Introdução à Teoria dos Grafos

Bacharelado em Ciência da Computação–UFMS, 2005

MENORES CAMINHOS E CAMINHOS DE CUSTO MÍNIMO

Resumo

Neste texto veremos aplicações para os conceitos básicos sobre grafos: o Problema do Menor Caminho e o Problema do Caminho de Custo Mínimo. No primeiro problema, dado um grafo G e um par de vértices queremos encontrar o caminho com a menor quantidade de arestas, ou o menor caminho, que conecta esses vértices. Quando associamos uma função custo $c\colon E_G \to \mathbb{Q}_{\geqslant}$ às arestas de G, queremos encontrar o caminho de menor custo entre um dado par de vértices. Veremos então algoritmos para solucionar esses problemas.

1 Distância em Grafos

Suponha que em um sistema distribuído uma mensagem deva ser enviada de um processador P_1 para um processador P_2 no menor tempo possível. Como isso pode ser feito? Esta rede de computadores pode ser modelada por um grafo. Todo vértice corresponde a um processador, e dois vértices são ligados por uma aresta se os processadores correspondentes podem se comunicar diretamente. Dois processadores com esta característica são ditos adjacentes. A figura 1 fornece um modelo hipotético de um sistema distribuído. Se o tempo de comunicação entre todo par de processadores é o mesmo, então o menor percurso para o envio de uma mensagem de P_1 para P_2 corresponde ao menor caminho entre P_1 e P_2. No nosso exemplo, P_1, P_3, P_5, P_4, P_2 e $P_1, P_{10}, P_6, P_9, P_2$ são dois menores caminhos entre P_1 e P_2.

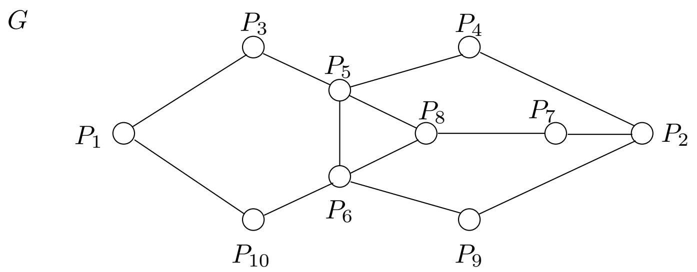

Figura 1: Um modelo de um computador multiprocessado.

Formalizaremos agora estas idéias. Para um grafo não trivial G e um par de vértices u, v de G, a **distância** $\text{dist}_G(u,v)$, ou $\text{dist}(u,v)$ se não existirem ambigüidades no contexto, entre u e v é o comprimento do menor caminho entre u e v em G, se tal caminho existe. Se G não contém um caminho de u para v, então definimos $\text{dist}(u,v) = \infty$. Por exemplo, nos grafos G_1 e G_2 da figura 2, $\text{dist}_{G_1}(u,v) = 2$ e $\text{dist}_{G_2}(u,v) = \infty$.

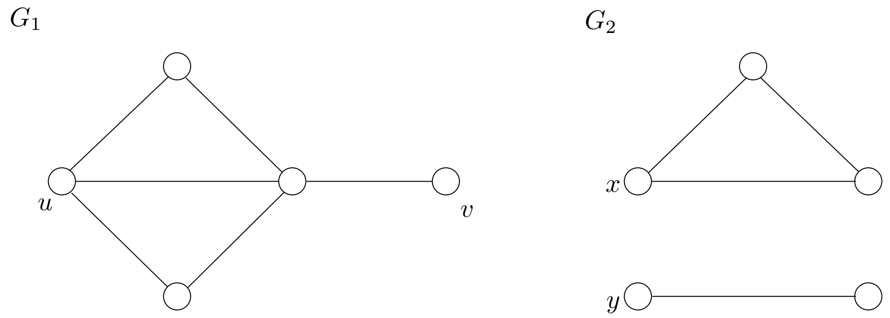

Figura 2: Distâncias em grafos.

