UNIVERSIDADE FEDERAL DE MINAS GERAIS

Departamento de Ciência da Computação

Projeto e Análise de Algoritmos

– *Trabalho Prático 1* –

Trabalho disponível em:
http://www.dcc.ufmg.br/∼renata/paa/tp1

Aluna: Renata Braga Araújo

Monitor: Fabiano Cupertino Botelho

Professor: Nivio Ziviani

Belo Horizonte
3 de abril de 2006

Sumário

1 Questão 1 - Avaliar as somas *(0,5 + 0,5 pontos)*

1.1 Item (a)

$$\sum_{i=1}^{n} i2^{i-1}$$

Solução:

Para resolver o somatório dado, primeiramente foi feita uma manipulação algébrica para retirar do mesmo os termos que não dependiam de i:

$$\sum_{i=1}^{n} i2^{i-1} = \sum_{i=1}^{n} i2^i 2^{-1} = \sum_{i=1}^{n} \frac{i2^i}{2} = \frac{1}{2}\sum_{i=1}^{n} i2^i$$

Dessa forma, é possível então aplicar a seguinte fórmula para resolvê-lo:

$$\sum_{i=1}^{n} ia^i = \frac{na^{n+2} - a^{n+1}(n+1) + a}{(1-a)^2}, \tag{1}$$

com $a \neq 1$.

Utilizando a fórmula 1, temos:

$$\begin{aligned}
&= \frac{1}{2}\left(\frac{n2^{n+2} - 2^{n+1}(n+1) + 2}{(1-2)^2}\right) \\
&= \frac{1}{2}(4n2^n - 2(n+1)2^n + 2) \\
&= \frac{4n2^n - 2n2^n - 2^n 2 + 2}{2} \\
&= \frac{2n2^n - 2^{n+1} + 2}{2} \\
&= n2^n - 2^n + 1
\end{aligned}$$

Colocando-se o termo 2^n em evidência, chegamos ao resultado final:

$$\boxed{\sum_{i=1}^{n} i2^{i-1} = 2^n(n-1) + 1}$$

Dedução da fórmula 1[1]:

Primeiramente, considere

$$S_n = \sum_{i=1}^{n} ia^i.$$

Expandindo essa série, temos:

$$S_n = a + 2a^2 + 3a^3 + ... + (n-1)a^{n-1} + na^n.$$

Multiplica-se, então, cada lado da equação por a:

$$aS_n = a^2 + 2a^3 + 3a^4 + ... + (n-1)a^n + na^{n+1}.$$

[1] Dedução baseada em [15], pág 504

Com a subtração de S_n por aS_n, obtém-se:

$$S_n - aS_n = \sum_{i=1}^{n} a^i - na^{n+1}.$$

Colocando S_n em evidência e utilizando o resultado do somatório a seguir:

$$\sum_{i=1}^{n} a^i = a\frac{a^n - 1}{a - 1} \tag{2}$$

temos:

$$S_n(1 - a) = a\frac{a^n - 1}{a - 1} - na^{n+1}.$$

Isolando, então, o termo S_n e aplicando algumas operações algébricas chegamos em:

$$S_n = \frac{na^{n+2} - a^{n+1}(n + 1) + a}{(1 - a)^2},$$

para $a \neq 1$ e

$$S_n = \sum_{i=1}^{n} i = \frac{(1 + n)n}{2},$$

para $a = 1$.

Dedução da fórmula 2[2]:

É utilizada uma metodologia análoga à que foi realizada para a prova mostrada anteriormente da fórmula 1. Assim, considere

$$S_n = \sum_{i=1}^{n} a^i.$$

Expandindo essa série, temos:

$$S_n = a + a^2 + a^3 + ... + a^{n-1} + a^n.$$

Multiplica-se cada lado da equação por a:

$$aS_n = a^2 + a^3 + a^4 + ... + a^n + a^{n+1}.$$

Realiza-se, então a subtração de aS_n por S_n:

$$aS_n - S_n = a^{n+1} - a.$$

Coloca-se então S_n em evidência, e isolando esse termo, temos:

$$S_n(a - 1) = a(a^n - 1)$$

$$S_n = a\frac{a^n - 1}{a - 1},$$

para $a \neq 1$.

[2]Dedução baseada em [15], págs 503 e 504

1.2 Item (b)

$$\sum_{i=1}^{n} \frac{1}{i}$$

Solução:

O somatório em questão pode ser expresso por $\sum_{i=m}^{n} f(i)$, onde $f(i)$ é uma função monotonicamente decrescente. É possível, então, segundo [1], fazer a aproximação do somatório por integrais:

$$\int_{m}^{n+1} f(x)dx \leq \sum_{i=m}^{n} f(i) \leq \int_{m-1}^{n} f(x)dx \tag{3}$$

A aproximação pela integral 3 fornece uma estimativa restrita para o n-ésimo número harmônico. Uma justificativa para essa aproximação é mostrada na Figura 1. O somatório é representado como as áreas dos retângulos na figura, e a integral é a área sob a curva.

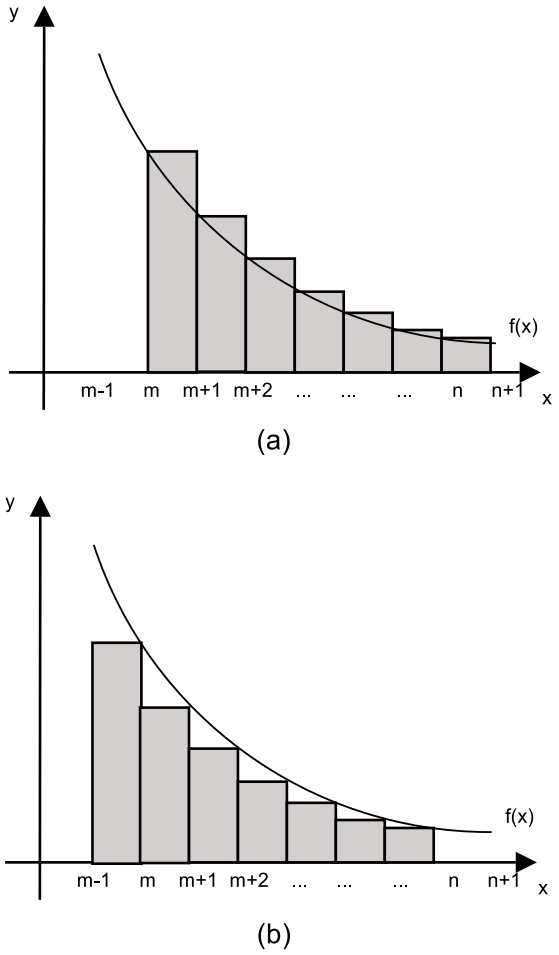

Figura 1: Aproximação de $\sum_{i=1}^{n} \frac{1}{i}$ por integrais. O valor do somatório é representado pela soma das áreas dos retângulos. A integral é representada pela área sob a curva. Comparando as áreas em (a), temos: $\int_{m}^{n+1} f(x)dx \leq \sum_{i=m}^{n} f(i)$, e movendo os retêngulos uma unidade para a esquerda, temos: $\int_{m-1}^{n} f(x)dx \geq \sum_{i=m}^{n} f(i)$.

