Grafos

Algoritmos e Estruturas de Dados
Prof. Dr. Luciano Demétrio Santos Pacífico
{luciano.pacifico@ufrpe.br}

Conteúdo

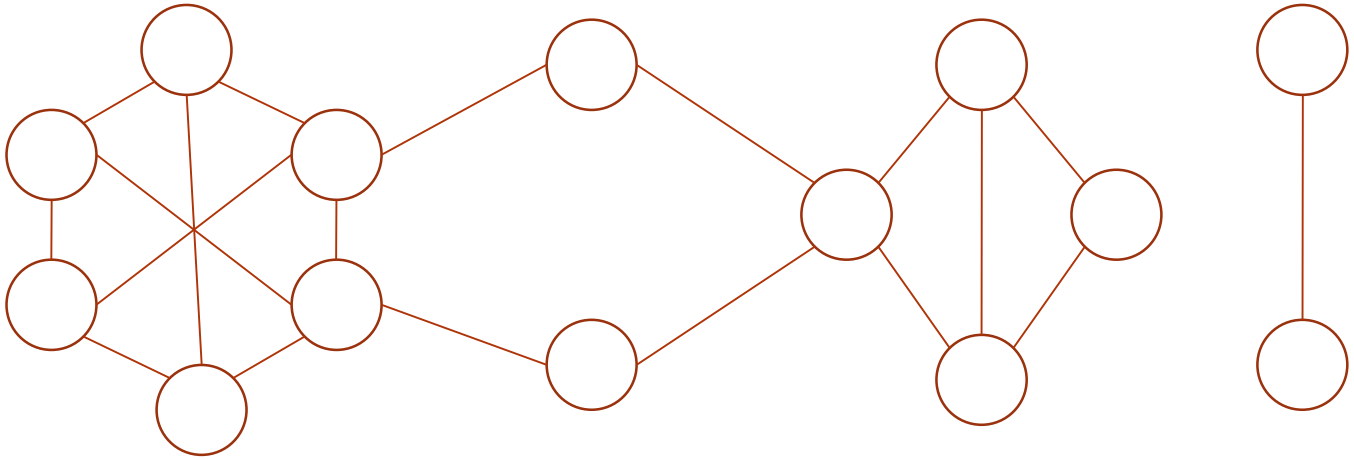
- **Introdução**
- **Árvore de Crescimento Mínimo**
- **Busca por Menor Caminho**

Introdução

UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Grafos

- Suponha que um grupo de computadores sejam interconectados como o diagrama abaixo:



- Esse diagrama é conhecido por **grafo**.

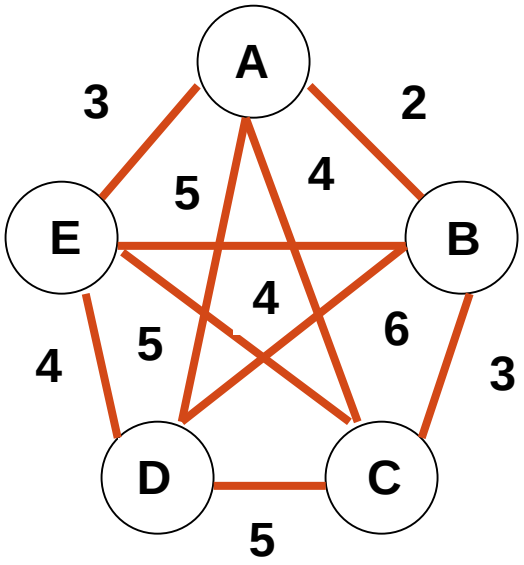
Introdução

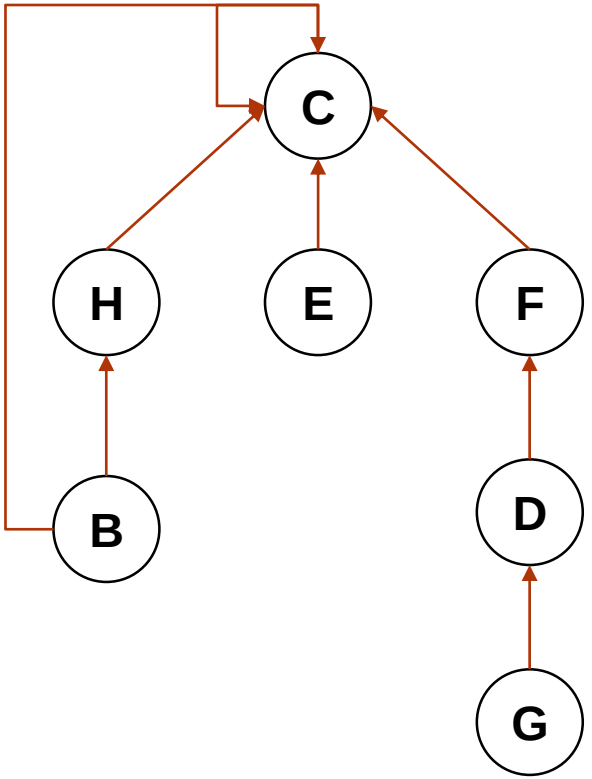
- Um grafo $G = (V, E)$ é uma representação de um conjunto de objetos (**vértices**) V e um conjunto de ligações (**arestas**) E.

- Grafos podem ser:
  - Ponderados ou não-ponderados;
  - Direcionados ou não-direcionados;
  - Cíclicos ou acíclicos;
  - Conectados ou não-conectados.

Problema do Caixeiro-Viajante

- Exemplos:



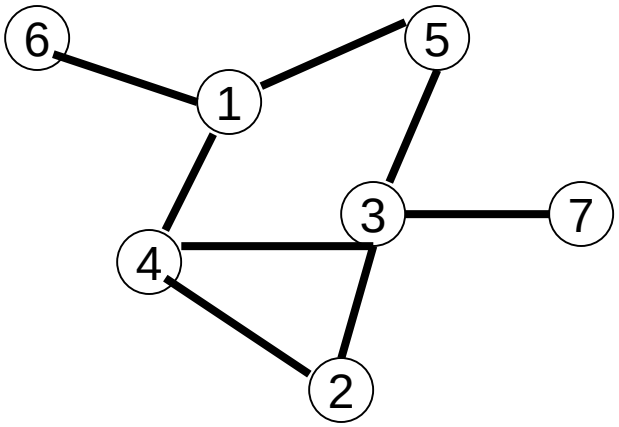


Introdução

- Outra forma para a representação de grafos é através de uma *matriz de conectividades*.
- Se o grafo for não direcionado, essa matriz *M* será simétrica.
- Quando não há conexão entre dois vértices *v* e *w*, $M_{vxw} = 0$.
- Se o grafo *G* não for ponderado, se dois vértices v e w forem conectados, $M_{vxw} = 1$.
- Se o grafo *G* for ponderado, a posição M_{vxw} da matriz M armazenará o peso (ou custo de caminho) entre os vértices *v* e *w*.

Introdução

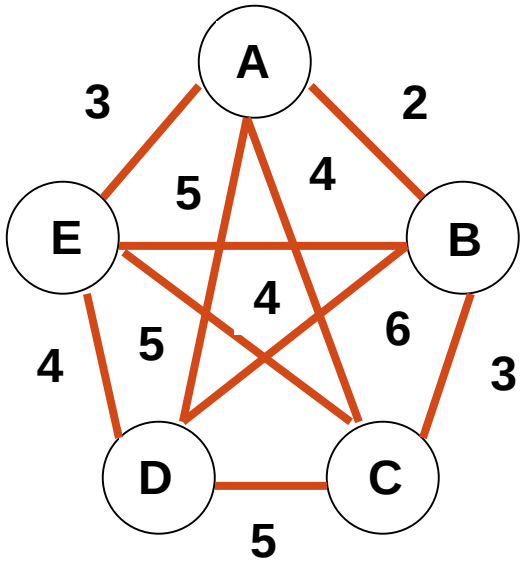
- Exemplo:



V	1	2	3	4	5	6	7
1	0	0	1	1	1	1	0
2	0	0	1	1	0	0	0
3	1	1	0	1	0	0	1
4	1	1	1	0	0	0	0
5	1	0	1	0	0	0	0
6	1	0	0	0	0	0	0
7	0	0	1	0	0	0	0

Introdução

- Exemplo 2:



V	A	B	C	D	E
A	0	2	4	5	3
B	2	0	3	6	5
C	4	3	0	5	5
D	5	6	5	0	4
E	3	5	5	4	0

Problemas em Grafos

- Os grafos são de grande importância para a ciência da computação e matemática teórica devido à existência de centenas de problemas computacionais interessantes que fazem uso de tais estruturas.

- Exemplos de problemas grafos:

  - Busca por Caminho;
  - Ciclos Hamiltonianos;
  - Caixeiro-Viajante;
  - Coloração de grafos,
  - ...

Árvore de Crescimento Mínimo

Introdução

- A modelagem de circuitos eletrônicos frequentemente precisa fazer os pinos de muitos componentes equivalentes eletronicamente, conectando-os juntos.

- Para interconectar um conjunto de *n* pinos, podemos usar um arranjo de *n* - 1 fios, cada um conectando dois pinos.

- De todos os arranjos possíveis, usamos aquele que fizer uso da menor quantidade de fios possível.

Introdução

- Podemos modelar o problema do posicionamento de elementos em um circuito através de um grafo $G(V, E)$ conectado e não-direcionado.

- Para o grafo acima, temos que V é o conjunto de pinos, E é o conjunto de possíveis interconexões entre dois pares de pinos e cada aresta $(u, v) \in E$ possui um custo de conexão (peso) associado $w(u, v) \in E$.

