MO417 Complexidade de Algoritmos - Ata

Redatora: Sheila Maricela Pinto Cáceres

2009

Exercicio 22.1-6

Quando uma representação de matriz de adjacências é usada, a maioria dos algoritmos de grafos exige o tempo $\Omega(V^2)$, mas existem algumas exceções. Mostre que detectar se um grafo orientado G contém um ***ralo universal*** - um vértice com grau de entrada $|V| - 1$ e grau de saída 0 - pode ser determinado em tempo $O(V)$, dada uma matriz de adjacências para G.

Solução:

Um ***ralo universal*** é um vértice em um grafo orientado G que tem uma aresta de chegada desde todos os outros vértices mas nenhuma aresta de saída.
Em uma matriz de adjacências de G, a linha que representa o vértice que é *ralo universal* estará preenchida com zeros e a coluna com uns excepto um campo (intersecção com a linha do ralo universal). A Tabela 1 mostra um exemplo de um grafo com *ralo universal* no vértice 4. Os elementos localizados na esquerda representam os vértices de origem e os elementos localizados na parte superior representam os vértices de destino. Assim cada número na tabela representa a aresta desde o vértice origem até o vértice destino.

	1	**2**	**3**	**4**	**5**
1	0	0	0	**1**	0
2	0	0	0	**1**	0
3	0	0	0	**1**	0
4	**0**	**0**	**0**	**0**	**0**
5	0	0	0	**1**	0

Tabela 1: Matriz de Adjacências de um grafo com ***Ralo Universal.***

Para determinar se G tem um *ralo universal*, basta seguir o seguinte algoritmo.

Algoritmo 1: "Ralo Universal"

```
1  i <- 1
2  j <- 2
3  while j <= n do
4          while ( j <= n e A[i,j]=0 ) do
5                  j <- j+1
6          if ( j <= n ) then
7                  i <- j
```

```
8                   j <- i+1
9   testaCandidato (A,i,n)
```

As duas primeiras linhas inicializam os índices i e j que representam o vértice de origem e o vértice de chegada de uma aresta respectivamente. A variável n representa o número de vértices.

O *while* da linha 3 verifica que j não exceda o número de vértices. O seguinte *while*, vai percorrendo os diferentes destinos das arestas com origem no vértice i enquanto se mantenha o valor de "0" em cada campo (característica do *ralo universal*). Os elementos percorridos neste anel são descartados dado que não houve aresta do vértice i ao vértice j. Em caso de encontrar um "1", nas linhas 7 e 8 continua-se a procura a partir da submatriz de elementos que ainda não foram percorridos e o processo começa de novo. O candidato é o vértice representado pelo valor atual de i. A linha 9 testa o candidato encontrado pelo algoritmo. A complexidade do algoritmo é $O(V)$ dado que a matriz é percorrida somente em j até que o número de vértices seja alcançado.

Algoritmo 2: "Testa Candidato"

```
1   testaCandidato (A,k,n)
2   for j <- 1 to n                    // procura algum 1 numa f i l a
3           do if A[k,j] = 1
4                   then return FALSE
5   for i <- 1 to n                    // procura algum 0 numa coluna e x c e to na diagonal
6           do if A[i,k] = 0 and i <> k
7                   then return FALSE
8   return TRUE
```

A função *testaCandidato* também apresenta tempo $O(V)$ dado que verifica se uma coluna tem uns é depois se uma fila tem zeros percorrendo os vértices duas vezes.

Referências

[1] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms.* MIT Press and McGraw-Hill, 1990.