Análise de Algoritmos - T2

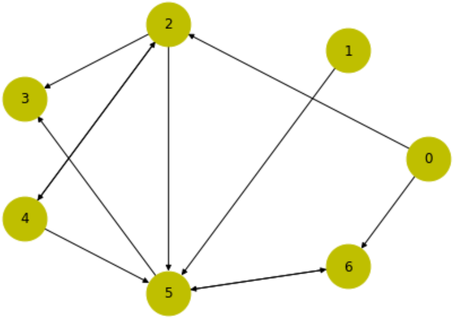
Renato Rodrigues   Vandré Leal

Universidade Federal de Uberlândia

08/11/2018

Introdução

Grafo: Estrutura em forma de diagrama que descreve a relação entre objetos de um determinado conjunto.



Grafos

Figura: Linha de metrô de Londres.

Problema

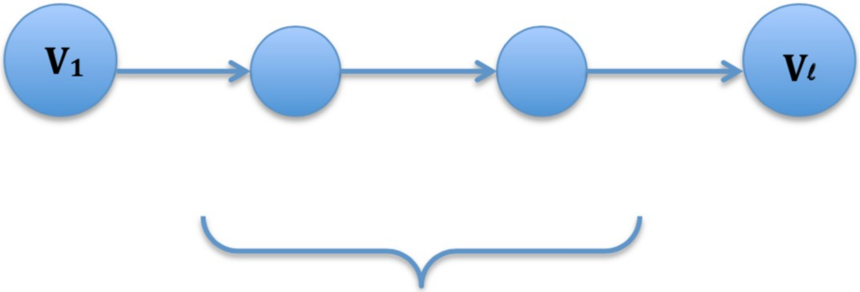
Encontrar o menor caminho entre todos os pares de vértices de um grafo ponderado.

Passos da Programação dinâmica

- Caracterizar a subestrutura ótima do problema.
- Definir a expressão recursiva para encontrar a solução ótima.
- Especificar um algoritmo para computar o valor da solução ótima.

Vértice intermediário

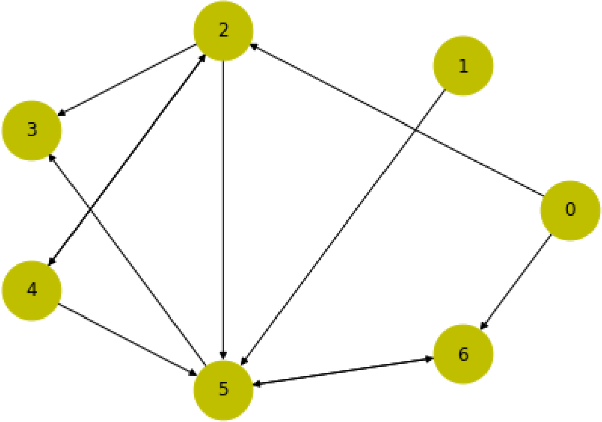
Um vértice intermediário de um caminho p $= \{v_1, v_2, ..., v_l\}$ é qualquer vértice pertencente ao caminho exceto v_1 e v_l, ou seja, $\{v_2, ..., v_{l-1}\}$



Subestrutura ótima

- O menor caminho não contém o mesmo vértice mais de uma vez.
- Para um menor caminho de i até j tal que quaisquer vértices intermediários no caminho são escolhidos do conjunto 1,2,...,k, há duas possibilidades:
  - k não é um vértice no caminho, então o menor caminho tem tamanho $d_{ij}^{(k-1)}$
  - k é um vértice no caminho, então o menor caminho é $d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

Subestrutura ótima



Expressão recursiva

O peso do menor caminho de um vértice i até um vértice j no qual todos os vértices intermediários estão no conjunto {1,2,...,k} é

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

Matrizes

Duas matrizes n x n:

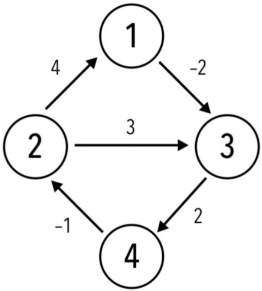
- Distâncias entre os pares de vértices.
- Antecessores de cada vértice.

Valores infinitos ou null significam que não há caminho entre dois pares de vértices ou o caminho ainda não foi encontrado.

