8 - Problema do Caminho Mínimo

Considere a rede:

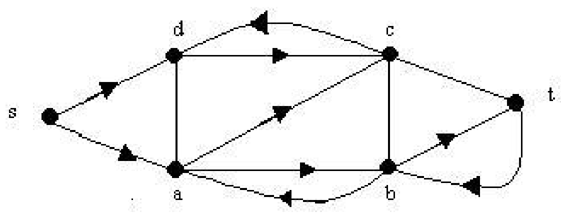


Figura 1

Dado dois vértices nesta rede, queremos determinar o menor caminho ente eles.

Uma primeira questão é como representar os valores associados às arestas neste grafo. Isto pode ser feito através da seguinte matriz de pesos: $W = [w_{ij}]$: w_{ij} = comprimento (distância, valor) da aresta orientada do vértice i ao vértice j,, e:

$w_{ij} \geq 0$ se existe a resta (i,j).

w_{ij} = 0, para i=j

$w_{ij} = \infty$ se não existe uma aresta orientada do vértice i para o vértice j.

Para o grafo da figura 1 acima a Matriz W é:

$$W = \begin{array}{c} \\ s \\ a \\ b \\ c \\ d \\ t \end{array} \begin{array}{c} \begin{array}{cccccc} s & a & b & c & d & t \end{array} \\ \begin{bmatrix} 0 & 15 & \infty & \infty & 9 & \infty \\ \infty & 0 & 35 & 3 & \infty & \infty \\ \infty & 16 & 0 & 6 & \infty & 21 \\ \infty & \infty & \infty & 0 & 2 & 7 \\ \infty & 4 & \infty & 2 & 0 & \infty \\ \infty & \infty & 5 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

Observe que:

Em geral $w_{ij} \neq w_{ji}$ e que $w_{ij} + w_{jk}$ pode ser menor ou igual a w_{ik}.

Exemplo: $w_{sd} + w_{da} \leq w_{sa} \Rightarrow (9 + 4 \leq 15)$

$w_{ab} \neq w_{ba}$

O algoritmo que nós vamos ver foi proposto por Dijkstra em 1959 e é um dos mais eficientes até hoje. Vamos supor então que queremos encontrar o caminho mínimo entre s e t na rede acima.

<u>Idéia do Algoritmo</u>: Rotular os vértices do digrafo.

A partir do vértice inicial s, proceder em direção ao vértice final t (seguindo as arestas orientadas) rotulando os vértices com as suas distâncias do vértice s, medida até aquele momento. A cada estágio do algoritmo teremos vértices que possuem rótulos temporários e outros rótulos permanentes.

O rótulo de um vértice *j* é feito permanente quando este rótulo representa a menor distância de *s* até *j*.

Começamos associando rótulo permanente igual a zero para o vértice *s*, e um rótulo temporário igual a ∞ para os outros $n - 1$ vértices do grafo. A cada iteração, um novo vértice recebe um rótulo permanente de acordo com as seguintes regras:

1) Cada vértice *j* com um rótulo temporário, recebe um novo rótulo temporário dado por:

   min { rótulo de *j*, (rótulo de $i + w_{ij}$)}

   onde *i* é o vértice que recebeu rótulo permanente na iteração anterior e w_{ij} é o valor da aresta entre o vértice *i* e o vértice *j*.

2) Encontre o menor valor entre os rótulos temporários. Este será o rótulo permanente do respectivo vértice. Em caso de empate selecione qualquer um dos candidatos e atribua rótulo permanente ao escolhido.

Repetir 1 e 2 até que o vértice destino, *t*, receba um rótulo permanente.