Dessa forma, no caso de um limite inferior, obtemos:

$$\sum_{i=1}^{n} \frac{1}{i} \geq \int_{1}^{n+1} \frac{dx}{x} = \ln(n+1)$$

Derivando, agora, a desigualdade para encontrar o limite superior, tem-se:

$$\sum_{i=2}^{n} \frac{1}{i} \leq \int_{i}^{n} \frac{dx}{x} = \ln n,$$

que produz o limite:

$$\sum_{i=1}^{n} \frac{1}{i} \leq \ln n + 1$$

Logo, obtemos os seguintes limites para o somatório:

$$\boxed{\ln(n+1) \leq \textstyle\sum_{i=1}^{n} \frac{1}{i} \leq \ln n + 1}$$

2 Questão 2 (*2 pontos*)

Sejam a, b e c constantes não negativas. Prove que a solução para

$$\begin{cases} T(1) = c & n = 1 \\ T(n) = aT(n/b) + cn & n > 1 \end{cases}$$

para n uma potência de b é

$$T(n) = \begin{cases} O(n), & \text{se } a < b, \\ O(n \log n), & \text{se } a = b, \\ O(n^{\log_b a}), & \text{se } a > b. \end{cases}$$

Solução[3]:

Para provar que essa é a solução da equação de recorrência, vamos primeiramente expandir essa relação da seguinte forma:

$$T(n) = aT\left(\frac{n}{b}\right) + cn$$

$$aT\left(\frac{n}{b}\right) = a^2 T\left(\frac{n}{b^2}\right) + ac\left(\frac{n}{b}\right)$$

$$a^2 T\left(\frac{n}{b^2}\right) = a^3 T\left(\frac{n}{b^3}\right) + a^2 c\left(\frac{n}{b^2}\right)$$

$$a^3 T\left(\frac{n}{b^3}\right) = a^4 T\left(\frac{n}{b^4}\right) + a^3 c\left(\frac{n}{b^3}\right)$$

Considerando $n = b^m$ (ou seja, n é uma potência de b), expande-se a fórmula até a última iteração apresentada a seguir:

$$a^{m-1} T\left(\frac{n}{b^{m-1}}\right) = a^m T\left(\frac{n}{b^m}\right) + a^{m-1} c\left(\frac{n}{b^{m-1}}\right)$$

[3]Solução baseada em [13, 1]

Adicionando os termos lado a lado, obtemos:

$$T(n) = a^m T\left(\frac{n}{b^m}\right) + c\sum_{i=0}^{m-1} a^i \frac{n}{b^i}$$

Substituindo-se na equação n por b^m, temos:

$$T(n) = a^m T(1) + c\sum_{i=0}^{m-1} a^i b^{m-i}$$

$$T(n) = a^m c + c\sum_{i=0}^{m-1} a^i b^{m-i}$$

É possível perceber, então, que o termo $a^m c$ representa extamente o último termo do somatório, caso seu índice superior fosse m, em vez de $m-1$. Fazendo-se a manipulação desse índice para incluir o termo independente no somatório, chega-se em:

$$T(n) = c\sum_{i=0}^{m} a^i b^{m-i}$$

Fazendo-se manipulações algébricas no somatório para chegar a uma fórmula mais simplificada, temos:

$$T(n) = c\sum_{i=0}^{m} a^{m-i} b^i$$

$$T(n) = c\sum_{i=0}^{m} a^m a^{-i} b^i$$

$$T(n) = ca^m \sum_{i=0}^{m} \left(\frac{b}{a}\right)^i$$

Essa é então uma simples série geométrica com base igual a $\frac{b}{a}$, que pode ser resolvida considerando os três casos em que $\frac{b}{a} < 1$, $\frac{b}{a} > 1$ ou $\frac{b}{a} = 1$.

- $a < b$

  Nesse caso, a razão $\frac{b}{a}$ que representa a base da série geométrica é maior que 1. Aplicando-se a fórmula para a série geométrica $\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$ (baseado em [1]) , temos:

  $$T(n) = a^m \frac{\left(\frac{b}{a}\right)^{m+1} - 1}{\frac{b}{a} - 1}$$

  Como estamos interessados no comportamento assintótico da função, temos:

  $$T(n) = O\left(a^m \left(\frac{b}{a}\right)^m\right) = O(b^m)$$

  Logo,

$$\boxed{T(n) = O(n)}$$

- $a = b$

  Nesse caso, a razão $\frac{b}{a}$ que representa a base da série geométrica é igual a 1, então temos:

$$T(n) = ca^m + \sum_{i=0}^{m} 1$$
$$T(n) = ca^m + m$$

  Como estamos interessados no comportamento assintótico da função, temos:

$$T(n) = O(a^m m)$$

  Como $a = b$, temos que $a^m = b^m = n$ e $m = \log_b n$. Dessa forma, temos o comportamento assintótico da função dado por:

$$\boxed{T(n) = O(n \log n)}$$

- $a > b$

  Nesse caso, a razão $\frac{b}{a}$ que representa a base da série geométrica é menor que 1, então a série converge para uma constante, mesmo que m tenda ao infinito. Dessa forma, temos que

$$T(n) = O(a^m)$$

  Como $m = \log_b n$, tem-se que $a^m = a^{\log_b n} = n^{\log_b a}$ (segundo as propriedades do logarítmo). Dessa forma, temos:

$$\boxed{T(n) = O(n^{\log_b a})}$$

3 Questão 3 - Resolva as equações de recorrência *(2 pontos)*

3.1 Item (a)

$$\begin{cases} T(n) = 2T(n/4) + n, \text{ para } n > 1 \\ T(1) = 27 \end{cases}$$

Solução:
Primeiramente, expande-se a equação da seguinte forma:

$$T(n) = 2T\left(\frac{n}{2^2}\right) + n$$

$$2T\left(\frac{n}{2^2}\right) = 2^2T\left(\frac{n}{2^4}\right) + \frac{n}{2}$$
$$2^2T\left(\frac{n}{2^4}\right) = 2^3T\left(\frac{n}{2^6}\right) + \frac{n}{2^2}$$
$$2^3T\left(\frac{n}{2^6}\right) = 2^4T\left(\frac{n}{2^8}\right) + \frac{n}{2^3}$$

Considerando $n = 2^{2k}$, expande-se a equação até a última iteração, mostrada a seguir:

$$2^{k-1}T\left(\frac{n}{2^{2(k-1)}}\right) = 2^kT\left(\frac{n}{2^{2k}}\right) + \frac{n}{2^{k-1}}$$

Adicionando-se lado a lado, temos:

$$T(n) = 2^kT\left(\frac{n}{2^{2k}}\right) + n\sum_{i=0}^{k-1}\left(\frac{1}{2}\right)^i$$

Utilizando, agora, a fórmula $\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}$ (baseado em [1]) para resolver o somatório encontrado, temos:

$$T(n) = 2^k27 + n\left(\frac{(\frac{1}{2})^k - 1}{\frac{1}{2} - 1}\right)$$

Como $n = 2^{2k} \Rightarrow \log n = \log 2^{2^k} \Rightarrow \log n = 2k \Rightarrow k = \frac{\log n}{2}$, pode-se substituir esses termos na equação encontrada, obtendo:

$$T(n) = 2^{\frac{\log n}{2}}27 + n\left(\frac{(\frac{1}{2})^{\frac{\log n}{2}} - 1}{-\frac{1}{2}}\right)$$
$$T(n) = \sqrt{2^{\log n}}27 + n\left(\frac{(\frac{1}{\sqrt{2^{\log n}}} - 1}{-\frac{1}{2}}\right)$$
$$T(n) = 27\sqrt{n} - 2n\left(\frac{1}{\sqrt{n}} - 1\right)$$
$$T(n) = 27\sqrt{n} - \frac{2n}{\sqrt{n}} + 2n$$
$$\boxed{T(n) = 25\sqrt{n} + 2n}$$

3.2 Item (b)

$$\begin{cases} T(n) = 7T(n/2) + n^2, \text{ para } n > 1 \\ T(1) = 0 \end{cases}$$

Solução:
Primeiramente, expande-se a equação da seguinte forma:

$$T(n) = 7T\left(\frac{n}{2}\right) + n^2$$

$$7T\left(\frac{n}{2}\right) = 7^2T\left(\frac{n}{2^2}\right) + 7\left(\frac{n}{2}\right)^2$$
$$7^2T\left(\frac{n}{2^2}\right) = 7^3T\left(\frac{n}{2^3}\right) + 7^2\left(\frac{n}{2^2}\right)^2$$
$$7^3T\left(\frac{n}{2^3}\right) = 7^4T\left(\frac{n}{2^4}\right) + 7^3\left(\frac{n}{2^3}\right)^2$$

Considerando $n = 2^k$, expande-se a equação até a última iteração, mostrada a seguir:

$$7^{k-1}T\left(\frac{n}{2^{k-1}}\right) = 7^kT\left(\frac{n}{2^k}\right) + 7^{k-1}\left(\frac{n^2}{2^{2(k-1)}}\right)$$