Introdução

- O objetivo é então encontrar um subconjunto acíclico $T \subseteq E$ que conecte todos os vértices e que o peso total

$$w(T) = \sum_{(u,v) \in T} w(u,v)$$

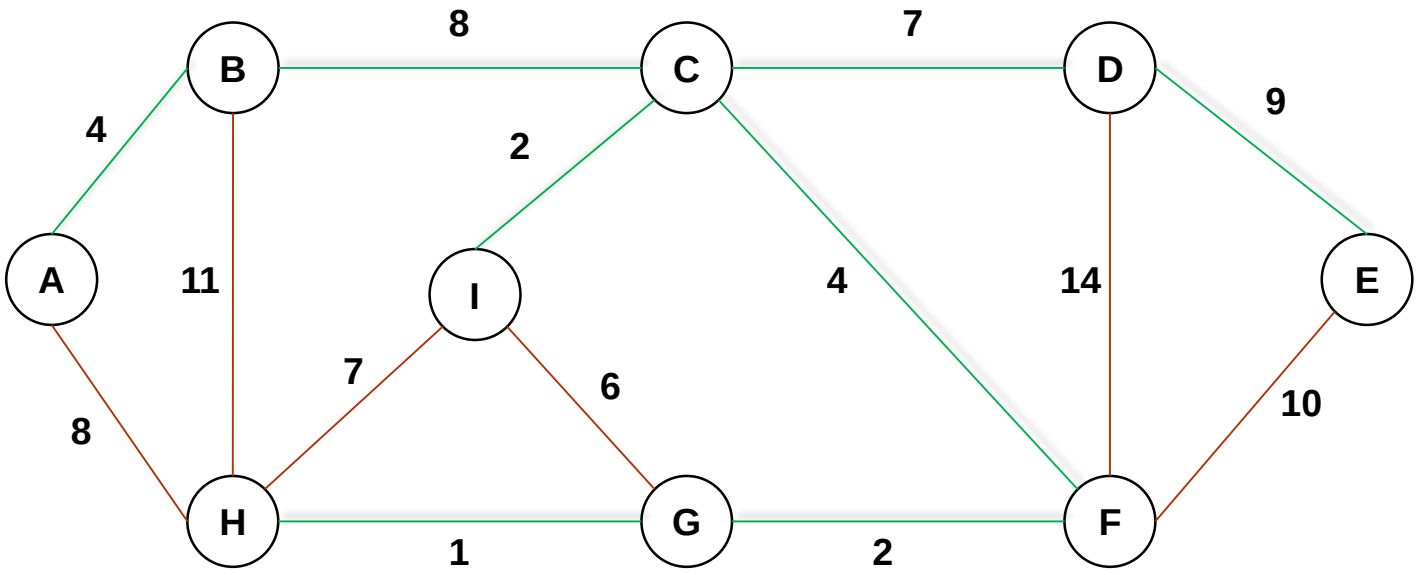
seja o menor possível.

Introdução

- Dado que *T* é acíclico e conecta todos os vértices, o mesmo pode ser representado como uma árvore, sendo essa árvore chamada de **árvore de crescimento**.

- O problema de determinar a árvore *T* é conhecido como o problema da **árvore de crescimento mínimo**.

Exemplo



Árvore de Crescimento Mínimo

- Suponha que temos um grafo $G = (V, E)$, conectado, **não-direcionado e ponderado** através de uma função $w: E \rightarrow \mathbb{R}$ e desejamos encontrar uma árvore de crescimento mínimo para G.

- Usando uma abordagem gulosa genérica, podemos fazer com que a árvore de crescimento mínimo seja montada a partir de uma aresta (u, v) por vez.

Árvore de Crescimento Mínimo

- A regra que será adota é:
  - No início de cada iteração do algoritmo, *A* é um subconjunto de uma árvore de crescimento mínimo.
- A cada etapa, determinamos uma aresta (*u*, *v*) para adicionarmos à *A* de forma que a regra acima não seja violada.
- Essa aresta é chamada de aresta segura pelo fato de que podemos adicioná-la à *A* sem que *A* deixe de ser subconjunto de uma árvore de crescimento mínimo.

Árvore de Crescimento Mínimo

procedimento *MST_generica*(*G*, *w*)

$A \leftarrow \emptyset$;

enquanto *A* não formar uma MST **faça**

Encontre uma aresta segura (*u*, *v*) para *A*;

$A \leftarrow A \cup \{(u, v)\}$;

Retorne A;

fim_MST_generica

Algoritmo Prim

- No algoritmo Prim as arestas em *A* sempre formarão **uma única árvore**.

- O algoritmo Prim começa sua execução a partir de um vértice arbitrário em *G*.*V*, sendo o crescimento em cada etapa representado pela inclusão em *A* de uma *aresta leve* que conecta A a um vértice isolado.

Algoritmo Prim

- Uma representação para o algoritmo Prim leva em consideração que todos os vértices ainda não pertencem à árvore se encontram em uma *lista de menor prioridade* *Q* baseada no atributo *chave*.

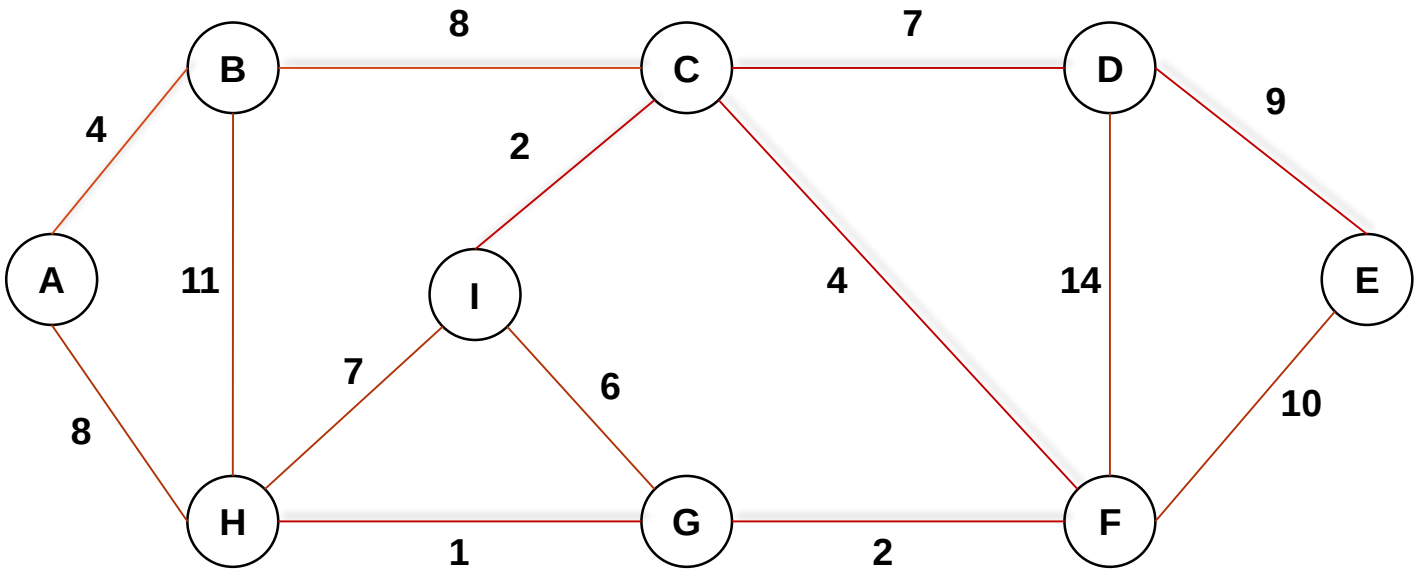
- Para cada vértice *v*, o atributo *v.chave* é o peso mínimo das arestas que conectam *v* à árvore.

- Se não houver conexão entre *v* e algum vértice na árvore, tem-se que *v.chave* = +∞.

Algoritmo Prim

```
1. //G -> é o grafo, composto de um conjunto de vértices G.V
2. //n -> número de vértices em G.V
3. //w -> Matriz simétrica representando os pesos das arestas
4. //r -> vértice que será usado como ponto de partida pelo Prim
5. procedimento encontrarPrimMST(G, w, r, n)
6.   para cada u em G.V
7.       u.chave = Inf
8.       u.pai = NIL
9.   r.chave = 0
10.  //vide Aula sobre Heaps para saber como criar uma Lista de Prioridade
11.  Q = criarListaPrioridadeMinima(G.V, n)
12.  enquanto Q.tamanhoHeap > 0
13.      u = extrairMinimo(Q, Q.tamanhoHeap)
14.      //vizinhança pode ser obtida por w[u][v] != 0
15.      para cada v em u.vizinhança
16.          se (v ∈ Q) e (w[u][v] < v.chave) //pseudocódigo simplificado por
17.              v.pai = u                    //questão de espaço
18.              v.chave = w[u][v]
```

