UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Caminhos mínimos de fonte única

Algoritmos em Grafos

Professor: Rennan Dantas

Universidade Federal do Ceará
Campus de Crateús

16, 18 e 23 de maio de 2022

Caminhos mínimos de fonte única

Introdução

- Um motorista deseja encontrar a rota mais curta do Rio de Janeiro a São Paulo
- Dado um mapa rodoviário do Brasil no qual a distância entre cada par de interseções adjacentes esteja marcada, como ele pode determinar essa rota mais curta?
- Um modo possível é enumerar todas as rotas do Rio de Janeiro a São Paulo, somar as distâncias em cada rota e selecionar a mais curta
- Número enorme de possibilidades
- Veremos como resolver tais problemas eficientemente

Introdução

- Em um problema de caminhos mínimos, temos um grafo dirigido ponderado $G = (V, E)$, com funções peso w: E $\rightarrow \mathbb{R}$ que mapeia as arestas para pesos de valores reais
- O peso do caminho $p =< v_0, v_1, ..., v_k >$ é a soma dos pesos de suas arestas constituintes:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}v_i)$$

- Definimos o peso do caminho mínimo de u a v por

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if there is a path from } u \text{ to } v \text{ ,} \\ \infty & \text{otherwise .} \end{cases}$$

Figura: Fonte: Livro Algoritmos - Cormen

Introdução

- Então um caminho mínimo do vértice u ao vértice v é definido como qualquer caminho p com peso $w(p) = \delta(u, v)$
- No exemplo da rota entre o Rio de Janeiro e São Paulo, podemos modelar o mapa rodoviário como um grafo: vértices representam interseções, arestas representam segmentos de estradas entre interseções e pesos de arestas representam distâncias rodoviárias
- Nossa meta é encontrar um caminho mínimo de um dado entroncamento de rodovias no Rio de Janeiro a um dado entroncamento de rodovias em São Paulo

Caminhos mínimos de fonte única

Introdução

- Pesos de arestas podem representar outras medidas que não sejam distâncias, como tempo, custos, multas, prejuízos ou qualquer outra quantidade que se acumule linearmente ao longo de um caminho e que seria interessante minimizar
- O algoritmo de busca em largura é um algoritmo de caminhos mínimos que funciona em grafos não ponderados, isto é, grafos nos quais cada aresta tem peso unitário

Caminhos mínimos de fonte única

Variantes

- Focalizaremos o problema de caminhos mínimos de fonte única: dado um grafo $G = (V, E)$, queremos encontrar um caminho mínimo de determinado vértice de origem $s \in V$ a todo vértice $v \in V$. O algoritmo para o problema da fonte única pode resolver muitos outros problemas, entre os quais as variantes apresentadas a seguir
- **Problemas de caminhos mínimos com um só destino:** Encontrar um caminho mínimo até um determinado vértice de destino t a partir de cada vértice v. Intervendo a direção de cada aresta no grafo, podemos reduzir esse problema a um problema de fonte única

Caminhos mínimos de fonte única

Variantes

- **Problema do caminho mínimo para um par:** Encontrar um caminho de u a v para vértices u e v dados. Se resolvemos o problema de fonte única com vértice de fonte u, também resolveremos esse problema. Além disso, todos os algoritmos conhecidos por esse problema têm o mesmo tempo de execução assintótica do pior caso que os melhores algoritmo de fonte única
- **Problemas de caminhos mínimos para todos os pares:** Encontrar um caminho mínimo de u a v para todo par de vértices u e v. Embora seja possível resolver esse problema executando um algoritmo de fonte única uma vez para cada vértice, em geral podemos resolvê-lo mais rapidamente. Além disso, sua estrutura é interessante por si só

