02 - Representações computacionais

Marco A L Barbosa malbarbo.pro.br

1. Exercício 22.1-2 de CLRS3 ou CLRS2.
2. Exercício 22.1-3 de CLRS3 ou CLRS2.
3. Exercício 22.1-4 de CLRS3 ou CLRS2.
4. Exercício 22.1-5 de CLRS3 ou CLRS2.
5. Exercício 22.1-6 de CLRS3 ou CLRS2.
6. Exercício 22.1-7 de CLRS3 ou CLRS2.
7. Exercício 22.1-8 de CLRS3 ou CLRS2.

Referências

- [CLRS2] - Thomas H. Cormen et al. Introduction to Algorithms. 2nd edition. Capítulo 22.1.
- [CLRS3] - Thomas H. Cormen et al. Introduction to Algorithms. 3rd edition. Capítulo 22.1.

Soluções

Exercício 3

Criamos uma lista de adjacência $G'.Adj$ de tamanho $|V|$. Para cada vértice u de G, examinamos $G.Adj[u]$ e adicionamos os vértices encontrados em $G'.Adj[u]$. Temos que achar uma maneira de evitar adicionar o vértice u (laço) e vértices repetidos (arestas múltiplas) em $G'.Adj[u]$. Vamos primeiro analisar uma opção menos eficiente.

Criamos um matriz $A = (a_{ij})$ de tamanho $|V| \times |V|$, e inicializamos cada valor a_{ij} com TRUE se $i \neq j$, e FALSE caso contrário ($A_{ij} ==$ TRUE significa que a aresta (i, j) é permitida em G'). Quando examinamos uma aresta (u, v), apenas adicionamos v em $G'.Adj[u]$ se $A_{uv} ==$ TRUE. Após a adição, fazemos $A_{uv} =$ FALSE, evitando desta forma que v seja adicionado novamente em $G'.Adj[u]$.

O tempo para analisar todos as arestas de G é $O(V + E)$. Inicializar a lista de adjacência de G' tem tempo $O(V)$. Adicionar um vértice na lista de adjacência de G' é constante (verificar e mudar um elemento de A também), portanto "preencher" as listas de adjacência de G' tem tempo $O(E)$. Poderíamos concluir então, de forma precipitada, que o tempo de execução para criar G' é $O(V + E)$. No entanto, não consideramos o tempo de inicialização de V, que é $O(V^2)$! Ou seja, o tempo para criar G' é $O(V^2)$, o que não atende o que foi pedido no exercício.

Podemos observar que cada linha i da matriz V é utilizada para evitar que laços e arestas múltiplas sejam adicionadas na lista de adjacência do vértice i em G'. Mas, uma vez que a lista de adjacência de i é explorada, a linha i da matriz não é mais utilizada. Desta forma, poderíamos ter apenas uma linha na matriz e reutilizar esta linha para todos os vértices. Precisamos apenas der o cuidado de deixar esta linha em um estado consistente antes de explorar a lista de adjacência de cada vértice.

Usamos então um atributo *permitido* nos vértices (equivalente a uma linha da matriz A), inicializado com TRUE. Antes de selecionar cada vértice u, assumimos que $v.permitido ==$ TRUE para todo $v \in G.V - \{u\}$, e $u.permitido ==$ FALSE, indicando que as arestas (u, v) são permitidas em G' e (u, u) não é. Quando examinamos uma aresta (u, v), apenas adicionamos v em $G'.Adj[u]$ se $v.permitido ==$ TRUE. Após a adição, fazemos $v.permitido =$ FALSE, evitando desta forma que v seja adicionado novamente em $G'.Adj[u]$. O código a seguir mostra este processo

```
MULTIGRAFO-PARA-GRAFO(G)
 1  Seja G'.Adj uma nova lista de adjacências de tamanho |V|
 2  for each vertex v ∈ G.V
 3       v.permitido = TRUE
 4  for each vertex u ∈ G.V
 5       u.permitido = FALSE
 6       for each vertex v ∈ G.Adj[u]
 7            if v.permitido == TRUE
 8                 Adiciona v em G'.adj[u]
 9                 v.permitido = FALSE
       // Precisamos restaurar o atributo permitido para a próxima iteração.
       // Ao invés de mudar o valor do atributo para todos os vértices (o que faria o tempo ser O(V^2)),
       // apenas mudamos para aqueles que o atributo foi alterado nas linha anteriores.
10     u.permitido = TRUE
11     for each vertex v ∈ G.Adj[u]
12          v.permitido = TRUE
13 return G'
```

A linha 1 e o laço das linhas 2 e 3 têm tempo $O(V)$ e o laço das linhas 4 a 12 tem tempo $O(V+E)$, portanto, o tempo de execução de MULTIGRAFO-PARA-GRAFO é $O(V + E)$, conforme o solicitado no exercício.