Análise e Síntese de Algoritmos

Caminhos Mais Curtos entre Todos os Pares [CLRS, Cap. 25]

2011/2012

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Árvores abrangentes
  - Caminhos mais curtos
  - Fluxos máximos
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica
  - Algoritmos greedy
- Tópicos Adicionais [CLRS, Cap.32-35]
  - Emparelhamento de Cadeias de Caracteres
  - Complexidade Computacional
  - Algoritmos de Aproximação

Resumo

Definições

Caminhos Mais Curtos Entre Todos os Pares

Encontrar caminhos mais curtos entre todos os pares de vértices

- Se pesos não negativos, utilizar algoritmo de Dijkstra, assumindo cada vértice como fonte: $O(V\ (V+E)\ lg\ V)$ (que é $O(V^3\ lg\ V)$ se o grafo é denso)
- Se existem pesos negativos, utilizar algoritmo de Bellman-Ford, assumindo cada vértice como fonte: $O(V^2E)$ (que é $O(V^4)$ se o grafo é denso)
- Objectivo: Encontrar algoritmos mais eficientes

Definições

Caminhos Mais Curtos Entre Todos os Pares

- Representação do grafo: utilização de matriz de adjacências
- Pesos dos arcos: matriz W $(n \times n)$

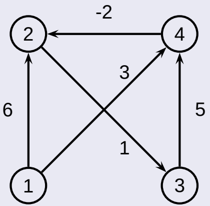
$$w_{ij} = \begin{cases} 0 & \text{se } i = j \\ \text{peso do arco } (i,j) & \text{se } i \neq j, (i,j) \in E \\ \infty & \text{se } i \neq j, (i,j) \notin E \end{cases}$$

- Representação dos caminhos mais curtos: matriz D $(n \times n)$
  - d_{ij} é o peso do caminho mais curto entre os vértices i e j
  - $d_{ij} = \delta(v_i, v_j)$

Definições

Exemplo Representação

Grafo



Matriz W

	1	2	3	4
1	0	6	∞	3
2	∞	0	1	∞
3	∞	∞	0	5
4	∞	-2	∞	0

Matriz D

	1	2	3	4
1	0	1	2	3
2	∞	0	1	6
3	∞	3	0	5
4	∞	-2	-1	0

Definições

Representação dos Caminhos Mais Curtos

- Representação dos predecessores: matriz Π $(n \times n)$
- $\pi_{ij} = \text{NIL}$ se $i = j$ ou não existe caminho de i para j
- Caso contrário: π_{ij} denota o predecessor de j num caminho mais curto de i para j
- Sub-grafo de predecessores $G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$:

$$V_{\pi,i} = \{j \in V : \pi_{ij} \neq \text{NIL}\} \cup \{i\}$$

$$E_{\pi_i} = \{(\pi_{ij}, j) \in E : j \in V_{\pi,i} - \{i\}\}$$

- Sub-grafo de predecessores $G_{\pi,i}$ é induzido pela linha i da matriz Π

Solução Recursiva

Solução Recursiva

- Propriedade de sub-estrutura óptima dos caminhos mais curtos: Sub-caminhos de caminhos mais curtos são também caminhos mais curtos
- $d_{ij}^{(m)}$: denota o peso mínimo dos caminhos do vértice i para o vértice j não contendo mais do que m arcos
- Com $m = 0$, existe caminho de i para j se e só se $i = j$

$$d_{ij}^{(0)} = \begin{cases} 0 & \text{se } i = j \\ \infty & \text{se } i \neq j \end{cases}$$

- Para $m \geq 1$:

$$d_{ij}^{(m)} = \min\{d_{ij}^{(m-1)}, \min_{1 \leq k \leq n} \{d_{ik}^{(m-1)} + w_{kj}\}\}$$

$$d_{ij}^{(m)} = \min_{1 \leq k \leq n} \{d_{ik}^{(m-1)} + w_{kj}\} \text{ porque } w_{jj} = 0$$

Solução Recursiva

Pseudo-Código

- Calcular sequência de matrizes $D^{(1)}, \ldots, D^{(n-1)}$, onde $D^{(n-1)}$ contém os pesos dos caminhos mais curtos
- Note-se que $D^{(1)} = W$

Extend-Shortest-Paths(D, W)

$n = rows[W]$
D': matriz $(n \times n)$
for $i = 1$ **to** n
    do for $j = 1$ to n
        do $d'_{ij} = \infty$
            for $k = 1$ **to** n
                do $d'_{ij} = \min(d'_{ij}, d_{ik} + w_{kj})$
return D'

- Complexidade: $\Theta(n^3)$ para cada matriz; Total: $\Theta(n^4)$

Solução Recursiva

Observações

- Genericamente: calcular $D^{(i)}$ em função de $D^{(i-1)}$ (e de W)
- Complexidade para cálculo de $D^{(n)}$: $\Theta(n^4)$
- É possível melhorar complexidade reduzindo número de matrizes calculadas: $O(n^3\ lg\ n)$
  - A cada iteração, calcular $D^{(2i)}$ em função de $D^{(i)}$ e de $D^{(i)}$

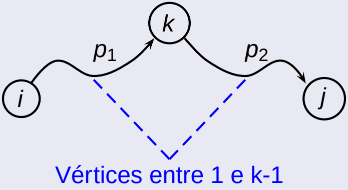
Floyd-Warshall

Conceitos

- Caracterização de um caminho mais curto $p =< v_1, v_2, \ldots, v_k >$
  - Vértices intermédios de caminho p são $\{v_2, \ldots, v_{k-1}\}$
- Considerar todos os caminhos entre i e j com vértices intermédios retirados do conjunto $\{1, \ldots, k\}$ e seja p um caminho mais curto (Nota: p é simples)
- Se k não é vértice intermédio de p, então todos os vértices intermédios de p estão em $\{1, \ldots, k-1\}$
- Se k é vértice intermédio de p, então existem caminhos p_1 e p_2, respectivamente de i para k e de k para j com vértices intermédios em $\{1, \ldots, k\}$
  - k não é vértice intermédio de p_1 e de p_2
  - p_1 e p_2 com vértices intermédios em $\{1, \ldots, k-1\}$

Floyd-Warshall

Formulação



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

Floyd-Warshall

Pseudo-Código

Floyd-Warshall(D, W)

$n = rows[W]$
$D^{(0)} = W$
for $k = 1$ **to** n
    do for $i = 1$ to n
        do for $j = 1$ **to** n
            do $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
return $D^{(n)}$

Complexidade: $\Theta(n^3)$

Floyd-Warshall

Observação

Podemos evitar uma matriz por cada passo do algoritmo. A linha e a coluna k não são alteradas na iteração k:

$$d_{ik}^{(k)} = \min(d_{ik}^{(k-1)}, d_{ik}^{(k-1)} + d_{kk}^{(k-1)})$$

$$d_{kj}^{(k)} = \min(d_{kj}^{(k-1)}, d_{kk}^{(k-1)} + d_{kj}^{(k-1)})$$

Nota: $d_{kk}^{(k-1)} = 0$

Floyd-Warshall

Pseudo-Código

Floyd-Warshall(D, W)

$n = rows[W]$
$D = W$
for $k = 1$ **to** n
    do for $i = 1$ to n
        do for $j = 1$ **to** n
            do $d_{ij} = \min(d_{ij}, d_{ik} + d_{kj})$
return D

Fecho Transitivo

Fecho Transitivo de um Grafo Dirigido

- Dado um grafo $G = (V, E)$ dirigido, o fecho transitivo é definido por $G^* = (V, E^*)$ tal que,

$$E^* = \{(i,j) : \text{ existe caminho de } i \text{ para } j \text{ em } G\}$$

- Algoritmo:
  - Atribuir a cada arco peso 1 e utilizar algoritmo de Floyd-Warshall
  - Se $d_{ij} \neq \infty$, então $(i,j) \in E^*$
  - Complexidade: $O(n^3)$

Algoritmo Johnson

Conceitos

- Utiliza algoritmos de Dijkstra e de Bellman-Ford
- Baseado em re-pesagem dos arcos
- Se arcos com pesos não negativos, utilizar Dijkstra para cada vértice
- Caso contrário, calcular novo conjunto de pesos não negativos w', tal que
  - Um caminho mais curto de u para v com função w é também caminho mais curto com função w'
  - Para cada arco (u, v) o peso $w'(u, v)$ é não negativo

Algoritmo Johnson

Re-pesagem dos arcos

- Dado $G = (V, E)$, com função de pesos w e de re-pesagem $h : V \to R$, seja $w'(u,v) = w(u,v) + h(u) - h(v)$
- Seja $p =< v_0, v_1, \ldots, v_k >$. Então $w(p) = \delta(v_0, v_k)$ se e só se $w'(p) = \delta'(v_0, v_k) = \delta(v_0, v_k) + h(v_0) - h(v_k)$
- Existe ciclo negativo com w se e só se existe ciclo negativo com w'
- Verificar que $w'(p) = w(p) + h(v_0) - h(v_k)$

$$
\begin{aligned}
w'(p) &= \sum_{i=1}^{k} w'(v_{i-1}, v_i) \\
&= \sum_{i=1}^{k} (w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i)) \\
&= \sum_{i=1}^{k} w(v_{i-1}, v_i) + h(v_0) - h(v_k) \\
&= w(p) + h(v_0) - h(v_k)
\end{aligned}
$$

Algoritmo Johnson

Propriedade de re-pesagem

Caminhos mais curtos mantém-se após a represagem. Se p é caminho mais curto com função de peso w, então também é caminho mais curto com função de peso w'

- Verificar que $w(p) = \delta(v_0, v_k) \rightarrow w'(p) = \delta'(v_0, v_k)$

Algoritmo Johnson

Propriedade de re-pesagem

Caminhos mais curtos mantém-se após a represagem. Se p é caminho mais curto com função de peso w, então também é caminho mais curto com função de peso w'

- Verificar que $w(p) = \delta(v_0, v_k) \to w'(p) = \delta'(v_0, v_k)$
- Hipótese: existe outro caminho mais curto p_z de v_0 para v_k após a re-pesagem em que $w'(p_z) < w'(p)$

Algoritmo Johnson

Propriedade de re-pesagem

Caminhos mais curtos mantém-se após a represagem. Se p é caminho mais curto com função de peso w, então também é caminho mais curto com função de peso w'

- Verificar que $w(p) = \delta(v_0, v_k) \rightarrow w'(p) = \delta'(v_0, v_k)$
- Hipótese: existe outro caminho mais curto p_z de v_0 para v_k após a re-pesagem em que $w'(p_z) < w'(p)$

$$w(p_z) + h(v_0) - h(v_k) = w'(p_z) < w'(p) = w(p) + h(v_0) - h(v_k)$$

Algoritmo Johnson

Propriedade de re-pesagem

Caminhos mais curtos mantém-se após a represagem. Se p é caminho mais curto com função de peso w, então também é caminho mais curto com função de peso w'

- Verificar que $w(p) = \delta(v_0, v_k) \rightarrow w'(p) = \delta'(v_0, v_k)$
- Hipótese: existe outro caminho mais curto p_z de v_0 para v_k após a re-pesagem em que $w'(p_z) < w'(p)$

$$w(p_z) + h(v_0) - h(v_k) = w'(p_z) < w'(p) = w(p) + h(v_0) - h(v_k)$$

- Para que isso seja verdade, temos que $w(p_z) < w(p)$, o que contradiz o facto de p ser caminho mais curto com função de peso w

Algoritmo Johnson

Propriedade de re-pesagem

Caminhos mais curtos mantém-se após a represagem. Se p é caminho mais curto com função de peso w, então também é caminho mais curto com função de peso w'

- Verificar que $w(p) = \delta(v_0, v_k) \rightarrow w'(p) = \delta'(v_0, v_k)$
- Hipótese: existe outro caminho mais curto p_z de v_0 para v_k após a re-pesagem em que $w'(p_z) < w'(p)$

$$w(p_z) + h(v_0) - h(v_k) = w'(p_z) < w'(p) = w(p) + h(v_0) - h(v_k)$$

- Para que isso seja verdade, temos que $w(p_z) < w(p)$, o que contradiz o facto de p ser caminho mais curto com função de peso w

Observação: Para quaisquer caminhos p_1, p_2 entre v_0 e v_k, verifica-se que $w(p_1) < w(p_2) \leftrightarrow w'(p_1) < w'(p_2)$

Algoritmo Johnson

Propriedade de re-pesagem

Prova de que $w'(p) = \delta'(v_0, v_k) \rightarrow w(p) = \delta(v_0, v_k)$ é muito semelhante à anterior em que se admite existir um caminho mais curto p_z com função de peso w

Algoritmo Johnson

Propriedade de re-pesagem

Prova de que $w'(p) = \delta'(v_0, v_k) \rightarrow w(p) = \delta(v_0, v_k)$ é muito semelhante à anterior em que se admite existir um caminho mais curto p_z com função de peso w

Ciclos Negativos

Existe ciclo negativo com w se e só se existe com w'

- Considere-se que o caminho $p_c =< v_0, \ldots, v_k >$ define um ciclo negativo. Então, $v_0 = v_k$ e $w(p_c) < 0$.
- $w'(p_c) = w(p_c) + h(v_0) - h(v_k) = w(p_c)$, dado que $v_0 = v_k$

Algoritmo Johnson

Propriedade de re-pesagem

Prova de que $w'(p) = \delta'(v_0, v_k) \rightarrow w(p) = \delta(v_0, v_k)$ é muito semelhante à anterior em que se admite existir um caminho mais curto p_z com função de peso w

Ciclos Negativos

Existe ciclo negativo com w se e só se existe com w'

- Considere-se que o caminho $p_c =< v_0, \ldots, v_k >$ define um ciclo negativo. Então, $v_0 = v_k$ e $w(p_c) < 0$.
- $w'(p_c) = w(p_c) + h(v_0) - h(v_k) = w(p_c)$, dado que $v_0 = v_k$

Resumo

Caminhos mais curtos e ciclos negativos não se alteram com a mudança da função de peso $w'(u,v) = w(u,v) + h(u) - h(v)$

Algoritmo Johnson

Organização

- Dado $G = (V, E)$, criar $G' = (V', E')$ definido do seguinte modo:
  - $V' = V \cup \{s\}$
  - $E' = E \cup \{(s, v) : v \in V\}$
  - $\forall v \in V : w(s, v) = 0$

Algoritmo Johnson

Organização

- Dado $G = (V, E)$, criar $G' = (V', E')$ definido do seguinte modo:
  - $V' = V \cup \{s\}$
  - $E' = E \cup \{(s, v) : v \in V\}$
  - $\forall v \in V : w(s, v) = 0$
- Ciclos negativos são detectados pela execução do algoritmo de Bellman-Ford aplicado a G'

Algoritmo Johnson

Organização

- Dado $G = (V,E)$, criar $G' = (V',E')$ definido do seguinte modo:
  - $V' = V \cup \{s\}$
  - $E' = E \cup \{(s,v) : v \in V\}$
  - $\forall v \in V : w(s,v) = 0$
- Ciclos negativos são detectados pela execução do algoritmo de Bellman-Ford aplicado a G'
- Se não existirem ciclos negativos:
  - Definir $h(v) = \delta(s,v)$
  - Pela propriedade dos caminhos mais curtos, para cada arco (u,v), temos que $h(v) \leq h(u) + w(u,v)$
  - Logo, $w'(u,v) = w(u,v) + h(u) - h(v) \geq 0$

Algoritmo Johnson

Organização

- Dado $G = (V, E)$, criar $G' = (V', E')$ definido do seguinte modo:
  - $V' = V \cup \{s\}$
  - $E' = E \cup \{(s, v) : v \in V\}$
  - $\forall v \in V : w(s, v) = 0$
- Ciclos negativos são detectados pela execução do algoritmo de Bellman-Ford aplicado a G'
- Se não existirem ciclos negativos:
  - Definir $h(v) = \delta(s, v)$
  - Pela propriedade dos caminhos mais curtos, para cada arco (u, v), temos que $h(v) \leq h(u) + w(u, v)$
  - Logo, $w'(u, v) = w(u, v) + h(u) - h(v) \geq 0$
- Executar Dijkstra para todo o vértice $u \in V$ com função de peso w'
  - Cálculo $\delta'(u, v)$ para todo o $u \in V$
  - Mas também $\delta(u, v) = \delta'(u, v) - h(u) + h(v)$

Algoritmo Johnson

Pseudo-Código

Johnson(G)

Representar G'
if Bellman-Ford(G', w, s) = FALSE
    then print “Indicar ciclo negativo”
    else atribuir $h(v) = \delta(s, v)$, calculado com Bellman-Ford
        calcular $w'(u,v) = w(u,v) + h(u) - h(v)$ para cada arco (u,v)
        for each $v \in V[G]$
            do executar Dijkstra(G, w', v); calcular $\delta'(u,v)$
                $d_{uv} = \delta'(u,v) + h(v) - h(u)$
return D

Algoritmo Johnson

Complexidade

- Bellman-Ford: $O(VE)$
- Executar Dijkstra para cada vértice: $O(V(V+E)\ lg\ V)$ (assumindo amontoado binário)

- Total: $O(V(V+E)\ lg\ V)$
- Útil para grafos esparsos