Adicionando-se lado a lado, temos:

$$T(n) = 7^k0 + n^2\sum_{i=0}^{k-1}\frac{7^i}{2^{2i}}$$
$$T(n) = n^2\sum_{i=0}^{k-1}\left(\frac{7}{4}\right)^i$$

Utilizando, agora, a fórmula $\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}$ (baseado em [1]) para resolver o somatório encontrado, temos:

$$T(n) = n^2\left(\frac{\left(\frac{7}{4}\right)^k - 1}{\frac{7}{4} - 1}\right)$$
$$T(n) = n^2\left(\frac{\frac{7^k}{4^k} - 1}{\frac{3}{4}}\right)$$

Como $n = 2^k \Rightarrow \log n = k$, pode-se substituir esses termos na equação encontrada, obtendo:

$$T(n) = \frac{4}{3}n^2\left(\frac{7^{\log n}}{4^{\log n}} - 1\right)$$
$$T(n) = \frac{4}{3}n^2\left(\frac{7^{\log n}}{2^{2\log n}} - 1\right)$$
$$T(n) = \frac{4}{3}n^2\left(\frac{7^{\log n}}{2^{\log n^2}} - 1\right)$$
$$T(n) = \frac{4}{3}n^2\left(\frac{7^{\log n}}{n^2} - 1\right)$$
$$T(n) = \frac{4}{3}7^{\log n} - \frac{4}{3}n^2$$

Fazendo-se a mudança de base, utilizando as propriedades do logarítmo:

$$\log_2 n = \log_7 n \log_2 7 \Rightarrow \log_2 = \log_7 n^{\log_2 7}$$

Logo,

$$T(n) = \frac{4}{3}7^{\log_7 n^{\log_2 7}} - \frac{4}{3}n^2$$

$$\boxed{T(n) = \tfrac{4}{3}n^{\log 7} - \tfrac{4}{3}n^2}$$

3.3 Item (c)

$$\begin{cases} T(n) = T(\sqrt{n}) + \log n, \text{ para } n \geq 1 \\ T(1) = 1 \end{cases}$$

Dica: use mudança de variáveis.

Solução:

Para resolver essa equação de recorrência, utilizaremos a seguinte mudança de variáveis:

$$m = \log m \Rightarrow n = 2^m$$

Dessa forma, obtemos:

$$T(2^m) = T(2^{\frac{m}{2}}) + m$$

Expandindo-se essa equação de recorrência, temos:

$$T(2^{\frac{m}{2}}) = T(2^{\frac{m}{2^2}}) + \frac{m}{2}$$
$$T(2^{\frac{m}{2^2}}) = T(2^{\frac{m}{2^3}}) + \frac{m}{2^2}$$
$$T(2^{\frac{m}{2^3}}) = T(2^{\frac{m}{2^4}}) + \frac{m}{2^3}$$
$$\vdots$$
$$T(2^{\frac{m}{2^{k-1}}}) = T(2^{\frac{m}{2^k}}) + \frac{m}{2^{k-1}}$$

Para que $T(2^{\frac{m}{2^k}}) = T(1)$, temos que $2^{\frac{m}{2^k}} = 1$. Assim, consideramos $k \to \infty$, temos que o expoente da potência tende a zero, fazendo com que o resultado tenda a 1, como desjado. Assim, temos:

$$T(2^m) = T(1) + m\sum_{i=0}^{\infty}\left(\frac{1}{2}\right)^i$$

Utilizando, agora, a seguinte fórmula $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$, para $|x| < 1$, (baseado em [1]) para resolver o somatório encontrado, temos:

$$T(2^m) = 1 + m\left(\frac{1}{1-\frac{1}{2}}\right)$$
$$T(2^m) = 1 + 2m$$

$$\boxed{T(n) = 1 + 2\log n}$$

3.4 Item (d)

$$\begin{cases} T(n) = T(n/9) + T(8n/9) + n, & \text{para } n \geq 1 \end{cases}$$

Dica: Use uma árvore de recursividade para enxergar a solução.

Solução:

A equação de recorrência em questão é característica de um problema que é resolvido através da partição do mesmo em outros subproblemas. Dessa forma, a árvore de recursão apresenta-se como uma boa técnica para solucionar essa equação. Assim, em uma árvore de recursão, cada nó representa o custo de um único subproblema em algum lugar no conjunto de invocações de funções recursivas. Somando-se os custos dentro de cada nível da árvore, obtém-se um conjunto de custos por nível, e então somando todos os custos por nível determinamos o custo total de todos os níveis da recursão.

A árvore de reursão para a equação $T(n) = T(n/9) + T(8n/9) + n$ é mostrada na Figura 2:

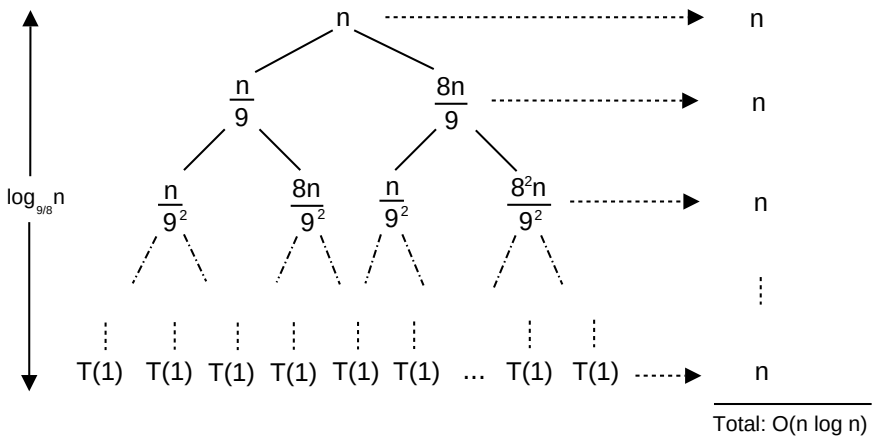

Figura 2: Árvore de recursão para a recorrência $T(n) = T(n/9) + T(8n/9) + n$

Percebe-se que, quando somamos os valores em todos os níveis da árvore de recursão, obtemos um valor n para cada nível. Como a árvore é desbalanceada, já que os subproblemas não possuem tamanhos iguais, temos que encontrar o caminho mais longo da raiz até uma folha da árvore para determinar sua altura. Percebe-se que o caminho mais longo é dado quando percorre-se $n \to \left(\frac{8}{9}\right) n \to \left(\frac{8}{9}\right)^2 n \to ... \to 1$. Como $\left(\frac{8}{9}\right)^k n = 1$, quando $k = \log_{\frac{9}{8}} n$, temos que a altura da árvore é $\log_{\frac{9}{8}} n$. Intuitivamente, esperamos que a solução para a recorrência seja no máximo o número de níveis da árvore vezes o custo de cada nível, ou seja:

$$T(n) = O(n \log_{\frac{9}{8}} n)$$

$$\boxed{T(n) = O(n \log n)}$$

Entretanto, é ainda necessário considerar o custo das folhas. Se a árvore fosse uma árvore binária completa de altura $\log_{\frac{9}{8}} n$, haveria $2^{\log_{\frac{9}{8}} n} = n^{\log_{\frac{9}{8}} 2}$. Como o

custo de cada folha é constante, o custo total de todas as folhas seria $\theta(n^{\log_{\frac{9}{8}} 2})$, que é $\omega(n \log n)$. Por outro lado, esse cálculo é feito considerando que essa fosse uma árvore binária completa, o que não é verdade já que a medida que descemos na árvore a partir da raiz, mais e mais nós internos estão ausentes. Assim, nem todos os níveis contribuem com um custo exatamente igual a n.

