Busca em largura

Esta página volta a tratar do problema de encontrar o território de um vértice em um digrafo. Ela introduz uma versão especializada do algoritmo TERRITÓRIO, conhecida como busca em largura (= *breadth-first search* = *BFS*). [Para justificar a palavra "busca", devemos imaginar que o algoritmo percorre o digrafo buscando vértices que pertencem ao território em questão.]

Nossa versão do algoritmo de busca em largura nada mais faz que pintar de preto todos os vértices do território de um vértice. Mas, diferentemente do algoritmo TERRITÓRIO, ela visita os vértices numa certa ordem específica, que chamaremos "ordem de busca em largura".

Se queremos apenas encontrar o território de um vértice, a ordem em que os vértices são visitados é irrelevante. Mas a ordem de visita torna-se essencial se desejamos determinar outras propriedades além da mera pertinência de vértices ao território. Por exemplo, a ordem de visita é importante se desejamos calcular a distância entre dois vértices, como veremos em outra página. Assim, o algoritmo de busca em largura a ser discutido abaixo é apenas um protótipo: ele será usado como modelo para algoritmos dedicados a vários outros problemas.

Esta página é inspirada na seção 2 do capítulo 22 do CLRS. Veja também o verbete *Breadth-first search* na Wikipedia.

Algoritmo de busca em largura

O algoritmo TERRITÓRIO tem um caráter "genérico": a ordem em que os vértices saem da lista manipulada pelo algoritmo é irrelevante. Já no algoritmo de busca em largura, a lista de vértices obedece a política FIFO: o vértice que sai da lista é sempre o que está lá há mais tempo. A lista se comporta, portanto, como uma fila FIFO.

O algoritmo pinta de preto todos os vértices do território de um vértice r. O código abaixo supõe que os vértices do digrafo são $1, 2, \ldots, n$ e que os arcos são representados pelo vetor $Adj[1..n]$ de listas de adjacência.

```
BUSCA-EM-LARGURA (n, Adj, r)
 1  para u ← 1 até n faça
 2      cor[u] ← branco
 3  cor[r] ← cinza
 4  F ← CRIA-FILA(r)
 5  enquanto F não está vazia faça
 6      u ← SAI-DA-FILA(F)
 7      para cada v em Adj[u] faça
 8          se cor[v] = branco
 9              então cor[v] ← cinza
10                    ENTRA-NA-FILA(v, F)
11      cor[u] ← preto
```

12 devolva $cor[1..n]$

O comando CRIA-FILA(r) cria uma fila com um só elemento igual a r. O comando SAI-DA-FILA(F) retira o primeiro elemento (ou seja, o elemento mais antigo) da fila F. O comando ENTRA-NA-FILA(v,F) insere v no fim da fila F.

Detalhes de implementação

A fila pode ser implementada, muito simplesmente, em um vetor $F[1..n]$. O primeiro elemento da fila será $F[i]$ e o último será $F[f-1]$.

A linha 4 do algoritmo (CRIA-FILA(r)) consiste simplesmente em

$i \leftarrow 1$
$f \leftarrow 2$
$F[1] \leftarrow r$

a linha 5 do algoritmo (enquanto F não estiver vazia faça) é traduzida por

enquanto $i < f$ faça

a linha 6 ($u \leftarrow$ SAI-DA-FILA( )) é implementada por

$u \leftarrow F[i]$
$i \leftarrow i+1$

a linha 10 (ENTRA-NA-FILA(v)) é implementada por

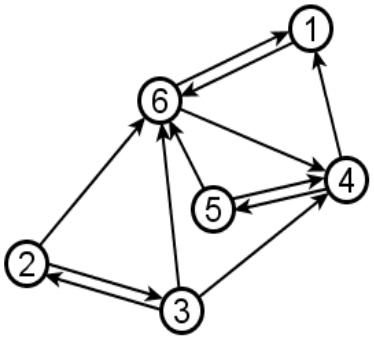
$F[f] \leftarrow v$
$f \leftarrow f+1$

Essa implementação da fila jamais sofre *overflow*, pois cada vértice do digrafo entra na fila no máximo uma vez.

Exemplo

Aplique o algoritmo BUSCA-EM-LARGURA ao digrafo definido pela figura com $r = 2$. Registre o estado do vetor cor e o estado da fila no início de cada iteração (ou seja, imediatamente antes de cada execução da linha 6).

u $Adj[u]$



1 (6)
2 (3, 6)
3 (2, 4, 6)
4 (1, 5)
5 (4, 6)
6 (1, 4)

As linhas da tabela abaixo dão o estado da fila $F[1..6]$ no início de sucessivas iterações: as casas pretas contêm os vértices que já saíram da fila; as casas cinza representam a fila propriamente dita.

2					
2	3	6			
2	3	6	4		
2	3	6	4	1	
2	3	6	4	1	5
2	3	6	4	1	5
2	3	6	4	1	5

Exercícios

1. [IMPORTANTE] Escreva uma variante do algoritmo BUSCA-EM-LARGURA que atribui um número de ordem (1, 2, …) a cada vértice no momento em que o vértice fica cinza. (Isso permite acompanhar a ordem em que o algoritmo visita os vértices.) Escreva outra variante que registre a ordem em que os vértices ficam pretos.

Consumo de tempo do algoritmo

Tal como TERRITÓRIO, o algoritmo BUSCA-EM-LARGURA é linear: o consumo de tempo do algoritmo é

$$O(n+m)\ .$$

Como de hábito, m é número de arcos do digrafo.

Veja "Busca em Largura" no sítio "Algoritmos para Grafos"

---

http://www.ime.usp.br/~pf/analise_de_algoritmos/

Last modified: Mon Apr 13 07:08:14 BRT 2015

Paulo Feofiloff
Departamento de Ciência da Computação
Instituto de Matemática e Estatística da USP