UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Árvores Geradores Mínimas

Algoritmos em Grafos

Professor: Rennan Dantas

Universidade Federal do Ceará
Campus de Crateús

02, 04 e 09 de maio de 2022

Introdução

- Em projeto de circuitos eletrônicos, muitas vezes, é necessário que os pinos de vários componentes se tornem eletronicamente equivalentes, o que é conseguido ligando-os uns aos outros
- Para interconectar um conjunto de n pinos, podemos usar um arranjo de $n - 1$ fios, cada qual conectando dois pinos
- De todos os arranjos possíveis, aquele que utiliza a mínima quantidade de fio é normalmente o mais desejável

Introdução

- Podemos modelar esse problema de fiação com um grafo conexo não dirigido $G = (V, E)$, onde V é um conjunto de pinos, E é o conjunto de interconexões possíveis entre pares de pinos e, para cada aresta $(u, v) \in E$, temos um peso $w(u, v)$ que especifica o custo (a quantidade necessária de fio) para conectar u e v
- Então, desejamos encontrar um subconjunto acíclico $T \subseteq E$ que conecte todos os vértices e cujo peso total

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

é minimizado

- Visto que T é acíclico e conecta todos os vértices, deve formar uma árvore, que denominaremos árvore geradora, já que "gera" o grafo G

Árvores Geradoras Mínimas

O problema de determinar a árvore T é denominado problema da árvore geradora mínima

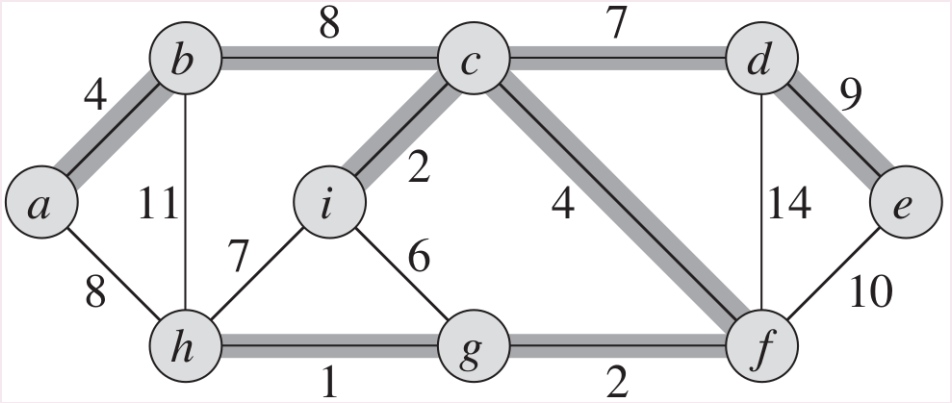

Figura: Fonte: Livro Algoritmos - Cormen

Introdução

- Veremos dois algoritmos gulosos que resolvem o problema
- Cada etapa de um algoritmo guloso deve fazer uma escolha entre várias opções possíveis
- A estratégia gulosa faz a escolha que é melhor no momento
- Em geral, tal estratégia não garante que sempre encontrará soluções globalmente ótimas para problemas

Introdução

- Porém, no caso da árvore geradora mínima, podemos provar que certas estratégias realmente produzem uma árvore geradora com o peso mínimo
- Veremos um algoritmo "genérico" de árvore geradora mínima que produz uma árvore geradora adicionando uma aresta por vez
- Por fim, veremos dois algoritmos que implementam o método genérico:
  - O algoritmo de Kruskal é semelhante ao algoritmo de componentes conexas
  - O algoritmo de Prim é semelhante ao algoritmo de caminhos mínimos de Dijkstra

Desenvolvendo uma árvore geradora mínima

- Suponha que temos um grafo conexo não dirigido $G = (V, E)$ com uma função peso $w : E \to \mathbb{R}$ e desejamos encontrar uma árvore geradora mínima para G
- Os dois algoritmos que consideraremos utilizam uma abordagem gulosa para o problema, embora os modos como aplicam essa abordagem sejam diferentes
- Essa estratégia gulosa é representada pelo método genérico apresentado a seguir, que desenvolve a árvore geradora mínima uma aresta por vez
- O método genérico administra um conjunto de arestas A, mantendo o seguinte invariante de laço:
  Antes de cada iteração, A é um subconjunto de alguma árvore geradora mínima

