

## MAC 6711 - Tópicos de Análise de Algoritmos

Departamento de Ciência da Computação

Primeiro semestre de 2010

### Análise Amortizada do Union-Find

Considere a seguinte implementação da estrutura de dados para armazenar uma coleção de conjuntos disjuntos, conhecida como *union-find*:

MAKESET( $x$ )	UNION( $x, y$ )
1. pai[ $x$ ] $\leftarrow x$	1. $x' \leftarrow \text{FINDSET}(x)$
2. rank[ $x$ ] $\leftarrow 0$	2. $y' \leftarrow \text{FINDSET}(y)$
	3. LINK( $x', y'$ )
FINDSET( $x$ )	LINK( $x, y$ )
1. <b>se</b> $x \neq \text{pai}[x]$	1. <b>se</b> rank[ $x$ ] > rank[ $y$ ]
2. <b>então</b> pai[ $x$ ] $\leftarrow \text{FINDSET}(\text{pai}[x])$	2. <b>então</b> pai[ $y$ ] $\leftarrow x$
3. <b>devolva</b> pai[ $x$ ]	3. <b>senão</b> pai[ $x$ ] $\leftarrow y$
	4. <b>se</b> rank[ $x$ ] = rank[ $y$ ]
	5. <b>então</b> rank[ $y$ ] $\leftarrow \text{rank}[y] + 1$

Seja  $\lg^{(1)} x = \lg x$  e, para  $i \geq 2$ , seja  $\lg^{(i)} = \lg(\lg^{(i-1)} x)$ . A função  $\lg^* n$  é definida da seguinte maneira:

$$\lg^* n = \min\{i : \lg^{(i)} n \leq 1\}.$$

Qualquer seqüência de operações MAKESET, FINDSET e UNION pode ser convertida em uma seqüência de operações MAKESET, FINDSET e LINK. Considere portanto uma seqüência de  $m$  operações MAKESET, FINDSET e LINK das quais  $n$  são MAKESET. Vamos mostrar que o custo amortizado de cada operação nessa seqüência é  $O(\lg^* n)$ .

Para um nó  $x$  da floresta de árvores enraizadas, associamos um conjunto de nós  $G(x) = \{y : \lg^*(\text{rank}[y]) = \lg^*(\text{rank}[x])\}$ , que é chamado de *grupo* de  $x$ . Usaremos a seguinte função potencial nesta análise amortizada:

$$\Phi = |\{x : G(x) = G(\text{pai}[x])\}|.$$

Note que  $\Phi \geq 0$ . Denote por  $\Phi_0$  o valor inicial de  $\Phi$ , ou seja, o valor de  $\Phi$  antes da primeira operação ser executada. Claro que  $\Phi_0 = 0$ , pois não há nenhum conjunto na coleção inicialmente.

Abaixo, vamos denotar por  $\Phi_d$  a função potencial *depois* da operação em questão ser executada e por  $\Phi_a$  a função potencial *antes* da operação ser executada.

O custo amortizado de cada operação é:

1. MAKESET( $x$ ):

$$\hat{c} = c + \Phi_d - \Phi_a = 1 + 1 = 2.$$

2. LINK( $x, y$ ):

$$\hat{c} = c + \Phi_d - \Phi_a \leq 1.$$

De fato, para esta operação,  $c = 1$  e de fato  $\Phi_d - \Phi_a \leq 0$  pois o campo pai é alterado para um nó que é raiz de uma subárvore e já contribuía com 1 para o potencial. Ou este nó deixa de contribuir e  $\Phi_d - \Phi_a = -1$ , ou continua contribuindo e  $\Phi_d - \Phi_a = 0$ . Por outro lado, devido ao possível incremento sofrido por rank[ $y$ ], a função potencial pode diminuir ainda mais numa execução da função LINK, ou seja,  $\Phi_d - \Phi_a \leq 0$ .

### 3. FINDSET( $x$ ):

Seja  $z = \text{FINDSET}(x)$  e  $k$  o comprimento do caminho entre  $x$  e  $z$ . O custo real de  $\text{FINDSET}(x)$  é  $k$ . Então temos que

$$\hat{c} = c + \Phi_d - \Phi_a = k + \Phi_d - \Phi_a.$$

Para determinar o valor de  $\Phi_d - \Phi_a$ , considere a seguinte partição das arestas da floresta corrente. Para cada vértice  $u$  da floresta, dizemos que

- a aresta entre  $u$  e  $\text{pai}[u]$  é *vermelha* se  $G(\text{pai}[u]) = G(u) = G(r)$ , onde  $r$  é a raiz da árvore em que  $u$  se encontra.
- a aresta entre  $u$  e  $\text{pai}[u]$  é *azul* se  $G(\text{pai}[u]) \neq G(u)$ .
- a aresta entre  $u$  e  $\text{pai}[u]$  é *amarela* se  $G(\text{pai}[u]) = G(u) \neq G(r)$ , onde  $r$  é a raiz da árvore em que  $u$  se encontra.