Caminhos mínimos de fonte única

Subestrutura ótima de um caminho mínimo

- Em geral, algoritmos de caminhos mínimos se baseiam na seguinte propriedade: um caminho mínimo entre dois vértices contém outros caminhos mínimos
- Lembre-se de que a subestrutura ótima é um dos indicadores fundamentais da possível aplicabilidade da programação dinâmica e do método guloso
- O algoritmo de Dijkstra, que veremos mais adiante, é um algoritmo guloso, e o algoritmo de Floyd-Warshall, que encontra caminhos mínimos entre todos os pares de vértices, é um algoritmo de programação dinâmica
- O lema a seguir enuncia com maior exatidão a propriedade de subestrutura ótima de caminhos mínimos

Lema

Dado um grafo dirigido ponderado $G = (V, E)$ com função peso $w : E \to \mathbb{R}$, seja $p =< v_1, v_2, ..., v_k >$ um caminho mínimo do vértice v_0 ao vértice v_k e, para quaisquer i e j tais que $1 \leq u \leq j \leq k$, seja $p_{ij} =< v_i, v_{i+1}, ..., v_j >$ o sub-caminho p do vértice v_i ao vértice v_j. Então, p_{ij} é um caminho mínimo de v_i a v_j.

Prova

Proof If we decompose path p into $v_0 \overset{p_{0i}}{\rightsquigarrow} v_i \overset{p_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$, then we have that $w(p) = w(p_{0i}) + w(p_{ij}) + w(p_{jk})$. Now, assume that there is a path p'_{ij} from v_i to v_j with weight $w(p'_{ij}) < w(p_{ij})$. Then, $v_0 \overset{p_{0i}}{\rightsquigarrow} v_i \overset{p'_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$ is a path from v_0 to v_k whose weight $w(p_{0i}) + w(p'_{ij}) + w(p_{jk})$ is less than $w(p)$, which contradicts the assumption that p is a shortest path from v_0 to v_k. ■

Figura: Fonte: Livro Algoritmos - Cormen

Arestas de peso negativo

- Algumas instâncias do problema de caminhos mínimos de fonte única podem incluir arestas cujos pesos são negativos
- Se o grafo $G = (V, E)$ não contém nenhum ciclo de peso negativo que possa ser alcançado da fonte s, então para todo $v \in V$, o peso do caminho mínimo $\delta(s, v)$ permanece bem definido, mesmo que tenha um peso negativo

Arestas de peso negativo

- Contudo, se o grafo contém um ciclo de peso negativo que possa ser alcançado a partir de s, os pesos de caminhos mínimos não são bem definidos
- Nenhum caminho de s a um vértice no ciclo pode ser um caminho mínimo - sempre podemos encontrar um caminho de peso menor seguindo um caminho "mínimo" proposto e depois percorrendo o ciclo de peso negativo
- Se houver um ciclo de peso negativo em algum caminho de s a v, definiremos $\delta(s, v) = -\infty$

Arestas de peso negativo

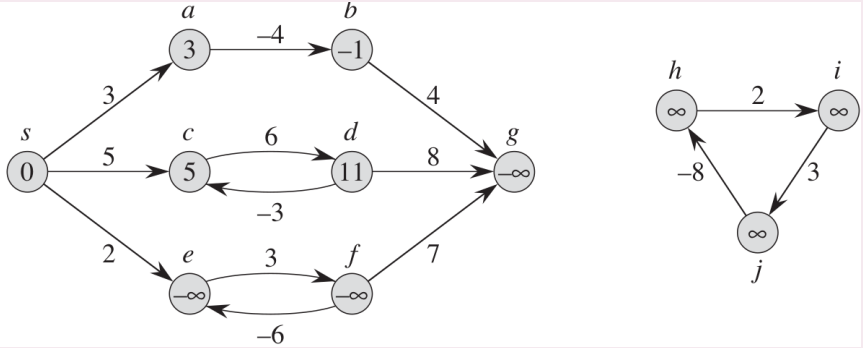

Figura: Fonte: Livro Algoritmos - Cormen

Ciclos

- Um caminho mínimo pode conter um ciclo?
- Ele não pode conter um ciclo de peso negativo, como podemos ver na figura anterior
- Nem pode conter um ciclo de peso positivo, pois remover o ciclo produz um caminho com os mesmos vértices de fonte e destino e um peso de caminho mais baixo
- Então podemos dizer, sem perda da generalidade, que quando estamos encontrando caminhos mínimos, eles não têm nenhum ciclo, isto é, são caminhos simples