Exemplo - Prim



Exemplo - Prim

- Conjunto de Vértices:
  - A, B, C, D, E, F, G, H, I

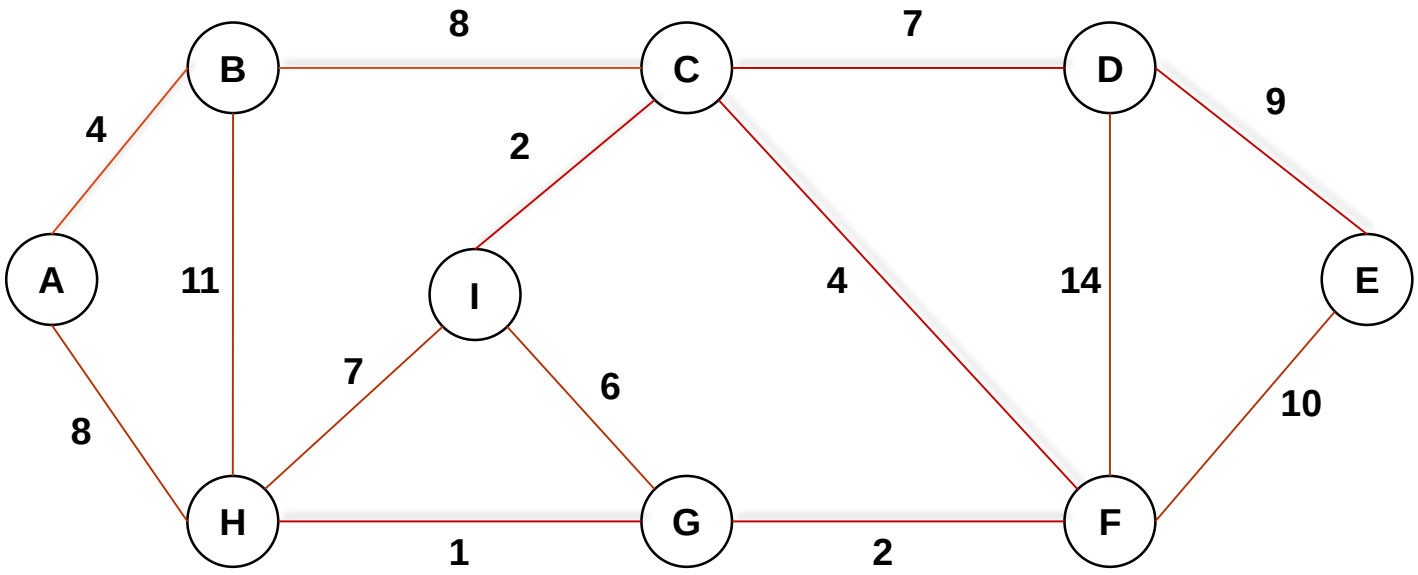
- Arestas e Pesos:
  - {A, B} = 4, {A, H} = 8,
  - {B, C} = 8, {B, H} = 11,
  - {C, D} = 7, {C, F} = 4, {C, I} = 2,
  - {D, E} = 9, {D, F} = 14,
  - {E, F} = 10,
  - {F, G} = 2,
  - {G, H} = 1, {G,I} = 6,
  - {H, I} = 7

Exemplo - Prim

- A cada iteração do laço **`enquanto`**, um vértice será removido de Q, e terá sua vizinhança explorada, na tentativa de aliviar os seus vizinhos que ainda estão em Q.

Iteração	1	2	3	4	5	6	7	8	9	10
A	Inf									NIL
B	Inf									NIL
C	Inf									NIL
D	Inf									NIL
E	Inf									NIL
F	Inf									NIL
G	Inf									NIL
H	Inf									NIL
I	Inf									NIL
Aresta	-									-

Exemplo - Prim

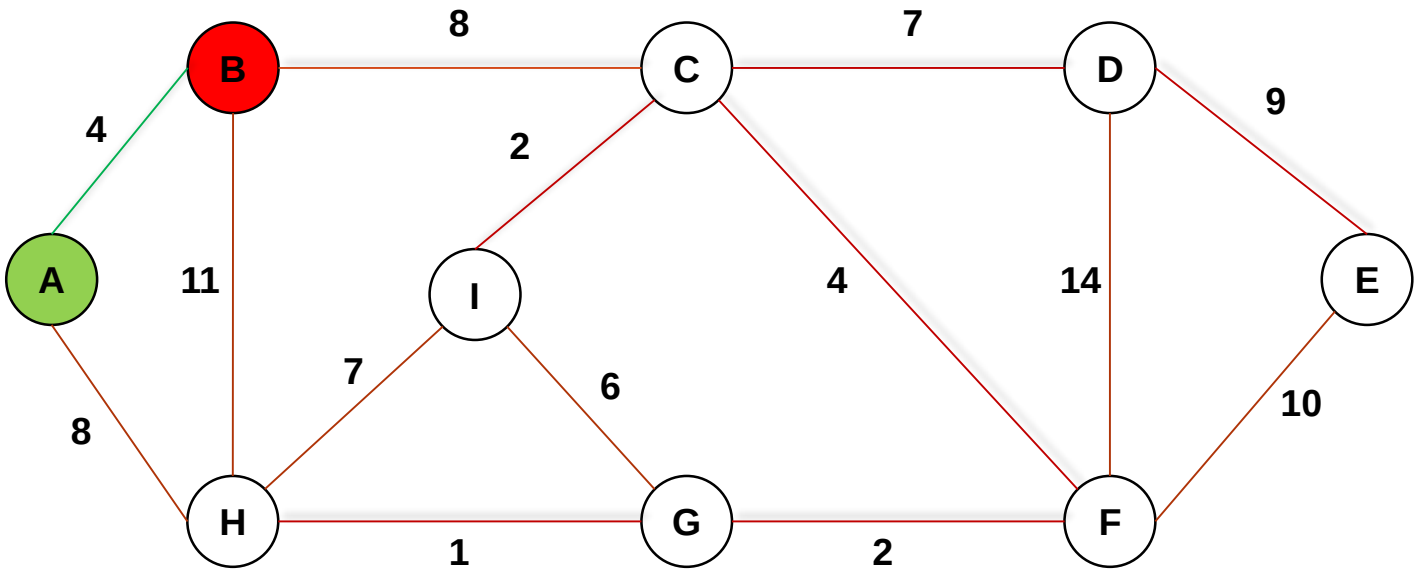


Exemplo - Prim

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	6	7	8	9	Pai
A*	0*									NIL*
B	4									A
C	Inf									NIL
D	Inf									NIL
E	Inf									NIL
F	Inf									NIL
G	Inf									NIL
H	8									A
I	Inf									NIL
Aresta	-									-

Exemplo - Prim

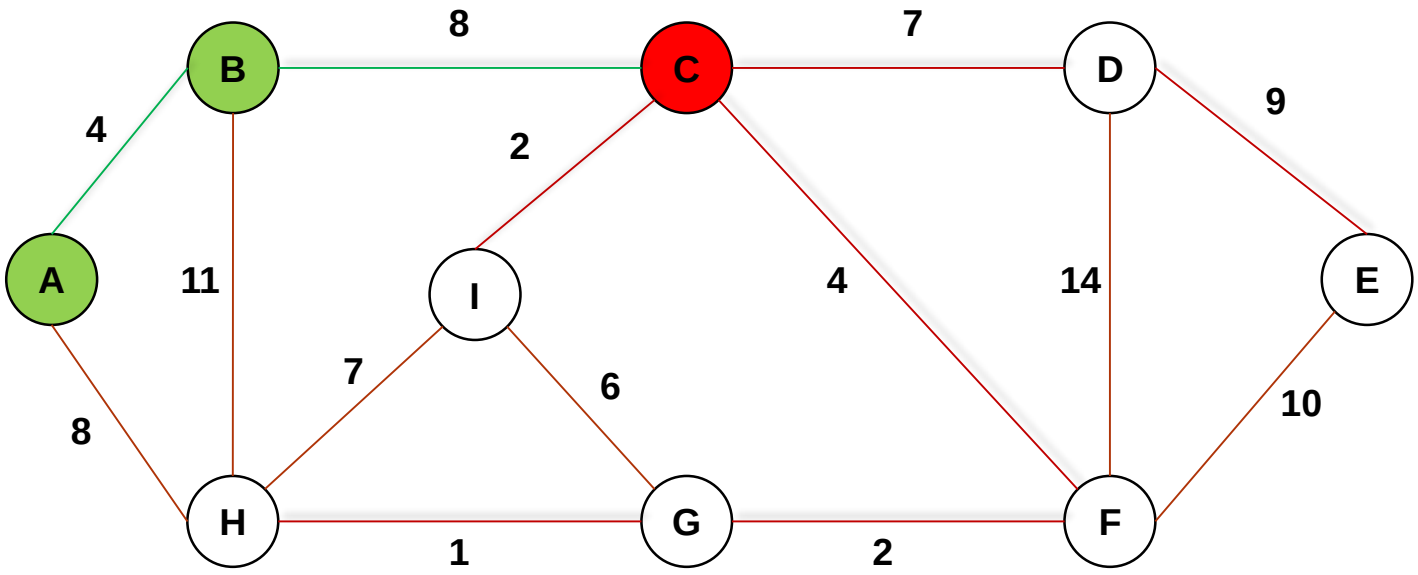


Exemplo - Prim

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	6	7	8	9	Pai
A*	0	0*								NIL*
B*	4	4*								A*
C	Inf	8								B
D	Inf	Inf								NIL
E	Inf	Inf								NIL
F	Inf	Inf								NIL
G	Inf	Inf								NIL
H	8	8								A
I	Inf	Inf								NIL
Aresta	-	{A,B}								-

