CAMINHOS MÍNIMOS EM GRAFOS: DIJKSTRA (PARTE I)

Prof. Daniel Kikuti

Universidade Estadual de Maringá

20 de abril de 2015

Sumário

Introdução

Aplicações

- **Rotas em mapas (Google Maps):** encontrar automaticamente menor distância entre dois pontos;

Introdução

Aplicações

- **Rotas em mapas (Google Maps):** encontrar automaticamente menor distância entre dois pontos;
- **Máquinas de estados finitos:** cada vértice descreve uma estado, cada aresta representa uma transição e peso da aresta representa o custo para mudar de um estado para outro (encontrar uma sequência ótima de escolhas que leve ao estado objetivo);

Introdução

Aplicações

- **Rotas em mapas (Google Maps):** encontrar automaticamente menor distância entre dois pontos;
- **Máquinas de estados finitos:** cada vértice descreve uma estado, cada aresta representa uma transição e peso da aresta representa o custo para mudar de um estado para outro (encontrar uma sequência ótima de escolhas que leve ao estado objetivo);
- **Outras aplicações:** redes de computadores (telecomunicação), *layout* de fábrica e instalações, robótica, transporte, etc.

Classificação dos problemas de caminho mínimo

Problemas que serão abordados em aula

- **Origem única:** Encontrar um caminho mínimo a partir de uma dada origem $s \in V$ até todo vértice $v \in V$.
- **Todos os pares:** Encontrar um caminho mínimo de u até v para todos os pares de vértices u e v.

Classificação dos problemas de caminho mínimo

Problemas que serão abordados em aula

- **Origem única:** Encontrar um caminho mínimo a partir de uma dada origem $s \in V$ até todo vértice $v \in V$.
- **Todos os pares:** Encontrar um caminho mínimo de u até v para todos os pares de vértices u e v.

Outras variantes

- **Destino único:** Encontrar um caminho mínimo até um determinado vértice de destino t a partir de cada vértice v.
- **Par único:** Encontrar o caminho mínimo de u até v.

Definições

Entrada (origem única)

Um grafo orientado $G = (V, E)$, uma função peso $w : E \to R$ e um vértice de origem s.

Definições

Entrada (origem única)

Um grafo orientado $G = (V, E)$, uma função peso $w : E \to R$ e um vértice de origem s.

Custo do caminho e caminho de custo mínimo

O **peso (ou custo) de um caminho** $p = \langle v_0, v_1, \ldots, v_k \rangle$ é a soma do peso das arestas no caminho

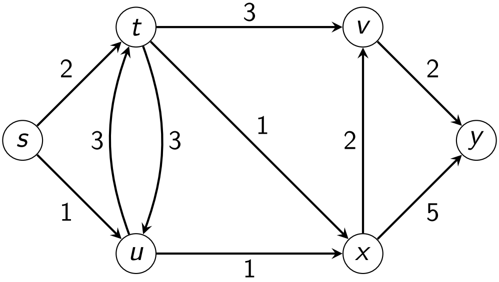
$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Definimos o **caminho de custo mínimo** de u a v como

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{se existe caminho de } u \text{ a } v \text{ e,} \\ \infty & \text{caso contrário.} \end{cases}$$

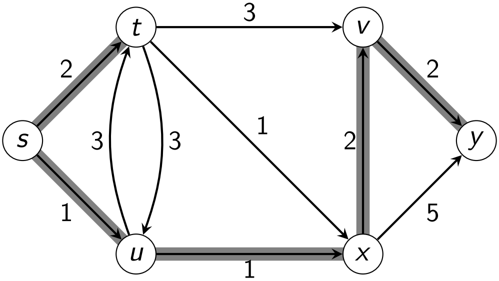
Um exemplo

Qual o caminho mais curto de s para os demais vértices?



Um exemplo

Qual o caminho mais curto de s para os demais vértices?



Propriedade importante

Subestrutura ótima (Lema 24.1)

Qualquer subcaminho de um caminho mínimo é um caminho mínimo.

Propriedade importante

Subestrutura ótima (Lema 24.1)

Qualquer subcaminho de um caminho mínimo é um caminho mínimo.

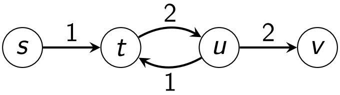
Demonstração

Seja $G = (V, E)$ um grafo direcionado com função peso $w : E \rightarrow R$, $p = \langle v_0, v_1, \ldots, v_k \rangle$ o caminho mínimo do vértice v_0 até v_k e, $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ um subcaminho de p tal que $0 \leq i \leq j \leq k$. O custo de p é então definido como $w(p) = w(p_{0i}) + w(p_{ij}) + w(p_{jk})$. Suponha que existe um caminho p'_{ij} de v_i até v_j tal que $w(p'_{ij}) < w(p_{ij})$. Então, $v_0 \overset{p_{0i}}{\leadsto} v_i \overset{p'_{ij}}{\leadsto} v_j \overset{p_{jk}}{\leadsto} v_k$ é um caminho de v_0 até v_k cujo custo $w(p_{0i}) + w(p'_{ij}) + w(p_{jk})$ é menor que $w(p)$, o que contradiz o fato de p ser um caminho mínimo de v_0 até v_k.

