UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
ESCOLA DE INFORMÁTICA APLICADA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Quinto Trabalho
Algoritmos em Grafos
Caminhos Mínimos - Algoritmo de Floyd-Warshall

Bruno Lírio Alves
Pedro Paulo Gouveia
Thales Veras

Vânia Felix

Rio de Janeiro, 26 de novembro de 2013.

2. Motivação
2.1 Definição do Problema

O algoritmo de Floyd-Warshall, encontra o menor caminho entre todos os pares de vértices de um grafo valorado. É um algoritmo que resolve o problema de encontrar o menor caminho entre todos os pares de vértices de um grafo orientado e ponderado. É apenas encontrar os valores de tais caminhos, não a sequencia de arestas a ser percorrida.

Algumas aplicações de definição do problema:

- Calcular o Fecho Transitivo de um grafo.
- Verificar se um grafo não-dirigido é bipartido.
- Achar um vértice central, isto é, que minimiza a distância máxima ou média entre todos os vértices.

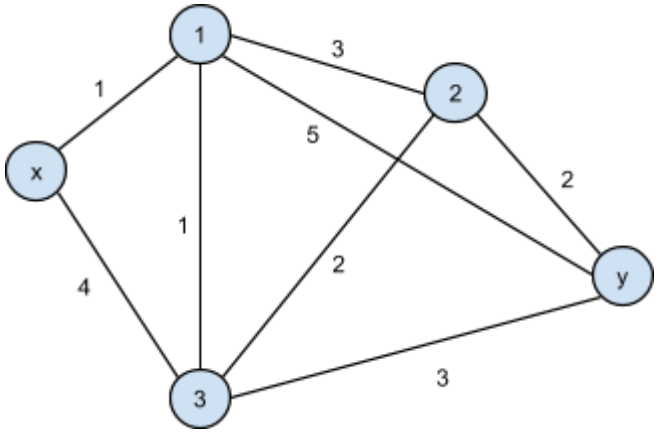
Poderíamos escolher uma descrição da ideia do algoritmo, por exemplo, em como avaliar o melhor local para instalarmos uma loja. De fato, podemos definir como melhor local aquele que diminui a distância entre a loja e locais estratégicos e importantes para ela.

- Um bairro onde o consumo dos produtos vendidos por ela é alto.
- Estabelecimentos que prestarão serviçoes para a loja.
- Um local onde se tenha uma grande concetração de um público alvo para a loja.

Definição Formal do Problema:
Entrada: Matriz de adjacência representando os pesos das arestas de um grafo orientado.
Questões: Calcula o caminho mínimo entre quaisquer dois vértices.
Saída: Uma matriz conterá todas as distâncias dos menores caminhos.

2.2. Exemplos de Instâncias do Problema

Dado um grafo G direcionado e ponderado, encontrar para todo par u, v de vértices um caminho mínimo de u a v.



O algoritmo de Floyd-Warshall tem objetivo para calcular o caminho mínimo entre cada par de vértices de um grafo

- O grafo pode conter arestas negativas

- Não pode conter ciclos negativos
- Utilizar técnica de programação dinânimca.

3. Algoritmo

3.1. Descrição da ideia do algoritmo

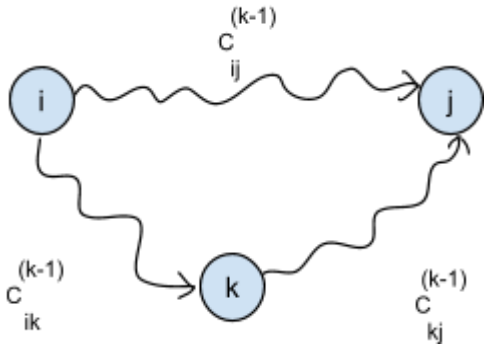
Para computar a distância do caminho minimo entre quaisquer dois vértices de um grafo, o algoritmo CaminhoMínimo poderia ser usado diversas vezes, com cada vértice como vértice origem de cada vez. Um problema de caminho mínimo entre "todos os pares". Uma descrição da ideia deste algoritmo, onde A é a matriz de adjacências do grafo com A[i,j] = 0 para todo i.

Um Algoritmo melhor segue o processo da programação dinâmica, então, devemos ter uma substrutura ótma e uma solução recursiva.

A substrutura ótima reside no fato de que os subcaminhos de caminhos mais curtos são caminhos mais curtos. Já a solução recursiva é dada abaixo:

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0, \\ \min\left(d_{ij}^{(k-1)},\ d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \right) & \text{if } k > 0. \end{cases}$$

Assim, considerando os vértices intermediários V = {1,2,...k}, a ideia geral é primeiro pensar no menor caminho sem vértices intermediários, ou seja, quando k =0; depois pensar no menor caminho que passa pelo vértice intermediário 1; depois pensar no menor caminho que passa pelos vértices intermediários 1 e 2. Desta forma a medida que você aumenta k, você só precisa pensar sobre o novo vértice adicionado, como mostra a figura abaixo:



3.2. A partir da ideia, mostrar exemplos* de solução para as instâncias apresentadas

O algoritmo preenche uma matriz bidimensional, ou seja, matriz de adjacência, caminho[][], onde caminho [a][b] é o tamanho do menor caminho entre os nós a e b. Além disso,

