Teoria dos Grafos Unidade 5: Caminhamento em Grafos

Prof. Dr. Paulo César Rodacki Gomes
rodacki@ifc-riodosul.edu.br

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CATARINENSE
Campus Rio do Sul

Tópicos

- Grafos e Ciclos Eulerianos
- Grafos e Ciclos Hamiltonianos
- Problema do caminho mínimo:
  - Algoritmo de Dijkstra
  - Algoritmo de Floyd
- Problema do Carteiro Chinês
- Problema do Caixeiro Viajante

Blibliografia

- Márcia A. Rabuske. **Introdução à Teoria dos Grafos**. Editora da UFSC. 1992.
- Joan M. Aldous, Robin J. Wilson. **Graphs and Applications**: as introductory approach. Springer. 2001
- Thomas Cormen et al. **Algoritmos**: teoria e prática. Ed. Campus. 2004.

Grafo Valorado

Definição

Um Grafo Valorado $G=(V, E)$ é formado por um conjunto V de vértices e um conjunto E de **arestas valoradas**, sendo que para cada aresta $(u, v) \in E$, temos um valor numérico $w(u, v)$ ou w_{uv}, chamado de custo da aresta (u, v).

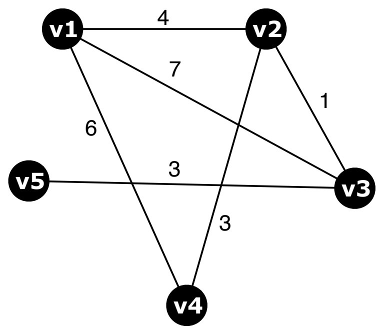
Este custo representa alguma grandeza numérica relevante para o problema, tal como distância, tempo, valor monetário, etc.

Grafo Valorado - Matriz de custos

Os custos das arestas de um grafo valorado G=(V, E) podem ser armazenados em uma matriz W, chamada de matriz de custos do grafo G, definida da seguinte forma:

$$w_{i,j} = \begin{cases} 0, & \text{se } v_i = v_j; \\ \infty, & \text{se } (v_i, v_j) \notin E; \\ \text{custo}, & \text{se } (v_i, v_j) \in E. \end{cases}$$

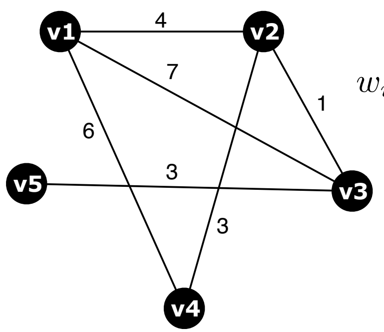
Exemplo



$$W = \begin{vmatrix} 0 & 4 & 7 & 6 & \infty \\ 4 & 0 & 1 & 3 & \infty \\ 7 & 1 & 0 & \infty & 3 \\ 6 & 3 & \infty & 0 & \infty \\ \infty & \infty & 3 & \infty & 0 \end{vmatrix}$$

Exemplo

Dado o grafo valorado abaixo, construa sua matriz de custos [W]:



$$w_{i,j} = \begin{cases} 0, & \text{se } v_i = v_j; \\ \infty, & \text{se } (v_i, v_j) \notin E; \\ \text{custo}, & \text{se } (v_i, v_j) \in E. \end{cases}$$

Caminho Mínimo

O **custo** de um caminho p = $\langle v_0, v_1, ..., v_k \rangle$

entre dois vértices v_0 e v_k é igual ao somatório dos custos de todas as arestas valoradas do caminho:

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

Caminho Mínimo

O **custo do caminho mínimo** do vértice u para o vértice v é definido por:

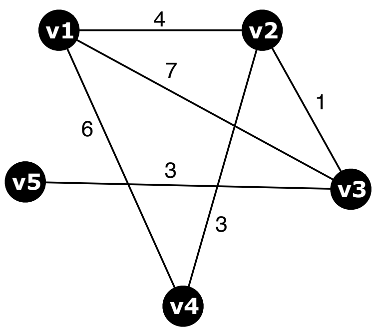
$$\delta_{u,v} = \delta(u,v) = \begin{cases} \min\{\ \mathrm{w(p)}:\ \mathrm{u} \rightsquigarrow \mathrm{v}\}, & \text{se } \exists \text{ caminho de u para v;} \\ \infty, & \text{caso contrário.} \end{cases}$$

O **caminho mínimo** entre dois vértices *u* e *v* é definido como qualquer caminho *p* com custo $w(p) = \delta(u, v)$.

Exemplo

Qual o caminho mínimo entre os vértices v1 e v3?

Qual o custo deste caminho mínimo?



Problema do caminho mínimo com uma origem

- Dado um grafo valorado *G=(V, E)*, queremos encontrar os caminhos mínimos de um dado vértice origem $s \in V$, até cada um dos vértices $v \in V$.
- O algoritmo que resolve este problema é o **Algoritmo de Dijkstra**

Algoritmo de Dijkstra

- Para armazenar os caminhos entre o vértice inicial *s* e os demais vértices, podemos usar uma árvore de caminhos mínimos, semelhante às árvores de busca.
- Dado o grafo valorado *G=(V, E)*, manteremos para cada vértice $v \in V$ um **predecessor** representado por $\pi[v]$ que contém ou um vértice ou NIL.

Algoritmo de Dijkstra

Elementos do algoritmo:

Elemento	Descrição
s	vértice inicial
v	quaisquer outros vértices
u	vértice "pivô", que pode representar uma mudança no caminho mínimo até o vértice v
d[v]	custo estimado do caminho mínimo de s até v
w(u,v)	custo da aresta (u, v), tem valor ∞ se não existir aresta (u, v)
π[v]	vértice predecessor do vértice v
Q	fila de prioridades mínimas de vértices (o vértice com menor valor de d[v] tem prioridade para sair da fila)
S	conjunto dos vértices cujo caminho mínimo já foi calculado

Algoritmo de Dijkstra

Inicialização dos atributos dos vértices:

```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ V[G]
      do d[v] ← ∞
         π[v] ← NIL
d[s] ← 0
```


Algoritmo de Dijkstra

Relaxamento da aresta (u, v): verifica se podemos melhorar a estimativa atual de caminho mínimo até o vértice v, incluindo o vértice v no caminho. Esta operação atualiza os valores de d[v] e *π[v]*.

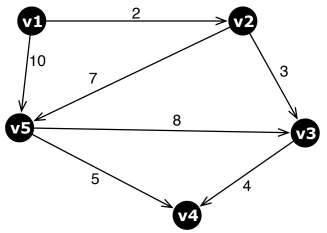
```
RELAX(u, v, w)
if d[v] > d[u] + w(u, v)
   then d[v] ← d[u] + w(u, v)
        π[v] ← u
```



```
RELAX(u, v, w)
if d[v] > d[u] + w(u, v)
   then d[v] ← d[u] + w(u, v)
        π[v] ← u
```

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← Ø
Q ← V[G]
while Q ≠ Ø
    do u ← EXTRACT-MIN(Q)
       S ← S ∪ {u}
       for each vertex v ∈ Adj[u]
           do RELAX(u, v, w)
```

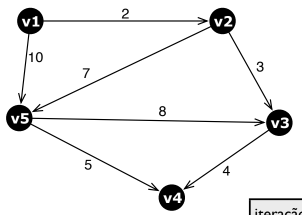
Exemplo



$$W = \begin{vmatrix} 0 & 4 & 7 & 6 & \infty \\ 4 & 0 & 1 & 3 & \infty \\ 7 & 1 & 0 & \infty & 3 \\ 6 & 3 & \infty & 0 & \infty \\ \infty & \infty & 3 & \infty & 0 \end{vmatrix}$$