Considerações sobre ciclos

Um caminho mínimo pode conter ciclos (arestas positivas)?

NÃO. Podemos obter um caminho mínimo eliminando o(s) ciclo(s). E se o ciclo for nulo, não há razão para usar tal ciclo.

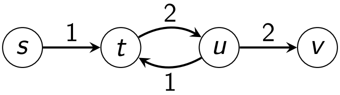


Qualquer caminho acíclico em um grafo $G = (V, E)$ contém no máximo $|V|$ vértices distintos e no máximo $|V| - 1$ arestas.

Considerações sobre ciclos

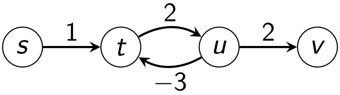
Um caminho mínimo pode conter ciclos (arestas positivas)?

NÃO. Podemos obter um caminho mínimo eliminando o(s) ciclo(s). E se o ciclo for nulo, não há razão para usar tal ciclo.



Qualquer caminho acíclico em um grafo $G = (V, E)$ contém no máximo $|V|$ vértices distintos e no máximo $|V| - 1$ arestas.

Ciclos negativos



Algoritmo de caminhos mínimos

Algoritmo de Edsger Dijkstra (1959)

- Usa estratégia gulosa
- Baseado no algoritmo de busca em largura
- Mantém dois conjuntos: S (vértices cujo caminho mínimo desde a origem já foi determinado) e $V - S$ (restante)
- O algoritmo seleciona repetidamente o vértice com a menor estimativa de caminhos mínimos ($v.d'$) e adiciona-o em S.

O algoritmo de Dijkstra em alto nível

Versão do livro de Kleinberg e Tardos

`dijkstra(G, w, s)`

1 Seja S o conjunto de vértices explorados, para cada $u \in S$ armazenaremos a distância $u.d$;

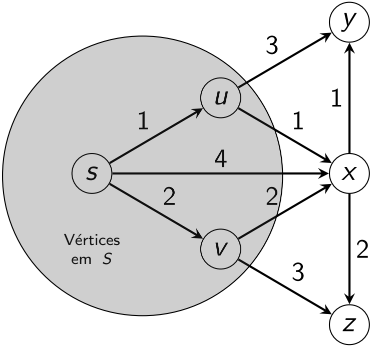
2 Inicialmente $S = \{s\}$ e $s.d = 0$;

3 enquanto $S \neq V$

4 Selecione um $v \notin S$ com pelo menos uma aresta de S para qual $v.d'$ é o menor possível, isto é: $v.d' = \min_{(u,v):\ u \in S} \{u.d + w(u,v)\}$;

5 Adicione v a S e defina $v.d = v.d'$

Uma figura explicando funcionamento



Análise do algoritmo

Correção – Invariante de laço

Considere o conjunto S em qualquer ponto de execução do algoritmo. Para cada $u \in S$, o caminho $s \overset{p}{\rightsquigarrow} u$ é um caminho mínimo.

Análise do algoritmo

Correção – Invariante de laço

Considere o conjunto S em qualquer ponto de execução do algoritmo. Para cada $u \in S$, o caminho $s \overset{p}{\rightsquigarrow} u$ é um caminho mínimo.

Demonstração por indução no tamanho de S

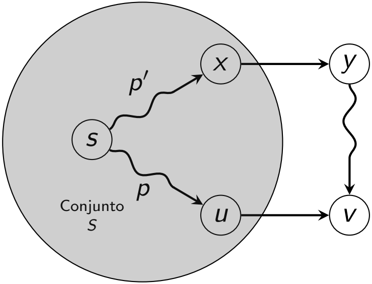
Base: $|S| = 1 \Rightarrow S = \{s\}$ e $s.d = 0$.
Hipótese: O invariante vale para $|S| = k$, $k \geq 1$.
Passo: Adicionaremos mais um vértice v e aumentaremos o tamanho de S para $k + 1$.

Análise do algoritmo – Continuação da correção

Demonstração do passo da indução

Seja (u, v) a aresta final do caminho $p_{sv} = \langle s \overset{p}{\leadsto} u \to v \rangle$. Pela hipótese, p é um caminho mínimo de s a u para cada $u \in S$.

