MO417 – Ata do Exercício 25.2-4

Marcos Vinícius Mussel Cirne

13 de junho de 2010

Enunciado: O algoritmo de Floyd-Warshall exige $O(n^3)$ de espaço de armazenamento, uma vez que computamos $d_{ij}^{(k)}$ para $i, j, k = 1, 2, ..., n$. Mostre que o procedimento a seguir (definido por FLOYD-WARSHALL'), que simplesmente elimina os sobrescritos, está correto, exigindo assim um espaço de $O(n^2)$.

O algoritmo original de Floyd-Warshall é definido da seguinte forma:

Algoritmo 1 FLOYD-WARSHALL(W)

```
1: n ← rows[W]
2: D^(0) ← W
3: for k ← 1 to n do
4:     for i ← 1 to n do
5:         for j ← 1 to n do
6:             d_ij^(k) ← min(d_ij^(k−1), d_ik^(k−1) + d_kj^(k−1))
7:         end for
8:     end for
9: end for
10: return D^(n)
```

E sua versão modificada, que elimina os sobrescritos, é definida através de uma simples modificação na linha 6 do algoritmo anterior:

Algoritmo 2 FLOYD-WARSHALL'(W)

```
1: n ← rows[W]
2: D ← W
3: for k ← 1 to n do
4:     for i ← 1 to n do
5:         for j ← 1 to n do
6:             d_ij ← min(d_ij, d_ik + d_kj)
7:         end for
8:     end for
9: end for
10: return D
```

A versão modificada do algoritmo de Floyd-Warshall simplesmente utiliza uma única matriz $n \times n$, onde $n = |V|$, para todas as n iterações do loop mais externo, ao invés de utilizar uma matriz diferente em cada iteração. Devemos então provar que os valores de d_{ik} e de d_{kj} não são modificados na iteração k, pois do contrário ocorreriam atualizações indevidas no custo de um caminho de um vértice i para um vértice j.

Na versão original do algoritmo, temos que $d_{ik}^{(k-1)}$ corresponde ao custo do caminho que parte do vértice i e chega no vértice k, contendo vértices intermediários entre 1 e $k-1$. O custo desse caminho na iteração $k-1$ será o mesmo atribuído a $d_{ik}^{(k)}$ na iteração k, pois inserir o vértice k em um caminho que já o contém não pode acarretar a diminuição do custo desse caminho (pressupondo que o grafo não contenha ciclos negativos). Em outras palavras, $d_{ik}^{(k)} = d_{ik}^{(k-1)}$.

Para provar que isso de fato ocorre, vamos analisar a expressão da linha 6 do algoritmo original para $d_{ik}^{(k)}$:

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

Na expressão acima, temos que $d_{kk}^{(k-1)} = 0$, pois não há ciclos negativos no grafo. Desta forma, temos:

$$\begin{aligned} d_{ik}^{(k)} &= \min(d_{ik}^{(k-1)}, d_{ik}^{(k-1)}) && (2) \\ &= d_{ik}^{(k-1)} && (3) \end{aligned}$$

O caso de $d_{kj}^{(k)}$ é análogo. Fazendo os respectivos cálculos, temos:

$$d_{kj}^{(k)} = \min(d_{kj}^{(k-1)}, \underbrace{d_{kk}^{(k-1)}}_{0} + d_{kj}^{(k-1)}) \tag{4}$$

$$= d_{kj}^{(k-1)} \tag{5}$$

Desta forma, podemos eliminar os sobrescritos da expressão da linha 6 do algoritmo original sem afetar a sua corretude, já que não é necessário recorrer a valores anteriores a uma determinada alteração. Logo, o algoritmo modificado está correto e podemos utilizar espaço $O(n^2)$ para guardar os valores dos caminhos ao longo de sua execução.