Ciclos

- Visto que qualquer caminho acíclico em um grafo $G = (V, E)$ contém no máximo $|V|$ vértices distintos, ele também contém no máximo $|V| - 1$ arestas
- Assim, podemos restringir nossa atenção a caminhos mínimos que tenham no máximo $|V| - 1$ arestas

Representação de caminhos mínimos

- Muitas vezes, desejamos calcular não apenas pesos de caminho mínimos, mas também os vértices nos caminhos mínimos
- A representação que usamos para caminhos mínimos é semelhante à que utilizamos para árvores em largura
- Dado um grafo $G = (V, E)$, mantemos para cada vértice $v \in V$ um predecessor $v.\pi$ que é um outro vértice ou NIL
- Os algoritmos que veremos definem os atributos π de modo que a cadeia de predecessores que se origina em um vértice v percorra um caminho mínimo de s a v em sentido contrário
- Assim, dado um vértice v para o qual $v.\pi \neq$ NIL o procedimento PRINT-PATH(G, s, v) imprimirá um caminho mínimo de s a v

Representação de caminhos mínimos

```
PRINT-PATH(G, s, v)
1  if v == s
2        print s
3  elseif v.π == NIL
4        print "no path from" s "to" v "exists"
5  else PRINT-PATH(G, s, v.π)
6        print v
```

Figura: Fonte: Livro Algoritmos - Cormen

Representação de caminhos mínimos

- Entretanto, durante a execução de um algoritmo de caminhos mínimos, os valores de π poderiam não indicar caminhos mínimos
- Estaremos interessados no **subgrafo dos predecessores** $G_\pi = (V_\pi, E_\pi)$ induzido pelos valores π
- Definimos o conjunto V_π como o conjunto de vértices de G com predecessores não NIL mais a fonte de s:

$$V_\pi = \{v \in V : v.\pi \neq NIL\} \cup \{s\}$$

- O conjunto de arestas dirigidas E_π é o conjunto de arestas induzidas pelos valores de π para os vértices em V_π:

$$E_\pi = \{(v.\pi, v) \in E : v \in V_\pi - \{s\}\}$$

Representação de caminhos mínimos

- É possível mostrar que os valores de π produzidos neste capítulo têm a seguinte propriedade: no término, G_π é uma "árvore de caminhos mínimos", informalmente, uma árvore enraizada que contém um caminho mínimo da fonte s a todo vértice que pode ser alcançado de s
- Uma árvore de caminhos mínimos com raiz em s é um subgrafo dirigido $G' = (V, E)$, onde $V' \subseteq V$ e $E' \subseteq E$, tal que
  1. V' é o conjunto de vértices que pode ser alcançados de s em G
  2. G' forma uma árvore enraizada com s e
  3. para todo $v \in V'$, o único caminho simples de s a v em G' é um caminho mínimo de s a v em G
- Caminhos mínimos não são necessariamente únicos nem são necessariamente únicas as árvores de caminhos mínimos

