# Dijkstra

Os nós do grafo da Figura 1 representam cidades, e os arcos, a presença de uma estrada ligando duas destas cidades. Os números ao lado dos arcos representam a distância medida em quilômetros. Neste caso temos um grafo ponderado, pois foi inserido um valor associado as arestas. Este valor é um peso que representa o custo para o caminhamento do nó A até B.

Figura 1 - Grafo do exemplo 1

1

3

2

4

5

6

100

180

101

120

90

200

45

15

40

Pode-se representar, numericamente, este grafo por uma variável indexada bidimensional, na qual a distância entre duas cidades i,j é indicada pelo elemento D(i,j); se i = j ou se não houver conexão entre i e j, D(i,j) será zero. Desta forma tem-se:

Figura 2 – Matriz de incidência do exemplo 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 0 | 100 | 15 | 0 | 0 | 0 |
| **2** | 100 | 0 | 40 | 180 | 200 | 0 |
| **3** | 15 | 40 | 0 | 45 | 90 | 0 |
| **4** | 0 | 180 | 45 | 0 | 0 | 101 |
| **5** | 0 | 200 | 90 | 0 | 0 | 120 |
| **6** | 0 | 0 | 0 | 101 | 120 | 0 |

O problema consiste, então, em se achar o caminho mais curto entre duas cidades quaisquer, Este problema foi resolvido por Dijkstra em 1971 e tem uma séria de aplicações em questões de otimização.

Além da matriz D das distâncias, considera-se a variável indexada unidimensional DA, cujo elemento DA[I] representa a distância acumulada em um caminho percorrido desde a origem até a cidade I. Cada uma destes elementos será iniciado com um valor bem grande, por exemplo 10000.

Ainda serão consideradas mais duas variáveis indexadas unidimensionais. A primeira, designada Ant, será tal que o seu elemento Ant[I] indica qual é a cidade antecedente de I no caminho considerado. A outra, ExpA, terá elementos lógicos, todos eles inicialmente com o valor falso, indicando que as cidades ainda não foram “expandidas”.

Partindo de uma cidade C inicialmente igual à origem, calcula-se a nova distância acumulada (NovaDA) de cada uma das cidades adjacentes a C ainda não expandidas. A nova distância prevalecerá sobre o valor anterior se lhe for inferior; neste caso, C será atribuído ao elemento Ant[i]. Quando terminar a expansão de C, registra-se que ExpA[C] é verdadeiro.

Em seguida, procura-se, dentre as cidades ainda não expandidas, aquela que tenha a menor distância acumulada. Esta será a nova cidade C, e a sua distância acumulada é, então, a menor que possa ser conseguida a partir da origem.

O processo será repetido até que a cidade C seja o destino ou que não se encontre nenhuma cidade ainda não expandida, cuja distância acumulada seja inferior a 10000. Neste último caso, isto significa que não existe caminho ligando a Origem ao Destino.

Abaixo o algoritmo do problema:

Algoritmo "Dijkstra"

// O problema consiste, então, em se achar o caminho mais curto entre duas cidades quaisquer,

// Este problema foi resolvido por Dijkstra em 1971 e tem uma séria de aplicações em questões de otimização.

Var

D : vetor[1..100,1..100] de Inteiro

DA, Ant : vetor[1..100] de Inteiro

ExpA : vetor[1..100] de Logico

N, Origem, Destino, i, j, C, NovaDA, Min : Inteiro

Inicio

Escreva("Digite a quantidade de Cidades:")

Leia(N)

Para i de 1 ate N Faca

Para j de 1 ate N Faca

Escreva("Digite a distancia da cidade ",i," até ",j,":")

Leia (D[i,j])

Fimpara

Fimpara

//Leitura da cidade de origem e destino

Escreva("Qual a cidade de origem:")

Leia (Origem)

Escreva("Qual a cidade de destino:")

Leia (Destino)

// O elemento DA[I] representa a distância acumulada em um caminho percorrido desde a origem até a cidade I.

// Cada uma destes elementos será iniciado com um valor bem grande, por exemplo 10000.

Para i de 1 ate N Faca

ExpA[i] <- FALSO

DA[i] <- 10000

Fimpara

// Ainda serão consideradas mais duas variáveis indexadas unidimensionais.

// A primeira, designada Ant, será tal que o seu elemento Ant[I] indica qual é a cidade antecedente de I no caminho considerado.

// A outra, ExpA, terá elementos lógicos, todos eles inicialmente com o valor falso, indicando que as cidades ainda não foram "expandidas".

C <- Origem

DA[C] <- 0

// Partindo de uma cidade C inicialmente igual à origem,

Enquanto ((C <> Destino) E (C <> 0)) Faca

Para i de 1 ate N Faca //Expansao de C

Se ((D[C,i] <> 0) E (NAO ExpA[i])) Entao

//calcula-se a nova distância acumulada (NovaDA) de cada uma das cidades adjacentes a C ainda não expandidas.

NovaDA <- DA[C] + D[C,i]

//A nova distância prevalecerá sobre o valor anterior se lhe for inferior; neste caso, C será atribuído ao elemento Ant[i].

Se (NovaDA < DA[i]) Entao

DA[i] <- NovaDA

Ant[i] <- C

Fimse

Fimse

Fimpara

// Quando terminar a expansão de C, registra-se que ExpA[C] é verdadeiro.

ExpA[C] <- VERDADEIRO

//Determinacao do Proximo C

Min <- 10000

C <- 0

Para i de 1 ate N Faca

//Em seguida, procura-se, dentre as cidades ainda não expandidas, aquela que tenha a menor distância acumulada.

Se( (NAO ExpA[i]) E (DA[i] < Min)) Entao

//Esta será a nova cidade C, e a sua distância acumulada é, então, a menor que possa ser conseguida a partir da origem.

Min <- DA[i]

C <- i

Fimse

Fimpara

// O processo será repetido até que a cidade C seja o destino ou que não se encontre nenhuma cidade

// ainda não expandida, cuja distância acumulada seja inferior a 10000.

// Neste último caso, isto significa que não existe caminho ligando a Origem ao Destino.

Fimenquanto

// Exibindo o caminho entre origem e destino

// Se C = Destino encontrei o caminho caso contrário não existe caminho entre as cidades.

Se (C = Destino) Entao

Escreva("Caminho mais curto de ",Origem," ate o destino ",Destino)

Enquanto (C <> Origem) Faca

C <- Ant[C]

Escreva(C)

Fimenquanto

Senao

Escreva("Nao existe caminho entre as cidades")

Fimse

Fimalgoritmo

# referência

http://pt.wikipedia.org/wiki/Algoritmo\_de\_Dijkstra