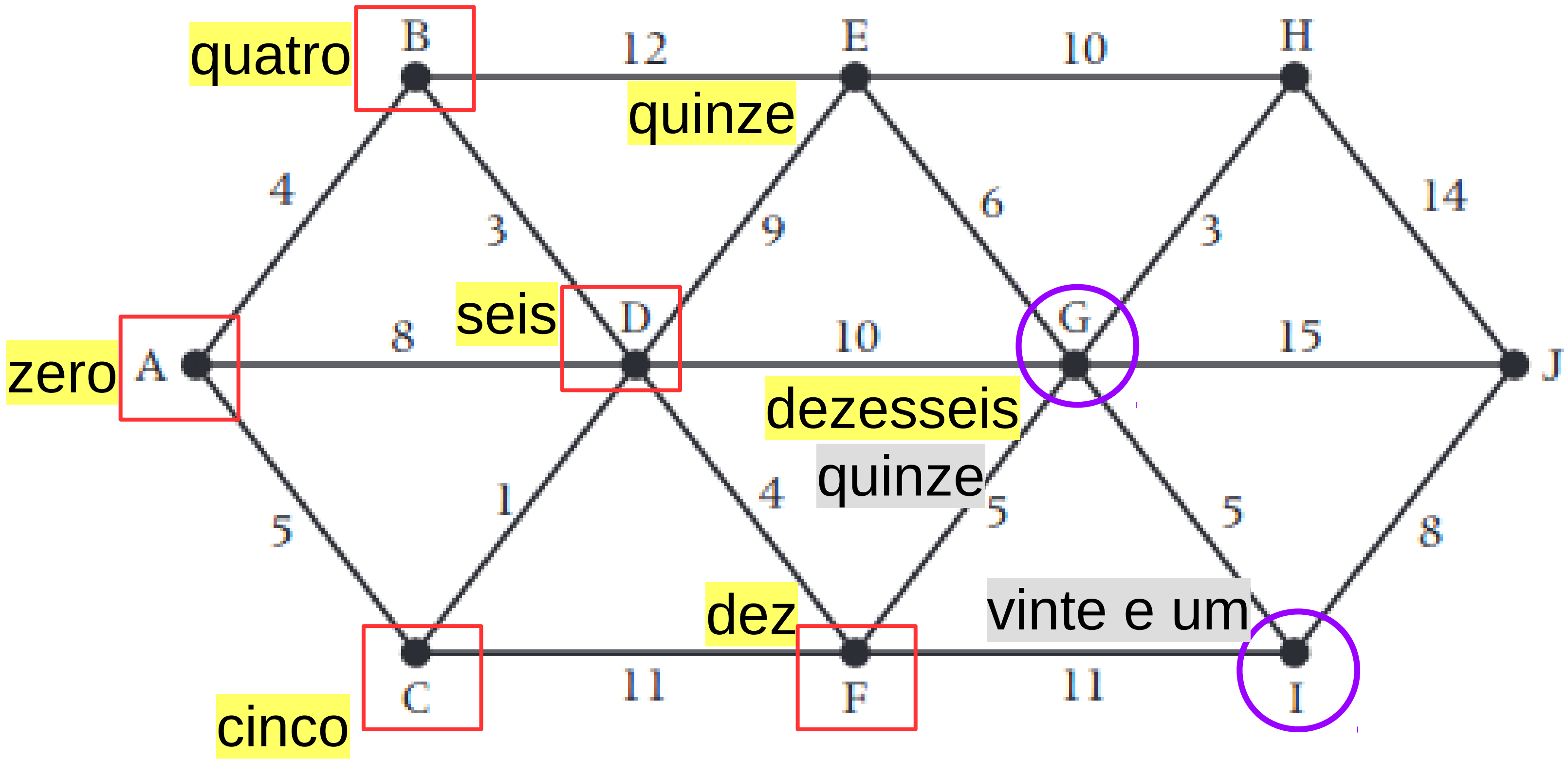
Exemplo 2

- Ache o caminho mais curto entre A e J:



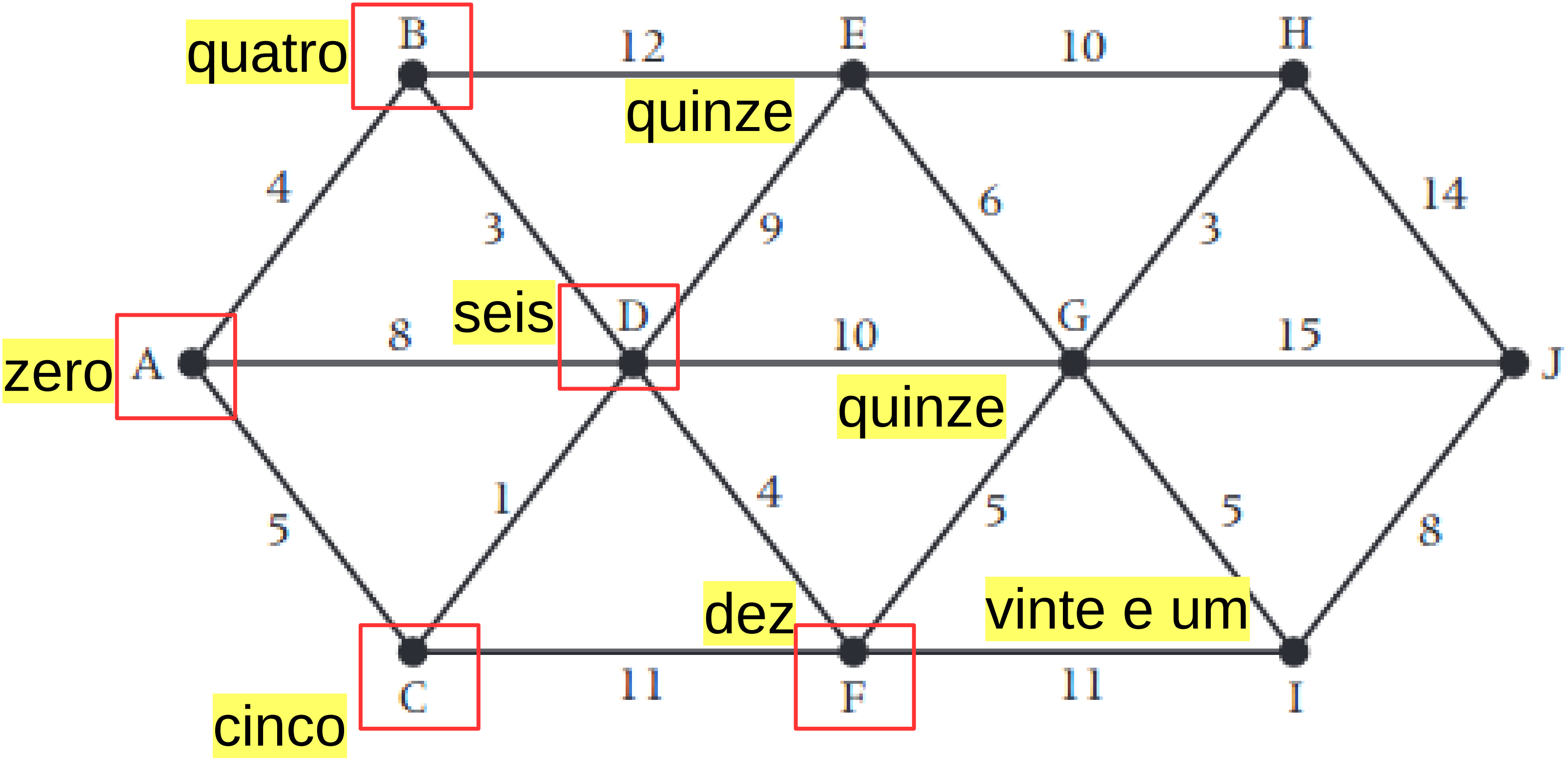
Exemplo 2

- Ache o caminho mais curto entre A e J:



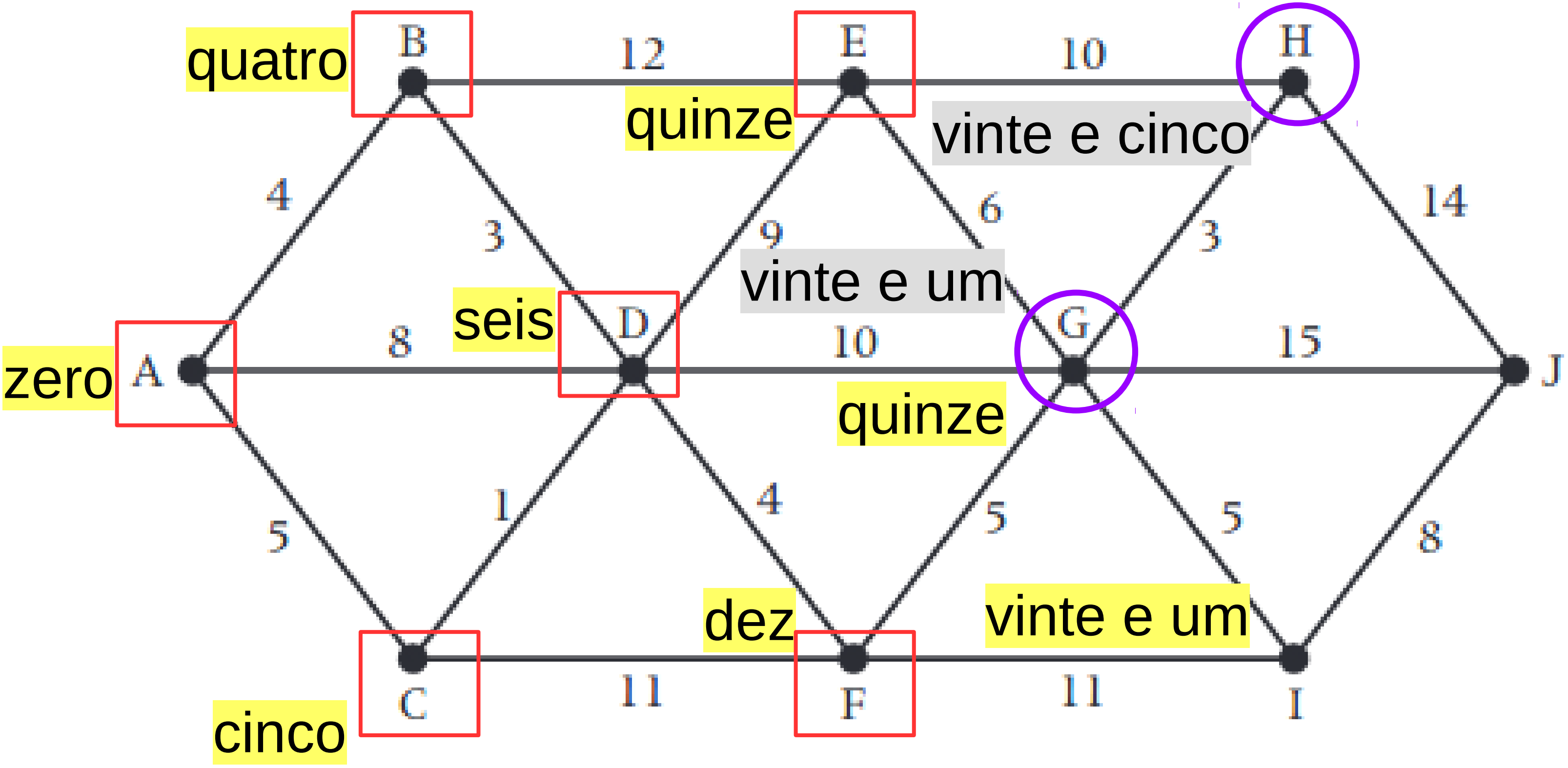
Exemplo 2

- Ache o caminho mais curto entre A e J:



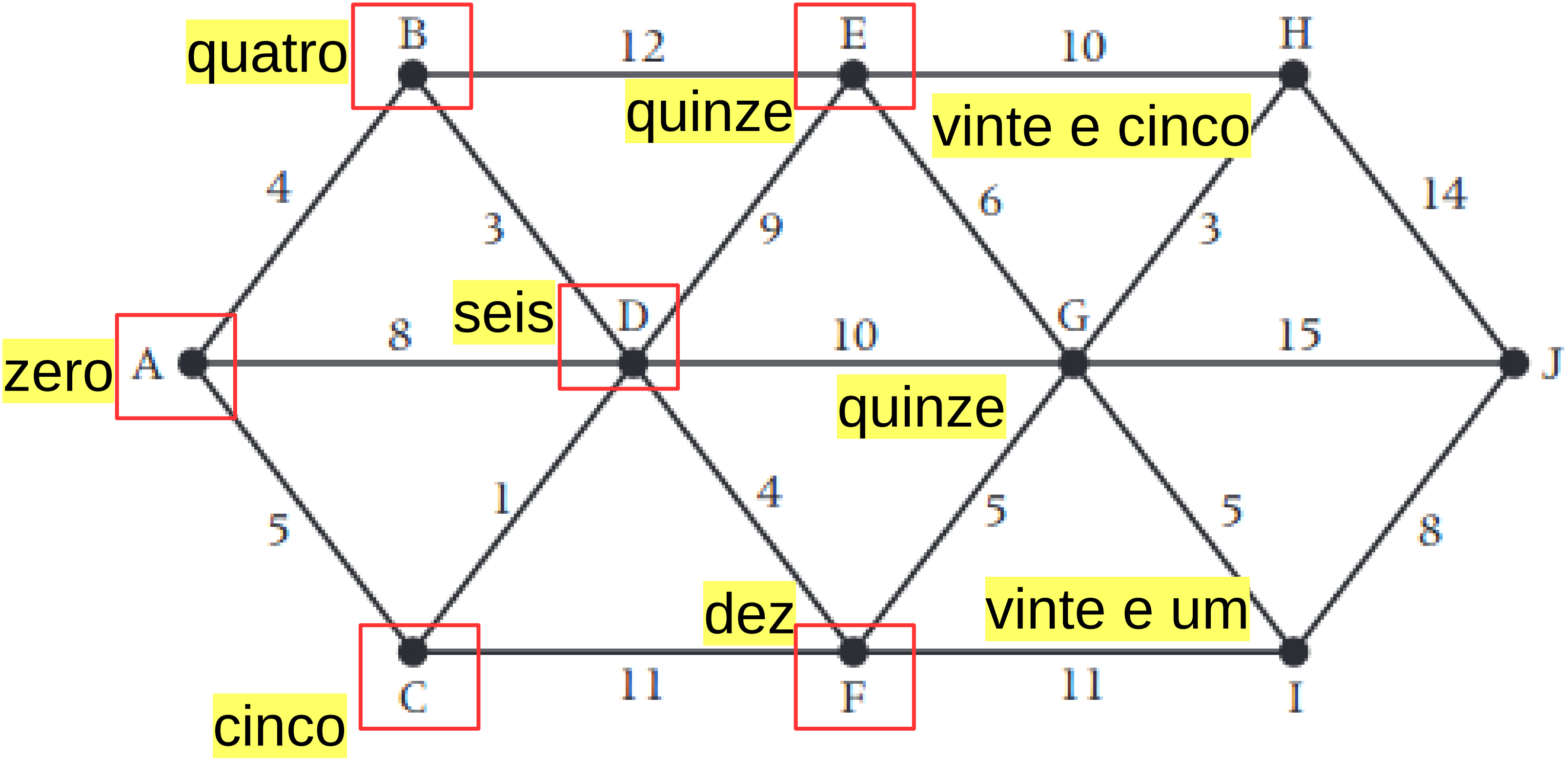
Exemplo 2

- Ache o caminho mais curto entre A e J:



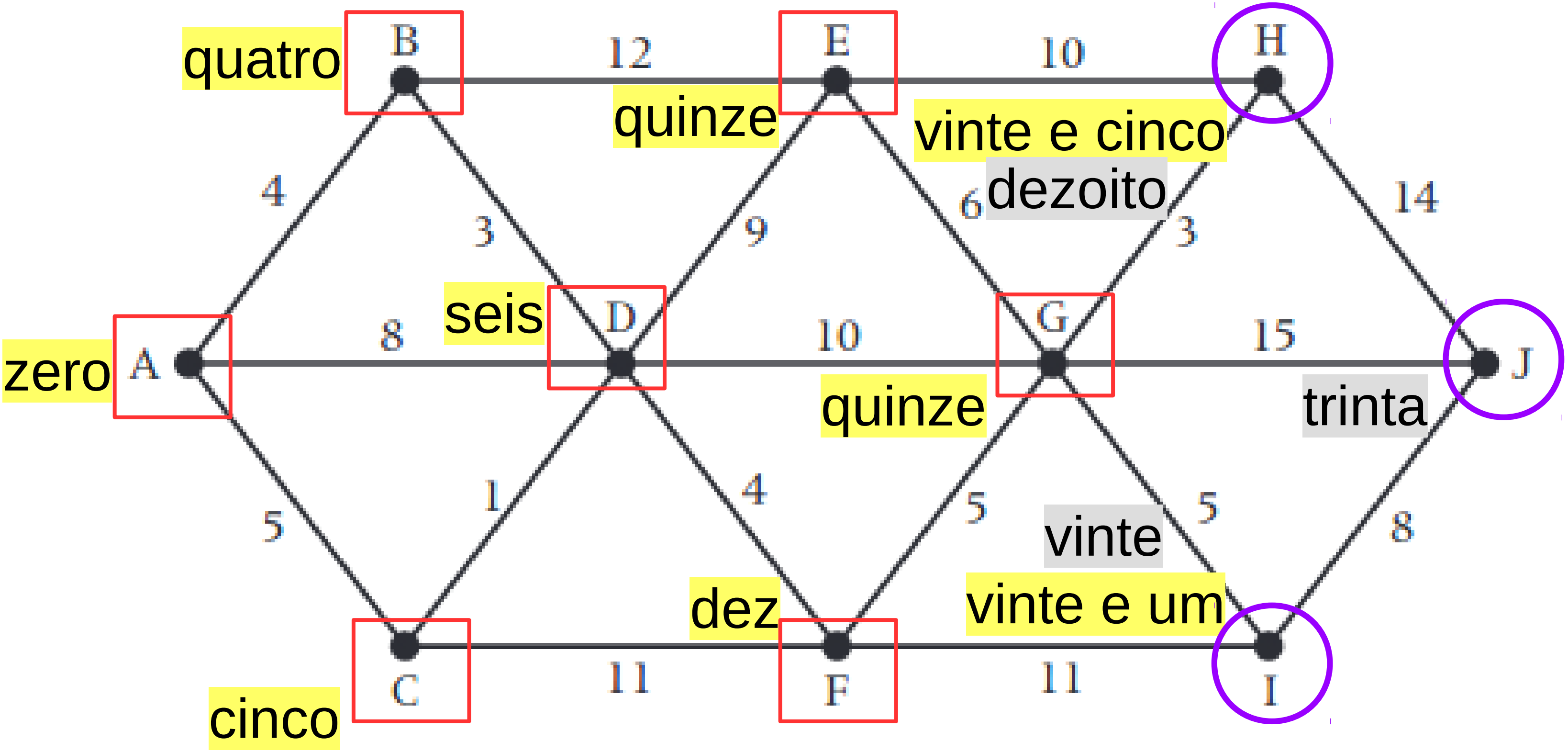
Exemplo 2

- Ache o caminho mais curto entre A e J:



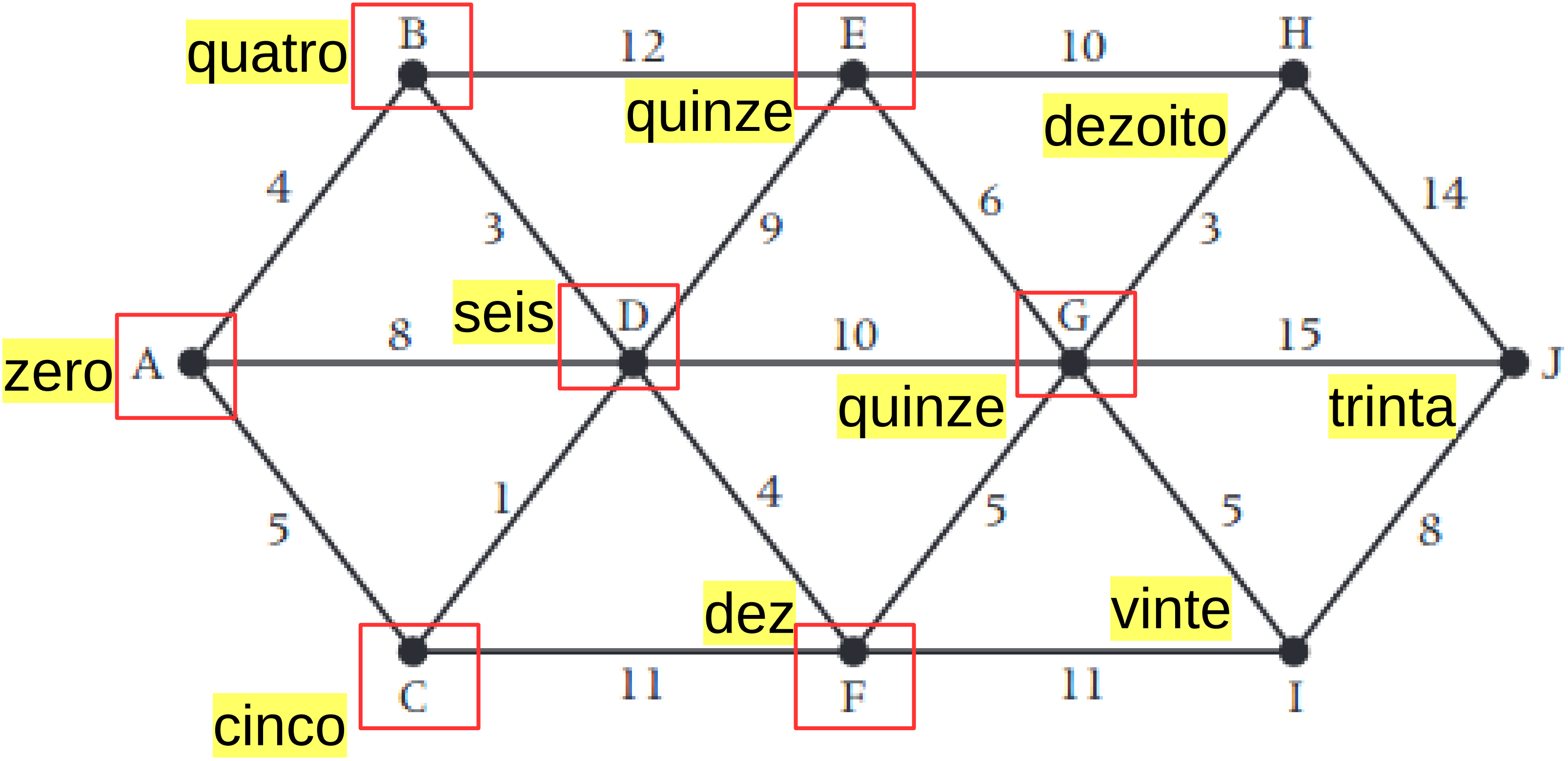
Exemplo 2

- Ache o caminho mais curto entre A e J:



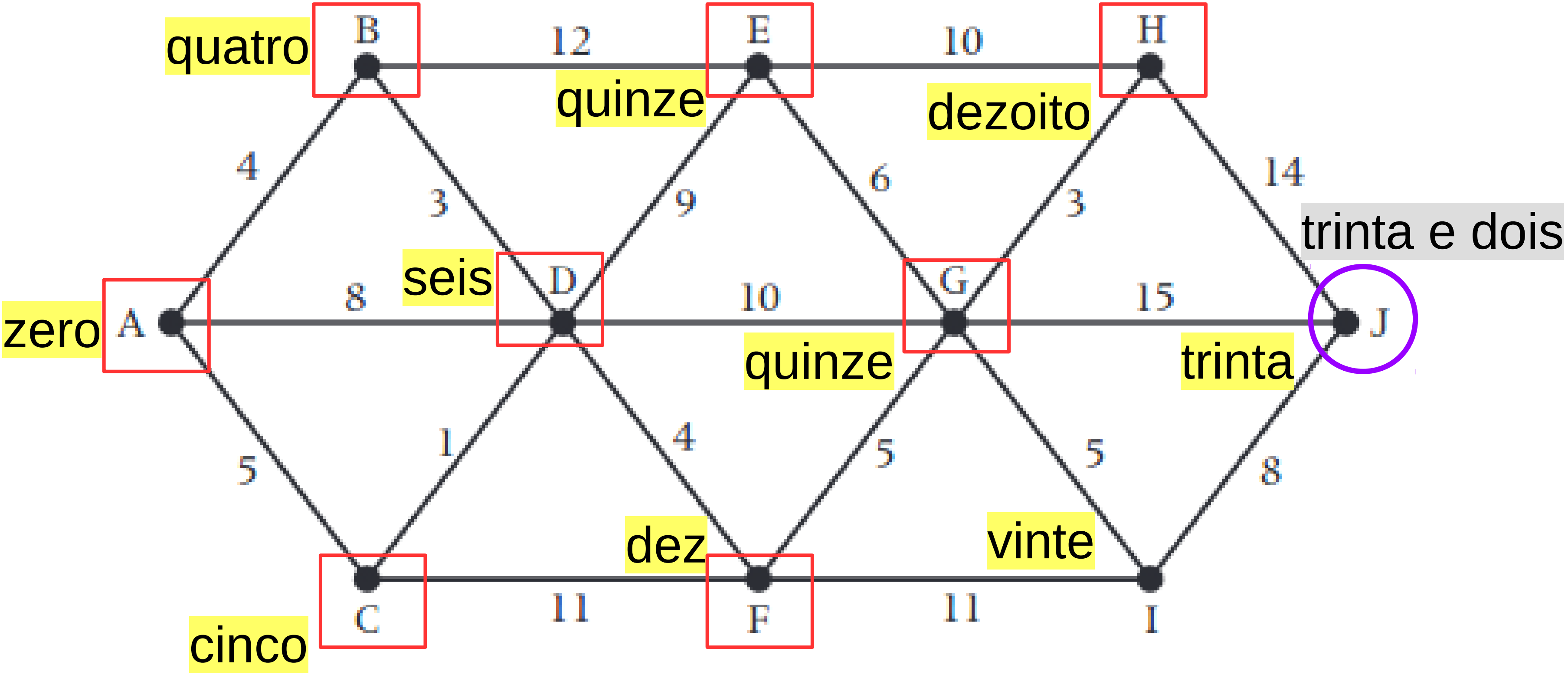
Exemplo 2

- Ache o caminho mais curto entre A e J:



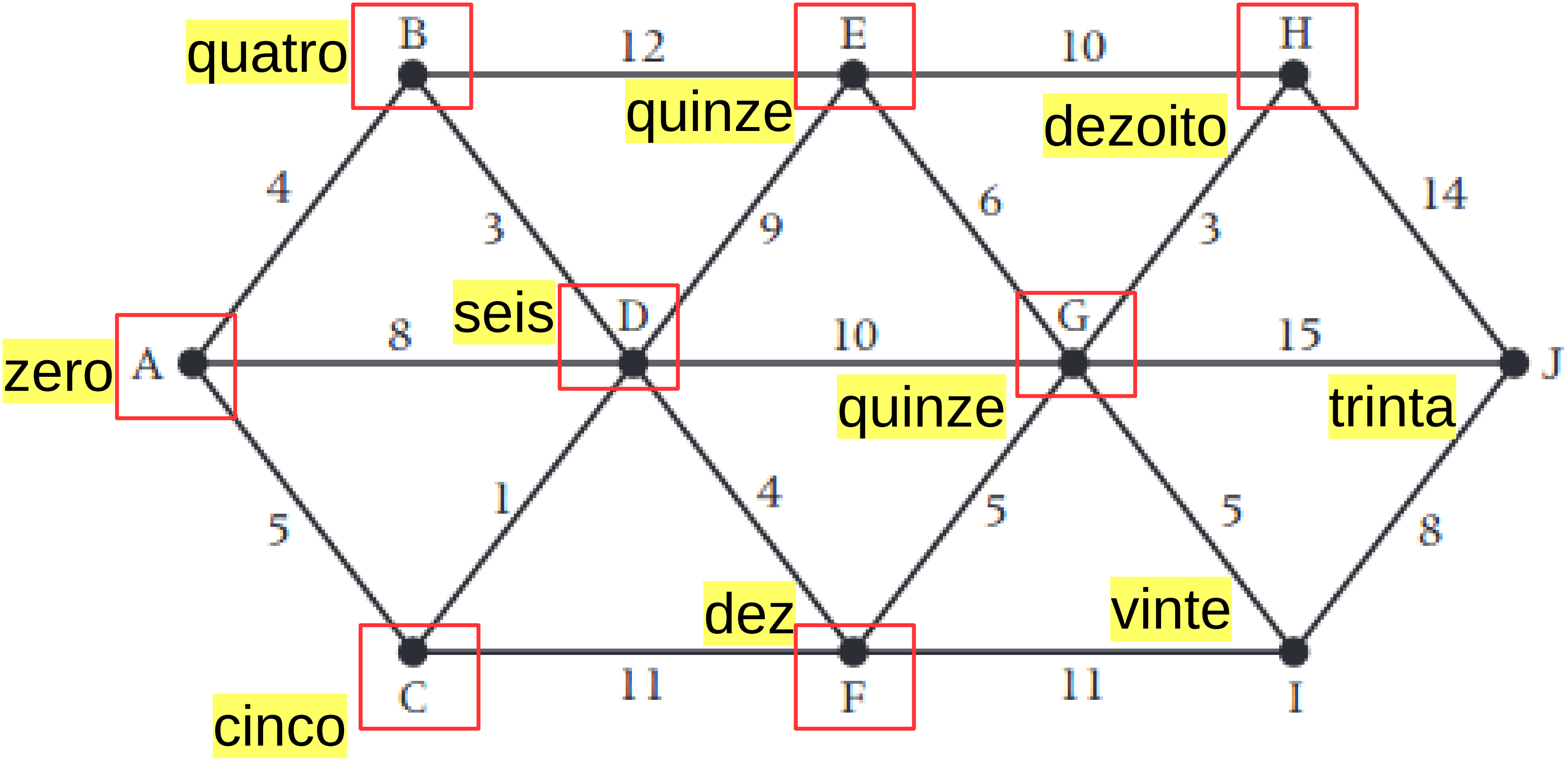
Exemplo 2

- Ache o caminho mais curto entre A e J:



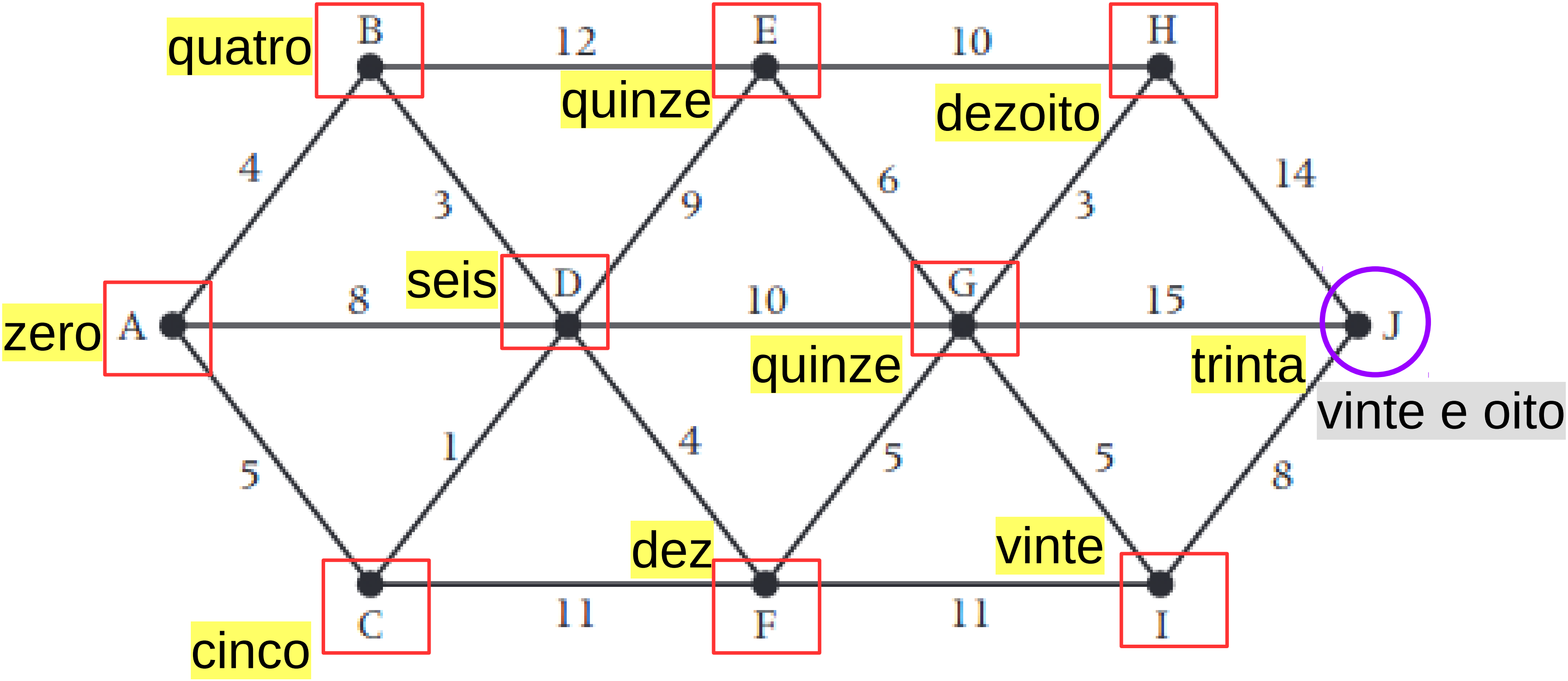
Exemplo 2

- Ache o caminho mais curto entre A e J:



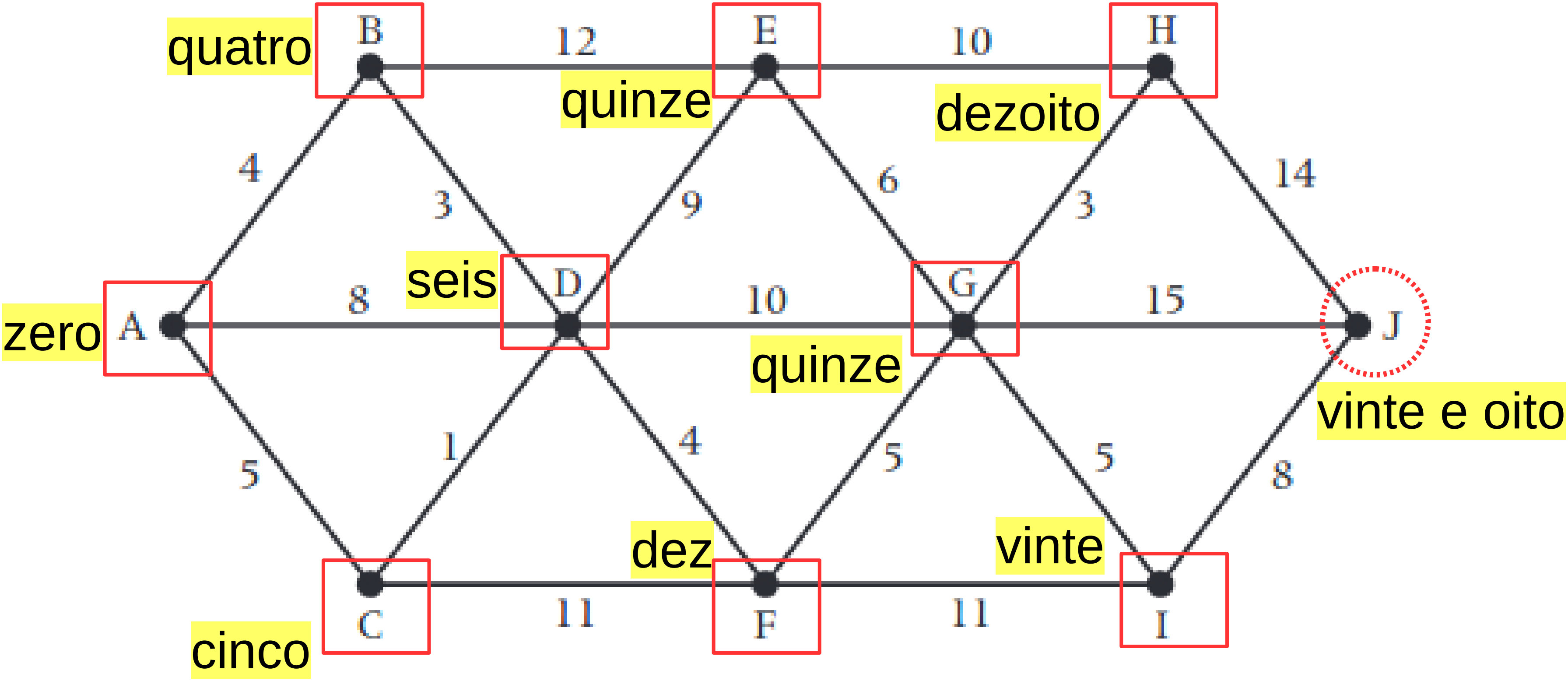
Exemplo 2

- Ache o caminho mais curto entre A e J:



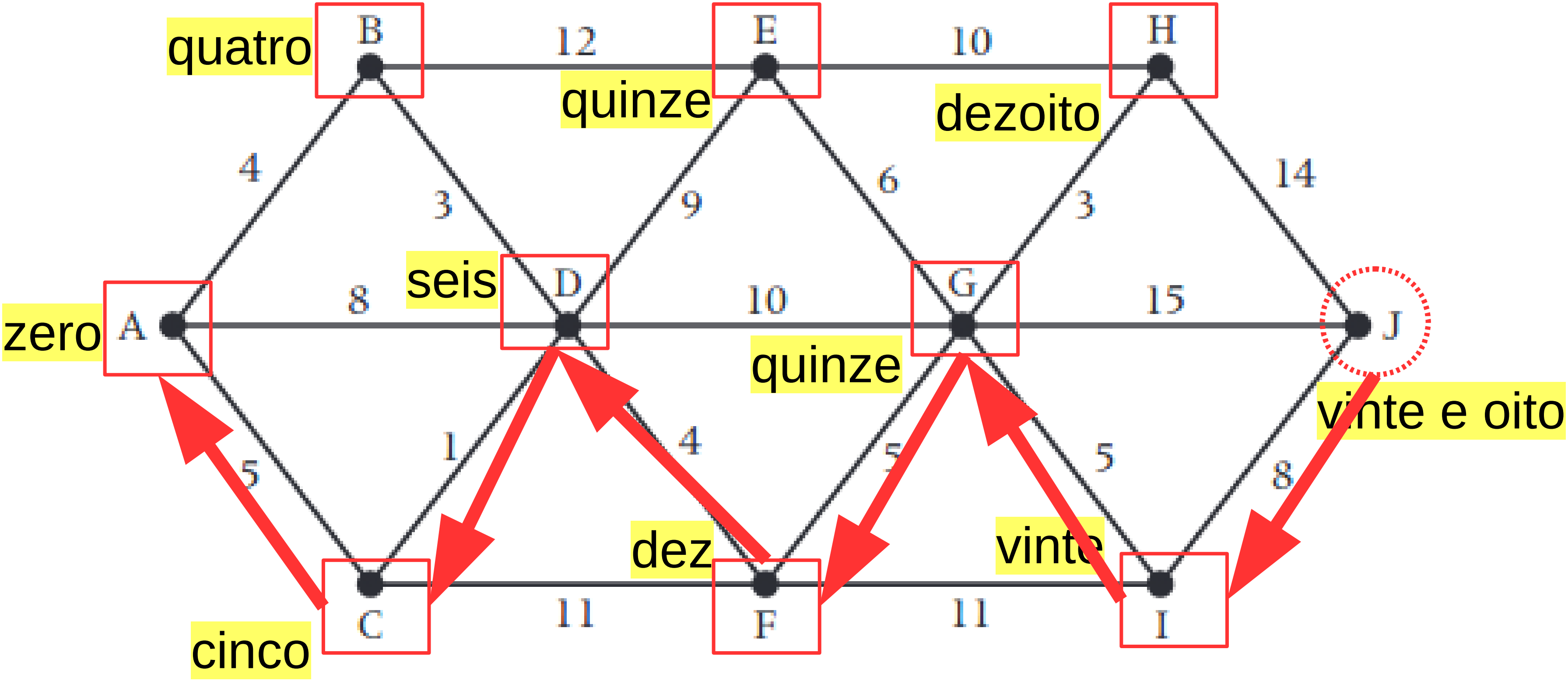
Exemplo 2

- Ache o caminho mais curto entre A e J:



Exemplo 2

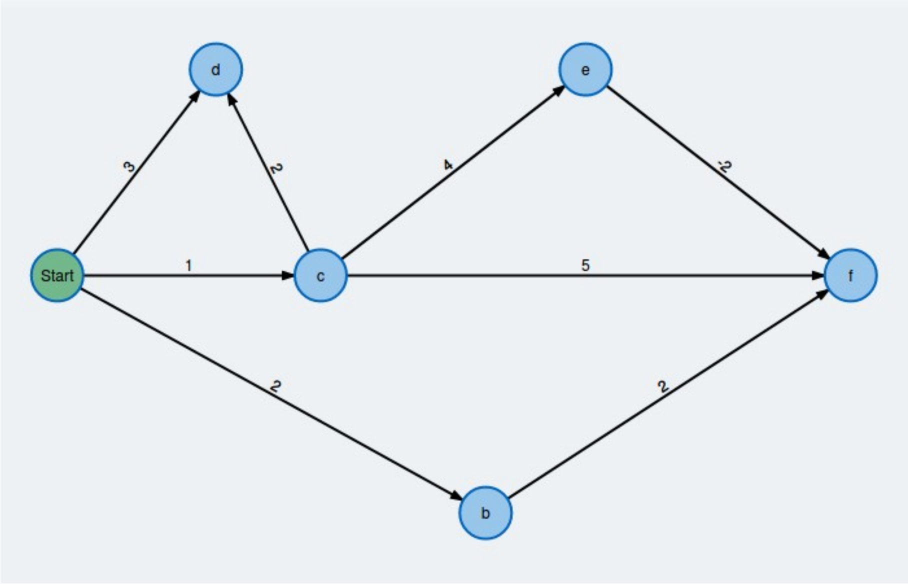
- Ache o caminho mais curto entre A e J:



Exercícios

Exercício

- Considere o grafo abaixo e calcule o custo do caminho usando o algoritmo de Dijkastra.



Exercício

- Demonstre passo a passo como calcular o menor caminho entre o ponto A e H.