Exemplo - Prim

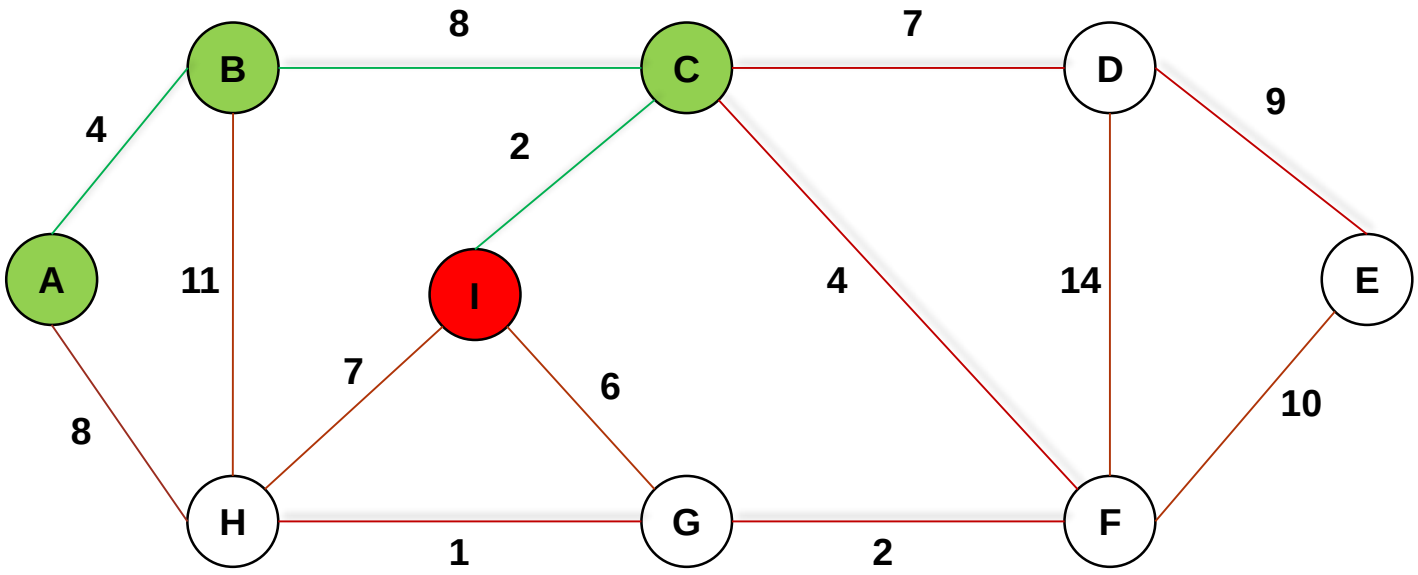


Exemplo - Prim

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	6	7	8	9	Pai
A*	0*	0*	0*							NIL*
B*	4	4*	4*							A*
C*	Inf	8	8*							B*
D	Inf	Inf	7							C
E	Inf	Inf	Inf							NIL
F	Inf	Inf	4							C
G	Inf	Inf	Inf							NIL
H	8	8	8							A
I	Inf	Inf	2							C
Aresta	-	{A,B}	{B,C}							-

Exemplo - Prim

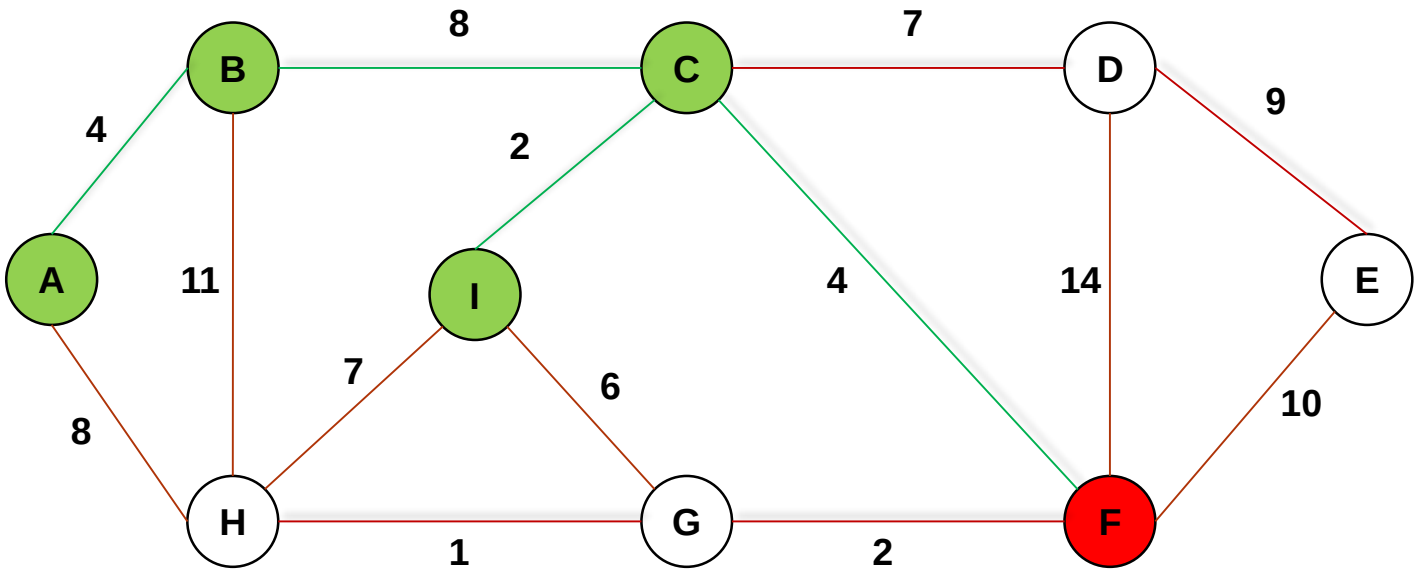


Exemplo - Prim

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	6	7	8	9	Pai
A*	0*	0*	0*	0*						NIL*
B*	4	4*	4*	4*						A*
C*	Inf	8	8*	8*						B*
D	Inf	Inf	7	7						C
E	Inf	Inf	Inf	Inf						NIL
F	Inf	Inf	4	4						C
G	Inf	Inf	Inf	6						I
H	8	8	8	7						I
I*	Inf	Inf	2	2*						C*
Aresta	-	{A,B}	{B,C}	{C,I}						-

Exemplo - Prim

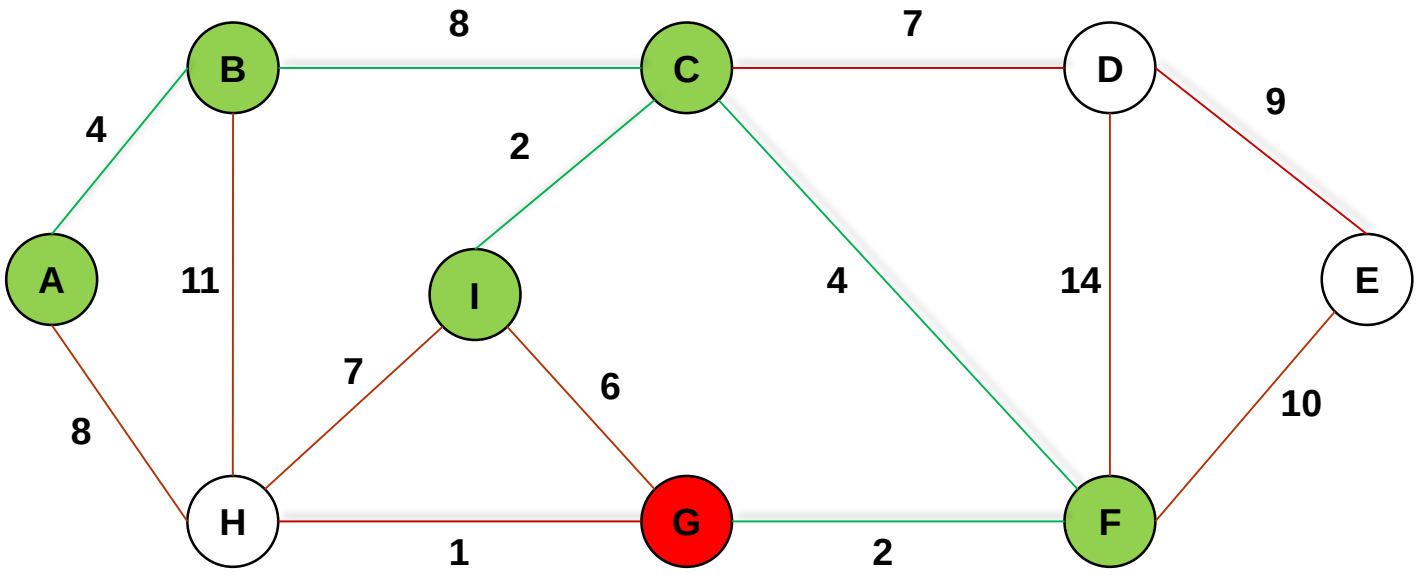


Exemplo - Prim

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	6	7	8	9	Pai
A*	0*	0*	0*	0*	0*					NIL*
B*	4	4*	4*	4*	4*					A*
C*	Inf	8	8*	8*	8*					B*
D	Inf	Inf	7	7	7					C
E	Inf	Inf	Inf	Inf	10					F
F*	Inf	Inf	4	4	4*					C*
G	Inf	Inf	Inf	6	2					F
H	8	8	8	7	7					I
I*	Inf	Inf	2	2*	2*					C*
Aresta	-	{A,B}	{B,C}	{C,I}	{C,F}					-

