CONSTRUÇÃO DE ÁRVORES BINÁRIAS DE BUSCA SEMI-ÓTIMAS ATRAVÉS DE ALGORITMOS HEURÍSTICOS

Francisco Cláudio Sampaio de Menezes

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA (M.Sc.)

Aprovada por:

Célio Guimarães
Presidente

Pierre-Jean Lavelle

Nélson Maculan Filho

ESTADO DO RIO DE JANEIRO - BRASIL

NOVEMBRO DE 1975

À
Marilde

AGRADECIMENTOS

Desejo expressar meu agradecimento:

- ao Prof. Célio Guimarães pela idéia, inúmeras sugestões e paciente orientação desta tese;
- aos meus pais, José Carlos de Menezes e Maria Vilani Sampaio de Menezes, pelo incentivo ao estudo que sempre me transmitiram;
- ao Banco do Nordeste do Brasil S.A., particularmente ao seu Departamento de Organização e Processamento(ORPRO), na pessoa do Dr. José Soares Nuto, pela excelente oportunidade profissional que me foi proporcionada, dentro de uma adequada política de aperfeiçoamento dos recursos humanos da Empresa;
- ao corpo docente da COPPE/UFRJ, em especial aos professores Nélson Maculan Filho e Fernando Yassuo Chiyoshi, pelo estímulo e amizade com que sempre me distinguiram;
- aos colegas do BNB que colaboraram na confecção da parte gráfica deste trabalho

O autor

RESUMO

Apresentam-se neste trabalho dois algoritmos para construção de árvores binárias de busca semi-ótimas, a partir da distribuição de probabilidades estacionária das consultas aos N nós e N+1 folhas da estrutura.

Os algoritmos utilizam uma técnica de construção e percurso de uma árvore auxiliar (não-binária) através de enumeração implícita, a partir de teoremas demonstrados.

As árvores obtidas foram superiores às resultantes de tentativas heurísticas anteriores existentes na literatura. Ambos os algoritmos consomem memória proporcional a N e tempo proporcional a $N.\log_2 N$.

Além disso, para arquivos em que não se conhece qualquer informação sobre o comportamento das consultas, foi elaborado um algoritmo adaptativo que permite inserções e consultas aleatórias e a otimização da árvore binária cada vez que é gerada uma transação.

ABSTRACT

Two algorithms are presented here for the construction of nearly optimum binary search trees, from the stationary probability distribution for access to the N nodes and N+1 leaves of the structure.

These algorithms use a technique of constructing and visiting the vertices of an auxiliary tree (non-binary) through an implicit enumeration, from proved theorems.

The constructed trees are in average better than those resultant from heuristical experiences made before, included in the literature. Experimental results show that both algorithms use storage proportional to N and time proportional to $N \log_2 N$.

Additionally, for keys from which no information about the access behavior is known, an adaptative algorithm was elaborated to permit random access or insertion and optimization of the binary tree each time a transation is performed.

SUMÁRIO

CONSTRUÇÃO DE ÁRVORES BINÁRIAS DE BUSCA SEMI-ÓTIMAS ATRAVÉS DE ALGORITMOS HEURÍSTICOS

0. INTRODUÇÃO

Dentre as técnicas para organização de arquivos com elevado número de chaves, árvores binárias de busca constituem método importante, porque são eficientes tanto para acesso aleatório como sequencial, bem como para modificação do arquivo.

Além disso e pelo fato de terem originado interessantes problemas matemáticos (em conexão com outros ramos de pesquisa, tais como teoria da codificação, classificação, árvores de decisão) as árvores binárias tem sido bastante investigadas nos últimos anos, havendo maior número de descobertas de suas propriedades matemáticas, em relação aos outros métodos de organização de arquivos.

No presente trabalho, são apresentados dois algoritmos para construção de árvores binárias ordenadas lexicograficamente, bem como um algoritmo adaptativo, que otimizam o valor esperado do número de comparações para busca de uma informação na estrutura construída.

0.0. Definições

Def.: 0.0 - Uma árvore binária pode ser definida, recursivamente, como segue:

a. uma árvore vazia, T_0, de zero nós, é uma árvore binária;
b. uma árvore binária T_N, de $N \geq 1$ nós, é um triplo ordenado $(L(v),v,R(v))$, onde v é um nó chamado raiz de T_N, $L(v)$ e $R(v)$ são árvores binárias de l e r nós, chamadas respectivamente de sub-árvores da esquerda e direita da raiz ($l \geq 0$, $r \geq 0$, $l + r = N - 1$).

Def.: 0.1 - Dadas N informações ou "chaves" em ordem lexicográfica ($K_1 < K_2 < \ldots < K_N$), uma árvore binária de busca é uma árvore binária, T_N, tal que para cada nó n as seguintes propriedades valem:

a. todas as chaves na sub-árvore esquerda de n precedem n na ordem lexicográfica dada;
b. todas as chaves na sub-árvore direita de n sucedem n na ordem lexicográfica dada.

Por exemplo:

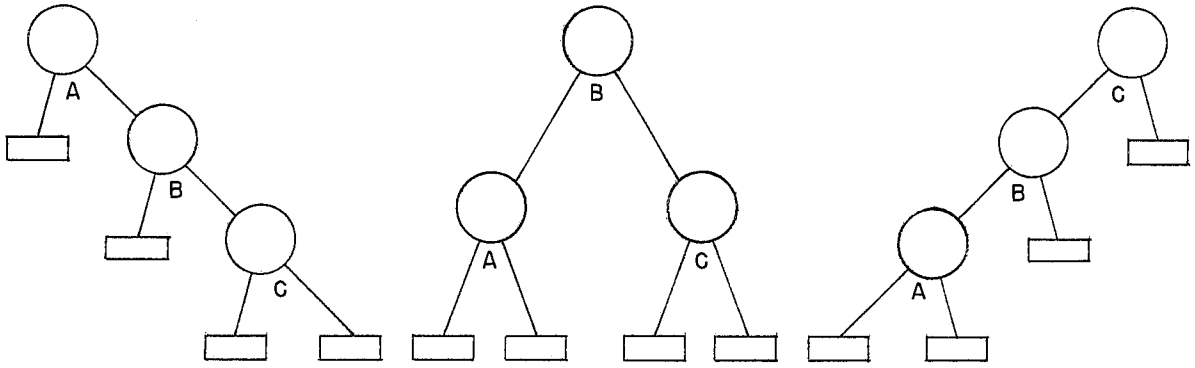


FIG. 0

Estas árvores serão chamadas simplesmente de árvores de busca.