A função distância de um grafo G é uma métrica, isto é, mapeia $V_G \times V_G$ para um conjunto de racionais não negativos e satisfaz as seguintes propriedades fundamentais:

(*i*) $\text{dist}(u, v) \geqslant 0$, e $\text{dist}(u, v) = 0$ se e somente se $u = v$;

(*ii*) $\text{dist}(u, v) = \text{dist}(v, u)$ para todo $u, v \in V_G$;

(*iii*) $\text{dist}(u, v) \geqslant \text{dist}(u, w) + \text{dist}(w, v)$ para todo $u, v, w \in V_G$, (**a desigualdade triangular**).

Para um grafo orientado D, a **distância (orientada)** $\text{dist}_D(u, v)$, ou $\text{dist}(u, v)$, de um vértice u para um vértice v de D é o comprimento de um menor caminho orientado u para v, se tal caminho existe, e é ∞ caso contrário. Portanto, se D é o grafo orientado da figura 3, então $\text{dist}(u, v) = \infty$ e $\text{dist}(u, z) = 3$.

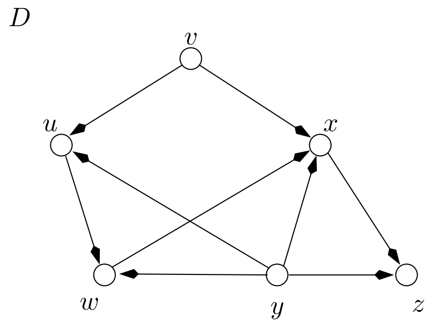

Figura 3: Distâncias em grafos orientados.

Um algoritmo eficiente, utilizando a idéia de busca em largura, foi proposto inicialmente por Moore [4], e nos permite encontrar menores caminhos e distâncias em grafos de forma sistemática. Como vimos também, o próprio algoritmo da busca em largura apresentado em sala mantém informações suficientes na estrutura que representa o grafo de tal forma que facilmente podemos obter o comprimento de um menor caminho entre um vértice inicial e outro vértice qualquer de V_G, assim como o próprio caminho[1]. Daqui em diante, estaremos mais interessados em uma generalização deste problema, que veremos a seguir.

2 Distância em Grafos com Custos nas Arestas

Diariamente, ambulâncias são chamadas em locais onde ocorrem acidentes. É de fato muito importante para uma ambulância alcançar um local no menor espaço de tempo, e assim a menor rota deve ser encontrada. A figura 4 mostra um grafo com custos que modela tal situação. O vértice v_h representa o hospital, o vértice v_a representa o local do acidente e os outros vértices representam cruzamentos de ruas entre o hospital e o local do acidente. Cada aresta representa uma rua e é rotulada com a quantidade de segundos que normalmente é necessária para atravessar esta rua. Ambos $v_h, v_1, v_4, v_5, v_7, v_8, v_a$ e v_h, v_3, v_6, v_8, v_a são menores percursos do hospital para o lugar do acidente.

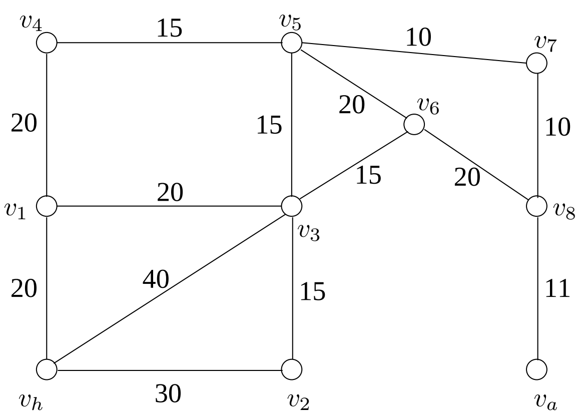

Figura 4: Um modelo de rotas possíveis de um hospital para um local de acidente.

Discutiremos um algoritmo eficiente que encontra caminhos de custo mínimo entre pares de vértices em um grafo com custos. Seja G um grafo com custos, onde cada aresta $e \in E_G$ tem um custo positivo $c(e)$. A **distância** dist(u, v) entre um par u, v de vértices de G é o custo de um caminho de u a v em G de menor custo, se tal caminho existe; caso contrário dist$(u, v) = \infty$. Para o grafo com custos G da figura 5, temos dist$(u, v) = 10$ e $P : u, x, y, v$ é um caminho custo mínimo de u para v. Note que o caminho $Q : u, y, v$ não é um caminho de custo mínimo de u para v em G, mesmo Q contendo menos arestas que P.

O algoritmo para encontrar um menor caminho de Moore foi projetado para encontrar o menor caminho entre um par de vértices em um grafo, mas não determina distâncias ou menores caminhos em grafos com custos. Dijkstra [3] desenvolveu um algoritmo eficiente para encontrar menores caminhos, se eles existem, de um vértice fixo u_0 para todos os outros vértices em um grafo com custos G. Discutiremos este procedimento a seguir.

[1] Veja o algoritmo BFS(G, s) e o algoritmo IMPRIME-CAMINHO(G, s, v).

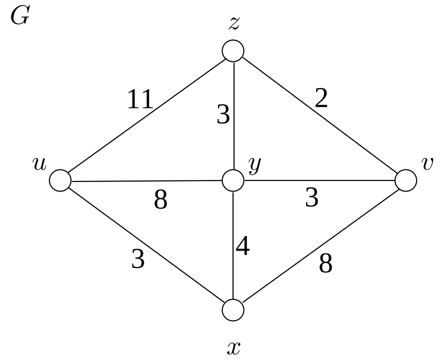

Figura 5: Distância em grafos com custos.

O algoritmo de Dijkstra rotula os vértices de G durante sua execução. No final, um vértice $v(\neq u_0)$ será rotulado com $l(v)$, onde $l(v) = \text{dist}(u_0, v)$. Inicialmente u_0 tem o rótulo $l(u_0) = 0$ e todos os outros vértices são rotulados com ∞. Para $v \neq u_0$, o rótulo $l(v)$ decrementa (possivelmente várias vezes) de ∞ para dist(u_0, v) quando esta distância é determinada. Em qualquer estágio no algoritmo, para todo vértice $v \neq u_0$, fazemos a variável PAI(v) denotar o vértice que precede v no menor caminho de u_0 para v detectado até aquele momento. Sempre que $l(v)$ é atualizado, um caminho de custo mínimo de u_0 a v é encontrado, e neste ponto PAI(v) é também atualizado para indicar qual o vértice que precede v neste caminho de custo mínimo de u_0 a v.

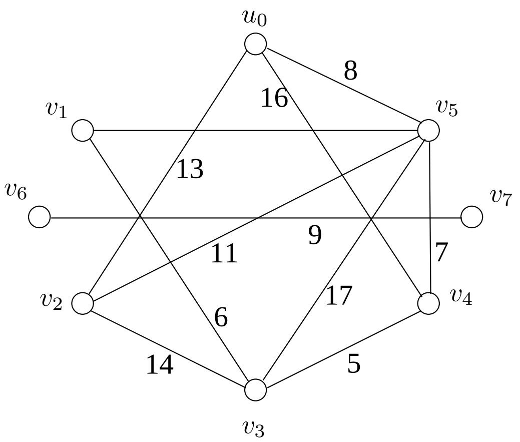

Figura 6: Encontrando distâncias em grafos com custos, utilizando o algoritmo de Dijkstra.

Os seguintes fatos preliminares nos ajudarão a compreender este algoritmo. Seja u_0 um vértice de um grafo com custos G e suponha que S é um subconjunto de V_G tal que $u_0 \in S$. Seja $\bar{S} = V_G \setminus S$ e defina a distância dist$(u_0, \bar{S})$ de u_0 para $\bar{S}$ por

$$\text{dist}(u_0, \bar{S}) = \min\{\text{dist}(u_0, x) : x \in \bar{S}\}.$$

Então, dist$(u_0, \bar{S}) = \infty$ se não existe caminho de u_0 para um vértice de $\bar{S}$. Caso contrário, existe pelo menos um vérice $v \in \bar{S}$ tal que $\text{dist}(u_0, v) = \text{dist}(u_0, \bar{S}) < \infty$. Além disso, se

$$P : u_0, u_1, u_2, \ldots, u_n, v$$

é um caminho de custo mínimo de u_0 para v em G, então $u_i \in S$ para $i = 0, 1, \ldots, n$, e $u_0, u_1, \ldots, u_n$ é um caminho de custo mínimo de u_0 para u_n. Ainda,

$$\text{dist}(u_0, \bar{S}) = \min\{\text{dist}(u_0, u) + c(uv) : u \in S, v \in \bar{S} \text{ e } uv \in E_G\}.$$

Finalmente, se este mínimo é obtido quando $u = x$ e $v = y$, então

$$\text{dist}(u_0, y) = \text{dist}(u_0, x) + c(xy),$$

que fornece uma expressão para a distância entre u_0 e y.

Podemos agora apresentar o algoritmo de Dijkstra.

DIJKSTRA(G, c, u_0): recebe um grafo G, uma função custo $c\colon E_G \to \mathbb{R}_{\geqslant 0}$ nas arestas de G e um vértice inicial u_0 e devolve a distância de u_0 a todo vértice $v \in V_G - u_0$.

1: $\{$*Inicialize os rótulos de todos os vértices $v \neq u_0$ de G para ∞ e $l(u_0) = 0$. O conjunto S é inicializado com o vértice u_0 e o conjunto $\bar{S}$ é inicializado com o conjunto $V \setminus \{u_0\}$.*$\}$
   $i \leftarrow 0$
   $S \leftarrow \{u_0\}$ e $\bar{S} \leftarrow V_G \setminus \{u_0\}$
   $l(u_0) \leftarrow 0$ e $l(v) \leftarrow \infty$, para todo $v \in \bar{S}$
   Se $p_G = 1$, então pare; caso contrário, continue.
2: $\{$*Atualiza os rótulos dos vértices $v \in \bar{S}$ que são adjacentes a u_i e* PAI(v) *é atribuído ao vértice u_i.*$\}$
   Para cada $v \in \bar{S}$ tal que $u_i v \in E_G$, proceda como segue:
   se $l(v) \leqslant l(u_i) + c(u_i v)$, então continue; caso contrário, $l(v) \leftarrow l(u_i) + c(u_i v)$ e PAI$(v) \leftarrow u_i$.
3: $\{$*Determina o próximo vértice u_{i+1} de $\bar{S}$ para o qual* $\text{dist}(u_0, u_{i+1})$ *foi encontrado.*$\}$
   Determine $m = \min\{l(v) : v \in \bar{S}\}$. Se $v_j \in \bar{S}$ é selecionado como um vértice com $l(v_j) = m$, então devolva m como a distância entre u_0 e v_j, e $u_{i+1} \leftarrow v_j$.
4: $\{$*Incrementa o conjunto S de vértices v, para os quais* $\text{dist}(u_0, v)$ *foi determinada e atualiza $\bar{S}$.*$\}$
   $S \leftarrow S \cup \{u_{i+1}\}$ e $\bar{S} \leftarrow \bar{S} \setminus \{u_{i+1}\}$.
5: $\{$*Atualiza o índice i e determina se o algoritmo terminou.*$\}$
   $i \leftarrow i + 1$. Se $i = n - 1$, então pare; caso contrário, vá para o Passo 2.

Como um exemplo, aplicamos o algoritmo de Dijkstra para o grafo G da figura 6. A tabela a seguir tem oito colunas, uma para cada vértice. Os pares ordenados na coluna correspondente ao vértice v_i indicam $(l(v_i), \text{PAI}(v_i))$, para $i = 1, 2, \ldots, 7$ em um dado ponto do algoritmo.