Exemplo - Prim

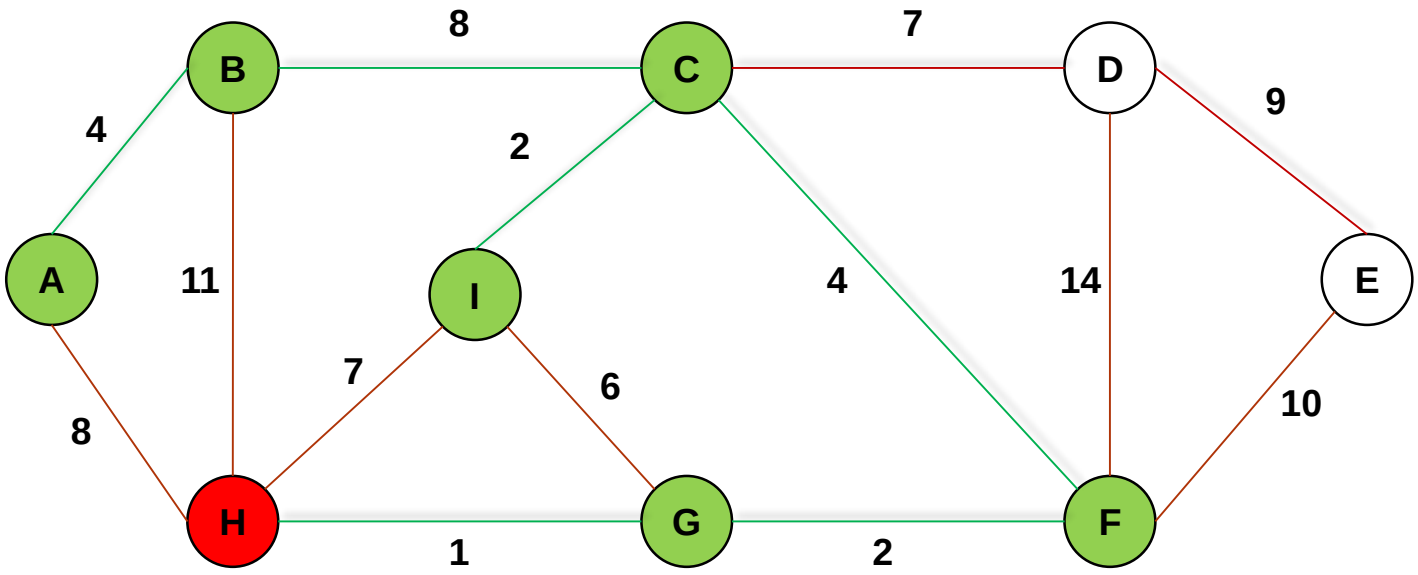


Exemplo - Prim

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	6	7	8	9	Pai
A*	0*	0*	0*	0*	0*	0*				NIL*
B*	4	4*	4*	4*	4*	4*				A*
C*	Inf	8	8*	8*	8*	8*				B*
D	Inf	Inf	7	7	7	7				C
E	Inf	Inf	Inf	Inf	10	10				F
F*	Inf	Inf	4	4	4*	4*				C*
G*	Inf	Inf	Inf	6	2	2*				F*
H	8	8	8	7	7	1				G
I*	Inf	Inf	2	2*	2*	2*				C*
Aresta	-	{A,B}	{B,C}	{C,I}	{C,F}	{F,G}				-

Exemplo - Prim

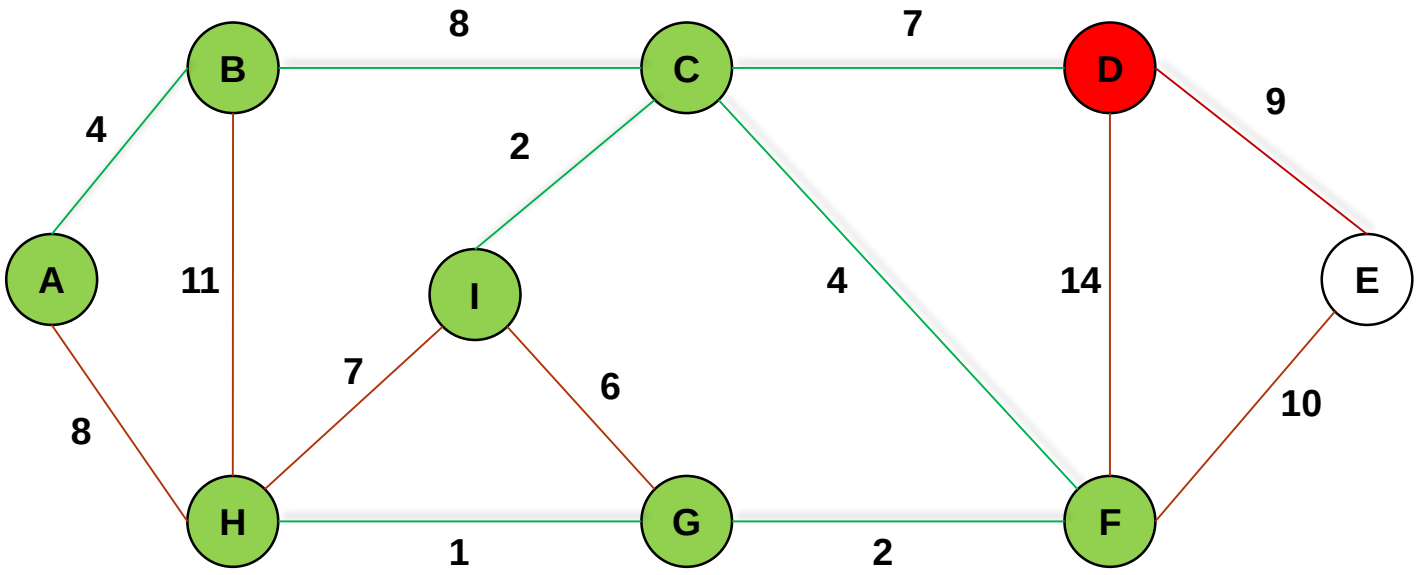


Exemplo - Prim

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	6	7	8	9	Pai
A*	0*	0*	0*	0*	0*	0*	0*			NIL*
B*	4	4*	4*	4*	4*	4*	4*			A*
C*	Inf	8	8*	8*	8*	8*	8*			B*
D	Inf	Inf	7	7	7	7	7			C
E	Inf	Inf	Inf	Inf	10	10	10			F
F*	Inf	Inf	4	4	4*	4*	4*			C*
G*	Inf	Inf	Inf	6	2	2*	2*			F*
H*	8	8	8	7	7	1	1*			G*
I*	Inf	Inf	2	2*	2*	2*	2*			C*
Aresta	-	{A,B}	{B,C}	{C,I}	{C,F}	{F,G}	{G,H}			-

Exemplo - Prim

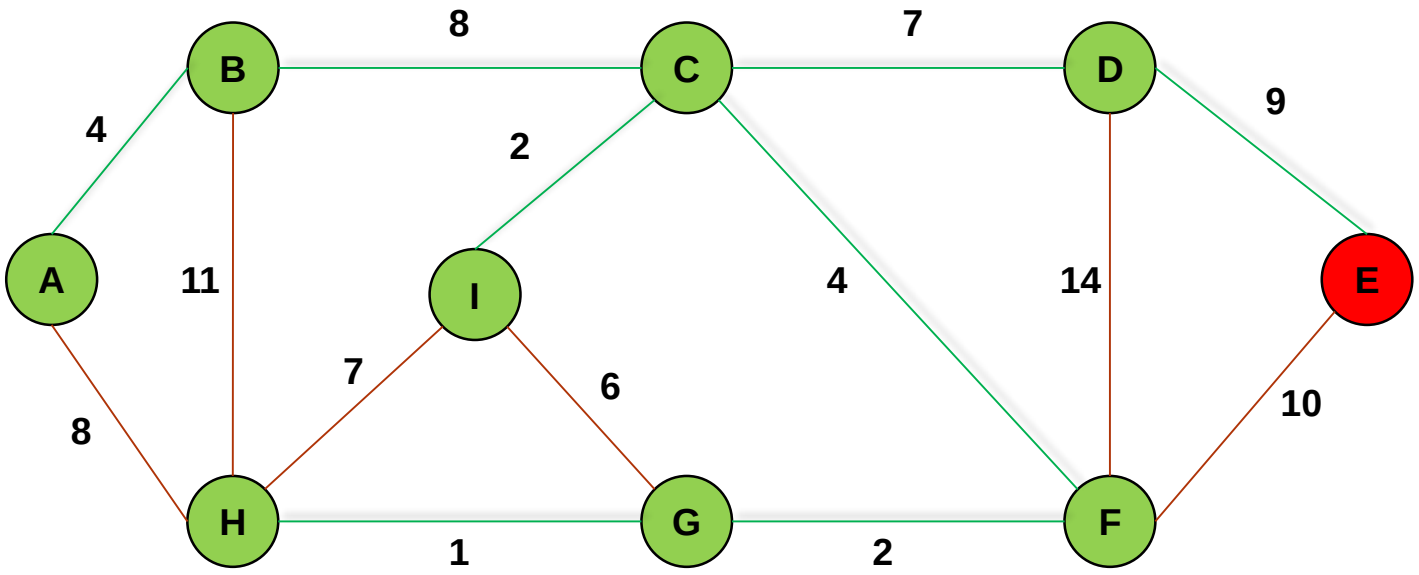


Exemplo - Prim

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	6	7	8	9	Pai
A*	0*	0*	0*	0*	0*	0*	0*	0*		NIL*
B*	4	4*	4*	4*	4*	4*	4*	4*		A*
C*	Inf	8	8*	8*	8*	8*	8*	8*		B*
D*	Inf	Inf	7	7	7	7	7	7*		C*
E	Inf	Inf	Inf	Inf	10	10	10	9		F
F*	Inf	Inf	4	4	4*	4*	4*	4*		C*
G*	Inf	Inf	Inf	6	2	2*	2*	2*		F*
H*	8	8	8	7	7	1	1*	1*		G*
I*	Inf	Inf	2	2*	2*	2*	2*	2*		C*
Aresta	-	{A,B}	{B,C}	{C,I}	{C,F}	{F,G}	{G,H}	{C,D}		-