1. BUSCA EM ÁRVORES BINÁRIAS

1.0. Formulação do Problema

O problema resolvido por Knuth [8,9], relativo à construção de árvores binárias ótimas para busca, pode ser enunciado matemáticamente da seguinte maneira:

Dadas N chaves $K_1 < K_2 < \ldots < K_N$ e as probabilidades

$p_i = P\{K_i \text{ seja o argumento de uma busca}\}$, $i = 1, \ldots, N$

$q_i = \{P\{K_i < \text{argumento da busca} < K_{i+1}\}$, $i = 1, \ldots, N-1$

$q_0 = P\{\text{argumento da busca} < K_1\}$

$q_N = P\{\text{argumento da busca} > K_N\}$ (1)

$$\sum_{i=1}^{N} p_i + \sum_{j=0}^{N} q_j = 1$$

Objetivo: Construir uma árvore binária T*, de busca, de modo que o valor esperado do número de comparações para localizar ou determinar a ausência de um argumento K, escolhido segundo a distribuição de probabilidade dada por (1), seja minimizado.

1.1. Custo Esperado

Como o número de árvores binárias constituídas de N nós é finito — $(\binom{2N}{N})/(N+1) \cong 4^N/N^{3/2}$ — então o mínimo existe.

Seja T_N a família das árvores binárias de busca com N nós e $T \in T_N$; o nível $l_{(j)}$ de um nó interno K_j é o nº de nós menos 1 do caminho desde a raiz ao nó K_j e $l_{(j)}$, nível da folha j, é o número de nós internos desde a raiz até a folha j.

Dado $K = K_j$, o número de comparações $C_{T|K_j}$ para localizar K_j será:

$$C_{T|K_j} = l_{(j)} + 1 \qquad (2)$$

e o número de comparações para detetar se um argumento de busca K satisfaz a $K_j < K < K_{j+1}$ é:

$$C_{T|K_j < K < K_{j+1}} = l_{(j)} \qquad (3)$$

A cada nó interno j da árvore associaremos uma chave K_j, a probabilidade p_j e outros valores que serão posteriormente apresentados. Esta árvore pode ser extendida com N+1 folhas, às quais associaremos as probabilidades $q_0, q_1, \ldots, q_N$ da seguinte forma: a folha q_j está localizada na sub-árvore direita do nó interno K_j, j = 1,2, ..., N e q_0 na sub-árvore esquerda do nó K_1.

Deste modo o evento (2), acima, ocorre com probabilidade p_j e o evento (3) ocorre com probabilidade q_j; sendo C_T o

número de comparações para localizar o argumento K distribuído de acordo com (1), C_T será uma variável aleatória tomando os valores $l_{(j)} + 1$ e $l_{[j]}$, com probabilidades p_j e q_j, respectivamente. Então o valor esperado de C_T é:

$$E(C_T) = \sum_{j=1}^{N} p_j(l_{(j)} + 1) + \sum_{j=0}^{N} q_j l_{[j]} \qquad (4)$$

Para simplificar a notação vamos denominar os nós internos $K_1, K_2, \ldots, K_N$, de $1,2,\ldots,N$; dada uma árvore de busca T construída com os nós $1,2,\ldots,N$, definimos:

T(i) — é a sub-árvore binária com raiz i;

L(i) — é a sub-árvore esquerda do nó i;

R(i) — é a sub-árvore direita do nó i;

$$W(i) \text{ — } W(T(i)) = \sum_{j \in T(i)} p_j + \sum_{l \in T(i)} q_l \qquad (5)$$

Ex.: T(2)

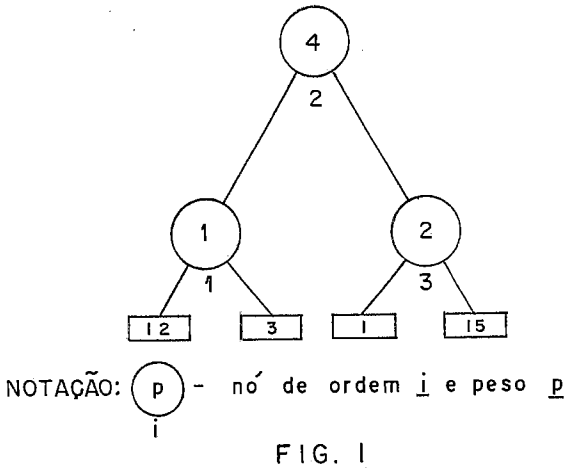


FIG. 1

W(2) = 38

É óbvio, pela definição de W(i), que:

$$W(i) = W(L(i)) + p_i + W(R(i)) \qquad (6)$$

Teorema 1.0 — $E(C_T) = \sum_{i=1}^{N} W(i)$ |7|

Prova (por indução): Se N = 1 é trivial, porque $E(C_T) = p_1 + q_0 + q_1 = W(1)$.

Suponhamos que o teorema vale para uma árvore qualquer com N - 1 nós. Seja T_N uma árvore binária qualquer com $N \geq 2$ nós, v um nó interno de distância (nível) máxima a raiz e W(v) a soma das probabilidades p_i e q_i, conforme (5).

Sem perda de generalidade, podemos admitir que o N-ésimo nó foi acrescentado à direita de v, produzindo a árvore $\hat{T}$.

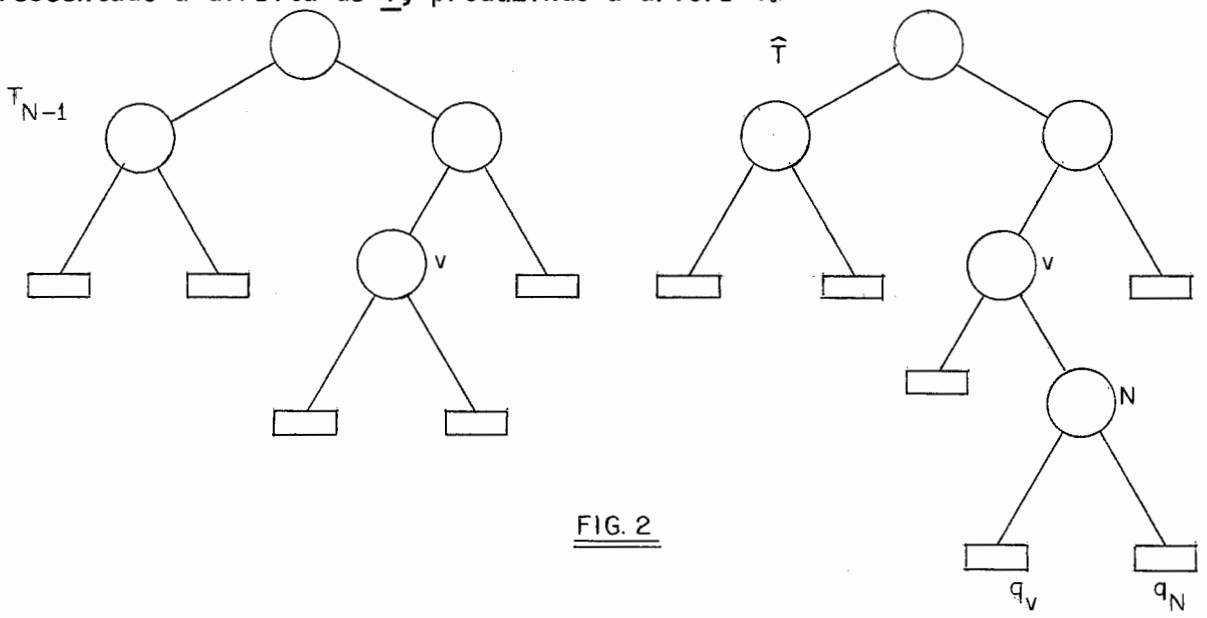


FIG. 2

Aplicando a definição: $E(C_{\hat{T}}) = \sum_{j=1}^{N} \hat{p}_j (l_{\hat{T}}(j)+1) + \sum_{j=0}^{N} \hat{q}_j l_{\hat{T}}(j)$.

