MO417 - Ata do exercício

Maikon Cismoski dos Santos - RA098365 - 19 de maio de 2010

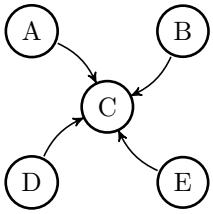
Exercício 22.1-6

Enunciado

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 **sorvedor universal** − um vértice com grau de entrada $|V| - 1$ e grau de saída 0 − é uma operação que pode ser realizada no tempo $O(V)$, dada uma matriz de adjacências para G.

Resolução

Um exemplo de grafo orientado que possui um sorvedor universal é apresentado na Figura 1. Do lado esquerdo da figura é ilustrado o grafo e do lado direito a matriz de adjacências. Nota-se que o vértice C é o sorvedor universal, ele possui grau de entrada $|V| - 1$ e grau de saída 0. Na matriz de adjacências, a linha referente ao vértice C armazena somente 0s, indicando que não existe nenhuma aresta que sai deste vértice para qualquer outro do grafo. Além disso, a coluna referente ao vértice C armazena somente 1s com exceção do próprio vértice C, informando que existem arestas saindo de todos os vértices e entrando em C, com exceção do próprio vértice C.



	A	B	C	D	E
A	0	0	1	0	0
B	0	0	1	0	0
C	0	0	0	0	0
D	0	0	1	0	0
E	0	0	1	0	0

Figura 1: Grafo orientado e sua matriz de adjacências.

O algoritmo Universal-Sink (Figura 2) foi proposto para determinar se um grafo orientado contém ou não um sorvedor universal. Os parâmetros de entrada são a matriz de adjacências e o número de vértices do grafo, representados por A e n, respectivamente. O algoritmo retorna *true* caso o grafo possua um sorvedor universal e *false* caso contrário.

No loop das linhas 3-6, um vértice candidado a ser sorvedor universal é encontrado. O índice do vértice candidato encontrado é armazenado na variável i. Após encontrar o candidato, o algoritmo faz uma verificação se o mesmo é válido nas linhas 7-14. Na linha 7, o comando *if* testa se o índice do vértice candidato é maior que o número de vértices. Caso afirmativo, o grafo não possui

um sorvedor universal, porque todos os vértices do grafo tem um grau de saída de pelo menos 1.

No loop das linhas 9-11, o grau de saída do vértice candidato é testado. Se o candidato tiver arestas saindo dele para qualquer outro vértice o algoritmo retorna *false*, uma vez que o vértice candidato deve possuir grau de saída 0. O grau de entrada do candidato é testado no loop das linhas 12-14, que verifica se existem arestas saindo de todos os vértices e entrando no candidato, com exceção do próprio vértice candidato.

Quando o vértice candidado passa por todos os testes de validação das linhas 7-14, conclui-se que o grafo possui um sorvedor universal e o algoritmo retorna *true* na linha 15.

```
UNIVERSAL-SINK(A, n)
 1  i ← 1
 2  j ← 1
 3  while i ≤ n and j ≤ n
 4        do if A[i][j] = 0
 5              then j ← j + 1
 6              else  i ← i  + 1
 7  if i > n
 8     then return false
 9  for j ← 1 to n
10        do if A[i][j] = 1
11              then return false
12  for k ← 1 to n
13        do if A[k][i] = 0 and i ≠ k
14              then return false
15  return true
```

Figura 2: Algoritmo para detectar se um grafo orientado G contém um sorvedor universal.

O tempo de execução do algoritmo Universal-Sink é $O(V)$. Dado que n é a quantidade de vértices, $n = V$, o *loop while* da linha 3 é executado no máximo $2n + 1$ vezes, uma vez que a cada iteração i ou j é incrementado em uma unidade. Os *loops* das linhas 9 e 12 executam $n + 1$ vezes cada um e as demais linhas do algoritmo tem custo constante. Então, conclui-se que o *loop while* da linha 3 possui custo dominante no algoritmo de $2n + 1$, que resulta no tempo de execução $O(V)$.