Algoritmo

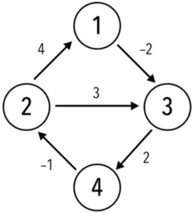
```
let V = number of vertices in graph
let dist = V × V array of minimum distances
for each vertex v
  dist [v][v] ← 0
for each edge (u,v)
  dist [u][v] ← weight(u,v)
for k from 1 to V
  for i from 1 to V
    for j from 1 to V
      if dist [i][j] > dist [i][k] + dist [k][j]
        dist [i][j] ← dist [i][k] + dist [k][j]
      end if
```

V = 4



Algoritmo

```
let V = number of vertices in graph
let dist = V × V array of minimum distances
for each vertex v
  dist [v][v] ← 0
for each edge (u,v)
  dist [u][v] ← weight(u,v)
for k from 1 to V
  for i from 1 to V
    for j from 1 to V
      if dist [i][j] > dist [i][k] + dist [k][j]
        dist [i][j] ← dist [i][k] + dist [k][j]
      end if
```



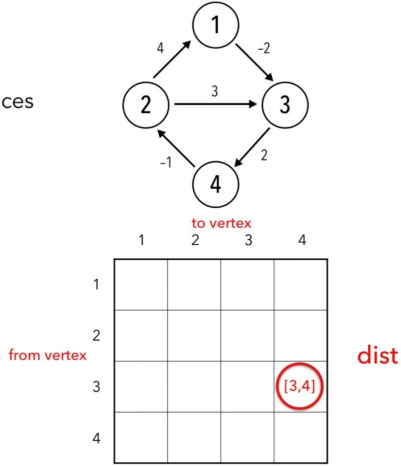
	1	2	3	4
1				
2				
3				
4				

dist

empty cells = ∞

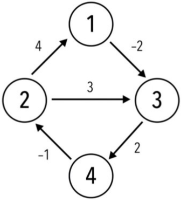
Algoritmo

```
let V = number of vertices in graph
let dist = V × V array of minimum distances
for each vertex v
  dist [v][v] ← 0
for each edge (u,v)
  dist [u][v] ← weight(u,v)
for k from 1 to V
  for i from 1 to V
    for j from 1 to V
      if dist [i][j] > dist [i][k] + dist [k][j]
        dist [i][j] ← dist [i][k] + dist [k][j]
      end if
```



Algoritmo

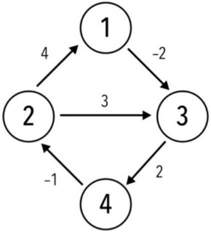
```
let V = number of vertices in graph
let dist = V × V array of minimum distances
for each vertex v
  dist [v][v] ← 0
for each edge (u,v)
  dist [u][v] ← weight(u,v)
for k from 1 to V
  for i from 1 to V
    for j from 1 to V
      if dist [i][j] > dist [i][k] + dist [k][j]
        dist [i][j] ← dist [i][k] + dist [k][j]
      end if
```



	1	2	3	4
1	0			
2		0		
3			0	
4				0

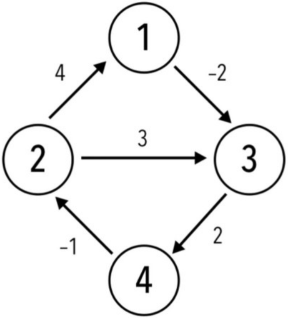
Algoritmo

```
let V = number of vertices in graph
let dist = V × V array of minimum distances
for each vertex v
  dist [v][v] ← 0
for each edge (u,v)
  dist [u][v] ← weight(u,v)
for k from 1 to V
  for i from 1 to V
    for j from 1 to V
      if dist [i][j] > dist [i][k] + dist [k][j]
        dist [i][j] ← dist [i][k] + dist [k][j]
      end if
```



	1	2	3	4
1	0			
2		0		
3			0	
4				0

Algoritmo



	1	2	3	4
1	0	-1	-2	0
2	4	0	2	4
3	5	1	0	2
4	3	-1	1	0

Algoritmo

A complexidade de tempo do algoritmo é $O(n^3)$ onde n representa o número de vértices do grafo.

Vantagens do algoritmo

- Apropriado para encontrar caminhos mais curtos entre nós de grafos densos.
- Apropriado para esta operação em grafos com pesos negativos.