Seja C o caminho que liga a raiz ao último nó incluído. Vemos que:

a) $\hat{p}_j = p_j$, $j=1,\ldots,N-1$; $\hat{q}_j = q_j$, $j=0,\ldots,N-1$; p_N e q_N são os valores incluídos;

b) Se $t \notin C$, então $\hat{W}(t)=W(t)$; $l_{\hat{T}}(t) = l_T(t)$; $l_{\hat{T}}(t) = l_T(t)$;

c) Se $t \in C$, então $\hat{W}(t)=W(t)+p_N+q_N$ para $t \neq N$; $W(n)=p_N+q_N+q_v$; $l_{\hat{T}}(j)=l_T(j)$ e $l_{\hat{T}}(j)=l_T(j)$ para $j \neq N,v$; $l_{\hat{T}}(N)=l_{\hat{T}}(v)=l_T(v)+1$.

Comparando $\hat{T}_N$ e T_{N-1}, podemos escrever:

$$E(C_{\hat{T}}) = \sum_{j=1}^{N-1} p_j (l_T(j)+1) + \sum_{j=0}^{N-1} q_j l_T(j) + q_v+q_N l_T(N)+p_N(l_T(N)+1) = \sum_{i=1}^{N-1} W(i) + q_v+q_N l_T(N)+p_N(l_T(N)+1)$$

$$= \sum_{i \notin C} W(i) + \sum_{\substack{j \in C \\ j \neq N}} W(j) + q_v+q_N l_T(N)+p_N(l_T(N)+1) = \sum_{i \notin C} W(i) + \sum_{j \in C} (W(j)+p_N+q_N) \quad \text{\#}$$

$$q_v+q_N+p_N = \sum_{i \notin C} \hat{W}(i) + \sum_{j \in C} \hat{W}(j) + W(N) = \sum_{k=1}^{N} W(k)$$

$$\therefore E(C_T) = \sum_{i \varepsilon T} W(i)$$

1.2. Árvore Ótima de Busca

Def.: 1.0 - Uma árvore binária de busca T* se diz ótima se

$E(C_{T*}) \leqslant E(C_T)$, $\forall\ T \varepsilon T_N$.

O conjunto das árvores ótimas pode ter mais de um elemento.

O princípio básico do algoritmo de construção de árvores ótimas (Apêndice 0 - Algoritmo K) é que se T* pertence ao conjunto das árvores ótimas, então todas as sub-árvores de T* são ótimas. Do contrário, a otimização de uma sub-árvore não-ótima causaria redução no valor de $E(C_{T*})$. Trata-se de um algoritmo "bottom-up".

Embora a árvore produzida seja ótima, o Algoritmo K tem como principal restrição a sobrecarga de memória e de tempo, da ordem de N^2 e N^3, respectivamente. Através de uma propriedade demonstrada em |9| é possível tornar o tempo de execução proporcional a N^2.

Para se ter idéia, um programa escrito em ALGOL-W |13| para N = 256 nós, consumiria aproximadamente 832 Kbytes de memória, na fase de execução. Se N = 16K nós, o tempo de execução seria esti

mado em torno de 460 minutos (mais de 7 horas), em um computador /370-mod. 165-IBM, e necessitaria de cerca de 3,22 megabytes nesta fase.

Outra limitação do "Algoritmo K" é não ser adaptativo, dificultando sua aplicação quando se desconhecem as probabilidades estacionárias de consulta às chaves e entre chaves consecutivas.

Cabe registro ainda sobre a existência de um algoritmo bastante complexo, devido a Hu e Tucker, que edifica uma árvore binária ótima quando $q_i = 0$, $i = 0,1, \ldots, N$ e requer da ordem de N unidades de memória e $N\log_2 N$ unidades de tempo |6|.

1.3. Métodos Heurísticos

Diante das dificuldades para executar o "Algoritmo K" quando N se eleva, tentativas tem sido feitas visando a construção de árvores semi-ótimas — da ordem de 2% ou menos de diferença para o valor ótimo de $E(C_T)$ — em cuja obtenção o consumo de tempo seja da ordem de $N\log_2 N$ e o uso de memória seja proporcional a N.

O trabalho de Walker & Gotlieb |14| mostra um algoritmo "top-down" que consome memória O(N) e tempo $O(N\log_2 N)$, produzindo árvores binárias em que $E(C_T) = 1,012E(C_{T_{\text{ótima}}})$, em média, com desvio padrão de 0,011 para os dados do Apêndice 1, implementados no citado estudo.

O princípio utilizado consiste na formação dos ní

veis 0,1, ... ,J, através da escolha de uma raiz inicial pelo método do centróide (*).

Como o centróide pode ter probabilidade de consulta muito pequena, busca-se numa vizinhança o nó de maior probabilidade, colocando-o na raiz da árvore. A vizinhança é definida por parâmetro externo. Quando o nº de nós das sub-árvores a serem formadas for inferior a N_0 — que é um indicador do nº de nós de uma sub-árvore que deve ser construída pelo "Algoritmo K"— constroem-se sub-árvores ótimas através do Algoritmo K.Do contrário,aplica-se novamente o método centróide às sub-árvores formadas até encontrar $N \leqslant N_0$. Este será chamado "Algoritmo GW" ou "Algoritmo G".

1.4.Rotações

J. Bruno e E.G. Coffman Jr. |2| utilizam o conceito de rotação dos nós (também chamadas promoções ou transformações) a fim de obter reduções sucessivas do $E(C_T)$, a partir de uma árvore inicial.

Estas transformações são explicadas a seguir, onde $t(i) = \bar{T}$ significa que a rotação t_i aplicada ao nó i de uma árvore binária T produziu a árvore $\bar{T}$.

(*) - Centróide é um nó, K_c, tal que o valor absoluto de $W(L(c)) - W(R(c))$ é mínimo.