Exemplo - Prim



Exemplo - Prim

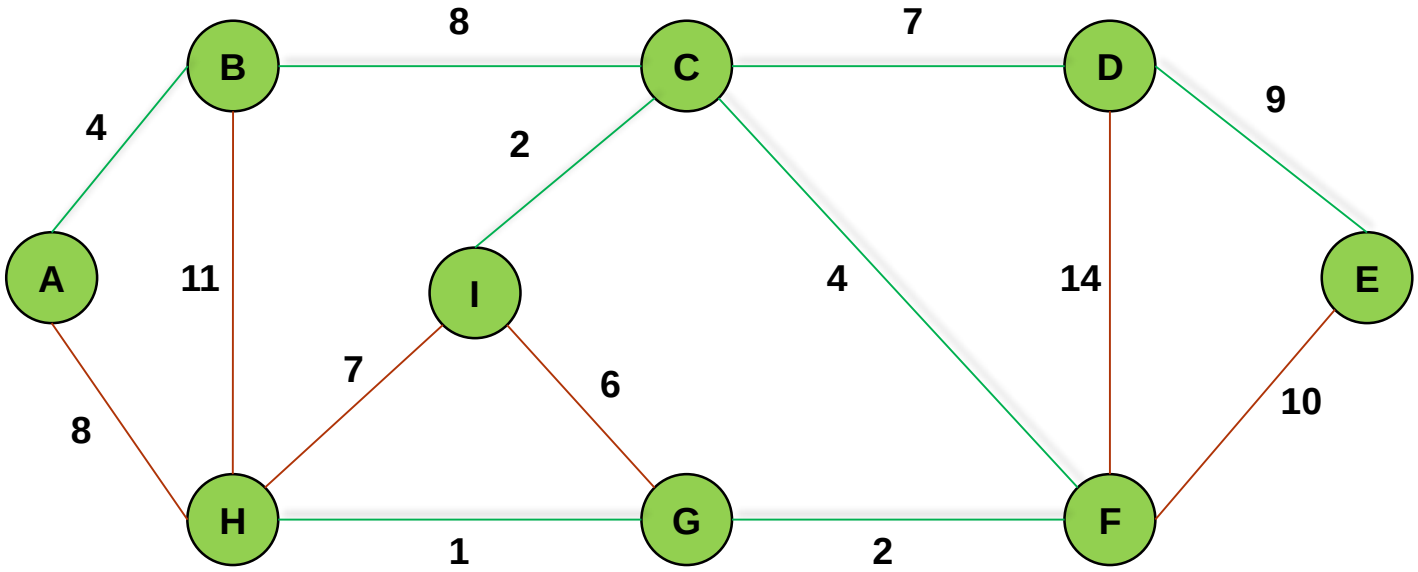
- O vértice `E` não alivia outros vértices, pois agora a lista `Q` está vazia... Mas o algoritmo ainda verifica sua vizinhança, como descrito no pseudocódigo.

Iteração	1	2	3	4	5	6	7	8	9	Pai
A*	0*	0*	0*	0*	0*	0*	0*	0*	0*	NIL*
B*	4	4*	4*	4*	4*	4*	4*	4*	4*	A*
C*	Inf	8	8*	8*	8*	8*	8*	8*	8*	B*
D*	Inf	Inf	7	7	7	7	7	7*	7*	C*
E*	Inf	Inf	Inf	Inf	10	10	10	9	9*	F*
F*	Inf	Inf	4	4	4*	4*	4*	4*	4*	C*
G*	Inf	Inf	Inf	6	2	2*	2*	2*	2*	F*
H*	8	8	8	7	7	1	1*	1*	1*	G*
I*	Inf	Inf	2	2*	2*	2*	2*	2*	2*	C*
Aresta	-	{A,B}	{B,C}	{C,I}	{C,F}	{F,G}	{G,H}	{C,D}	{F,E}	-

Exemplo - Prim

- Saída do laço após `Q.tamanhoHeap == 0`

- É possível encontrar o conjunto final de arestas (árvore geradora mínima) observando-se o pai de cada um dos vértices.



Algoritmo Prim

- Complexidade do algoritmo Prim (considerando uma implementação com Heap):
  - Construção do heap - $O(|E|)$
  - Loop - $O(|V| \log|E| + |E| \log|E|)$
    $= O(|E| \log|E|)$
- Custo Total: $O(|E| \log|E|)$

Busca por Menor Caminho

Busca por Menor Caminho

- Um caminho *C* num grafo *G* é mínimo se não existe outro caminho com mesma origem e mesmo término que *C*, mas comprimento menor que o de *C*.

- A distância de um vértice *s* a um vértice *t* em um grafo *G* é o comprimento de um caminho mínimo de *s* a *t*. Se não existe caminho algum de *s* a *t*, a distância de *s* a *t* é infinita.

Busca por Menor Caminho

- A distância de *s* a *t* é *d,* se, e somente se:
  - Existe um caminho de comprimento *d* de *s* a t;
  - Nenhum caminho de *s* a *t* tem comprimento menor que *d*.
- Em geral, em um grafo direcionado, a distância de um vértice *s* a um vértice *t* é diferente da distância de *t* a *s*. Se o grafo é não-direcionado, entretanto, as duas distâncias são iguais.

Busca por Menor Caminho

- Para a busca de caminhos mínimos em grafos há algoritmos específicos que executam a tarefa.

- Entretanto há formas específicas para tratar o problema, sendo diferentes quando busca-se caminhos mínimos a partir de um dado vértice ou quando se buscam os caminhos mínimos entre todos os pares de vértices.

Busca por Menor Caminho

- Problema dos Caminhos Mínimos com Origem Fixa: Dado um vértice *s* de um grafo com custos nos arcos, encontrar, para cada vértice *t* que pode ser alcançado a partir de *s*, um caminho mínimo simples de *s* a *t*.

- Todos os algoritmos para esses problemas exploram a seguinte propriedade básica:

  - Propriedade Triangular: Para quaisquer vértices *x*, *y* e *z* de um grafo com custos não-negativos nos arcos, tem-se:

$$d(x, z) \leq d(x, y)+d(y, z)$$

sendo $d(i, j)$ a distância de *i* a *j*.

Busca por Menor Caminho

- Um algoritmo eficiente para a obtenção do caminho mínimo em grafos com custos não-negativos é o chamado algoritmo de Dijkstra.

- O algoritmo pode ser usado, em particular, para encontrar um caminho de custo mínimo de um dado vértice a outro, ou a todos os outros vértices.

Busca por Menor Caminho

1. Atribui-se uma distância infinita para todos os pares de vértices, exceto o vértice origem.

2. Marque todos os vértices como não visitados e defina o vértice inicial como vértice corrente.

3. Para este vértice corrente, considere todos os seus vértices vizinhos não visitados e calcule a distância a partir do vértice. Se a distância for menor do que a definida anteriormente, substitua a distância.

4. Quando todos os vizinhos do vértice corrente forem visitados, marque-o como visitado, o que fará com que ele não seja mais analisado (sua distância é mínima e final).

5. Eleja o vértice não visitado com a menor distância (a partir do vértice inicial) como o vértice corrente e continue a partir do passo 3.

Busca por Menor Caminho

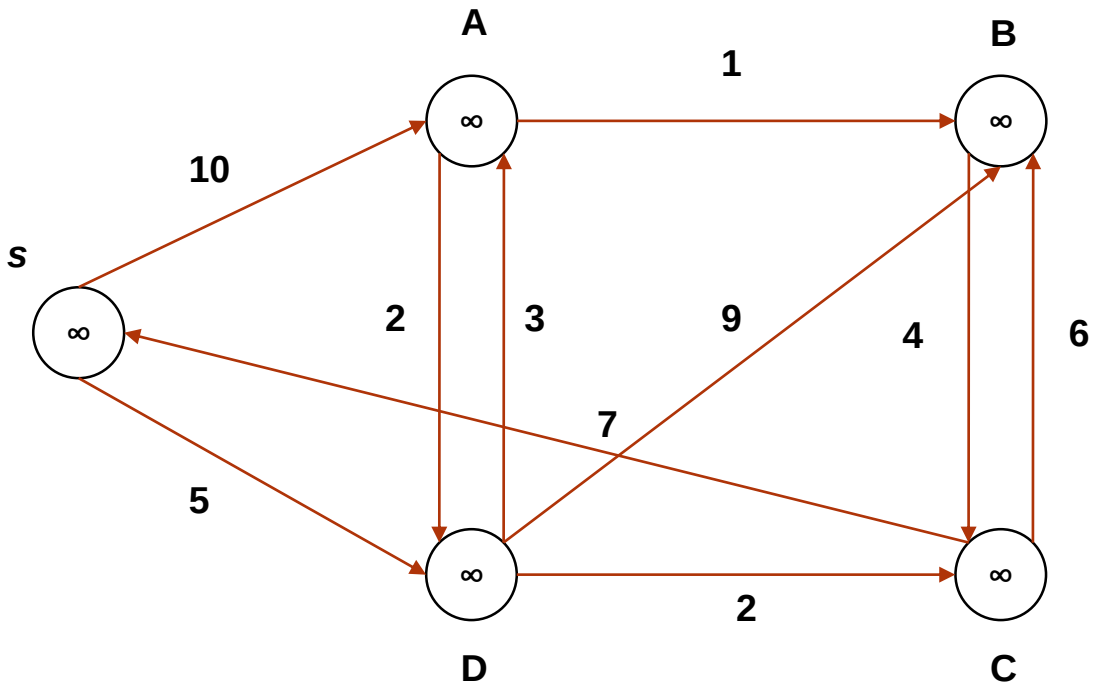
```
1. //G -> Grafo
2. //s -> vértice escolhido como fonte
3. procedimento iniciarFonteUnica(G, s)
4.     para cada v em G.V
5.         v.d = Inf
6.         v.pai = NIL
7.     s.d = 0

1. //w -> matriz de pesos das arestas
2. procedimento relaxarNo(u, v, w)
3.     se v.d > u.d + w[u][v]
4.         v.d = u.d + w[u][v]
5.         v.pai = u
```

Busca por Menor Caminho

```
1.  //G -> Grafo
2.  //w -> matriz de pesos das arestas
3.  //s -> vértice escolhido como fonte
4.  //n -> número de vértices no grafo
5.  procedimento Dijkstra(G, w, s, n)
6.      iniciarFonteUnica(G, s)
7.      Q = criarListaPrioridadeMinima(G.V, n)
8.      enquanto Q.tamanhoHeap > 0
9.          u = extrairMinimo(Q, Q.tamanhoHeap)
10.         //vizinhança pode ser obtida por w[u][v] != 0
11.         para cada v em u.vizinhança
12.             relaxarNo(u, v, w)
```

Exemplo - Dijkstra

- Para a busca de caminho mais curto, o grafo pode ser representado por uma matriz que **não será necessariamente simétrica**, dado que o grafo é **direcionado**.