Exemplo: Vamos aplicar a idéia acima à rede da figura 1.

$$
\begin{array}{lcccccc}
 & s & a & b & c & d & t \\
it = 0 & \underline{0} & \infty & \infty & \infty & \infty & \infty \\
it = 1 & 0 & 15 & \infty & \infty & 9 & \infty \\
 & \underline{0} & 15 & \infty & \infty & \underline{9^*} & \infty \\
it = 2 & \underline{0} & 13 & \infty & 11 & \underline{9} & \infty \\
 & \underline{0} & 13 & \infty & \underline{11^*} & \underline{9} & \infty \\
it = 3 & \underline{0} & 13 & \infty & \underline{11} & \underline{9} & 18 \\
 & \underline{0} & \underline{13^*} & \infty & \underline{11} & \underline{9} & 18 \\
it = 4 & \underline{0} & \underline{13} & 48 & \underline{11} & \underline{9} & 18 \\
 & \underline{0} & \underline{13} & 48 & \underline{11} & \underline{9} & \underline{18^*}
\end{array}
$$

it=1

Re-rotular os vértices

Regra 1 – Novo rótulo

Novo rot (a) = $\min \{\infty, 0 + w_{sa}\}$

$= \min \{\infty, 15\} = 15$

novo rot (b) $= \min \{\infty, 0 + w_{sb}\}$

Regra 2 – Rótulo permanente

$\min \{15, \infty, \infty, 9, \infty\} = 9 \Rightarrow$ vertice d

it =2

A partir do vértice d (último a receber rótulo permanente) vamos re-rotular os vértices.

a: $\min \{15, 9 + w_{da}\} =$

$\min \{15, 9 + 4\} = 13$

c: $\min \{\infty, 9 + w_{dc}\} = 11$

vértice a receber rótulo permanente

$\min \{13, \infty, 11, \infty\} = 11 \Rightarrow verticeC$

it =3

a: $\min \{13, 11 + w_{ca}\} = 13$

b: $\min\{\infty, 11 + w_{cb}\} = \infty$
t: $\min\{\infty, u + w_{ct}\} = 18$
rótulo permanente
$\min\{13, \infty, 18\} = 13 \Rightarrow \text{vertice a}$

it =4
b: $\min\{\infty, 13 + w_{ab}\} = 48$
rótulo permanente
$\min\{48, 18\} = 18 \Rightarrow \text{vertice t}$

Podemos parar pois o vértice destino, t, recebeu rótulo permanente. Assim o menor caminho entre *s* e *t* tem comprimento igual a 18. Como recuperar o caminho? A partir do vértice destino *t*, verificamos o vértice com rótulo permanente usado na obtenção do rótulo de *t*. No nosso exemplo, o vértice *c*. Repetimos o processo a partir de *c* até alcançar o vértice inicial *s*. Assim tem *t, c, d, s.* O caminho é obtido invertendo a ordem obtida: *s,d,c,t*.

Em uma possível implementação deste algoritmo vamos precisar armazenar as seguintes informações:

- Indicação se um vértice k possui rótulo permanente ou temporário
- Guardar a menor distância entre o vértice inicial *s* e o vértice k.
- Guardar o vértice com rótulo permanente que deu origem a um novo rótulo (importante para recuperar o caminho).

Assim vamos podemos definir a seguinte estrutura de dados:

1) Entrada de Dados: Matriz de pesos *W*

$$0(n^2)$$

2) A dificuldade é distinguir, a cada iteração os vértices com rótulos permanentes, e os vértices com rótulo temporário.
Utilizando um vetor lógico (ou binário) *n*-dimensional

$$Final(v) = \begin{cases} true & \text{se o rótulo do vértice v é permanente} \\ false & \text{se o rótulo é temporário} \end{cases}$$

b) Precisamos também de um vetor *n*-dimensional para guardar as distâncias acumuladas do vértice inicial *s* ao vértice *i.* Vamos chamar este vetor de dist.

c) Como recuperar o caminho? Sabemos que a menor distância será dada por dist(t). Mas qual é este caminho?
Cada vez que o rótulo de um vértice é modificado precisamos saber a partir de que vértice foi calculado o novo rótulo. Mantendo um vetor *n*-dimensional PRED tal que:
PRED (v)= indica o vértice com rótulo permanente que deu origem ao rótulo do vétice v, e se v for o vértice inicial, então Pred(s) = -1, temos que o menor caminho é dado por:
S, pred(pred(. . .)), . . ., pred(pred(t), pred(t), t.