Assim, podemos usar o método de substituição para verificar que $O(n \log n)$ é um limite superior para a solução da recorrência. Mostramos que $T(n) \leq dn \log n$, onde d é uma constante positiva apropriada. Assim, temos:

$$T(n) \leq T\left(\frac{n}{9}\right) + T\left(\frac{8n}{9}\right) + n$$

$$T(n) \leq d\left(\frac{n}{9}\right)\log\left(\frac{n}{9}\right) + d\left(\frac{8n}{9}\right)\log\left(\frac{8n}{9}\right) + n$$

$$T(n) \leq \left(d\left(\frac{n}{9}\right)\log n - d\left(\frac{n}{9}\right)\log 9\right) + \left(d\left(\frac{8n}{9}\right)\log n - d\left(\frac{8n}{9}\right)\log\frac{9}{8}\right) + n$$

$$T(n) \leq dn\log n - d\left(\left(\frac{n}{9}\right)\log 9 + \left(\frac{8n}{9}\right)\log\frac{9}{8}\right) + n$$

$$T(n) \leq dn\log n - d\left(\left(\frac{n}{9}\right)\log 9 + \left(\frac{8n}{9}\right)\log 9 - \frac{8n}{9}\log 9\right) + n$$

$$T(n) \leq dn\log n - dn\left(\log 9 - \frac{9}{8}\right) + n$$

$$\boxed{T(n) \leq dn\log n}$$

desde que $d \geq 1/(\log 9 - \frac{9}{8})$.

4 Limite inferior *(2 pontos)*

Apresente o limite inferior para intercalar dois conjuntos de inteiros $A(1 : m)$ e $B(1 : n)$ onde os itens em A e os itens em B estão ordenados. Assuma que todos os números $m + n$ são distintos.

Solução[4]:

Para estabelecer o limite inferior desse problema podemos utilizar um oráculo. O oráculo é uma técnica que, dado um modelo de computação que expresse o comportamento do algoritmo, informa o resultado de cada passo possível. Assim, visando determinar um bom limite inferior, ele faz com que o algoritmo trabalhe o máximo, sempre escolhendo o resultado para o próximo teste que cause o maior trabalho possível para determinar a resposta final. Assim, contabilizando o trabalho que é realizado podemos derivar o limite inferior do problema.

Modelando esse problema em uma árvore de comparação, o oráculo determinará como percorrer essa árvore para alcançar o objetivo desejado. Assim, cada nodo dessa árvore de comparação representará um estado do problema, sendo que o conjunto de suas folhas representam todas as maneiras possíveis de se encontrar a solução do mesmo. Ao caminhar nessa árvore, é decidido a cada passo o elemento de qual conjunto (A ou B) será retirado, para compor a solução final nesse momento. A Figura 3 ilustra a árvore de comparação descrita.

[4] Baseado em [7, 9, 13]

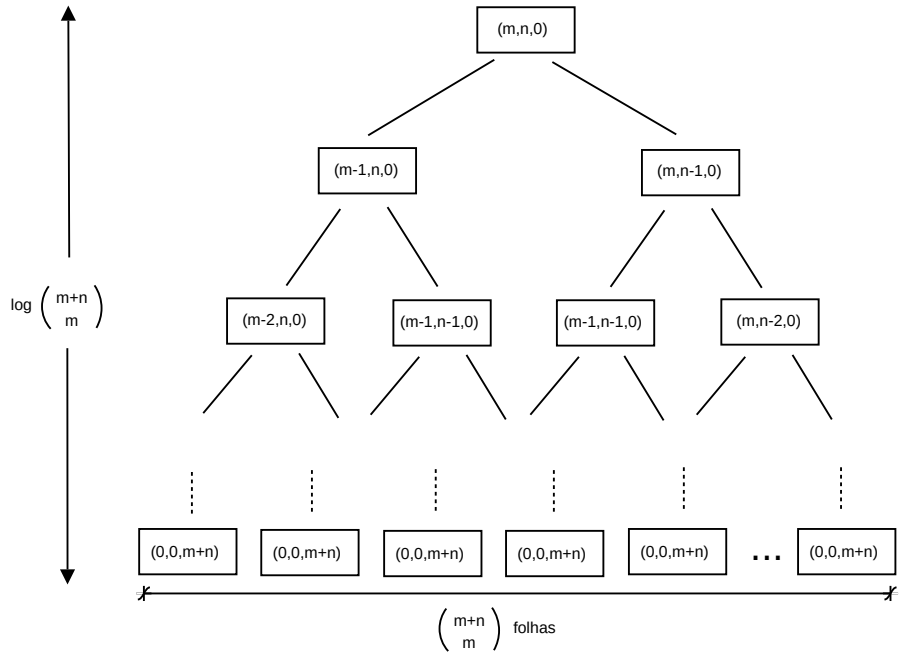

Figura 3: Árvore de comparação para o problema de intercalar dois conjuntos de inteiros $A(1:m)$ e $B(1:n)$ onde os itens em A e os itens em B estão ordenados.

Utilizando-se de conceitos combinatórios, temos que existem $\left(\begin{array}{c} m+n \\ m \end{array}\right)$ maneiras de se intercalar os elementos de dois conjuntos A e B de forma a preservar a ordem dos elementos. No nosso modelo de computação, temos então que a árvore de comparação terá $\left(\begin{array}{c} m+n \\ m \end{array}\right)$ folhas. Dessa forma, podemos concluir que a árvore de comparação terá uma altura de, no mínimo:

$$\lceil \log \left(\begin{array}{c} m+n \\ m \end{array}\right) \rceil$$

Ou seja, serão necessárias no mínimo $\lceil \log \left(\begin{array}{c} m+n \\ m \end{array}\right) \rceil$ comparações para se chegar a uma solução.

Além disso, sabemos que o pior caso de um procedimento de *merging* necesita de $m+n-1$ comparações. Esse pior caso, é facilmente visualizado e se dá quando os dois maiores elementos entre os dois conjuntos se encontram em conjuntos diferentes.

Se considerarmos que $MERGE(m,n)$ é o número mínimo de comparações para intercalar m elementos com n elementos, temos a inegualdade:

$$\boxed{\lceil \log \left(\begin{array}{c} m+n \\ m \end{array}\right) \rceil \leq MERGE(m,n) \leq m+n-1}$$

Sabemos, então, que os limites superior e inferior podem ficar arbitrariamente distantes quando m se torma muito menor que n. Isso não é surpresa já que o algoritmo funcioma melhor quando m e n são aproximadamente iguais. No caso

extremo em que $m = 1$, nós observamos uma inserção binária que iria requerer o número mínimo de comparações para intercalar $A(1)$ em $B(1), ..., B(n)$.

Quando m e n são iguais, então o limite inferior dado pelo modelo da árvore de comparação é na verdade muito pequeno. Assim, apresentamos um teorema a seguir que melhora esse o inferior para esse caso específico.

Teorema: Para todo $m \geq 1$, $MERGE(m, m) \geq 2m - 1$

Demostração: Considere um algoritmo genérico que intercala as duas seqüências $A[1..m]$ e $B[1..m]$ e suponha que essas seqüências são tais que o resultado do algoritmo é

$$B[1] < A[1] < B[2] < A[2] < ..... < B[m-1] < A[m-1] < B[m] < A[m]$$

Portanto, qualquer $i < m$, o valo de $A[i]$ deve necessariamente ser comparado com $B[i]$ e com $B[i+1]$, pois caso contrário, se $A[i]$ não for comparado com $B[i]$ então não há como o algoritmo distinguir entre as duas saídas abaixo:

$$B[1] < A[1] < ... < B[i] < A[i] < B[i+1]..... < B[m] < A[m]$$
$$B[1] < A[1] < ... < A[i] < B[i] < B[i+1]..... < B[m] < A[m]$$

O mesmo raciocínio vale para a comparação entre $A[i]$ e $B[i+1]$. Então, no pior caso, todos os elementos de A, exceto $A[m]$ têm que ser comparados pelo menos duas vezes. No caso de $A[m]$, este elemento é comparado pelo menos uma vez (com $B[m]$). Logo,

$$\boxed{MERGE(m, m) \geq 2(m-1) + 1 = 2m - 1}$$

Visto que o algoritmo convencional de intercalação realiza $m+n-1$ comparações, então podemos concluir que, para $n = m$, isto é, se as duas seqüências têm o mesmo tamanho, então esse algoritmo é ótimo. Por sua vez o limite inferior é justo.