Exemplo - Dijkstra

- Conjunto de Vértices:
  - s, A, B, C, D

- Arestas e Pesos:
  - {s, A} = 10, {s, D} = 5
  - {A, B} = 1, {A, D} = 2
  - {B, C} = 4
  - {C, B} = 6, {C, s} = 7
  - {D, A} = 3, {D, B} = 9, {D, C} = 2.

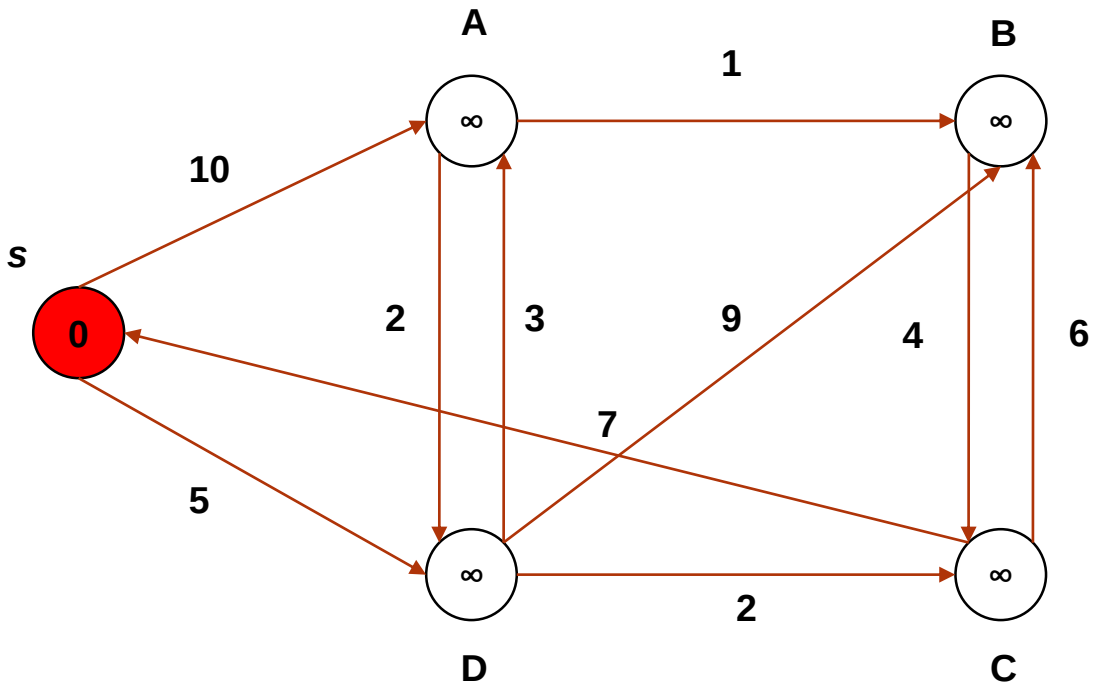
Exemplo - Dijkstra

- A cada iteração do laço **`enquanto`**, um vértice será removido de Q, e terá sua vizinhança explorada, na tentativa de aliviar os seus vizinhos que ainda estão em Q.

Iteração						Pai
S	Inf					NIL
A	Inf					NIL
B	Inf					NIL
C	Inf					NIL
D	Inf					NIL
Aresta	-					-

Exemplo - Dijkstra

- Para a busca de caminho mais curto, o grafo pode ser representado por uma matriz que **não será necessariamente simétrica**, dado que o grafo é **direcionado**.

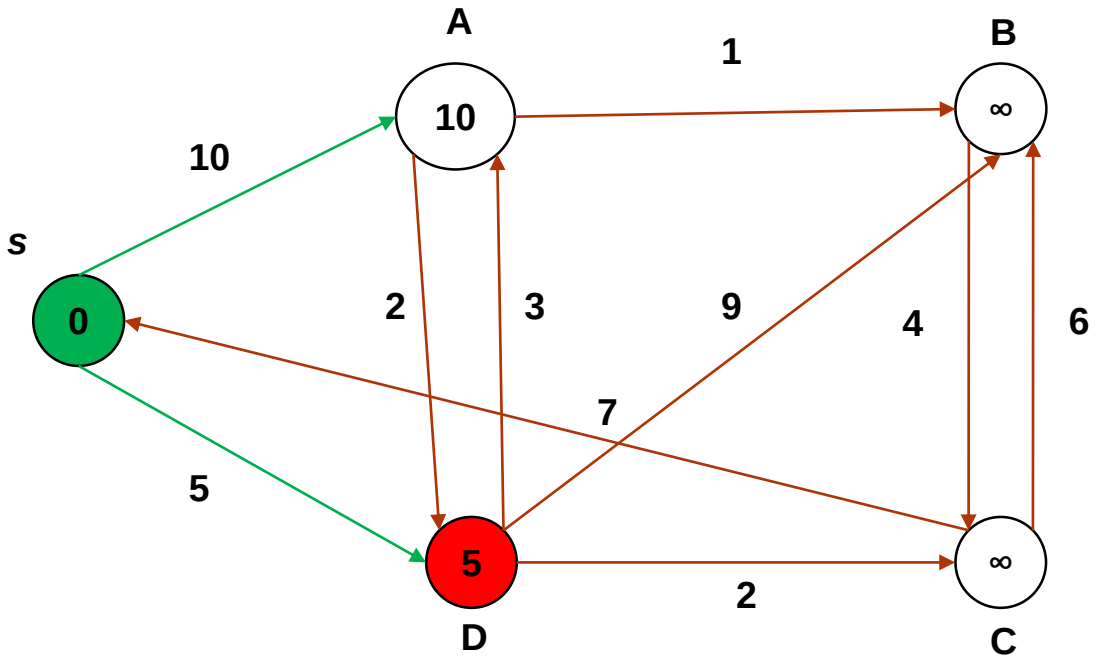


Exemplo - Dijkstra

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	Pai
S*	0*					NIL*
A	10					A
B	Inf					NIL
C	Inf					NIL
D	5					A
Aresta	-					-

Exemplo - Dijkstra

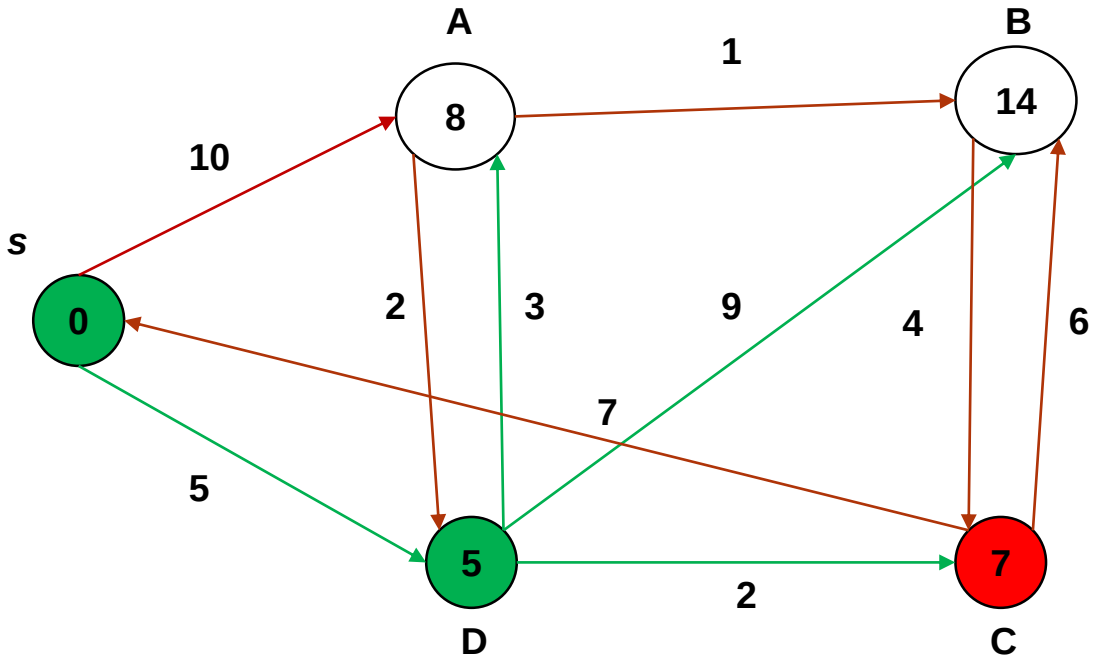


Exemplo - Dijkstra

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	Pai
S*	0*	0*				NIL*
A	10	8				D
B	Inf	14				D
C	Inf	7				D
D*	5	5*				A*
Aresta	-	{A,D}				-