```
RELAX(u, v, w)
if d[v] > d[u] + w(u, v)
    then d[v] ← d[u] + w(u, v)
         π[v] ← u
```

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← ∅
Q ← V[G]
while Q ≠ ∅
    do u ← EXTRACT-MIN(Q)
       S ← S ∪ {u}
       for each vertex v ∈ Adj[u]
           do RELAX(u, v, w)
```



iteração	S	u	d(u)	d(v1)	d(v2)	d(v3)	d(v4)	d(v5)
inicio	vazio	---	---	0	∞	∞	∞	∞
0	v1	v1	0	---	2	∞	∞	10
1	v1, v2	v2	2	---	---	5	∞	9
2	v1, v2, v3	v3	5	---	---	---	9	9
3	v1, v2, v3, v5	v5	9	---	---	---	9	---
4	v1, v2, v3, v5, v4	v4	9	---	---	---	---	---

Problema do caminho mínimo com várias origens

- Dado um grafo valorado *G=(V, E)*, queremos encontrar os caminhos mínimos entre todos os pares de vértices
- O algoritmo **Floyd-Warshall** calcula esses caminhos através de operações recursivas em matrizes de distância (D), de tamaho n x n
- O algoritmo calcula n matrizes de distância, e o resutado final com as caminhos mínimos surge na última matriz calculada

Algoritmo de Floyd-Warshall

- Como este algoritmo usa a estrutura de matriz de adjacência, a entrada do algoritmo é a matriz de custos do grafo, definida da seguinte forma:

$$w_{i,j} = \begin{cases} 0, & \text{se } v_i = v_j; \\ \infty, & \text{se } (v_i, v_j) \notin E; \\ \text{custo}, & \text{se } (v_i, v_j) \in E. \end{cases}$$

Algoritmo de Floyd-Warshall

- Inicialização: $D^0 \leftarrow W$
- Demais matrizes:

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

Onde:

i: linha (índice do vértice origem)
j: coluna (índice do vértice destino)
k: iteração (índice da matriz)

Algoritmo de Floyd-Warshall

Algoritmo: Foyd-Warshall(G, W)

$D^0 \leftarrow W$;
para cada $k \leftarrow 1$ **até** n **faça**
  para cada $i \leftarrow 1$ **até** n **faça**
    para cada $j \leftarrow 1$ **até** n **faça**
      $d_{ij}^{(k)} \leftarrow min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$;
retorna D^n

Algoritmo de Floyd-Warshall

Os caminhos podem ser armazenados em uma **Matrizes de roteamento** (Π), construídas a partir das matrizes de distância (D), da seguinte forma:

Inicialização:
$$\pi_{ij}^{(0)} = \begin{cases} NIL, & \text{se } v_i = v_j \text{ ou } w_{ij} = \infty, \\ v_i, & \text{se } v_i \neq v_j \text{ e } w_{ij} < \infty. \end{cases}$$

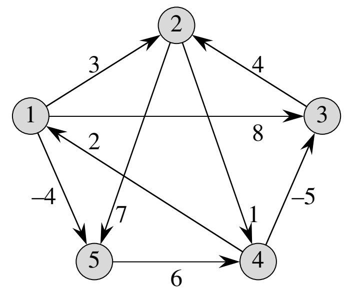
Demais matrizes:
$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)}, & \text{se } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}, \\ \pi_{kj}^{(k-1)}, & \text{se } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}. \end{cases}$$

Obs: π_{ij} armazena o predecessor de v_j num caminho iniciado em v_i

Algoritmo: Foyd-Warshall(G, W)