5 Árvore B *(0,5 + 2 + 0,5 pontos)*

O objetivo deste trabalho é realizar um estudo do algoritmo árvore B (Ziviani, 2004).

5.1 Item (a)

Obtenha analiticamente o valor da altura h de uma árvore B de ordem m para o melhor caso e o pior caso.

Solução:

Uma árvore B de ordem m, por definição, é uma árvore que:

- cada página contém no mínimo m registros (e $m+1$ descendentes) e no máximo $2m$ registros (e $2m + 1$ descendentes), exceto a página raiz que pode conter entre 1 e $2m$ registros;

- todas as páginas folha aparecem no mesmo nível.

Tendo em vista essas propriedades, podemos fazer agora um estudo analítico do valor de sua altura h para o melhor caso e o pior caso.

⇒ **Estudo do melhor caso**

No melhor caso, os registros vão ocupar a capacidade total de cada página da árvore B, ou seja, $2m$ registros por página, sem nenhum desperdício de espaço, fazendo com que a altura da árvore seja a menor possível. Dessa forma, a Figura 4 representa a distribuição desses registros para o melhor caso, assim como a relação do número de páginas por nível da árvore.

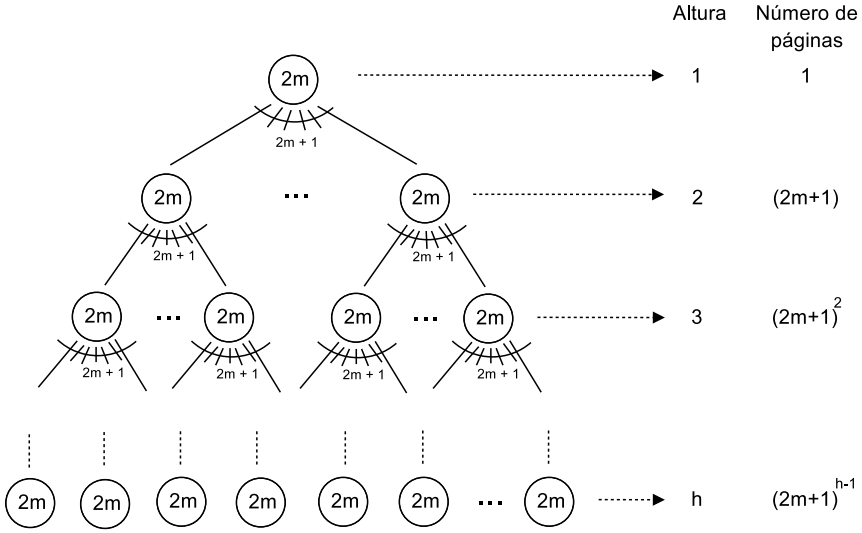

Figura 4: Estudo do melhor caso de uma Árvore B de ordem m

Através da Figura 4, podemos perceber que o número total de páginas (p) da árvore é:

$$p = \sum_{i=0}^{h-1} (2m+1)^i$$

Utilizando a seguinte fórmula para progressão geométrica:

$$\sum_{i=0}^{n} \frac{x^{i+1} - 1}{a - 1}$$

Podemos resolver o somatório da seguinte forma:

$$p = \frac{(2m+1)^{(h-1)+1} - 1}{(2m-1)+1}$$

$$p = \frac{(2m+1)^h - 1}{2m}$$

Sabemos também que o número de registros total da árvore (N) é dado pelo número total de páginas (p) vezes o número de máximo de registros por página ($2m$), ou seja:

$$N = 2mp$$

$$N = 2m\frac{(2m+1)^h - 1}{2m}$$

$$N = (2m+1)^h - 1$$

Isolando-se agora o h para obtermos o limite mínimo da altura da árvore, temos:

$$N + 1 = (2m+1)^h$$

$$log_{2m+1}(N+1) = log_{2m+1}(2m+1)^h$$

$$log_{2m+1}(N+1) = h$$

Assim, o limite inferior da altura h, obitdo analisando o melhor caso é:

$$\boxed{h \geq \log_{2m+1}(N+1)}$$

⇒ **Estudo do pior caso**

No pior caso, os registros vão ocupar a capacidade mínima de cada página da árvore B, ou seja, 1 registro na página raiz e m registros em todas as outras páginas, fazendo com que a altura da árvore seja a maior possível. Dessa forma, a Figura 5 representa a distribuição desses registros para o pior caso, assim como a relação do número de páginas por nível da árvore.

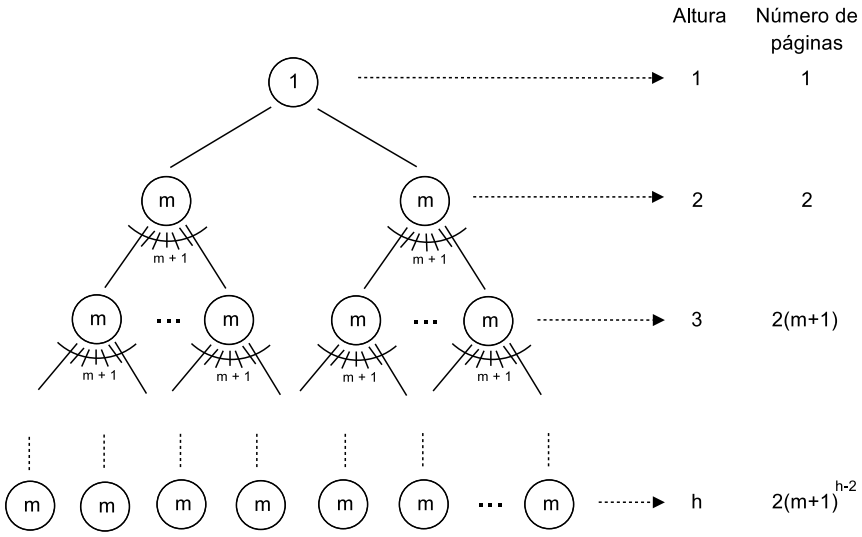

Figura 5: Estudo do pior caso de uma Árvore B de ordem m

Através da Figura 5, podemos perceber que o número total de páginas (p) da árvore, excluindo a página raiz que será lidada separadamente, é:

$$p = \sum_{i=0}^{h-2} 2(m+1)^i$$

$$p = 2\sum_{i=0}^{h-2}(m+1)^i$$

Utilizando a seguinte fórmula para progressão geométrica:

$$\sum_{i=0}^{n}\frac{x^{i+1}-1}{a-1}$$

Podemos resolver o somatório da seguinte forma:

$$p = 2\left(\frac{(m+1)^{(h-2)+1}-1}{(m+1)-1}\right)$$

$$p = 2\left(\frac{(m+1)^{h-1}-1}{m}\right)$$

Podemos obter agora o número de registros total da árvore (N) que é dado pelo número total de páginas (p), excluída a raiz, vezes o número de máximo de registros por página (m), mais 1 registro que é aquele que se encontra na raiz e foi ignorado a princípio, ou seja:

$$N = mp + 1$$

$$N = m2\left(\frac{(m+1)^{h-1}-1}{m}\right) + 1$$

$$N = 2\left((m+1)^{h-1}-1\right) + 1$$

$$N = 2(m+1)^{h-1} - 2 + 1$$

$$N = 2(m+1)^{h-1} - 1$$

Isolando-se agora o h para obtermos o limite máximo da altura da árvore, temos:

$$\frac{N+1}{2} = (m+1)^{h-1}$$

$$\log_{m+1}\frac{N+1}{2} = \log_{m+1}(m+1)^{h-1}$$

$$\log_{m+1}\frac{N+1}{2} = h-1$$

$$\log_{m+1}\frac{N+1}{2} + 1 = h$$

Assim, o limite superior da altura h, obitdo analisando o pior caso é:

$$\boxed{h \leq \log_{m+1}\left(\tfrac{N+1}{2}\right) + 1}$$

5.2 Item (b)

Obtenha empiricamente os resultados a seguir. Utilize valores distintos de $m = 1$, $m = 50$ e $m = 100$. Utilize arquivos de diferentes tamanhos com chaves geradas randomicamente para cada valor de m. Use o programa *Permut.c* para obter uma permutação randômica de n valores (Ziviani, 2004, pag. 427). Repita cada experimento algumas vezes e obtenha a média para cada medida de complexidade.