Desenvolvendo uma árvore geradora mínima

- Em cada etapa, determinamos uma aresta (u, v) que pode ser adicionada a A sem violar esse invariante, no sentido de que $A \cup \{(u, v)\}$ também é um subconjunto de uma árvore geradora mínima
- Denominamos tal aresta aresta segura para A, já que ela pode ser adicionada com segurança a A e, ao mesmo tempo, manter o invariante
- É claro que a parte complicada é encontrar uma aresta segura na linha 3
- Deve existir uma, já que, quando a linha 3 é executada, o invariante estabelece que existe uma árvore geradora T tal que $A \subseteq T$
- Dentro do corpo do laço while, A deve ser um subconjunto próprio de T, e portanto deve haver uma aresta $(u, v) \in T$ tal que $(u, v) \notin A$ e (u, v) é segura para A

Desenvolvendo uma árvore geradora mínima

```
GENERIC-MST(G, w)
1  A = ∅
2  while A does not form a spanning tree
3       find an edge (u, v) that is safe for A
4       A = A ∪ {(u, v)}
5  return A
```

Figura: Fonte: Livro Algoritmos - Cormen

Desenvolvendo uma árvore geradora mínima

- Daremos uma regra (Teorema 1) para reconhecer arestas seguras
- Mais a frente, veremos dois algoritmos que usam essa regra para encontrar arestas seguras eficientemente
- Primeiro, precisamos de algumas definições
- Um corte $(S, V - S)$ de um grafo não dirigido $G = (V, E)$ é uma partição de V
- A figura abaixo ilustra essa noção

Desenvolvendo uma árvore geradora mínima

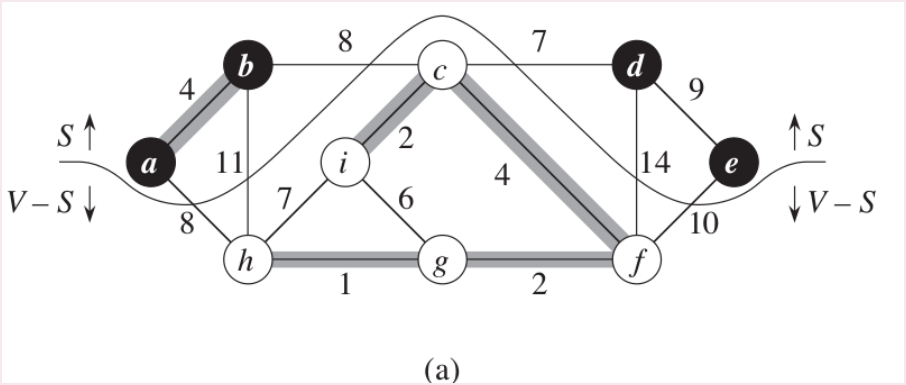

Figura: Fonte: Livro Algoritmos - Cormen

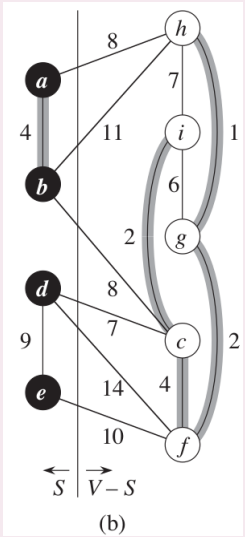

Figura: Fonte: Livro Algoritmos - Cormen

Desenvolvendo uma árvore geradora mínima

- Dizemos que uma aresta $(u, v) \in E$ cruza o corte $(S, V - S)$ se um de seus pontos extremos está em S e o outro está em $V - S$
- Dizemos que um corte respeita um conjunto A de arestas se nenhuma aresta em A cruza o corte
- Uma aresta é uma aresta leve que cruza um corte se o seu peso é o mínimo de qualquer aresta que cruza o corte, podendo haver mais de uma
- Nossa regra para reconhecer arestas seguras é dada pelo seguinte teorema

Teorema 1

Seja $G = (V, E)$ um grafo conexo não dirigido com uma função peso de valores reais w definida em E. Sejam A um subconjunto de E que está incluído em alguma árvore geradora mínima para G, seja $(S, V - S)$ qualquer corte de G que respeita A e seja (u, v) uma aresta leve que cruza $(S, V - S)$. Então, a aresta (u, v) é segura para A.

Prova

- Seja T uma árvore geradora mínima que inclui A, e suponha que T não contenha a aresta leve (u, v) já que, se contiver, terminamos
- Construiremos uma outra árvore geradora mínima T' que inclui $A \cup \{(u, v)\}$ usando uma técnica de recortar e colar, mostrando assim que (u, v) é uma aresta segura para A
- A aresta (u, v) forma um ciclo com as arestas no caminho simples p de u a v em T, como mostra a imagem abaixo

Desenvolvendo uma árvore geradora mínima

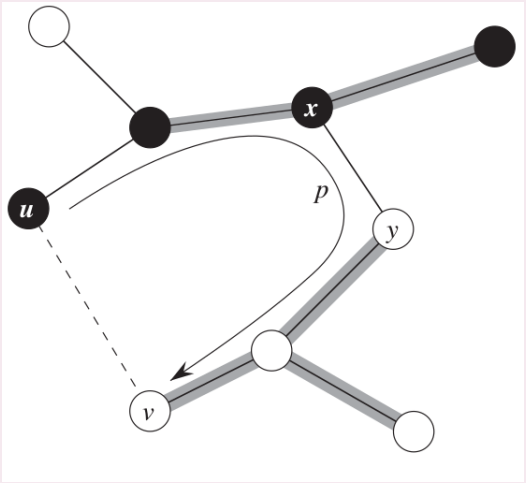

Figura: Fonte: Livro Algoritmos - Cormen

Prova

- Visto que u e v estão em lados opostos do corte $(S, V - S)$, no mínimo uma aresta em T se encontra no caminho simples p e também cruza o corte
- Seja (x, y) qualquer dessas arestas
- A aresta (x, y) não está em A porque o corte respeita A
- Como (x, y) está no único caminho simples de u a v em T, remover (x, y) separa T em duas componentes
- A adição de (u, v) reconecta as duas componentes e forma uma nova árvore geradora $T' = T - (x, y) \cup \{(u, v)\}$
- Em seguida, mostramos que T' é uma árvore geradora mínima

Prova

- Visto que (u, v) é uma aresta leve que cruza $(S, V - S)$ e (x, y) também cruza esse corte, $w(u, v) \leq w(x, y)$
- Então,

$$\begin{aligned} w(T') &= w(T) - w(x, y) + w(u, v) \\ &\leq w(T) \end{aligned}$$

- Porém, T é uma árvore geradora mínima, de modo que $w(T) \leq w(T')$;
- Assim, T' também deve ser uma árvore geradora mínima

Prova

- Resta mostrar que (u, v) é realmente uma aresta segura para A
- Temos $A \subseteq T'$, já que $A \subseteq T$ e $(x, y) \notin A$
- Assim, A $\cup \{(u, v)\} \subseteq T'$
- Consequentemente, como T' é uma árvore geradora mínima, (u, v) é segura para A $\square$

Árvores Geradoras Mínimas

- O teorema nos permite compreender melhor o funcionamento do método genérico
- À medida que o algoritmo progride, o conjunto A é sempre acíclico; caso contrário, uma árvore geradora mínima incluindo A conteria um ciclo, o que é uma contradição
- Em qualquer ponto na execução, o grafo $G_A = (V, A)$ é uma floresta, e cada uma das componentes conexas de G_A é uma árvore
- Além disso, qualquer aresta segura (u, v) para A conecta componentes distintos de G_A, já que A $\cup\{(u, v)\}$ deve ser acíclico

Corolário

Seja $G = (V, E)$ um grafo conexo não dirigido com uma função peso de valor real w definida em E. Seja A um subconjunto de E que está incluído em alguma árvore geradora mínima para G, e seja C=(V_c,E_c) uma componente conexa (árvore) na floresta $G_A = (V, A)$. Se (u, v) é uma aresta leve que conecta C a alguma outra componente em G_A, então (u, v) é segura para A

Prova

O corte $(V_C, V - V_C)$ respeita A, e (u, v) é então uma aresta leve para esse corte. Portanto, (u, v) é segura para A. $\square$

