Busca em Largura / Busca em Profundidade

Teoria dos Grafos - 2020

Prof. Roberto C. de Araujo

Um algoritmo de busca em um grafo é um algoritmo para percorrer os vértices de um grafo, a partir de um vértice inicial, até que seja atingida uma condição de parada. Dentre as condições de parada, destacam-se:

- percorrer todos os vértices do grafo,
- percorrer todos os vértices do grafo cuja distância ao vértice inicial seja menor que k,
- atingir algum vértice específico do grafo,
- etc.

Dentre tantos detalhes, os algoritmos de buscas em grafo devem fazer uma "**marcação**" nos vértices para evitar que o algoritmo caia em um loop infinito (ao repetir sempre os mesmos vértices durante a busca).

Para evitar isso, uma estratégia comum é usar um sistema de cores nos vértices:

- um vértice BRANCO é aquele que ainda não foi atingido pela busca;
- um vértice CINZA é aquele que já foi atingido, mas, a partir dele, ainda pode ser usado para expandir a busca; e
- um vértice é PRETO se ele já foi atingido e todas as possibilidades de expandir a busca a apartir dele já foram consideradas

1. Busca em Largura

Algoritmo simples que expande os vértices na mesma ordem em que eles são atingidos. É útil para obter o ***caminho mais curto*** a partir do vértice inicial da busca. Algoritmo publicado no livro do "Cormen", 3a. edição, página 595:

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

2. Busca em Profundidade

Algoritmo que, como estratégia, expande primeiro o vértice que foi atingido por último no processo de busca. Algoritmo publicado no livro do "Cormen", 3a. edição, página 604:

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```