1. A altura esperada h de uma árvore B de ordem m randômica com n chaves, para os valores de m indicados acima. Assuma que a altura esperada é aproximadamente $\log_x n + 1$. Determine empiricamento o valor de x.

2. A probabilidade de que a página segura mais profunda esteja no primeiro nível da árvore.

3. O valor da taxa de utilização de memória.

Solução:

A resolução dessa questão será dividida nas seguintes partes: primeiramente será descrita a metodologia dos experimentos realizados para obter os resultados empíricos solicitados. E, em seguida, será apresentado e analizado os valores obtidos para cada item desse exercício, para as árvores de ordem 1, ordem 50 e ordem 100.

Metodologia dos testes

Para a implementação da árvore B e execução dos testes, foram utilizados como base os arquivos em C disponibilizados em [15]. Foi feita uma instrumentação desses códigos para que os mesmos fossem capaz de medir o valor da altura da árvore, a probabilidade de a página segura mais profunda estar no primeiro nível da árvore e a taxa de utilização da memória. Assim, foram feitas pequenas modificações também no formato de saída para uma melhor coleta dos dados.

A configuração dos testes realizados foi a seguinte: variou-se o número de registros a serem inseridos na árvore entre $n = 100$ e $n = 1000000$, para cada ordem m da árvore B ($m = 1$, $m = 50$ e $m = 100$). Para cada valor de n, foi ainda geradas 1000 permutações, sendo os dados colhidos para cada uma dessas execuções. Assim, ao final das mil execuções era feita a média para cada valor de n.

Os teste foram executados em uma máquina do laboratório *Speed - DCC - UFMG*, que possui a seguinte configuração: processador `Pentium IV`, $1GB$ de memória RAM, e sistema operacional *Linux*.

Item 1 - Estudo da altura h de uma Árvore B

Para obter a altura média das árvores B, foi feito o seguinte procedimento (válido tanto para as árvores de ordem $m = 1$, $m = 50$ e $m = 100$): mediu-se a altura para cada árvore gerada, e para cada valor de n foi feita a média da altura a partir das mil permutações testadas.

Como a altura esperada de uma árvore B pode ser dada por:

$$h_{esperada} = \log_x(n + 1),$$

o objetivo dessa análise era encontrar o valor de x, já que possuimos os valores experimentais de h e n, colhidos.

Para determinar o valor de x, entretanto, foi necessário primeiramente fazer uma manipulação algébrica de mudança de base do logarítmo, mostrada a seguir:

$$h_{esperada} = \frac{\log(n+1)}{\log x}$$

Através dessa equação, foi então possível utilizar a ferramenta *Gnuplot* [3] para fazer um *fitting*, que nos levaria ao valor de x procurado. Assim, foi usado o seguinte comando do *Gnuplot*:

```
f(n)= log(n+1)/log(x)
fit [:] f(n) "dados_experimentais.txt"using 1:2 via n
```

Os valores de x obtidos com essa regressão para cada uma das ordens m da árvore B estão apresentados na Tabela 1.

m	**x**
1	2.31231
50	34.8393
100	57.7618

Tabela 1: Valores de x encontrados para cada uma das ordens m da Árvore B

Após ter encontrado esses valores de x, foi feita então a construção de três gráficos (um para cada ordem $m = 1$, $m = 50$ e $m = 100$). Cada um desses gráficos apresenta três curvas: uma que descreve o limite inferior para a altura de uma árvore B, uma que descreve o limite superior para a altura de uma árvore B, e outra que foi obtida através dos pontos experimentais (também plotados no gráfico) das alturas médias das árvores e o ajuste de $\log_x(n+1)$ para o x encontrado. Os limites inferior e superior foram obtidos através da análise analítica realizada no item (a) dessa questão.

A Figura 6 corresponde ao gráfico obtido para as árvores B de ordem $m = 1$. Com base nos limites analíticos inferior e superior e no valor de x encontrado, temos que:

$$\text{Limite inferior} = 1 + \log_2 \tfrac{n+1}{2}$$
$$\text{Limite superior} = \log_3(n+1)$$
$$\text{Altura média} = \log_{2.31231}(n+1)$$

É importante, ainda ressaltar que foi colocada uma escala logarítimica no eixo x já que o número de registros oscila entre um valor muito grande. Dessa forma, as três curvas aparentam a forma de uma reta, como era esperado devido a essa transformação de escala. Sendo que isso ocorre na análise dos três gráficos gerados.

Analisando-se, então esse gráfico da Figura 6, temos que as alturas média das árvores B (para os vários n medidos) se mantiveram dentro dos limites inferior e superior, e além disso, a uma distância quase simétrica entre eles, o que representa um ótimo resultado dos experimentos realizados. Além disso, percebe-se também que mesmo para uma entrada muito grande (1000000 de registros) a altura da árvore não passa de 20.

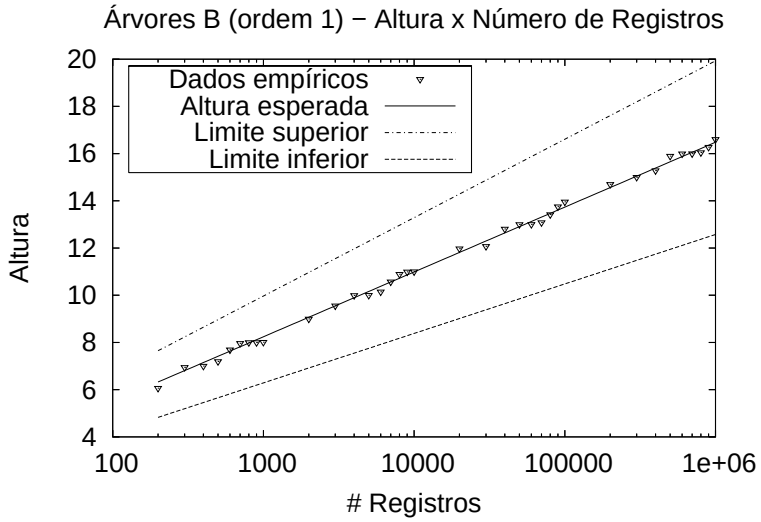

Figura 6: Gráfico correspondente a árvores B de ordem 1 - Estudo da altura

Outro ponto importante é o valor encontrado para x ($x = 2.31231$). Esse valor apresentou-se muito próximo ao valor da altura média encontrado na literatura [15], que é dado por:

$$\bar{h}(n) \approx \log_{\frac{7}{3}}(n+1)$$

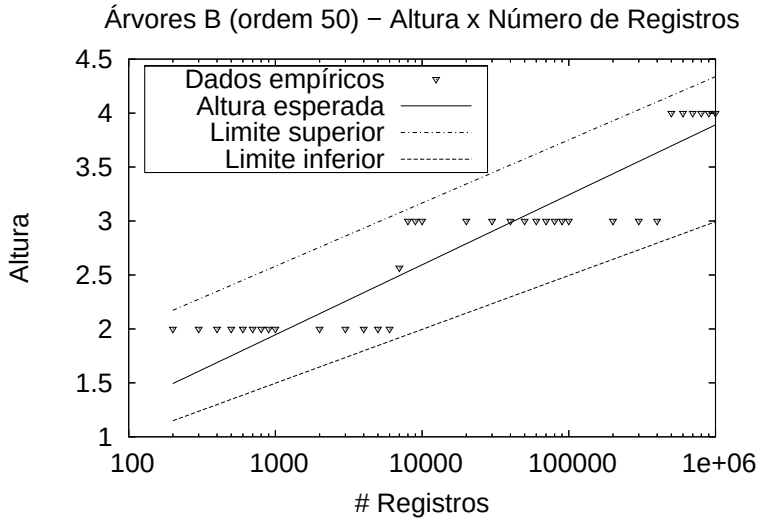

Figura 7: Gráfico correspondente a árvores B de ordem 50 - Estudo da altura

O gráfico da Figura 7 corresponde a árvores B de ordem $m = 50$. Com base nos limites analíticos inferior e superior e no valor de x encontrado, temos que:

$$\begin{gathered}\text{Limite inferior} = 1 + \log_{51} \tfrac{n+1}{2}, \\ \text{Limite superior} = \log_{101} n + 1 \\ \text{Altura média} = \log_{34.8393}(n+1)\end{gathered}$$