Algoritmo de Kruskal

- Os dois algoritmos de árvore geradora mínima que veremos aperfeiçoam o método genérico
- Cada um deles utiliza uma regra especifica para determinar uma aresta segura na linha 3 de Generic-MST
- No algoritmo de Kruskal, o conjunto A é uma floresta cujos vértices são todos os vértices do grafo dado
- A aresta segura adicionada a A é sempre uma aresta de peso mínimo no grafo que conecta duas componentes distintas
- No algoritmo de Prim, o conjunto A forma uma árvore única
- A aresta segura adicionada a A é sempre uma aresta de peso mínimo que conecta a árvore a um vértice não presente na árvore

Algoritmo de Kruskal

- O algoritmo de Kruskal acha uma aresta segura para adicionar à floresta que está sendo desenvolvida encontrando, entre todas as arestas que conectam quaisquer duas árvores na floresta, uma aresta (u,v) de peso mínimo
- Sejam C_1 e C_2 as duas árvores que são conectadas por (u,v)
- Visto que (u,v) deve ser uma aresta leve que conecta C_1 a alguma outra árvore, o corolário anterior implica que (u,v) é uma aresta segura para C_1
- O Algoritmo de Kruskal se qualifica como algoritmo guloso porque em cada etapa ele adiciona à floresta uma aresta de menor peso possível

Algoritmo de Kruskal

- Para entendermos o algoritmo, vamos definir:
  - A operação Find-set(u) retorna um elemento representativo do conjunto que contém u
  - Assim, podemos determinar se dois vértices pertencem à mesma árvore testando se Find-set(u) é igual a Find-set(v)
  - Para combinar as árvores, o algoritmo Kruskal chama o procedimento Union