Considere agora qualquer outro caminho $p'_{sv} = \langle s \overset{p'}{\leadsto} x \to y \leadsto v \rangle$ de s a v. Mostraremos que p'_{sv} é pelo menos tão longo quanto p_{sv}. A figura a seguir ilustra esta situação.



O caminho alternativo p'_{sv} passando por x e y já é pelo menos tão longo quanto p_{sv} no momento em que deixa o conjunto S.

Análise do algoritmo – Continuação da correção

Término do passo indutivo

Na iteração $k+1$, o algoritmo considerou adicionar y ao conjunto S via aresta (x, y), mas rejeitou-o em favor de v. Isto significa que não existe caminho de s a y via x que seja menor que p_{sv}, pois o subcaminho p' de s até y é pelo menos tão longo quanto p_{sv}. Como não existem arestas com pesos negativos, o caminho p'_{sv} será também tão longo quanto ou maior que p_{sv}.

Análise do algoritmo - Complexidade

Qual o custo de tempo?

dijkstra(G, w, s)

1 Seja S o conjunto de vértices explorados, para cada $u \in S$ armazenaremos a distância $u.d$;

2 Inicialmente $S = \{s\}$ e $s.d = 0$;

3 enquanto $S \neq V$

4 Selecione um $v \notin S$ com pelo menos uma aresta de S para qual $v.d'$ é o menor possível, isto é: $v.d' = \min_{(u,v):\ u \in S} \{u.d + w(u,v)\}$;

5 Adicione v a S e defina $v.d = v.d'$

Uma versão mais baixo nível

Detalhes do algoritmo

- Usa uma fila de prioridades, em que as chaves são os pesos estimados dos caminhos mínimos (v.d)
- $v.pred$ = predecessor de v no caminho mínimo a partir de s. Se não existe predecessor, então v.pred = NIL. *pred* induz uma árvore, a árvore de caminhos mínimos
- Mantém dois conjuntos de vértices: S (vértices cujo caminho mínimo desde a origem já foi determinado) e $Q = V - S$ (fila de prioridades)
- Inicialmente $s.d = 0$ e demais vértices $v.d = \infty$.
- O algoritmo seleciona repetidamente o vértice $u \in Q$, com a menor estimativa de caminhos mínimos, adiciona-o em S e relaxa todas as arestas que saem de u.

Funções auxiliares

Inicializa vértices

```
initialize-single-source(G, s)
para cada vértice v ∈ G.V
    v.d = ∞
    v.pred = NIL
s.d = 0
```

Funções auxiliares

Inicializa vértices

```
initialize-single-source(G, s)
para cada vértice v ∈ G.V
    v.d = ∞
    v.pred = NIL
s.d = 0
```

Relaxa

```
relax(u, v, w)
se v.d > u.d + w(u,v)
    v.d = u.d + w(u,v)
    v.pred = u.
```

O Algoritmo

Algoritmo de Dijkstra versão Cormen

```
dijkstra(G, w, s)
initialize-single-source(G, s)
S = ∅
Q = G.V
enquanto Q != ∅
   u = extract-min(Q)
   S = S U u
   para cada vértice v em u.adj
      relax(u, v, w)
```

Análise do algoritmo - Complexidade

Operações de fila

- `insert` implícita na linha 3 (executado uma vez para cada vértice)
- `extract-min` na linha 5 (executado uma vez para cada vértice)
- `decrease-key` implícita em relax (executado no máximo de $|E|$ vezes, uma vez para cada aresta relaxada)

Análise do algoritmo - Complexidade

Operações de fila

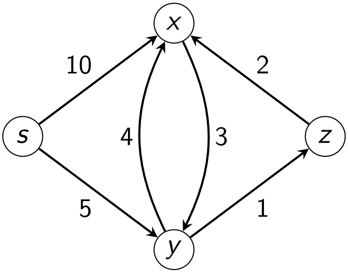
- `insert` implícita na linha 3 (executado uma vez para cada vértice)
- `extract-min` na linha 5 (executado uma vez para cada vértice)
- `decrease-key` implícita em relax (executado no máximo de $|E|$ vezes, uma vez para cada aresta relaxada)

Total usando heap

Tempo total de $O((V + E) \lg V + V) = O(E \lg V)$.

Exercício 1

Faça uma execução passo a passo do algoritmo de Dijkstra para a figura abaixo.



Exercício 2 [24.3.2 - Cormen]

Forneça um exemplo simples de um grafo orientado com arestas de peso negativo para o qual o algoritmo de Dijkstra produz respostas incorretas. Por que a demonstração de correção do algoritmo vista em aula não é válida quando são permitidas arestas de peso negativo?