Analisando-se, então esse gráfico da Figura 7, temos que as alturas média das árvores B (para os vários n medidos) novamente se mantiveram dentro dos limites inferior e superior. Entretanto, os pontos empíricos não se comportaram de maneira tão regular à altura média esperada quanto os do gráfico da Figura 6. Isso pode ser justificado devido ao fato de essas árvores terem uma ordem muito grande ($m = 50$). Dessa forma, mesmo variando o número de registros é bastante provável que, durante um intervalo grande, a altura da árvore seja conservada. Pode-se ainda notar que mesmo para uma entrada de 1000000 de registros a altura máxima alcançada foi 4.

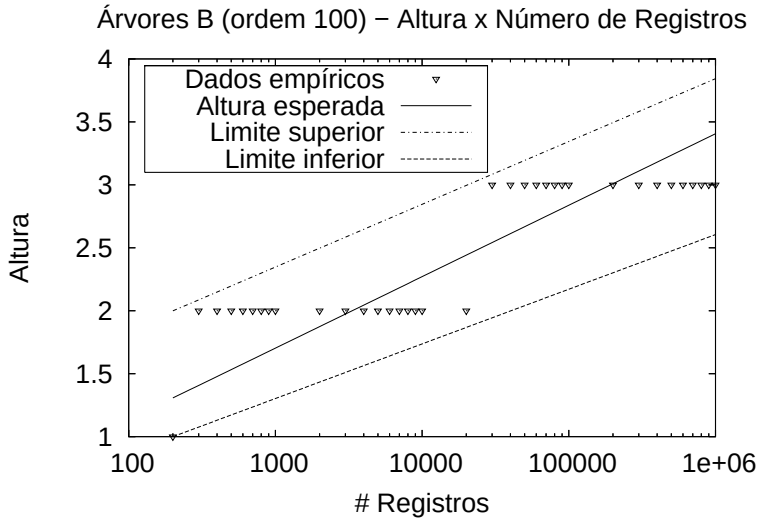

Figura 8: Gráfico correspondente a árvores B de ordem 100 - Estudo da altura

Por fim, o gráfico da Figura 8 corresponde a árvores B de ordem $m = 100$. Com base nos limites analíticos inferior e superior e no valor de x encontrado, temos que:

$$\text{Limite inferior} = 1 + \log_{101} \tfrac{n+1}{2},$$
$$\text{Limite superior} = \log_{201} n + 1$$
$$\text{Altura média} = \log_{57.7618}(n + 1)$$

A análise do gráfico da Figura 8, nos mostra que as alturas média das árvores B (para os vários n medidos) novamente se mantiveram dentro dos limites inferior e superior. Entretanto, assim como o gráfico da Figura 7, os pontos empíricos não se comportaram de maneira tão regular à altura média esperada quanto os do gráfico da Figura 6. A justificativa para tal fato é a mesma explicada anteriormente para as árvores de ordem $m = 50$. E, além disso, a altura máxima alcançada foi 3, mesmo para 1000000 de registros.

Item 2 - Estudo da página segura mais profunda

Vamos fazer agora um estudo sobre a probabilidade da página segura mais profunda se encontrar no primeiro nível da árvore. Para a realização dos experimentos para medir essa probabilidade, foi adotado o seguinte procedimento: ao colocar o último registro na árvore, verica-se se ele provoca ou não um remanejamento da mesma. Caso provoque, significa que a página profunda mais segura encontra-se no

primeiro nível da árvore, e então o algoritmo retorna o valor 1 (verdadeiro). Caso não provoque, então a página segura mais profunda não se encontra no primeiro nível da árvore, retornando 0 (falso). Dessa forma, soma-se as vezes que o algoritmo retorna o valor 1 e divide-se pelo número total de permutações da entrada.

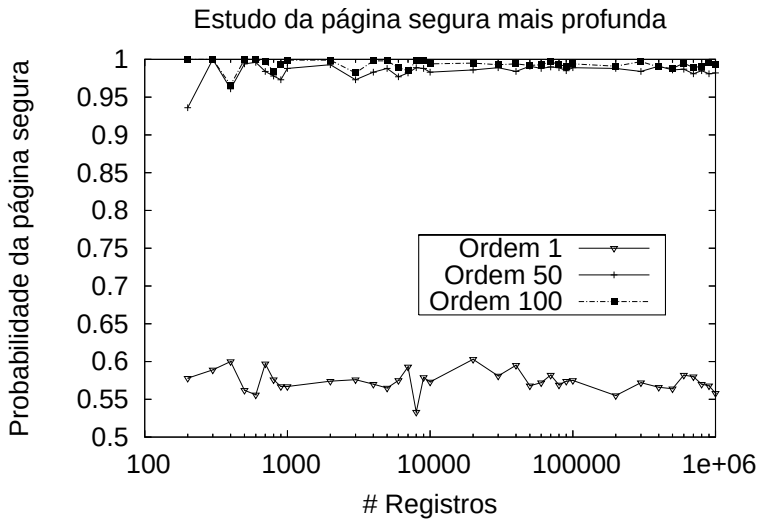

Figura 9: Gráfico correspondente a árvores B de ordem 1, 50 e 100 - Estudo da página segura mais profunda

Através dos dados colhidos, foi feito então um gráfico que apresenta a probabilidade da página segura mais profunda estar no primeiro nível da árvore para as ordens $m = 1$, $m = 50$ e $m = 100$. Esse gráfico pode ser visualizado na Figura 9.

Para fazer a análise do gráfico da Figura 9, entretanto, é necessário primeiro consultar a literatura, para podermos comparar os resultados. Dessa forma, de acordo com [15], temos que para uma árvore B de ordem $m = 1$:

$$Pr\{\text{Psmp esteja no primeiro nível}\} = \frac{4}{7}$$

Ou seja, essa probabilidade é em torno de 57%. Temos ainda que a probabilidade de a página segura mais profunda de uma árvore B de ordem $m = 70$ se encontrar no primeiro nível é de 99%. Por fim, tem-se ainda que o valor dessa probabilidade independe do número de registros da árvore n, dependendo somente de sua ordem.

Com bases nessas informações, percebe-se, então, pelo gráfico que os dados colhidos ficaram muito próximos dessas porcentagens. Para a árvore B de ordem $m = 1$, a maioria dos pontos colhidos ficaram bem próximos de 57%. Para as árvores de ordem $m = 50$ e $m = 100$, os valores obtidos também encontram-se bem razoáveis já que para $m = 50$, a probabilidade fica em torno de 97% que é próximo dos 98% para $m = 70$, e a probabilidade para $m = 100$ fica bem próxima de 100%, como era esperado. Por fim, percebe-se que realmente a probabilidade da página segura mais profunda estar no primeiro nível não depende do número de registros n.

Item 3 - Estudo da utilização de memória

Para fazer o estudo do valor da utilização da memória (T_m) foi adotado o seguinte procedimento para a realização dos testes: contabilizava-se o número de páginas (p) presenta na árvore, multiplicava-se esse número pela capacidade máxima de cada árvore ($2m$) e dividia-se o número de registros (n) inseridos por esse valor. Ou seja:

$$T_m = \frac{n}{2mp}$$

Através dos dados colhidos, foi feito então um gráfico que apresenta a taxa de utilização da memória para as ordens $m = 1$, $m = 50$ e $m = 100$. Esse gráfico pode ser visualizado na Figura 10.

Para fazer a análise do gráfico da Figura 10, entretanto, é necessário primeiro consultar a literatura, para podermos comparar os resultados. Dessa forma, de acordo com [15, 9], temos que a utilização da memória é cerca de $\ln 2$ para o algoritmo original proposto por Bayer e McCreight (1972). Isso significa que as páginas têm uma ocupação de aproximadamente 69% da área reservada após n inserções randômicas em uma árvore B inicialmente vazia. Temos ainda que a taxa de utilização de memória não é menor que 50%, já que todas as páginas devem ter pelo menos m registros, com exceção página da raiz (segundo[10]).

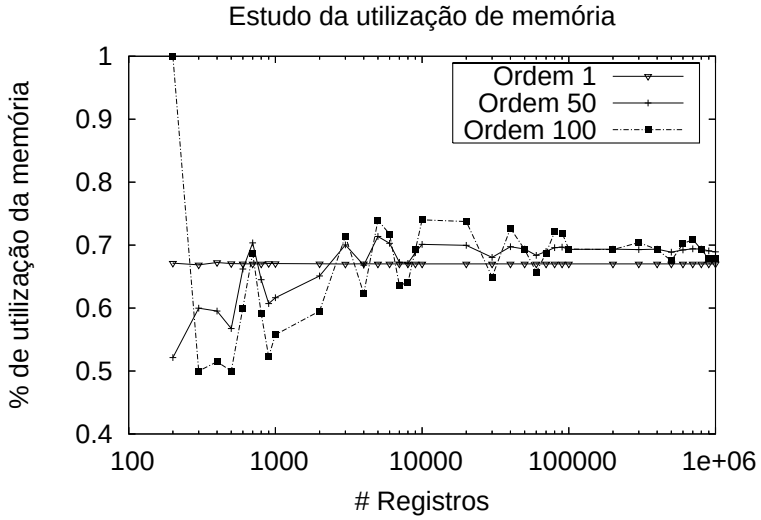

Figura 10: Gráfico correspondente a árvores B de ordem 1, 50 e 100 - Estudo da utilização da memória

Com bases nessas informações, percebe-se, então, pelo gráfico que os dados colhidos ficaram muito próximos dessas porcentagens discutidas. Para a árvore B de ordem $m = 1$, a utilização da memória foi constante e em torno 69%, como era esperado. Para as árvores de $m = 50$ e $m = 100$ os dados não apresentaram um comportamneto tão regular quanto a de ordem $m = 1$. Entretanto, apesar de algumas oscilações para valores de n pequenos, percebe-se uma tendência a medida em o n cresce de que as curvas venham a se convergir perto do valor de utilização 69%, como também se espera. Podemos explicar essas oscilações para as árvores de ordem grandes (como $m = 50$ e $m = 100$) para um número de registros n pequenos, devido ao fato que que o desperdício só é evitado com um número grande de registros.

5.3 Item (c)

Procure trabalhos relacionados mais recentes na literatura com resultados analíticos e descreva suscintamente esses trabalhos. Como os resultados experimentais que você obteve comparam com os resultados analíticos de Ziviani (2004) e/ou encontrados na literatura?

Solução:

A comparação dos resultados experimentais com os resultados analíticos encontrados na literatura foi realizada no item (b) (com análises e críticas para os resultados encontrados) e, portanto, será feita aqui somente uma breve conclusão dos resultados obtidos, e a seguir uma discussão de alguns trabalhos mais recentes sobre árvore B encontrados na literatura.

Em geral, temos que a metodologia de testes utilizada foi bastante satisfatória já que os resultados experimentais obtidos coincidiram em grande parte com aqueles encontrados na literatura. Nas pouca vezes que houve alguma diferença, foi possível explicar a razão da mesma, o que não comprometeu os testes realizados e permitiu até uma análise mais crítica de todo o processo.

Nos concentraremos, agora, portanto, em descrever alguns trabalhos relacionados encontrados na literatura. Atualmente, percebe-se que a árvore B ainda é utilizada em um número considerável de aplicações, sendo bastante interessante levantar um estudo sobre elas. Percebe-se, por exemplo, que ela é usada em indexação de objetos na *Web*, ou como idexação de dados em sistemas de gerência de banco de dados. Dessa forma, alguns trabalhos nesse sentido, que avaliam e tentam melhorar o desempenho da árvore B, como [11], [14], [6], e [2]. Ainda como indexadores, a árvore B é utilizada também na indexação de cache. Alguns trabalhos apresentam o impacto que o tamanho do nodo tem sobre o desempenho de consistência da cache, como [5], avaliando qual seria esse tamanho mais apropriado que proporcionaria o melhor desempenho da mesma. Há ainda alguns trabalhos que apresentam uma indexação temporal de dados utilizando árvores B, como [4], e outros que trabalham em variações da versão original para casos específicos, como [12]. É importante ressaltar também os modelos analíticos de desempenho de algoritmos de árvore B recentes apresentados em [8].

Além dos trabalhos citados, existem diversos outros que utilizam árvore B em várias aplicações, sendo que o conjunto deles demostra que a árvore B ainda é uma estrutura significativa em diversas áreas da computação.

6 Contribuições e agradecimentos

Alguns colegas de sala dessa disciplina foram fundamentais no entendimento dos problemas desse trabalho. Podem ser citados os alunos: Guilherme H. T. Ferreira, Alex Borges, Elisa Tuler, Leonardo Chaves, Marcelo Maia, Frederico Quintão, Bruno Coutinho, entre outros. Foram diversas as discussões sobre os resultados obtidos, assim como sobre referências bibliográficas e decisões de implementação que contribuiram positivamente para a análise do custo de funções recursivas e das questões que envolvem árvores B.

Referências

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge Mass., second edition, 2001.

[2] Michael Freeston. A general solution of the n-dimensional b-tree problem. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 80–91, New York, NY, USA, 1995. ACM Press.

[3] Gnuplot 3.7.3. http://www.gnuplot.info.

[4] Cheng Hian Goh, Hongjun Lu, Beng Chin Ooi, and Kian-Lee Tan. Indexing temporal data using existing b+-trees. *Data Knowl. Eng.*, 18(2):147–165, 1996.

[5] Richard A. Hankins and Jignesh M. Patel. Effect of node size on the performance of cache-conscious b+-trees.

[6] Joseph M. Hellerstein, Jeffrey F. Naughton, and Avi Pfeffer. Generalized search trees for database systems. In Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors, *VLDB*, pages 562–573. Morgan Kaufmann, 1995.

[7] Ellis Horowitz and Sartaj Sahni. *Fundamentals of Computer Algorithms.* Computer Science Press, 1978.

[8] Theodore Johnson and Dennis Shasha. The performance of current b-tree algorithms. *ACM Trans. Database Syst.*, 18(1):51–101, 1993.

[9] Donald E. Knuth. *The art of computer programming, volume 3: sorting and searching.* Addison-Wesley, Reading, 1st edition, 1973.

[10] Carl E. Langenhop and William E. Wright. An efficient model for representing and analyzing b-trees. In *ACM '85: Proceedings of the 1985 ACM annual conference on The range of computing : mid-80's perspective*, pages 35–40, New York, NY, USA, 1985. ACM Press.

[11] D. B. Lomet. Digital BTrees. In *Proceedings of the 7th Conference on Very Large Databases, Morgan Kaufman pubs. (Los Altos CA), Zaniolo and Delobel(eds)*, 1981.

[12] David B. Lomet and Betty Salzberg. The hb-tree: a multiattribute indexing method with good guaranteed performance. *ACM Trans. Database Syst.*, 15(4):625–658, 1990.

[13] Udi Manber. *Introduction to Algorithms. A Creative Approach.* Addison-Wesley, 1989. MAN u 89:1 P-Ex.

[14] Yehoshua Sagiv. Concurrent operations on b-trees with overtaking. In *PODS '85: Proceedings of the fourth ACM SIGACT-SIGMOD symposium on Principles of database systems*, pages 28–37, New York, NY, USA, 1985. ACM Press.

[15] N. Ziviani. *Projeto de Algoritmos com Implementações em Pascal e C.* Pioneira Thomson Learning, São Paulo., second edition, 2004.