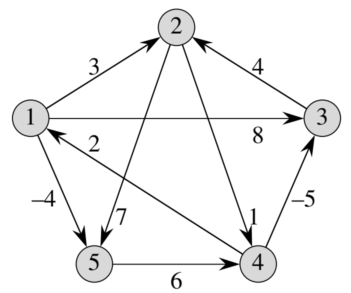
para cada $i \leftarrow 1$ **até** n **faça**
  para cada $j \leftarrow 1$ **até** n **faça**
    $d_{ij}^0 \leftarrow w_{ij}$;
    se $v_i \neq v_j$ *e* $w_{ij} < \infty$ **então**
      $\pi_{ij}^0 \leftarrow v_i$;
    senão
      $\pi_{ij}^0 \leftarrow NIL$;
para cada $k \leftarrow 1$ **até** n **faça**
  para cada $i \leftarrow 1$ **até** n **faça**
    para cada $j \leftarrow 1$ **até** n **faça**
      se $(d_{ik}^{k-1} + d_{kj}^{k-1}) < d_{ij}^{k-1}$ **então**
        $d_{ij}^k \leftarrow d_{ik}^{k-1} + d_{kj}^{k-1}$;
        $\pi_{ij}^k \leftarrow \pi_{kj}^{k-1}$;
      senão
        $d_{ij}^k \leftarrow d_{ij}^{k-1}$;
        $\pi_{ij}^k \leftarrow \pi_{ij}^{k-1}$;
retorna D^n, Π^n;

Exemplo



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

Exemplo



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

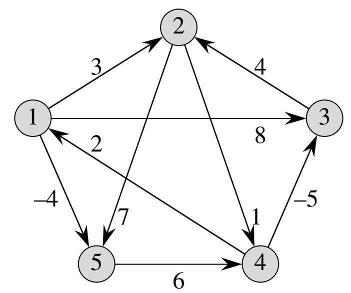
$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL}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

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

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

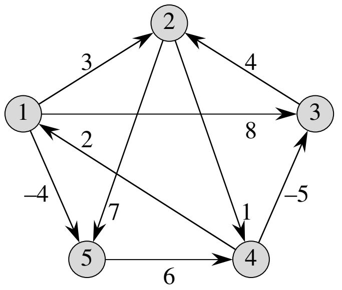
$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1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

Exemplo



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

```
PRINT-ALL-PAIRS-SHORTEST-PATH(Π, i, j)
if i = j
   then print i
   else if π_ij = NIL
           then print "no path from" i "to" j "exists"
           else PRINT-ALL-PAIRS-SHORTEST-PATH(Π, i, π_ij)
                print j
```



$$\Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

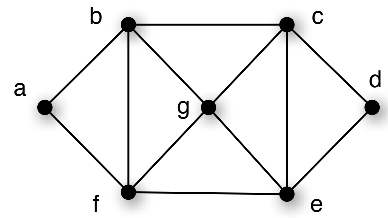
Grafos e ciclos eulerianos e hamiltonianos

Começamos explorando dois tipos de problemas a rotas conectando um conjunto de cidades em um mapa:

- **Problema do explorador**: um explorador deseja encontrar um roteiro que passe por todas as estradas do mapa somente uma vez e retorne ao ponto de partida
- **Problema do turista**: um turista deseja encontrar um roteiro que passe em cada cidade somente uma vez, e retorne a cidade de partida.

Grafos e ciclos eulerianos e hamiltonianos

Exemplo:



Grafos e ciclos eulerianos e hamiltonianos

Definições:

- Um grafo conexo é **Euleriano** se ele contiver um caminho simples fechado que inclua cada uma de suas arestas. Tal caminho é chamado de **Ciclo euleriano**
- Um grafo conexo é **Hamiltoniano** se contiver um caminho simples fechado que inclua cada um dos vértices do grafo. Tal ciclo é chamado de **Ciclo hamiltoniano**

Exercício

Dados os grafos abaixo, verifique se cada um deles é Euleriano e/ou Hamiltoniano. Escreva os ciclos eulerianos e hamiltonianos sempre que possível: