

PROJETO E ANÁLISE DE ALGORITMOS (INF 2926)

Lista Obrigatória 3

Entrega: 23 de junho de 2006

1. Considere o algoritmo de *Preflow-Push* para resolver o problema de Fluxo Máximo do vértice s ao vértice t em um Grafo $G = (V, E)$ com capacidades u_e para todos os arcos $e \in E$.
 - (a) Apresente uma análise que demonstre que a complexidade deste algoritmo é $O(n^3)$ quando a escolha do vértice ativo para se fazer a operação de *Push/Relabel* é feita pelo critério FIFO. Explique primeiro como este critério é implementado no algoritmo.
 - (b) Apresente uma análise que demonstre que a complexidade deste algoritmo é $O(n^3)$ quando a escolha do vértice ativo para se fazer a operação de *Push/Relabel* é feita pelo critério que escolhe sempre o **vértice de maior rótulo**. Explique primeiro como este critério é implementado no algoritmo.

Sugestão: Estas análises estão feitas nas seções 7.7 e 7.8 do livro *Network Flows*, Ahuja, Magnanti, Orlin. Espera-se uma apresentação objetiva e clara e que mostre a compreensão do que está lá.

2. Considere uma árvore binária de busca onde a sub-árvore com raiz em um nó x possui $size[x]$ elementos. Seja α uma constante tal que $1/2 \leq \alpha < 1$. Um nó x da árvore é dito balanceado se $size[left[x]] \leq \alpha \cdot size[x]$ e $size[right[x]] \leq \alpha \cdot size[x]$ onde $left[x]$ e $right[x]$ são os nós filhos à esquerda e à direita do nó x , respectivamente. A árvore de busca é **α -balanceada** se todos os seus nós são α -balanceados.
 - (a) Dado um nó x , arbitrário, mostre como reconstruir sua sub-árvore de modo que fique 1/2-balanceada. O algoritmo deve executar em $O(size[x])$ (que é $\Theta(size[x])$). Lembre que a sub-árvore de x já é uma sub-árvore de busca e $size[v]$ diz quantos elementos estão na sub-árvore de v .
 - (b) Mostre que a operação de busca em uma árvore **α -balanceada** com n elementos é feita em $O(\log n)$. (Fácil !!)
 - (c) ESTE ITEM NÃO É PARA SER RESPONDIDO ! A partir deste item assumo que $\alpha > 1/2$ (ou seja é estritamente maior que 1/2). Suponha que as operações de *Insert* e *Delete* são implementadas da forma habitual, sem rotações, exceto que quando algum nó não está mais α -balanceado, a operação do item a) (reconstrução para deixar 1/2-balanceada) é feita no nó desbalanceado de maior nível.

A idéia é analisar este esquema de reconstrução utilizando o método do potencial para análise *amortizada*. Para um nó x na árvore binária T define-se:

$$\Delta(x) = |size[left[x]] - size[right[x]]|$$

Define-se também o potencial na árvore T , $\Phi(T)$ como:

$$\Phi(T) = c \sum_{x \in T | \Delta(x) \geq 2} \Delta(x)$$

onde c é uma constante suficientemente grande que depende de α .

- (d) Argumente que qualquer árvore binária de busca tem potencial não-negativo e que uma árvore 1/2-balanceada tem potencial igual a ZERO.
- (e) Suponha que m unidades de potencial podem pagar pela reconstrução de uma sub-árvore com m nós. Que valor, em função de α , deve ter c para que a operação de reconstrução tome tempo *amortizado* constante ($O(1)$), quando é aplicada sobre uma sub-árvore que não está α -balanceada ?
- (f) Mostre que as operações de *Insert* e *Delete* tomam tempo *amortizado* $O(\log n)$.