03 - Busca em largura

Marco A L Barbosa malbarbo.pro.br

Sequência de etapas para construção de algoritmos e implementação de funções.

- Nomear o algoritmo/função e definir as entradas e saídas;
- Descrever de forma sucinta e precisa o que o algoritmo/função faz;
- Criar diversos exemplos de entrada e saída. Aqui é importante não escrever a saída "de cabeça", mas pensar sobre e realizar um processo que encontre a saída a partir da entrada. Este processo pode começar simples para exemplos simples e ser estendido para funcionar para os demais exemplos;
- Baseado nas etapas anteriores, escrever o corpo do algoritmo/função;
- Argumentar sobre a corretude do algoritmo/função e executar testes (automatizados no caso de funções e manuais no caso de algoritmos em pseudocódigo);
- Corrigir eventuais erros e repetir o passo anterior;
- Simplificar o algoritmo/função.

Exercícios

1. Exercício 22.2-1 de CLRS3 ou CLRS2.
2. Exercício 22.2-2 de CLRS3 ou CLRS2.
3. Exercício 22.2-3 de CLRS3. (De acordo com a errata, é para considerar a remoção da linha 18, e não das linhas 5 e 14)
4. Exercício 22.2-4 CLRS3 ou 22.2-3 CLRS2.
5. Exercício 22.2-5 CLRS3 ou 22.2-4 CLRS2.
6. Exercício 22.2-6 CLRS3 ou 22.2-5 CLRS2.
7. Usando como base o BFS explique como verificar se um grafo não orientado é conexo.
8. Usando como base o BFS explique como identificar quantas componentes conexas um grafo não orientado tem.
9. Exercício 22.2-7 CLRS3 ou 22.2-6 CLRS2.

Veja a solução de alguns exercícios no final do arquivo.

Exercícios de implementação

Considere que o grafo de entrada é representado por uma lista de adjacências.

1. Projete uma função que receba como entrada um grafo g e dois vértices u e v de g e encontre um caminho entre u e v, se existir.
2. Projete uma função que receba como entrada um grafo g e um vértice inicial s de g e encontre os vértices acessíveis a partir de s que estão mais distantes dele.

3. Projete uma função que receba como entrada um grafo não orientado conexo acíclico (árvore) g e calcule o seu diâmetro. (Veja a definição de diâmetro no exercício 22.2-8 e pesquise um algoritmo para calcular o diâmetro de uma árvore).

Referências

- [CLRS2] - Thomas H. Cormen et al. Introduction to Algorithms. 2^{nd} edition. Capítulo 22.2.
- [CLRS3] - Thomas H. Cormen et al. Introduction to Algorithms. 3^{rd} edition. Capítulo 22.2.

Soluções

Exercício 9 (22.2-7 CLRS3 ou 22.2-6 CLRS2)

Representamos os lutadores e as rivalidades por um grafo não orientado $G = (V, E)$. Os lutadores são os vértices e as rivalidades as arestas. Desta forma $n = |V|$ e $r = |E|$. Temos que verificar se é possível atribuir a cada vértice o rótulo "bom" ou "mau" de forma que cada aresta conecte vértices com rótulos diferentes. De fato, os rótulos em si não são importantes, precisamos é classificar o vértices em duas classes de forma que não existam arestas entre vértices da mesma classe. Este é o problema de verificar se um grafo é bipartido.

Podemos usar os valores $v.d$ obtidos pela execução do BFS para particionar os vértices. O vértice com $d = 0$, fica na primeira partição (dos "bons"), os vértices com $d = 1$ precisam ficar na segunda partição (dos "maus") pois estão conectados com vértices na primeira partição. Os vértices com $d = 2$ devem ficar na primeira partição (pois estão conectados com vértices da segunda partição) e assim por diante. Esta classificação é feita analisando apenas as arestas da árvore da busca em profundidade, no final precisamos verificar se as demais arestas conectam vértices em partições diferentes. A execução do BFS é $O(n + r)$ e o custo de verificar é arestas é $O(n + r)$ (no caso de lista de adjacências), portando o tempo de execução do algoritmo é $O(n + r)$. Segue o pseudo código do algoritmo.

```
BONS-E-MAUS(G)
1  Execute BFS no grafo G partindo de um vértice qualquer
2  for u ∈ G.V
3      for v ∈ G.Adj[u]
4          if paridade de u.d == paridade de v.d
5              return FALSE
6  Designe cada vértice v com v.d par como "bom"
7  Designe cada vértice v com v.d ímpar como "mau"
8  return TRUE
```