Representação de caminhos mínimos

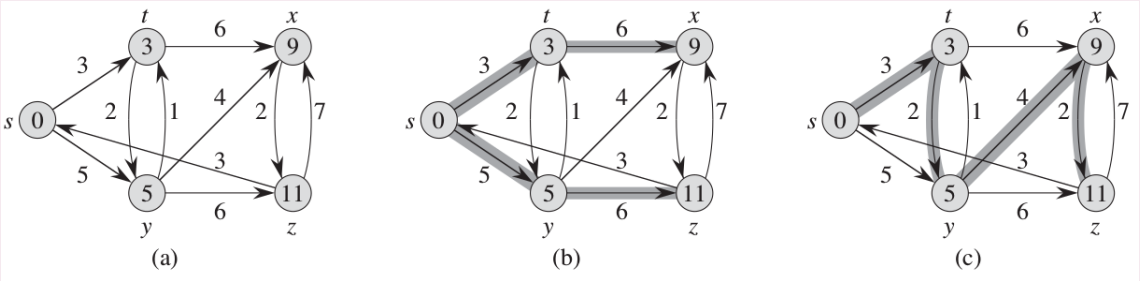

Figura: Fonte: Livro Algoritmos - Cormen

Relaxamento

- Os algoritmos que veremos usam a técnica de relaxamento
- Para cada atributo $v \in V$, mantemos um atributo $v.d$, que é um limite superior para o peso de um caminho mínimo da fonte s a v
- Denominamos $v.d$ uma estimativa de caminho mínimo
- Inicializamos as estimativas de caminhos mínimos e predecessores pelo seguinte algoritmo de tempo $\Theta(V)$:

Relaxamento

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  for each vertex v ∈ G.V
2      v.d = ∞
3      v.π = NIL
4  s.d = 0
```

Figura: Fonte: Livro Algoritmos - Cormen

Relaxamento

- O processo de relaxar uma aresta (u, v) consiste em tentar melhorar o caminho mínimo até v que encontramos até agora passando por u, em caso positivo, atualizar $v.d$ e $v.\pi$

Relaxamento

```
RELAX(u, v, w)
1   if v.d > u.d + w(u, v)
2       v.d = u.d + w(u, v)
3       v.π = u
```

Figura: Fonte: Livro Algoritmos - Cormen

Relaxamento

- Os algoritmos que veremos chamam o INITIALIZE-SINGLE-SOURCE e depois relaxam arestas repetidamente

Relaxamento

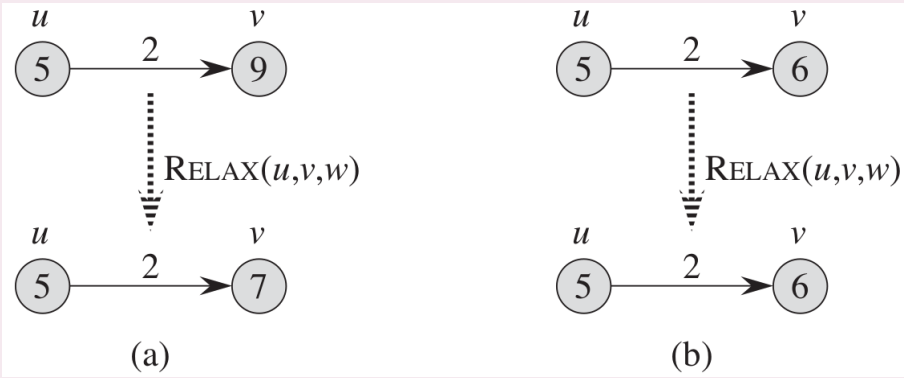

Figura: Fonte: Livro Algoritmos - Cormen

O algoritmo de Bellman-Ford

- O algoritmo de Bellman-Ford resolve o problema de caminhos mínimos de fonte única no caso geral no qual os pesos das arestas podem ser negativos
- Dado um grafo dirigido ponderado $G = (V, E)$ com fonte s e função peso w:E $\rightarrow \mathbb{R}$, o algoritmo de Bellman-Ford devolve um valor booleano que indica se existe ou não um ciclo de peso negativo que pode ser alcançado da fonte
- Se tal ciclo existe, o algoritmo indica que não há nenhuma solução
- Se tal ciclo não existe, o algoritmo produz os caminhos mínimos e seus pesos

O algoritmo de Bellman-Ford

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3       for each edge (u, v) ∈ G.E
4            RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6       if v.d > u.d + w(u, v)
7            return FALSE
8  return TRUE
```