Def.: 1.1 - Se i = raiz de T, então t(i) = T, isto é, a árvore T não se altera; caso contrário, definimos:

a - rotação para a direita de um nó i em torno de um nó j, $t_r(i)$, onde i é a raiz de L(j), a seguinte transformação de T:

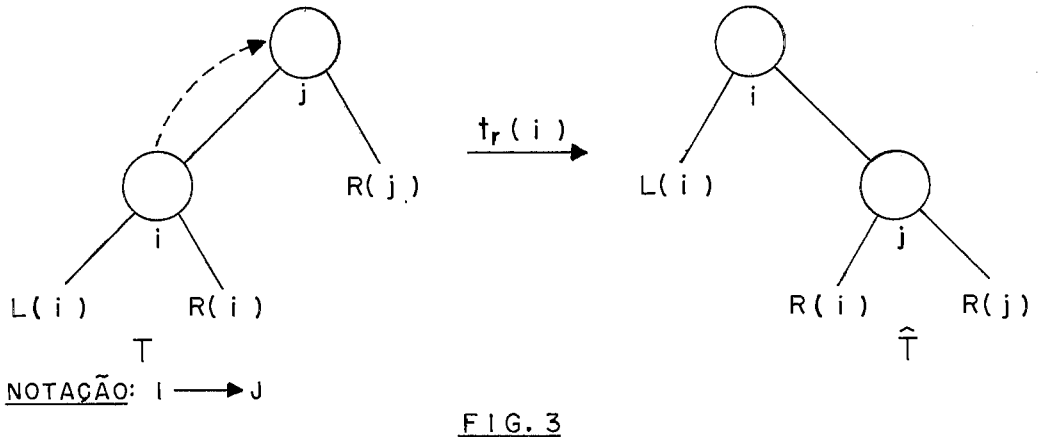


FIG. 3

b - rotação para a esquerda de um nó j em torno de um nó i, $t_l(j)$, onde j é a raiz de R(i), a seguinte transformação:

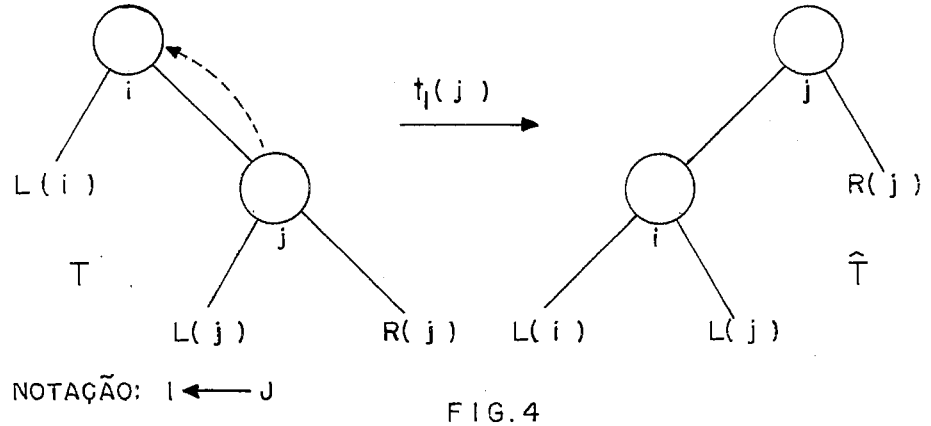


FIG. 4

Constata-se que se $T \varepsilon T_N$ e $\bar{T} = t(i)$, então $\bar{T} \varepsilon T_N$.

Por conveniência adotemos a convenção de que $\xi < T$ e $\xi > T$, onde ξ = árvore binária vazia e >/< significa maior/menor que qualquer nó de T e podemos escrever:

$L(i) < i < R(i) < j < R(j)$ — antes da rotação

$L(i) < i < R(i) < j < R(j)$ — depois da rotação. Logo $\bar{T} \varepsilon T_N$.

Desta forma a ordem lexicográfica é mantida quando os nós são visitados em ordem simétrica |7|.

2. ALGORITMO I - Inserção à Direita

2.0. Fundamentos

Dada uma árvore binária de busca T e um nó $i \in T$, define-se:

$E(i) = p_i + W(L(i))$, peso esquerdo de i

$D(i) = p_i + W(R(i))$, peso direito de i.

Da fórmula (6) se segue que $W(i) = E(i) + D(i) - p_i$

Teorema 2.0 | 2 |: Seja $\bar{T}$ a árvore obtida através de uma rotação para a direita de um nó i em torno de um nó j de uma árvore T. Então:

$E(C_{\bar{T}}) - E(C_T) = D(j) - E(i)$

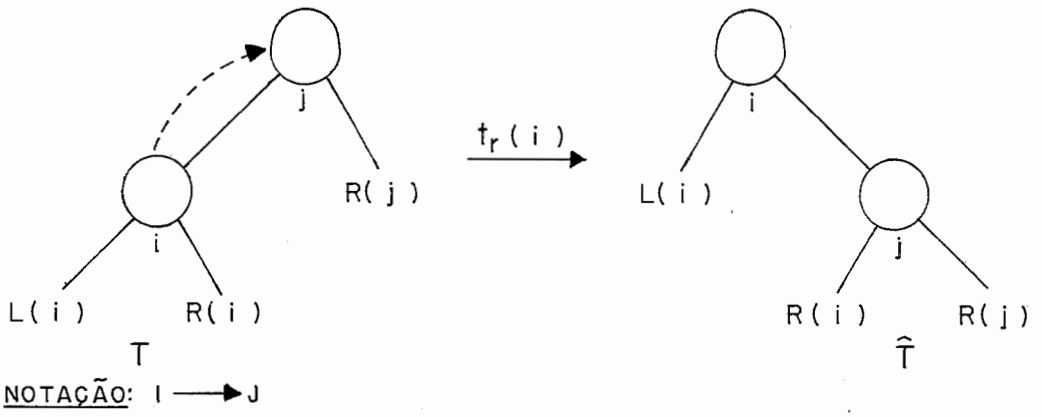


FIG. 5

Prova: Temos, na figura acima:

$\hat{W}(v) = W(v), \quad v \neq i, \quad v \neq j$

$\hat{W}(j) = p_j + W(R(i)) + W(R(j))$

$\hat{W}(i) = p_i + W(L(i)) + \hat{W}(j)$

$W(j) = p_j + W(R(j)) + W(i)$

$E(C_{\hat{T}}) - E(C_T) = \hat{W}(i) + \hat{W}(j) - W(i) - W(j)$

Mas $\hat{W}(i) = W(j)$ por construção e

$E(C_{\hat{T}}) - E(C_T) = p_j + W(R(j)) + W(R(i)) - (p_i + W(L(i)) + W(R(i)))$

$E(C_{\hat{T}}) - E(C_T) = D(j) - E(i)$

Teorema 2.1 | 2 | - Seja $\hat{T}$ a árvore obtida através de uma rotação para a esquerda de um nó i em torno de um nó j. Então,

$E(C_{\hat{T}}) - E(C_T) = E(j) - D(i)$

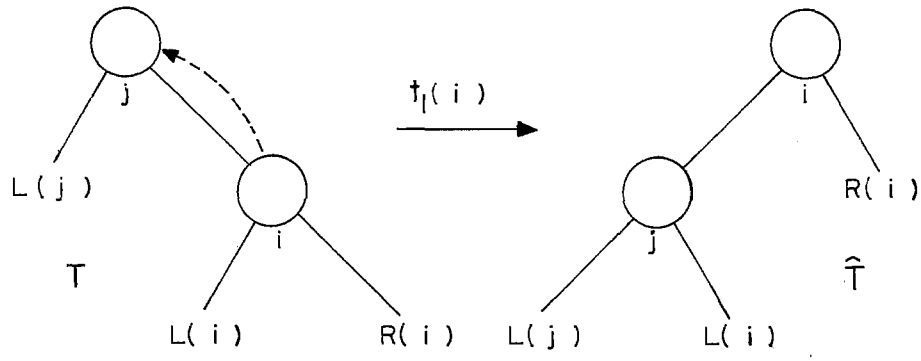


FIG.6

Na figura acima, vemos que:

$\hat{W}(v) = W(v) \quad v \neq i, \quad v \neq j$

$\hat{W}(j) = W(L(j)) + p_j + W(L(i))$

$\bar{W}(j) = \hat{W}(j) + p_i + W(R(i))$

$W(i) = W(L(i)) + p_i + W(R(i))$

$E(C_{\hat{T}}) - E(C_T) = \bar{W}(i) + \bar{W}(j) - W(i) - W(j)$

Mas $\bar{W}(j) = W(j)$, logo

$E(C_{\hat{T}}) - E(C_T) = \bar{W}(j) - W(i)$

$= W(L(j)) + p_j + W(L(i)) - (W(L(i)) + p_i + W(R(i)))$

$E(C_{\hat{T}}) - E(C_T) = p_j + W(L(j)) - (p_i + W(R(i)))$

$E(C_{\hat{T}}) - E(C_T) = E(j) - D(i)$

Def.: 2.0 — Uma rotação é dita viável se

$E(C_{\hat{T}}) - E(C_T) < 0$

Teorema 2.2 - Uma rotação $i \rightarrow j$ é viável se e somente se $E(i) > D(j)$.

Prova: Pelo Teorema 2.0, $E(C_{T'}) < E(C_T)$ se $D(j) - E(i) < 0$ que implica $E(i) > D(j)$. Reciprocamente, se $E(i) > D(j)$, então $D(j) - E(i) < 0$ e $E(C_{T'}) < E(C_T)$.

Teorema 2.3 - Uma rotação $j \leftarrow i$ é viável se e somente se $D(i) > E(j)$.

Prova: Pelo Teorema 2.1, $E(C_{T'}) < E(C_T)$ se $E(j) - D(i) < 0$ que implica $D(i) > E(j)$. Reciprocamente, se $D(i) > E(j)$, então $E(j) - D(i) < 0$ e $E(C_{T'}) < E(C_T)$.

Def.: — Último caminho: Dada uma árvore binária T, o seu último caminho é o conjunto de nós tais que:

1. a raiz pertence ao último caminho:
2. o filho da direita de qualquer nó do último caminho pertence ao último caminho.

Em consequência, dada a raiz de T, está determinado o seu último caminho, constituído pela raiz e todos os filhos da direita, de acordo com a linha pontilhada na figura 7.

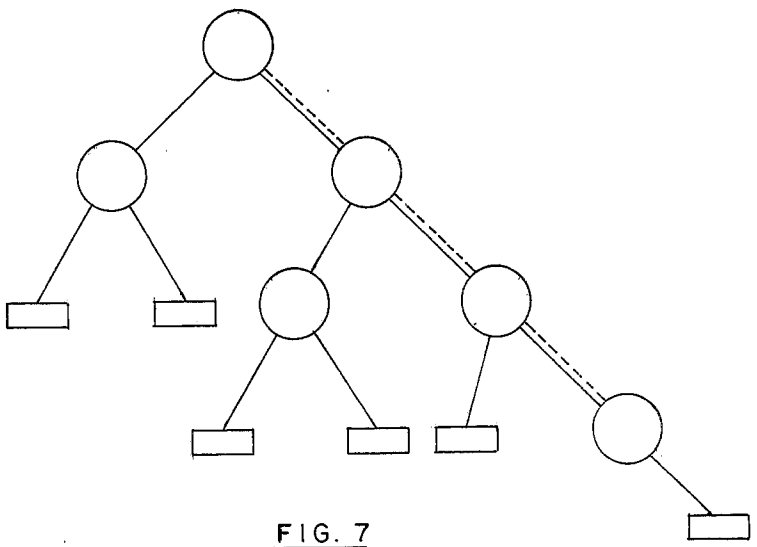


FIG. 7

Se T for vazia, o último caminho é vazio.

2.1. Descrição do Algoritmo I

A partir de uma árvore binária vazia, faz-se a inserção dos nós na ordem lexicográfica. Cada vez que é feita uma inserção procura-se melhorar a árvore através de sucessivas rotações para a esquerda, no último caminho, tendo em vista que a inclusão de $p_i + q_i$ não provoca desbalanceamento nos pesos esquerdo e direito, E(j) e D(j), dos nós não pertencentes ao último caminho. Admite-se, assim, que todas as rotações viáveis surgidas após a inclusão de um nó em T se encontram no último caminho. É claro que antes da inclusão poderia haver rotações viáveis em outros caminhos da

árvore binária.

Com a inclusão, os pesos da direita dos nós do último caminho sofrem aumento.

A escolha da sequência de rotações a ser efetuada é feita após o exame de todas as sequências possíveis, selecionando aquela que nos dá a maior redução em $E(C_T)$. O exame de todas as sequências de rotações decorre da necessidade de analisar a redução ou aumento de $E(C_T)$ de todas as árvores binárias que podem ser obtidas a partir das sequências de rotações no último caminho, uma vez que cada sequência produz uma árvore diferente (à exceção das sequências redundantes, de acordo com o parágrafo 2.3.3).

A investigação deste problema mostrou que todas as sequências de rotações podem ser organizadas em forma de uma arvore não-binária, que chamaremos de árvore auxiliar, cujos vértices podem ser percorridos através da técnica de enumeração implícita, a ser explicada no parágrafo 2.3.1. Ao final deste exame, dispõe-se da sequência dos nós que devem ser rebaixados de nível no último caminho, de modo a reduzir o custo esperado $E(C_T)$ ao máximo. Isto é feito com o auxílio de uma tabela que contem o peso da direita D(i) e o peso da esquerda E(i), para todo nó pertencente ao último caminho, a qual será chamada tabela de pesos.