$l(u_0)$	v_1	v_2	v_3	v_4	v_5	v_6	v_7	adicionado a S
0	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	$(\infty, -)$	u_0
	$(\infty, -)$	$(13, u_0)$	$(\infty, -)$	$(16, u_0)$	$(8, u_0)$	$(\infty, -)$	$(\infty, -)$	v_5
	$(18, v_5)$	$(13, u_0)$	$(25, v_5)$	$(15, v_5)$		$(\infty, -)$	$(\infty, -)$	v_2
	$(18, v_5)$		$(25, v_5)$	$(15, v_5)$		$(\infty, -)$	$(\infty, -)$	v_4
	$(18, v_5)$		$(20, v_4)$			$(\infty, -)$	$(\infty, -)$	v_3
			$(20, v_4)$			$(\infty, -)$	$(\infty, -)$	v_1
						$(\infty, -)$	$(\infty, -)$	

Usando a figura 6 e a tabela acima, obtemos, para cada $i = 1, 2, \ldots, 5$, os seguintes caminhos de custo mínimo Q_i de u_0 a v_i:

$$\begin{aligned}
&Q_1 : u_0, v_5, v_1;\\
&Q_2 : u_0, v_2;\\
&Q_3 : u_0, v_5, v_4, v_3;\\
&Q_4 : u_0, v_5, v_4;\\
&Q_5 : u_0, v_5.
\end{aligned}$$

Observe que um grafo qualquer pode ser considerado como um grafo com custos, onde cada aresta tem custo 1. Assim, o algoritmo de Dijkstra pode também ser utilizado para encontrar distâncias e caminhos de custo mínimo nos grafos. Note que se o algoritmo de Dijkstra é utilizado para encontrar distâncias em grafos, então o rótulo de um vértice é modificado no máximo uma vez. Esta quantidade de trocas segue diretamente do fato de que um caminho de custo mínimo de u para v em um grafo com custos, onde todas as arestas têm o mesmo custo, é o caminho com menor número de arestas.

A corretude do algoritmo de Dijkstra é mostrada no seguinte teorema.

TEOREMA 2.1 *Seja G um grafo com custos, com n vértices. O algoritmo de Dijkstra determina a distância de um vértice fixo u_0 de G para todo vértice de G. Ou seja, quando o algoritmo termina,*

$$l(v) = \text{dist}(u_0, v), \text{ para todo } v \in V_G.$$

Ainda, se $l(v) \neq \infty$ e $v \neq u_0$, então

$$u_0, w_0, w_1, w_2, \ldots, w_k = v$$

é o caminho de custo mínimo de u_0 para v, onde $w_{i-1} = \text{PAI}(w_i)$, para $i = 1, 2, \ldots, k$.

□

Completaremos esta seção discutindo a complexidade do algoritmo de Dijkstra. Os passos mais caros do algoritmo são os passos 2 e 3. Depois da execução do algoritmo, uma aresta de G foi consultada no máximo uma vez no passo 2. A complexidade do passo 2 é portanto $O(q)$. Cada vez que o passo 3 é executado, o menor rótulo dos elementos de $\bar{S}$ deve ser determinado. Isto pode ser feito em $|\bar{S}| - 1$ comparações e então $|\bar{S}| < p$. O passo 3 é executado $p - 1$ vezes, e assim, o tempo total gasto no passo 3 é $O(p^2)$. Como $q \leqslant p(p-1)/2$, o algoritmo todo tem complexidade $O(p^2)$.

Referências

Este texto foi produzido com a consulta às referências [1] e [2] a seguir. As referências [3] e [4] são os artigos originais sobre o assunto.

[1] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, The Macmillan Press LTD, 1977.

[2] G. Chartrand and O. R. Oellermann, *Applied and Algorithmic Graph Theory*, McGraw-Hill, Inc., 1993.

[3] E. W. Dijkstra, *A note on two problems in connection with graphs*, Numerische Math, *1*, pp. 269-271, 1959.

[4] E. F. Moore, *The shortest path through a maze*, Proc. International Symposium on Switching Theory, Harvard University Press, Cambridge, pp. 285-292, 1959.