

PUC-Rio

Departamento de Informática

Prof. Marcus Vinicius S. Poggi de Aragão

Período: 2004.2

Horário: 3as-feiras e 5as-feiras de 13 às 15 horas - Sala 422L

## PROJETO E ANÁLISE DE ALGORITMOS (INF 2926)

### Lista 4

1. Considere o problema de Fluxo Máximo do vértice  $s$  ao vértice  $t$  em um grafo orientado  $G = (V, E)$  com a restrição de que as capacidades de todos os arcos no grafo é 1.
  - (a) Proponha um algoritmo com complexidade  $O(nm)$  para encontrar o fluxo máximo destes grafos e discuta a sua implementação. (Descreva o algoritmo informalmente, apresente as estruturas que devem ser utilizadas e as operações necessárias, reescreva o algoritmo em função destas operações e analise cuidadosamente sua complexidade).
  - (b) Hoje, o algoritmo de complexidade mais baixa para este problema tem complexidade  $O(\min\{n^{2/3}m, m^{3/2}\})$  (ver Ahuja seção 8.2). Qual o gargalo do seu algoritmo ? Onde o seu algoritmo pode ser melhorado ?
  - (c) Considere agora que os arcos da rede possuem capacidades quaisquer. Explique como este algoritmo pode ser utilizado para se obter um algoritmo para o problema genérico de Fluxo Máximo de complexidade polinomial dada por  $O(n.m.\log U)$ , onde  $U$  é a capacidade do arco com maior capacidade.
  
2. Considere uma árvore binária de busca onde a sub-árvore com raiz em um nó  $x$  possui  $size[x]$  elementos. Seja  $\alpha$  uma constante tal que  $1/2 \leq \alpha < 1$ . Um nó  $x$  da árvore é dito balanceado se  $size[left[x]] \leq \alpha.size[x]$  e  $size[right[x]] \leq \alpha.size[x]$  onde  $left[x]$  e  $right[x]$  são os nós filhos à esquerda e à direita do nó  $x$ , respectivamente. A árvore de busca é  **$\alpha$ -balanceada** se todos os seus nós são  $\alpha$ -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  **$\alpha$ -balanceada** com  $n$  elementos é feita em  $O(\log n)$ . (Fácil !!)
  - (c) A partir deste item assuma 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  $\alpha$ -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 pontencial 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  $\alpha$ .

- (d) Argumente que qualquer árvore binária de busca tem potencial não-negativo e que uma árvore 1/2-balanceada tem potencial 0.
- (e) Suponha que  $m$  unidades de potencial podem pagar pela reconstrução de uma sub-árvore com  $m$  nós. Que valor, em função de  $\alpha$ , 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  $\alpha$ -balanceada ?
- (f) Mostre que as operações de *Insert* e *Delete* tomam tempo *amortizado*  $O(\log n)$ .