Exemplo - Dijkstra

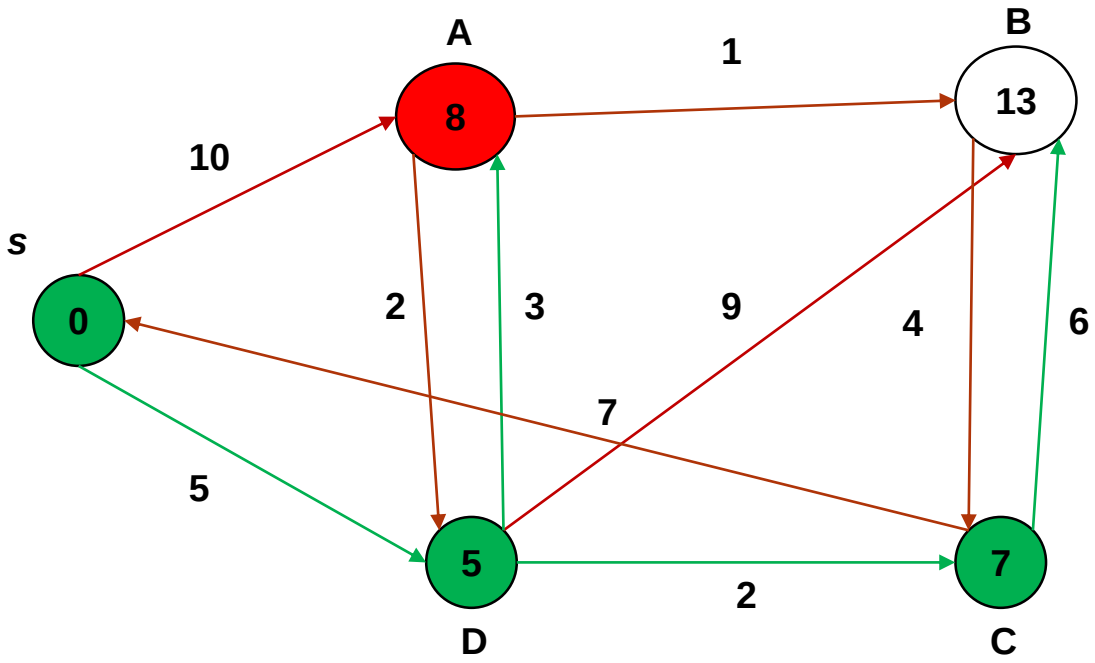


Exemplo - Dijkstra

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	Pai
S*	0*	0*	0*			NIL*
A	10	8	8			D
B	Inf	14	13			C
C*	Inf	7	7*			D*
D*	5	5*	5*			A*
Aresta	-	{A,D}	{D,C}			-

Exemplo - Dijkstra

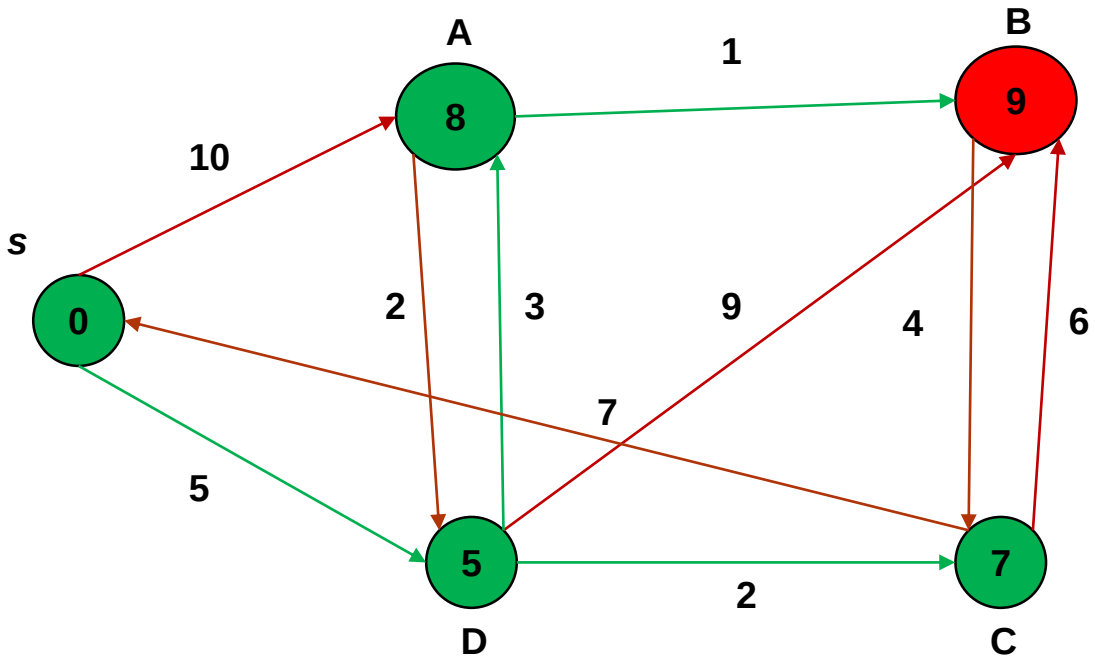


Exemplo - Dijkstra

- Vértices já removidos serão marcados com um *. Vértice em uso estará em vermelho. Empates serão decididos por escolha **lexicográfica**.

Iteração	1	2	3	4	5	Pai
S*	0*	0*	0*	0*		NIL*
A*	10	8	8	8*		D*
B	Inf	14	13	9		A
C*	Inf	7	7*	7*		D*
D*	5	5*	5*	5*		A*
Aresta	-	{A,D}	{D,C}	{D,A}		-

Exemplo - Dijkstra



Exemplo - Dijkstra

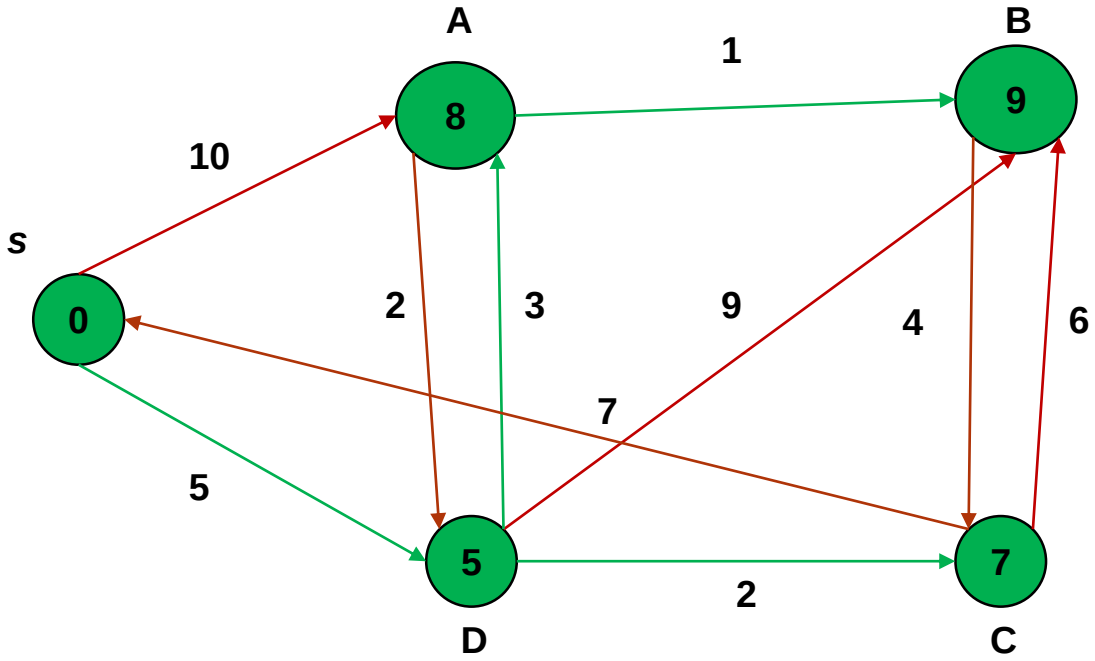
- O vértice `B` não alivia outros vértices, pois agora a lista `Q` está vazia... Mas o algoritmo ainda verifica sua vizinhança, como descrito no pseudocódigo.

Iteração	1	2	3	4	5	Pai
S*	0*	0*	0*	0*	0*	NIL*
A*	10	8	8	8*	8*	D*
B*	Inf	14	13	9	9*	A*
C*	Inf	7	7*	7*	7*	D*
D*	5	5*	5*	5*	5*	A*
Aresta	-	{A,D}	{D,C}	{D,A}	{A,B}	-

Exemplo - Dijkstra

- Saída do laço após `Q.tamanhoHeap == 0`

- É possível encontrar o conjunto final de arestas (caminhos da fonte até cada um dos demais vértices) observando-se o pai de cada um dos vértices.



Busca por Menor Caminho

- Complexidade do algoritmo de Dijkstra:
  - Inicialização - $O(|V|)$
  - Loop - $O((|V| + |E|) \log|E|)$
  - Existem $|V|$ deleções do heap (extrair o mínimo)
  - Existem no máximo $|E|$ atualizações (cada aresta só é analisada uma vez)
- Custo Total: $O((|V| + |E|) \log|E|)$

Referências

- CORMEN, H. T.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Introduction to Algorithms, 3rd ed., *Boston: MIT Press*, 2009.

- FEOFILOFF, Paulo. Algoritmos em Linguagem C. Editora Campus/Elsevier, 2009.

Grafos

Algoritmos e Estruturas de Dados
Prof. Dr. Luciano Demétrio Santos Pacífico
{luciano.pacifico@ufrpe.br}