Figura: Fonte: Livro Algoritmos - Cormen

O algoritmo de Bellman-Ford

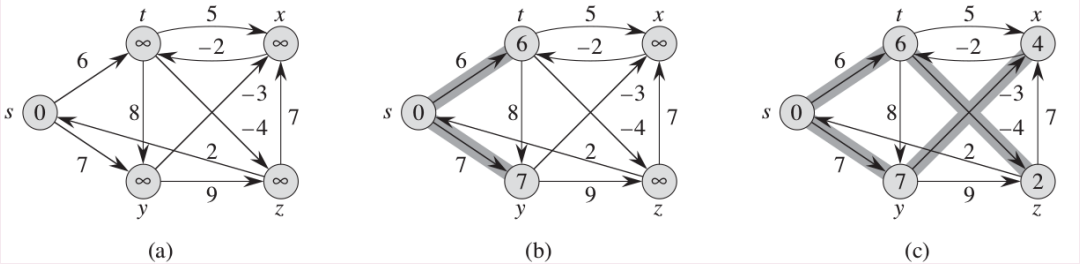

Figura: Fonte: Livro Algoritmos - Cormen

O algoritmo de Bellman-Ford

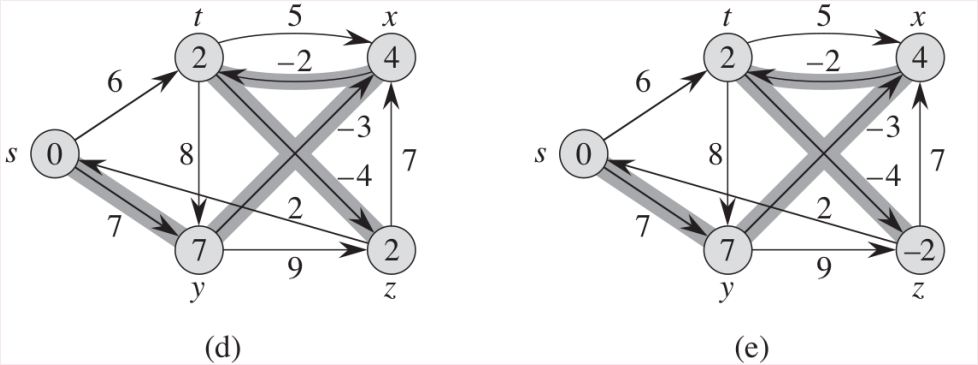

Figura: Fonte: Livro Algoritmos - Cormen

Caminhos mínimos de fonte única

Caminhos mínimos de fonte única em grafos acíclicos dirigidos

- Relaxando as arestas de um gad ponderado de acordo com uma ordenação topológica de seus vértices, podemos calcular caminhos mínimos de uma fonte única no tempo $\Theta(V + E)$
- Caminhos mínimos são sempre bem definidos em um gad já que, mesmo que existam arestas de peso negativo, não deve existir nenhum ciclo de peso negativo

Caminhos mínimos de fonte única em grafos acíclicos dirigidos

```
DAG-SHORTEST-PATHS(G, w, s)
1  topologically sort the vertices of G
2  INITIALIZE-SINGLE-SOURCE(G, s)
3  for each vertex u, taken in topologically sorted order
4       for each vertex v ∈ G.Adj[u]
5            RELAX(u, v, w)
```