É importante notar que ao ser feita uma promoção para a esquerda realizam-se as seguintes modificações na árvore e na tabela de pesos:

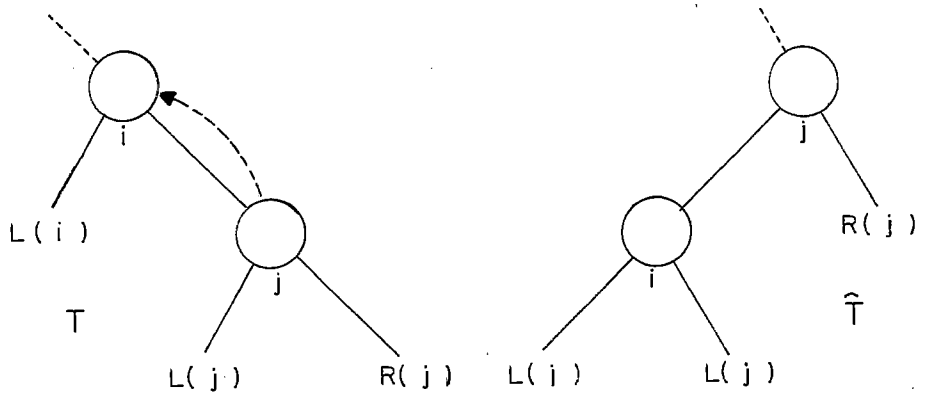


FIG.8

Tabela de Pesos 2.0

Nó	Antes da Rotação E	D	Depois da Rotação Ê	D̂
i	E(i)	D(i)	E(i)	D(i)-E(j)
j	E(j)	D(j)	E(i)+E(j)	D(j)

Após i ← j, D(i) é reduzido, já que perdeu o ramo direito de j, e E(j) aumenta porque ganhou i e L(i). De forma que, após a transformação tem-se: $\hat{D}(i) = D(i) - D(j)$ e $\hat{E}(j) = E(j) + E(i)$, onde o símbolo ^ indica os valores depois da rotação.

2.2. Inserção

Por conveniência, vamos numerar os nós do último

caminho de 1 a m, a partir da raiz. Por definição, $D(i) = p_i + W(R(i))$.

Feita a inserção do nó m+1, $m \neq 0$, juntamente com a folha q_{m+1}, conforme a fig. 9, os novos pesos $\hat{W}$, $\hat{E}$ e $\hat{D}$ serão dados por:

$$\hat{W}(R(m)) = W(R(m)) + p_{m+1} + q_{m+1}$$

$$\hat{D}(m) = D(m) + p_{m+1} + q_{m+1}$$

$$\hat{E}(m) = E(m)$$

Por indução,

$$\hat{D}(j) = D(j) + p_{m+1} + q_{m+1}, \quad j = 1, \ldots, m$$

$$\hat{E}(j) = E(j), \quad j = 1, \ldots, m$$

$$\hat{E}(m+1) = p_{m+1} + q_m$$

$$\hat{D}(m+1) = p_{m+1} + q_{m+1}$$

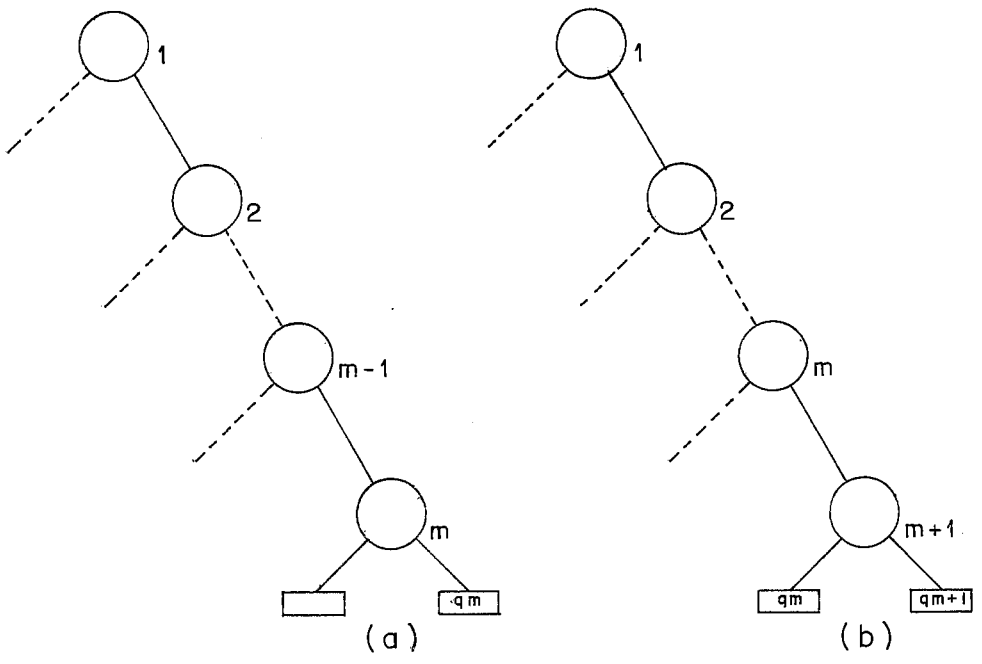


FIG. 9

No caso do 1º nó inserido, inclui-se também a folha associada a q_0, produzindo a árvore abaixo

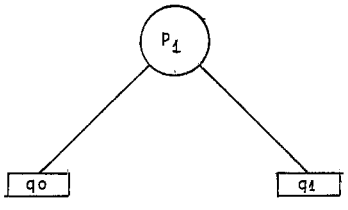


FIG. 10

2.3. Geração e Percurso da Árvore Auxiliar

Dado o último caminho de uma árvore binária T e uma tabela de pesos deste caminho, um algoritmo de geração e percurso de uma árvore auxiliar produz a sequência de nós que devem ser rebaixados de nível a fim de reduzir o valor de $E(C_T)$ ao mínimo.

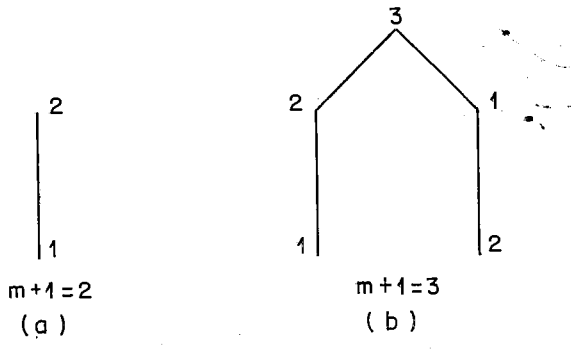
Por conveniência, numeramos de 1 até m+1 os nós do último caminho, sendo 1 associado a raiz e m+1 associado ao último nó incluído. Para distinguir da árvore binária, chamaremos de vértices aos nós desta árvore auxiliar. Representaremos estes vértices pelo número dos nós da árvore binária que baixam de nível, na ordem especificada a seguir. Por analogia, chama-se nível de um vértice ao número de arcos do caminho desde a raiz até cada vértice. A raiz tem nível 0.