Rastrear o algorítmo de Dijkstra usando o digrafo cuja matriz de pesos associada é:

$$
W = \begin{array}{c} \\ S \\ A \\ B \\ C \\ D \\ T \end{array}
\begin{array}{c} \begin{array}{cccccc} S & A & B & C & D & T \end{array} \\
\begin{bmatrix} 0 & 7 & 4 & 9 & 7 & \infty \\ 7 & 0 & 1 & \infty & \infty & 6 \\ 4 & 1 & 0 & 3 & \infty & \infty \\ 9 & \infty & 3 & 0 & 1 & 3 \\ 7 & \infty & \infty & 1 & 0 & 5 \\ \infty & 6 & \infty & 3 & 5 & 0 \end{bmatrix} \end{array}
$$

dist

S	A	B	C	D	T
∞	∞	∞	∞	∞	∞
0	7	4	9	7	11
	5		7		10

final

S	A	B	C	D	T
F	F	F	F	F	F
T	T	T	T	T	T
	it=2	it=1	it=3	it=4	it=5

pred

S	A	B	C	D	T
-1	-1	-1	-1	-1	-1
	S	S	S	S	A
	B		B		C

It=1
Recent= s
Para todos os sucessores v de recent se final(v)=F:

$v = A$: new label $\Leftarrow$ dest(s) + $w_{recent,v}$

new label $\Leftarrow 0 + w_{sA} = 0 + 7 = 7$

se new label < dist(A) então

7 $< \infty$

dest(A) $\Leftarrow$ 7

pred (A) = s

$v = B$: new label $\Leftarrow 0 + w_{sB} = 0 + 4 = 4$

se 4<∞

dest (B) $\Leftarrow 4$

pred (B) = s

$v = C$: new label $\Leftarrow 0 + w_{sC} = 0 + 9$

se 9 < ∞

dest (C) $\Leftarrow 9$

pred (C) = s

$v = D$: new label $\Leftarrow\ 0 + w_{sD} = 0 + 7$

se 7 < ∞

dest (D) $\Leftarrow 7$

pred (D) = *s*

Determine o vértice com menor rótulo temporário

Y = B

Final (B) = T

Recent = B

It=2

recent = B – Vamos calcular novos rótulos para todos os vértices adjacentes a B e que não possuem rótulo

s $\Rightarrow$ já possui rótulo permanente

A: new label $\Leftarrow$ dist(B) + $w_{BA} = 4 + 1 = 5$

5<7

dist(A) = 5

pred(A) = B

C: new label $\Leftarrow 4 +\ w_{BC} = 4 + 3 = 7$

7<9

dist(C) = 7

pred(C) = B

Vértice com menor rótulo temporário: y=A

Final(A) = T

Recent = A

It=3

Sucessores de A com rótulo temporário

T: new label $\Leftarrow$ dist(A) + $w_{AT} = 5 + 11$

11< ∞

dist(T) = 11

pred(T) = A

Vamos escolher vértice com menor rótulo temporário $\Rightarrow$ y = V ou D

Final (C) = T

Recent = C

It=4

Sucessores de C com vértice temporário

T: new label $\Leftarrow$ dist(C) + $w_{ED} = 7 + 1 = 8$

8</7 não

new label $\Leftarrow$ dist(C) + $w_{ED} = 7 + 3 = 10$

10<11 não

dist(T) = 10

pred(T) = C

y = D

Final (D) = True

Recent = D

It=5
T: new label $\Leftarrow$ dist(D) = $7 + w_{DT} = 7 + 5 = 12$
$12 \not< 10$
y = T
Final (D) = True
Recent = T