Algoritmo de Kruskal

```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Kruskal

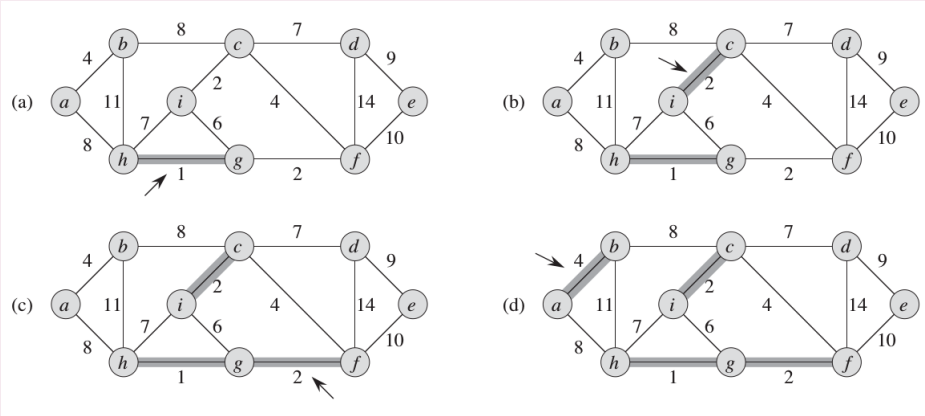

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Kruskal

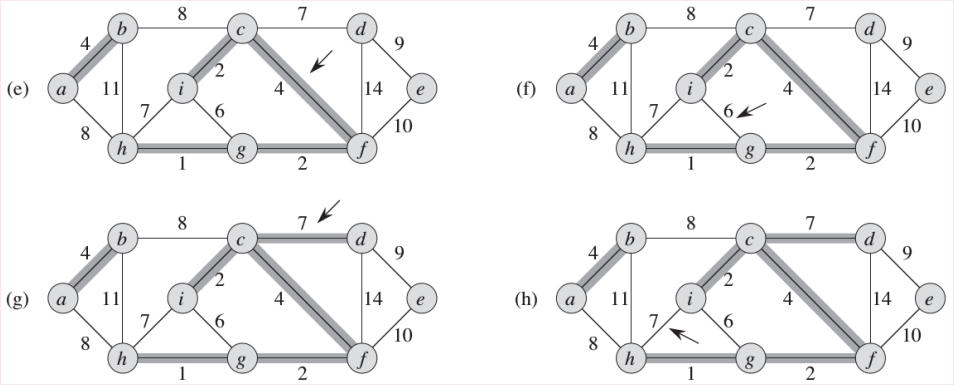

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Kruskal

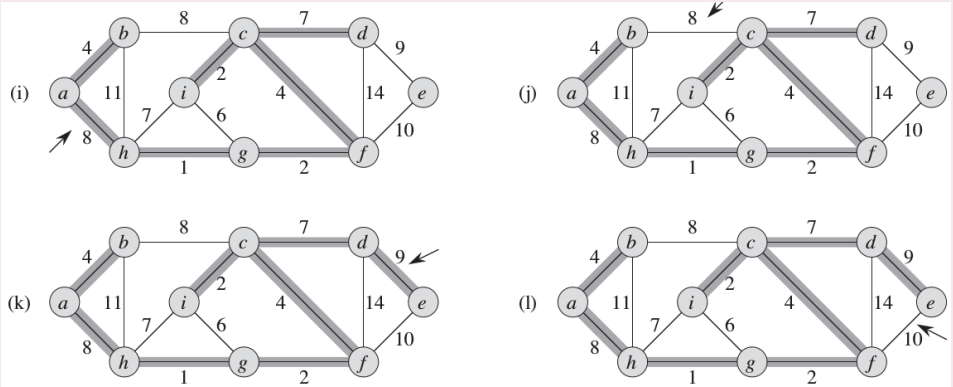

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Kruskal

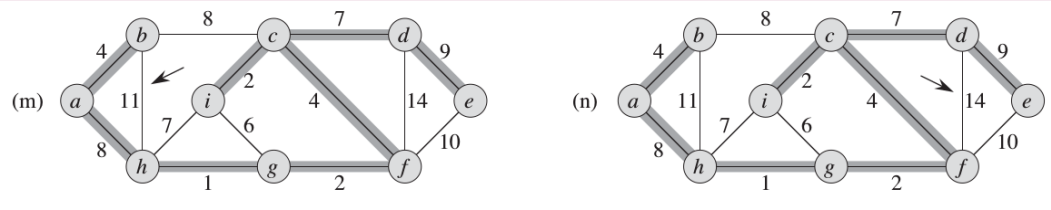

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Prim

- O algoritmo de Prim tem as seguintes propriedades: as arestas no conjunto A sempre formam uma árvore única
- Como podemos ver no algoritmo, a árvore começa em um vértice arbitrário r e aumenta até que a árvore abranja todos os vértices em V
- Cada etapa adiciona à árvore A uma aresta leve que conecta A a um vértice isolado - um vértice no qual nenhuma aresta de A incide
- Pelo corolário, essa regra adiciona apenas arestas que são seguras para A; portanto, quando o algoritmo termina, as arestas em A formam uma árvore geradora mínima
- Essa estratégia se qualifica como gulosa, já que a cada etapa ela adiciona à árvore uma aresta que contribui com a mínima quantidade possível para o peso da árvore

Algoritmo de Prim

- Durante a execução do algoritmo, todos os vértices que não estão na árvore residem em uma fila de prioridade mínima Q baseada em um atributo chave
- Para cada vértice v, o atributo v.chave é o peso mínimo de qualquer aresta que conecta v a um vértice na árvore; por convenção, v.chave= ∞ se não existe nenhuma aresta desse tipo
- O atributo v.π nomeia o pai de v na árvore
- O algoritmo mantém implicitamente o conjunto A de Generic-MST como

Algoritmo de Prim

$A = \{(v, v.\pi) : v \in V - \{r\} - Q\}$.

When the algorithm terminates, the min-priority queue Q is empty; the minimum spanning tree A for G is thus

$A = \{(v, v.\pi) : v \in V - \{r\}\}$.

MST-PRIM(G, w, r)