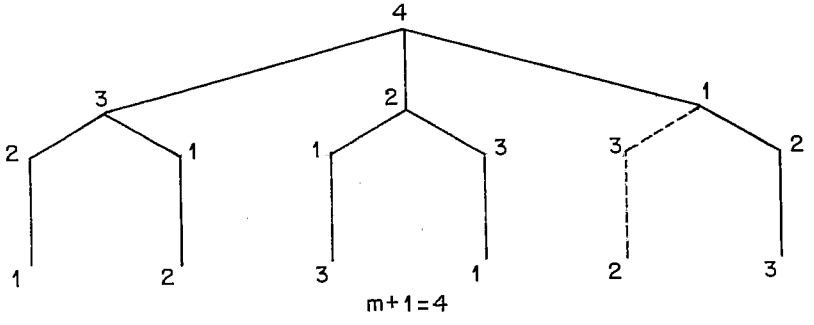
A sequência de enumeração dos vértices pode ser feita por nível, do seguinte modo:

1. a raiz tem número m+1 e possui m filhos no nível 1 da árvore auxiliar, que são numerados m, m-1, ... , 2, 1, da esquerda para a direita;
2. cada vértice v, v $\neq$ raiz, produz n_1-1 filhos, onde n_1 = nº de vértices do nível de v, na árvore auxiliar. Estes filhos são numerados da esquerda para a direita com o número dos irmãos de v, tomados da esquerda para a direita, a partir do 1º irmão a direi

ta de v, ciclicamente, isto é: se não houver mais irmãos à direita, continua-se a enumeração a partir do irmão mais a esquerda, até completar os n_1-1 filhos de cada vértice.

Por exemplo, temos as seguintes árvores auxiliares para 2, 3, 4 e 5 nós no caminho sob exame:





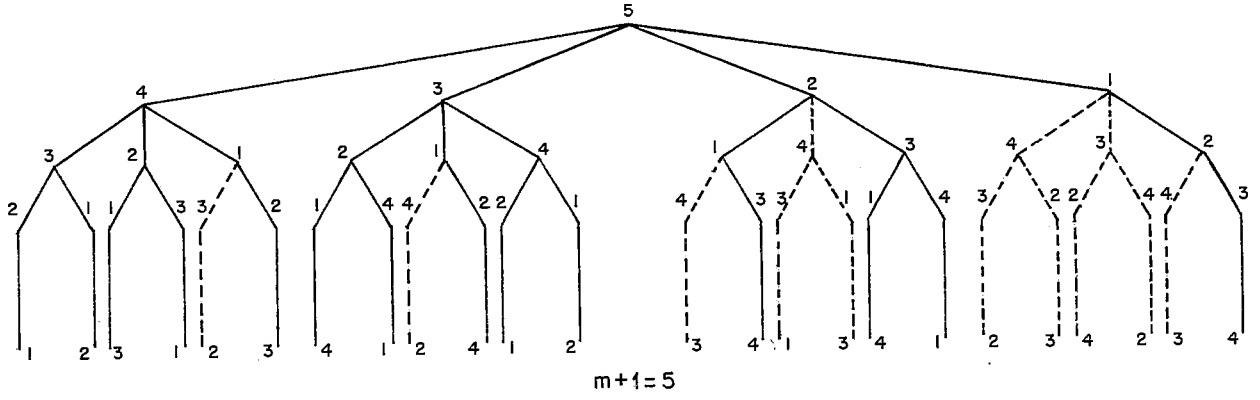


FIG. 11

Este tipo de enumeração assegura que as sequências de rotações iniciadas pelo nó de nº m serão as primeiras a serem examinadas na ordem adotada para visita dos vértices da árvore auxiliar. Este procedimento parece ser adequado vez que se pode admitir que os nós dos níveis mais baixos sofrem um maior desbalanceamento quando se faz a inclusão do (m+1)-ésimo nó.

Em consequência, o nº de vértices da árvore auxiliar é

$D = \sum_{r=0}^{m} A_m^r$, onde m = nº de nós do último caminho menos 1 e A_m^r = arranjos de m, r a r.

Então,

$$D = 1 + m + m(m-1) + m(m-1)(m-2) + \ldots + \underbrace{m(m-1)(m-2)\ldots}_{m-1 \text{ termos}} + \underbrace{m(m-1)(m-2)\ldots}_{m \text{ termos}}$$

O percurso da árvore auxiliar a fim de obter a sequência que reduz o valor de $E(C_T)$ ao máximo é feito em pré-ordem, qual seja: primeiro é visitada a raiz, depois a sub-árvore auxiliar da esquerda e em seguida a sub-árvore auxiliar da direita, recursivamente. Isto é feito com o auxílio de uma pilha, conforme programas anexos (Apêndice 6).

2.3.1. Enumeração Implícita e Retorno

À medida que m cresce o número de vértices da árvore auxiliar, D, torna-se muito grande — da ordem de m^m — o que implicaria em um consumo elevado de tempo para visitar todos os vértices na citada pré-ordem. Procurou-se, então, estabelecer um critério satisfatório para obter a sequência de promoções no último caminho que provoque a maior redução de $E(C_T)$, sem percorrer todos os vértices da árvore auxiliar. As definições, lemas e teoremas que se seguem provam como isto pode ser realizado.

Def.: 2.2 - $G(j) = D(j+1) - E(j)$, $j = 1, \ldots, m$ chama-se ganho obtido com o rebaixamento do nó j pelo nó j+1, em um caminho qualquer de uma árvore binária T.

Def.: 2.3 - $G_i(n_i)$ é o ganho obtido quando n_i é rebaixado após a execução das i-1 primeiras rotações.

Def.: 2.4 - Dada uma sequência de rotações $n_1, n_2, \ldots, n_k$, o ganho acumulado da sequência é dado por

$$F(n_1, n_2, \ldots, n_k) = \sum_{j=1}^{k} G_j(n_j)$$

Def.: 2.5 - Seja $n_1, n_2, \ldots, n_k$ uma sequência de nós a serem rebaixados. Diz-se que $n_1, n_2, \ldots, n_k$ é uma sequência de ganho máximo, ou sequência ótima, se $F(n_1, n_2, \ldots, n_k) \geq F(\underset{\sim}{s})$, onde $\underset{\sim}{s}$ é qualquer sequência de nós a serem rebaixados diferente de $n_1, n_2, \ldots, n_k$.