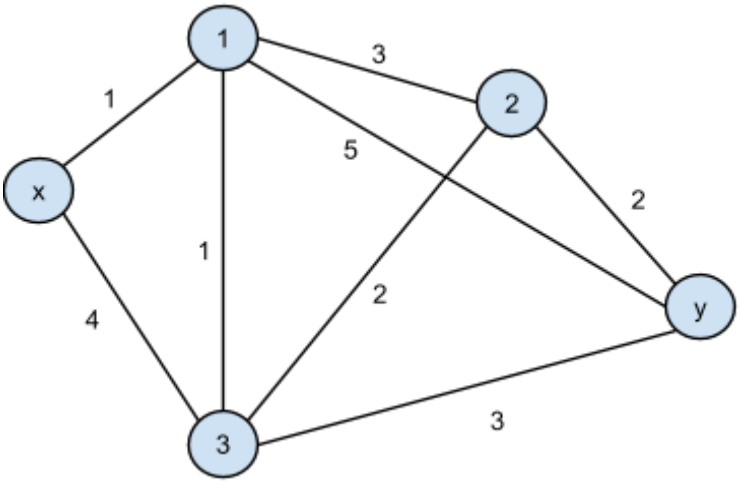
assume-se que esta matriz está inicialmente preenchida com o valor de cada aresta ou infinito, caso não haja uma aresta entre dois vértices.

Começamos fixando um vértice k do grafo; para cada par (i,j) de vértices, então, verificamos se o menor caminho já conhecido entre (i,j) supera a soma do tamanho do caminho de i para k com o de k para j. Caso supere, o tamanho do menor caminho passa a ser essa soma.

3.3. Descrição Formal do Algoritmo

```
CaminhoMínimoTodosOsPares (A: matriz n x n);
// Algoritmo de Floyd - calcula o caminho mínimo entre
//quaisquer dois vértices; inicialmente A é a matriz de
//adjacências; ao fim, A conterá todas as distâncias dos
//menores caminhos.
var i,j,k: integer
for k:= 1 to n do
      for i:= 1 to n do
            for j:= 1 to n do
                  A[i,j] := min (A[i,j], A[i,k] + A[k,j]);
```

3.4. Etapas da aplicação da técnica



Inicial: A, após k = x

	X	1	2	3	Y
x	0	1	∞	4	∞
1	1	0	3	1	5
2	∞	3	0	2	2
3	4	1	2	0	3
y	∞	5	2	3	0

Após k = 1 e k = 2

	X	1	2	3	Y
x	0	1	4	2	6
1	1	0	3	1	5
2	4	3	0	2	2
3	2	1	2	0	3
y	6	5	2	3	0

Após k = 3 e k = y

	X	1	2	3	Y
x	0	1	4	2	5
1	1	0	3	1	4
2	4	3	0	2	2
3	2	1	2	0	3
y	5	4	2	3	0

4. Análise do Algoritmo

4.1. Prova de corretude

Considerando os vértices intermediários entre i e j, digamos o subconjunto de vértices V = {1,2,...,k}, o algoritmo primeiramente considera o menor caminho entre i e j quando k = 0 se houver, atribuindo a esta distância então, o valor da única aresta entre estes vértices. Isto feito, o algoritmo segue verificando a cada novo vértice intermediário adicionado qual a menor distância, se a que já calculamos ou a distância com a adição do novo vértice, e segue fazendo esta verificação para os possíveis valores de k.

Assim, ao final das verificações, certamente teremos a menor distância uma vez que verificamos a menor distância para os possíveis vértices intermediários, guardamos a menor e descartamos as outras.

4.2. Demonstração da complexidade/tempo do algoritmo

```
for k:= 1 to n do
      for i:= 1 to n do
            for j:= 1 to n do
                  A[i,j] := min (A[i,j], A[i,k] + A[k,j]);
```

n n n

Notamos que o algoritmo de Floyd-Warshall possui complexidade O (n^3), onde é o número de vértices do grafo fornecido. O laço principal é executado n vezes e o laço interno considera cada um dos O (n^2) pares de vértices, realizando um operação de tempo constante para cada par. Se usarmos uma estrutura de dados como a matriz de adjacência, temos um tempo de execução total de O (n^3).

5. Conclusão e Discussões

Apesar do algoritmo não fornecer a sequência de arestas dos caminhos mínimos, é muito vantajoso termos a menor distância entre todos os pares de nós de um grafo o que nos possibilita resolver uma série de problemas relacionados a suas aplicações. Um deles, e de grande importância é o fecho transitivo que é uma matriz de 1s e 0s, 1s para os vértices i e j para os quais há um caminho entre eles e 0s paras os quais não há. Assim, basta verificar a matriz resultante do algoritmo Floyd-Warshall, se houver algum número para i e j então há um caminho entre eles e a correspondente matriz do fecho transitivo recebe 1 em ij, caso contrário, se for infinito a matriz de fecho transitivo recebe 0.

5.1. Discutir as vantagens e desvantagens do procedimento adotado.

Vantagens:

- Tem como saída uma matriz de caminhos mínimos
- Pode trabalhar com arestas de pesos negativos

Desvantagens:

- Não fornece a sequência de arestas dos caminhos mínimos

5.2. É possível ter um algoritmo melhor?

Não para um algoritmo que pretende dar a menor distância entre todos os pares de nós.

6. Referências Bibliográficas

Algoritmo de Floyd Warshall (2013). WIKIPEDIA. ***Site.*** Disponível em: <http://pt.wikipedia.org/wiki/Algoritmo_de_Floyd-Warshall>. Acesso em: 20 nov. 2013

CORMEN, Thomas H.. **Algoritmos: teória e prática.** Rio de Janeiro: Editora Campus, 2002.

SZWARCFITER, Jayme Luiz **Estruturas de Dados e Seus Algoritmos.** Rio de Janeiro Editora LTC, 2010.

GERSTING, Judith L. **Fundamentos Matemáticos para a Ciência da Computaçã.** Rio de Janeiro Editora LTC, 3ª edição, 1995.