Notas de Aula - AED2 – Grafos: caminhos mínimos
Prof. Jefferson T. Oliva

Ver slide 3: suponha que gostaríamos de ir de Pato Branco até Foz do Iguaçu. Como vocês podem ver, existem vários caminhos até o destino, mas qual seria o caminho mais perto? Aplicar algoritmo de Prim ou de Kruskal não seria interessante, pois ambos algoritmos podem não gerar o menor caminho entre ambas cidades, já que em cada processo de construção da árvore geradora mínima, é considerada a aresta de corte mais leve, ou seja, um vértice que esteja mais próximo à arvore não significa que acarretará na menor distância entre as cidades de origem e destino. Em outras palavras, os algoritmos computam a menor distância para todas as outras cidades, mas não em relação a uma única origem.

Problema do(s) Caminho(s) Mínimo(s) e a sua Representação

Seja **G** um grafo orientado e suponha que para cada aresta **(u, v)** associamos um peso (custo, distância) **(u, v)**. Usaremos a notação **(G, w).**
- Problema do Caminho Mínimo entre Dois Vértices: Dados dois vértices **s** e **t** em **(G, w)**, encontrar um caminho (de peso) mínimo de **s** a **t**
- Aparentemente, este problema não é mais fácil do que o Problema dos Caminhos Mínimos com Mesma Origem: Dados **(G, w)** e **s ∈ V[G]**, encontrar para cada vértice **v** de **G**, um caminho mínimo de **s** a **v**.

Ver teorema no slide 5

P = (v1, v2, . . . , vk): caminho mínimo de **v1** a **vk**

1 ≤ i ≤ j ≤ k

Pij = (vi, vi+1, . . . , vj): caminho mínimo de **vi** a **vj**

Usamos uma ideia similar à usada em Busca em Largura no algoritmo de caminhos mínimos que veremos.

Para cada vértice **v ∈ V[G]** associamos um predecessor **π[v]**.

Ao final do algoritmo obtemos uma Árvore de Caminhos Mínimos com raiz **s**.

Um caminho de **s** a **v** nesta árvore é um caminho mínimo de **s** a **v** em **(G, w)**.

Estimativa de distâncias
- Para cada **v ∈ V [G]** queremos determinar **dist(s, v)**, o peso de um caminho mínimo de **s** a **v** em **(G, w)** (distância de **s** a **v**.)
- Os algoritmos de caminhos mínimos associam a cada **v ∈ V[G]** um valor **d[v]** que é uma estimativa da distância **dist(s, v)**.

Algoritmos Auxiliares

- inicialização
→ O valor **d[v]** é uma estimativa superior para o peso de um caminho mínimo de **s** a **v**
→ Ele indica que o algoritmo encontrou até aquele momento um caminho de **s** a **v** com peso **d[v]**
→ O caminho pode ser recuperado por meio dos predecessores **π[]**.

```
INITIALIZE-SINGLE-SOURCE(G, s)
1  para cada vértice v ∈ V[G] faça
2      d[v] ← ∞
3      π[v] ← NIL
4  d[s] ← 0
```

- relaxação: **ver slides 9 e 10**
→ O relaxamento de uma aresta **(u, v)** consiste em verificar se é possível melhorar o melhor caminho até **v** obtido até o momento se passarmos por **u**
→ Se isto acontecer, **d[v]** e **π[v]** devem ser atualizados

```
RELAX(u, v, w)
1  se d[v] > d[u] + w(u, v)
2      então d[v] ← d[u] + w(u, v)
3            π[v] ← u
```

No exemplo do slide 9:
- no vértice **u**, o peso é 5
- no vértice **v**, o peso é 9
- o peso da aresta **(u, v)** é 2
- o peso do vértice **u** + o peso da aresta **(u, v)** é menor em relação ao peso do vértice **v**
- ou seja, 5 + 2 < 9
- então, o peso do vértice **v** é atualizado para 7 e o seu pai passará a ser o vértice **u**

Caminhos mínimos: existem três algoritmos baseados em relaxação para tipos de instâncias diferentes de Problemas de Caminhos Mínimos
- **G** é acíclico: aplicação de ordenação topológica (uma ordem linear de seus nós em que cada nó vem antes de todos nós para os quais este tenha arestas de saída – Wikipédia)
- **(G, w)** não tem arestas de peso negativo: **algoritmo de Dijkstra**
- **(G, w)** tem arestas de peso negativo, mas não contém ciclos negativos: algoritmo de Bellman-Ford

Algoritmo de Dijkstra

Objetivo: encontrar o caminho de distância mínima de um vértice origem **x** a um vértice destino **y**.
- resolve o problema do caminho mínimo em um grafo direcionado ou não direcionado com arestas de peso não negativo
- É semelhante ao algoritmo busca em largura, utilizando uma estratégia gulosa

Entrada: um grafo orientado (ou não) **(G, w)** (sem arestas de peso negativo) e um vértice **s** de **G**

Saída:
- para cada **v** $\in$ **V[G]**, o peso de um caminho mínimo de **s** a **v**
- uma árvore de caminhos mínimos com raiz **s**. Um caminho de **s** a **v** nesta árvore é um caminho mínimo de **s** a **v** em **(G, w)**.

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S ← ∅
3  Q ← V[G]
4  enquanto Q ≠ ∅ faça
5      u ← EXTRACT-MIN(Q)
6      S ← S ∪ {u}
7      para cada vértice v ∈ Adj[u] faça
8          RELAX(u, v, w)
```

O conjunto Q é implementado como uma fila de prioridade.

O conjunto S não é realmente necessário, mas simplifica a análise do algoritmo.

Em cada iteração, o algoritmo de Dijkstra
- escolhe um vértice **u** fora do conjunto **S** que esteja mais próximo a esse e acrescenta-o a **S**
- atualiza as distâncias estimadas dos vizinhos de **u** e **v**
- atualiza a Arvore dos Caminhos Mínimos

Ver slides de 17 ao 23

Exercício no slide 24

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. Introduction to Algorithms. Third edition, The MIT Press, 2009.

Marin, L. O. Grafos: Caminhos Mínimos. AE23CP - Algoritmos e Estrutura de Dados II. Slides. Engenharia de Computação. Dainf/UTFPR/Pato Branco, 2017.

Tenenbaum, A.; Langsam, Y. Estruturas de Dados usando C. Pearson, 1995.

Ziviani, N. Projeto de Algoritmos - com implementações em Java e C++. Thomson, 2007.