Figura: Fonte: Livro Algoritmos - Cormen

Caminhos mínimos de fonte única em grafos acíclicos dirigidos

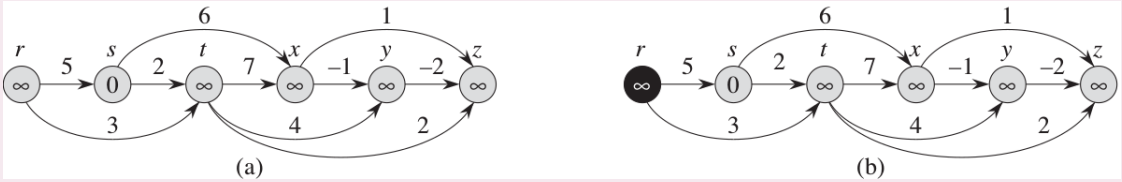

Figura: Fonte: Livro Algoritmos - Cormen

Caminhos mínimos de fonte única em grafos acíclicos dirigidos

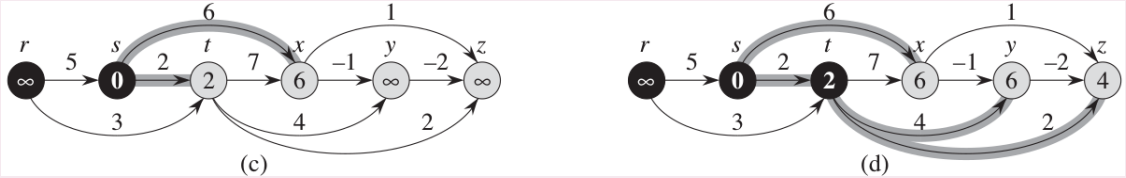

Figura: Fonte: Livro Algoritmos - Cormen

Caminhos mínimos de fonte única em grafos acíclicos dirigidos

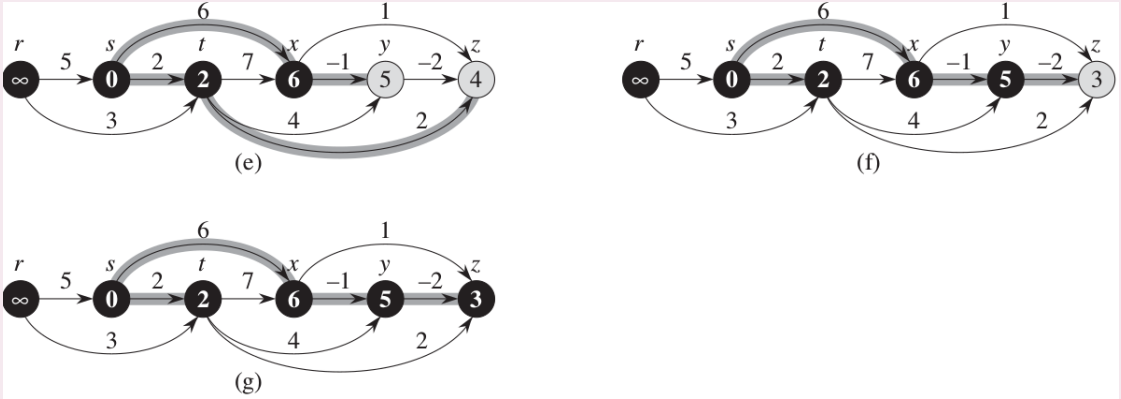

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Dijkstra

- O algoritmo de Dijkstra resolve o problema de caminhos mínimos de fonte única em um grafo dirigido ponderado $G = (V, E)$ para o caso no qual todos os pesos de arestas são não negativos
- Então, nesta seção iremos supor que $w(u, v) \geq 0$ para cada arestas $(u, v) \in E$
- Como veremos, com uma boa implementação, o tempo de execução do algoritmo de Dijkstra é inferior ao do algoritmo de Bellman-Ford

Algoritmo de Dijkstra

- O algoritmo de Dijkstra mantém um conjunto S de vértices cujos pesos finais de caminhos mínimos que partem da fonte s já foram determinados
- O algoritmo seleciona repetidamente o vértice $u \in V - \{s\}$ que tem a mínima estimativa do caminho mínimo, adiciona u a S e relaxa todas as arestas que saem de u