Como, ao final da execução de  $\text{FINDSET}(x)$ , todos os nós do caminho entre  $x$  e  $z$  têm  $z$  como pai, temos que o valor de  $\Phi$  decresce exatamente do número de arestas amarelas no caminho de  $x$  a  $z$ . Mas então  $\hat{c} = v + a$ , onde  $v$  é o número de arestas vermelhas e  $a$  é o número de arestas azuis no caminho de  $x$  a  $z$ .

A função rank cresce estritamente a medida que subimos nas árvores. Uma aresta azul indica que passamos de um grupo para outro. Portanto, o número de arestas azuis no caminho de qualquer nó até uma raiz é no máximo o número de grupos, que é não mais que  $1 + \lg^* n$ . De fato, a função rank assume valores entre 0 e  $\lg n$ , o que implica que os grupos vão de 0 até  $\lg^*(\lg n) \leq \lg^* n$ . Ou seja,  $a \leq 1 + \lg^* n$  e o custo amortizado do  $\text{FINDSET}$  fica

$$\hat{c} = v + a \leq v + 1 + \lg^* n.$$

Vamos usar essas delimitações superiores nos custos amortizados para demonstrar que

$$\sum_{i=1}^m \hat{c}_i = O(m \lg^* n), \quad (1)$$

onde  $\hat{c}_i$  é o custo amortizado da  $i$ -ésima operação da seqüência. Assim, teremos que o custo amortizado por operação da seqüência é  $O(\lg^* n)$ , como queríamos.

Lembre-se que, denotando-se por  $c_i$  o custo real da  $i$ -ésima operação,

$$\sum_{i=1}^m \hat{c}_i = \sum_{i=1}^m c_i + \Phi_f - \Phi_0,$$

onde  $\Phi_f$  é o potencial final da floresta. Como  $\Phi_0 = 0$  e  $\Phi_f \geq 0$ , temos que  $\sum_{i=1}^m c_i \leq \sum_{i=1}^m \hat{c}_i$ .

Das deduções anteriores, temos que

$$\sum_{i=1}^m \hat{c}_i \leq 2(m - f) + f(1 + \lg^* n) + \sum_{i=1}^m v_i,$$

onde  $f$  é o número de operações  $\text{FINDSET}$  na seqüência,  $v_i = 0$  se a  $i$ -ésima operação não é um  $\text{FINDSET}$  e é o número de arestas vermelhas no caminho de  $x$  a  $z$  se a  $i$ -ésima operação é um  $\text{FINDSET}(x)$  que devolve  $z$  como resposta.

Para calcular  $\sum_{i=1}^m v_i$ , observe que, se um nó deixa de ser ponta inferior de uma aresta vermelha, nunca mais ele volta a ser ponta inferior de uma aresta vermelha.

Todos os nós visitados num  $\text{FINDSET}$ , exceto os dois últimos (a raiz e o filho dela no caminho), têm o seu campo pai alterado no  $\text{FINDSET}$ . Toda vez que o campo pai de um nó é alterado, o valor do

rank do seu pai aumenta. Isso porque a função rank cresce estritamente a medida que subimos nas árvores da floresta.

Quantas vezes um nó  $y$  que é ponta inferior de uma aresta vermelha pode sofrer alteração do seu campo pai num FINDSET? Se  $g = \lg^*(\text{rank}[y])$ , então isso pode ocorrer no máximo  $t(g+1) - t(g) \leq t(g+1)$  vezes, onde  $t(s) = \min\{k \in Z : \lg^* k \geq s\}$ . (A função  $t(s)$  é uma espécie de inversa da função  $\lg^* n$ .)

Assim sendo,

$$\sum_{i=1}^m v_i \leq 2f + \sum_{g=0}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1).$$

Lembre-se agora que o número de nós  $y$  tais que  $\text{rank}[y] = r$  é no máximo  $n/2^r$ . Além disso,  $t(g) = \min\{k \in Z : \lg^* k \geq g\} \leq \text{rank}[y]$ . Disso, tiramos que

$$\begin{aligned} |\{y : \lg^*(\text{rank}[y]) = g\}| &\leq \sum_{r=t(g)}^{\infty} |\{y : \text{rank}[y] = r\}| \\ &\leq \sum_{r=t(g)}^{\infty} \frac{n}{2^r} \\ &= \frac{n}{2^{t(g)}} \sum_{r=0}^{\infty} \frac{1}{2^r} \\ &= \frac{2n}{2^{t(g)}}. \end{aligned}$$

Portanto concluímos que

$$\sum_{i=1}^m v_i \leq 2f + \sum_{g=0}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1) \leq 2f + 2n \sum_{g=0}^{\lg^* n} \frac{t(g+1)}{2^{t(g)}}.$$

Mas, note que  $t(g+1) \leq 2^{t(g)}$ . Ou seja,

$$\sum_{i=1}^m v_i \leq 2f + 2n(1 + \lg^* n).$$

Então podemos concluir que  $\sum_{i=1}^m c_i \leq \sum_{i=1}^m \hat{c}_i \leq 2m + 2n + 2m \lg^* n \leq 4m + 2m \lg^* n = O(m \lg^* n)$ . Assim sendo, o custo amortizado por operação é  $O(\lg^* n)$ .