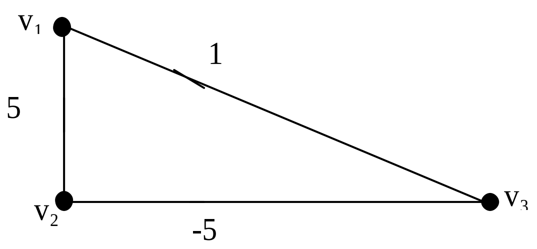
Como final (T) = True $\Rightarrow$ Fim

Qual é o valor do menor caminho?
Dist(T) = 10
Qual é este caminho?
T, Pred(T)
T, C, Pred (C)
T, C, B, Pred (B)
T, C, B, S, Pred(S)
T, C, B, S, -1
O menor caminho é {S,B,C,T}

E se quisermos o menor caminho entre S e C? Podemos aproveitar os cálculos anteriores .
Como C possui rótulo permanente, temos que:
Dist(C) = 7
E o caminho é:
C, Pred(C)
C, B, Pred(B)
C, B, S e o menor caminho é então: {S,B,C}

Observação:

1) A complexidade em termos de tempo computacional do algoritmo acima é dada por:
O laço principal deste algoritmo, no pior caso, é executado (n-1) vezes. Isto acontece quando o vértice final, t, é o último a receber um rótulo permanente. Para cada execução deste laço, precisamos examinar uma linha da matriz de pesos, e atualizar os vetores dist e pred, ou seja um tempo proporcional a *n*. Assim o tempo total é da ordem $2n\ (n\text{-}1)$. A complexidade é $0(n^2)$, observe que independe do número de arestas no grafo.
2) O algoritmo de Dijskstra funciona apenas se $w_{ij} \geq 0, \forall i, j$. Verifique este fato aplicado o algoritmo á seguinte rede para encontrar o menor caminho entre v_1 e v_3:



3) É possível, usando o algoritmo de Dijkstra encontrar o menor caminho entre o vértice *s* e todos os outros vértices do grafo. O que deve ser modificado?
4) Outros Algoritmos para problemas de Caminho Mínimo em grafos podem ser encontrados por exemplo na pg. 283 de: Otimização Combinatória e Programação Linear, M.C. Goldbarg e H.P.L. Luna, Ed. Campus, 2000.

Algoritmo - Caminho Mínimo em Grafos

Considere um digrafo G(V, A), com n vértices e sua matriz de pesos W = [w_{ij}], nxn, tal que:

$$w_{ij} = \begin{cases} \text{p, valor da aresta (i, j)} \\ 0, \quad i = j \\ \infty, \text{se não existe a aresta (i, j)} \end{cases}.$$

Queremos encontrar o menor caminho entre o vértice S e o vértice T no digrafo G.

Defina os vetores:
final (i) – indica se o vértice i recebeu rótulo permanente (potencial) ou não
dist(i) – indica a distância acumulada do vértice inicial S até o vértice i
pred(i) – indica o vértice com rótulo permanente que deu origem ao rótulo do vétice i.

Algoritmo de Djkstra [1]

Inicialização
para todo $v \in V$ **faça**
    início
        dist(v) = ∞
        final(v) = falso
        pred(v) = -1
    fim
dist(S) = 0
final(S) = verdadeiro
recente = S

Iteração Principal

enquanto final(T) = falso **faça**
    início
        para todo vértice j tal que exista a aresta (*recente*,*j*) e tal que final (j) = falso **faça**
            início {atualização dos rótulos}
                rótulo = dist(*recente*) + $w_{recente, j}$
                {o rótulo de j é modificado se houver um caminho menor de S a j através do vértice *recente*}
                se rótulo < dist(j) **então**
                    início
                        dist(j) = rótulo
                        pred(j) = *recente*
                    fim
            fim
    {encontre o vértice com menor rótulo temporário}
    Seja y o vértice com menor rótulo temporário, tal que dist(y) ≠ ∞
        inicio {o rótulo do vértice y se torna permanente}
            final(y) = verdadeiro
            recente = y
        fim
    fim

[1] – Discrete Optimization Algorithms – M. Syslo, N. Deo, J.S. Kowallk