Algoritmo de Dijkstra

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Dijkstra

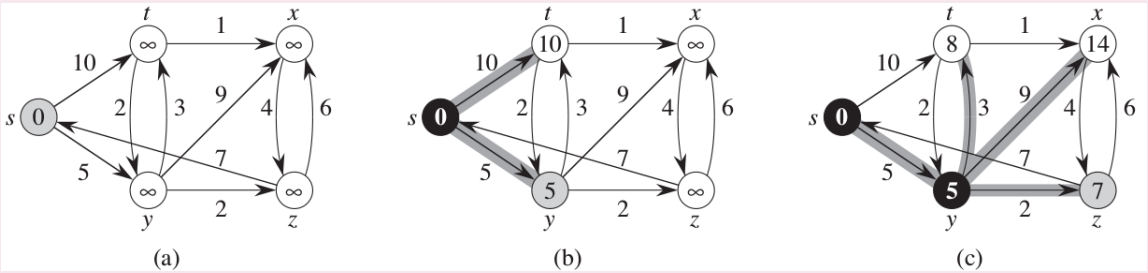

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Dijkstra

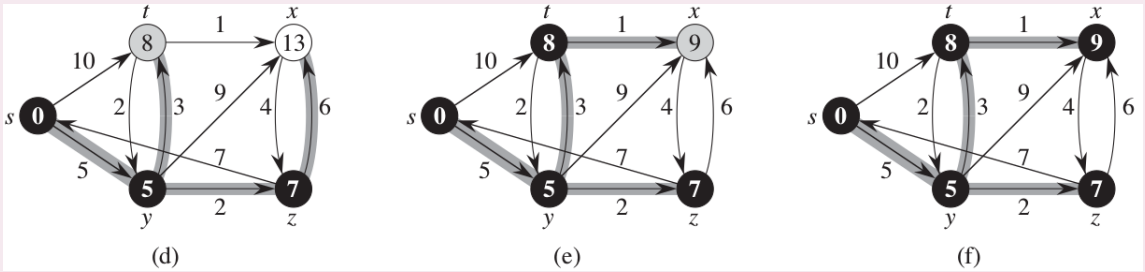

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Floyd-Warshall

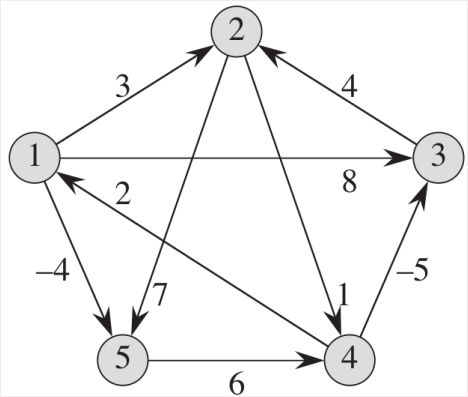

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Floyd-Warshall

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0\,, \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1\,. \end{cases}$$

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Floyd-Warshall

$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty\,, \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty\,. \end{cases}$$

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Floyd-Warshall

```
FLOYD-WARSHALL(W)
1  n = W.rows
2  D^(0) = W
3  for k = 1 to n
4       let D^(k) = (d_ij^(k)) be a new n × n matrix
5       for i = 1 to n
6            for j = 1 to n
7                 d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
8  return D^(n)
```

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Floyd-Warshall

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Floyd-Warshall

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

Figura: Fonte: Livro Algoritmos - Cormen

Algoritmo de Floyd-Warshall

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Figura: Fonte: Livro Algoritmos - Cormen

Próxima Aula

O que vem por aí?

- Revisão/Tira dúvidas
- Teste 2
- Revisão/Tira dúvidas
- Avaliação parcial 2

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Caminhos mínimos de fonte única

Algoritmos em Grafos

Professor: Rennan Dantas

Universidade Federal do Ceará
Campus de Crateús

16, 18 e 23 de maio de 2022