```
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Prim

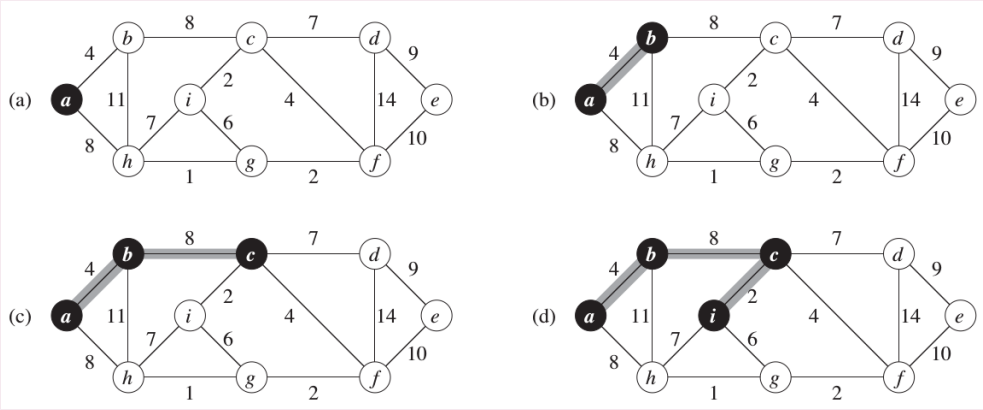

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Prim

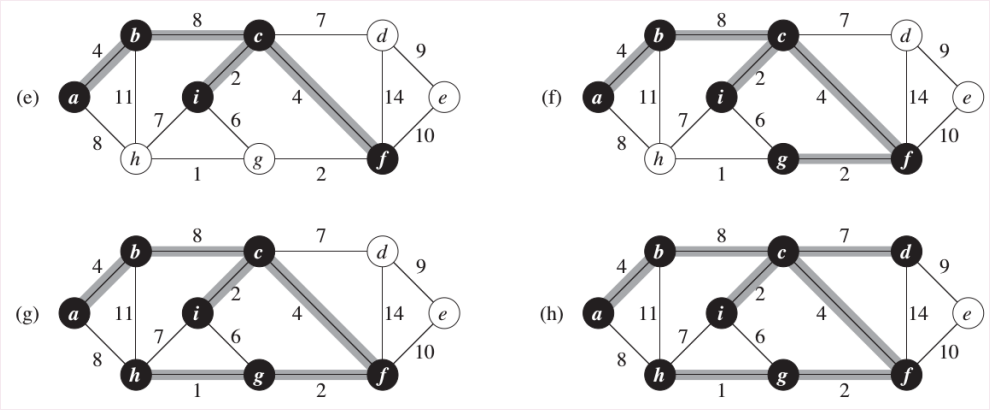

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Prim

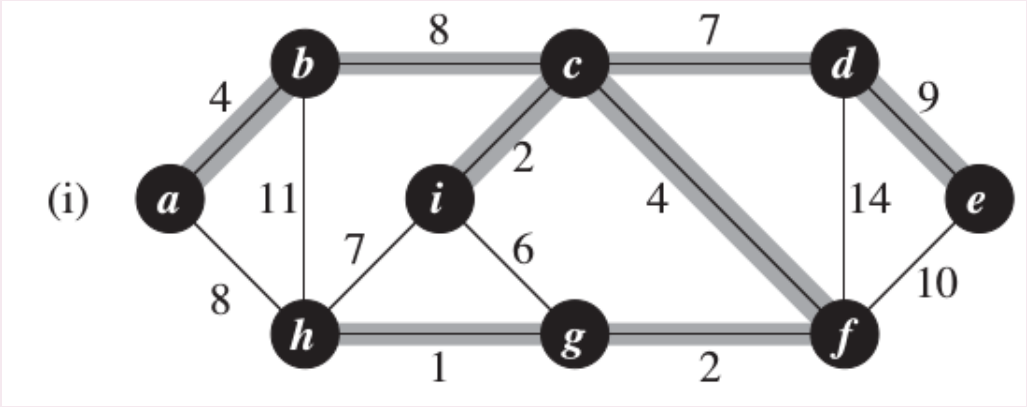

Figura: Fonte: Livro Algoritmos - Cormen

Próxima Aula

O que vem por aí?

- Caminhos mínimos
- Algoritmo de Bellman-Ford
- Algoritmo de Dijkstra
- Tira dúvidas/Exercícios
- Teste 2
- Avaliação parcial 2

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Árvores Geradores Mínimas

Algoritmos em Grafos

Professor: Rennan Dantas

Universidade Federal do Ceará
Campus de Crateús

02, 04 e 09 de maio de 2022