2.3.2. Princípio de Otimalidade da Sequência de Ganho Máximo

Lema 2.0: Se $n_1, n_2, \ldots, n_k$ é uma sequência de ganho máximo para o último caminho 1, 2, ... , m+1, então $n_2, n_3, \ldots, n_k$ é uma sequência de ganho máximo para o caminho resultante, de m nós após n_1 ser rebaixado.

Prova: Por hipótese, $n_1, n_2, \ldots, n_k$ é uma sequência de ganho máximo. Se $n_2, \ldots, n_k$ não for uma sequência de ganho máximo para o caminho de m nós significa que existe uma outra sequência $n_2', \ldots, n_k'$, de ganho máximo. Desta maneira, $F(n_1, n_2', \ldots, n_k') > F(n_1, n_2, \ldots, n_k)$ contrariando a hipótese. Logo, $n_1, n_2', \ldots, n_k'$ também é uma sequência de ganho máximo e então $n_2, \ldots, n_k$ é uma sequência de ganho máximo para o caminho resultante de m nós.

Lema 2.1: Se para toda sequência de ganho máximo $n_1, n_2, \ldots, n_k$ valer $G(n_1) > 0$, então $G_i(n_i) > 0$, $i = 2, \ldots, k$.

Prova: Do lema 2.0 se segue que $n_2, n_3, \ldots, n_k$ é uma sequência de ganho máximo para os m nós pertencentes ao último caminho após a rotação de n_1. Logo $G_2(n_2) > 0$. Aplicando novamente o lema 2.0, obtemos $G_3(n_3) > 0$ e assim sucessivamente até $G_k(n_k)$.

Teorema 2.4 - Se $G(j) \leq 0$ para $j = 1,2, \ldots,m$ então a sequência de ganho máximo é vazia.

Prova: Se $m = 2$, é válido porque $G(1) \leq 0$ e então a promoção não é viável. Admite-se que o teorema seja válido para $m \geq 2$. Consideremos agora um último caminho com m+1 nós tal que $G(j) \leq 0$ para $j = 1,2, \ldots,m$ e suponhamos que existe uma sequência de ganho máximo $n_1, n_2, \ldots, n_k$ não vazia neste novo caminho.

Sejam $\bar{E}(i)$ e $\bar{D}(i)$ os novos pesos no último caminho após a remoção de n_1 onde $n_1 = j$ para algum $1 \leq j \leq m$.

Então $\bar{D}(i) < \bar{E}(i+1)$, $i \neq j$ para o novo caminho $1,2, \ldots,j-1,j+1, \ldots,m+1$, de m nós, porque $G(j) \leq 0$. Consequentemente, $n_2, n_3, \ldots, n_k$ é vazia e como $G(j) = G(n_1) < 0$ chegamos a uma contradição, isto é, $n_1, n_2, \ldots, n_k$ é vazia.

A tabela a seguir mostra os pesos esquerdo e direito antes e depois do rebaixamento de j.

Tabela de Pesos 2.1

Nó	E	D	$\bar{E}$	$\bar{D}$
1	E(1)	D(1)	E(1)	D(1)
⋮				
j-1	E(j-1)	D(j-1)	E(j-1)	D(j-1)
j	E(j)	D(j)		
j+1	E(j+1)	D(j+1)	E(j)+E(j+1)	D(j+1)
⋮				
m	E(m)	D(m)	E(m)	D(m)
m+1	E(m+1)	D(m+1)	E(m+1)	D(m+1)

Def.: 2.6 - $F_i(e_1,e_2, \ldots,e_{m-i})$ = ganho máximo para um caminho constituído pelos nós 1,2, ...,m-i após serem rebaixados os nós $n_1,n_2, \ldots,n_i$. O conjunto $\{e_1,e_2, \ldots,e_{m-i}\}$ = $\{1,2, \ldots,m\} - \{n_1,n_2, \ldots,n_i\}$, ou seja, os n's são rebaixados, os e's são os nós promovidos e que permaneceram no caminho.

Teorema 2.5 - Se $n_1,n_2, \ldots,n_k$ é uma sequência ótima (não vazia) para um caminho de m nós então $G(n_1) > 0$.

Prova: Se $m = 2$, então $G(n_1) > 0$ e o teorema é válido. Suponhamos que vale também para $m \geq 2$ e consideremos um caminho com m+1 nós e uma sequência ótima não vazia $n_1, \ldots,n_k$ tal que

$G(n_1) < 0$, $n_1 = j$ para algum $1 \leq j \leq m$.

A tabela resultante da remoção de j será:

Tabela de Pesos 2.2

Nó	E	D
1	E(1)	D(1)
⋮		
j-1	E(j-1)	D(j-1)
j+1	E(j)+E(j+1)	D(j+1)
j+2	E(j+2)	D(j+2)
⋮		
m+1	E(m+1)	D(m+1)

Como, por hipótese, a sequência $n_1, n_2, \ldots, n_k$ é ótima, o princípio de otimalidade garante que $n_2, \ldots, n_k$ é ótima e pelo lema 2.1, nenhuma de suas promoções dá perda. Logo, como $G(j) < 0$ e $D(j) \geq D(j+1)$, $j = 1, \ldots, m$ então $D(k) \leq E(j)+E(j+1)$, $k \geq j+2$. Em consequência, nenhuma promoção da sequência $n_2, \ldots, n_k$ é do tipo $j+1 \leftarrow k$, $k \geq j+2$.

Temos, então:

$$F_1(1,2,\ldots,j-1,j+1,\ldots,m+1) = F_1(1,2,\ldots,j-1,j+1)+F_1(j+2,\ldots,m+1)$$

Porém:

$F_1(1,2,\ldots,j-1,j+1) \leq F(1,2,\ldots,j)$ porque $D(i) > D(i+1)$, para $i = 1,2,\ldots,m$, e

$F_1(j+2,\ldots,m+1) = F(j+2,\ldots,m+1)$

Logo,

$G(j) + F_1(1,2,\ldots,j-1,j+1,\ldots,m+1) < F(1,2,\ldots,j)+F(j+2,\ldots,m+1)$.

Encontrou-se então uma sequência que não rebaixa j e dá ganho acumulado maior do que $n_1,n_2,\ldots,n_k$, o que é uma contradição. Logo, $G(n_1) \geq 0$.

Os teoremas 2.4 e 2.5 acima permitem podar a árvore auxiliar quando se encontra uma rotação em que $G_j(j) < 0$, evitando o prosseguimento do exame de sequências em que há alguma promoção não viável.

Pelo visto nos teoremas 2.2 e 2.3 podemos decidir se uma rotação é viável ou não com base nos pesos da esquerda e da direita de cada nó, dispensando o armazenamento e cálculo dos W(i). Desta forma, cada nó da árvore binária tem os seguintes campos:

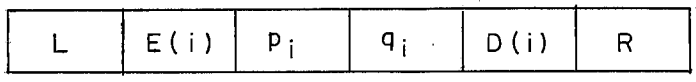


FIG.12

onde L é o apontador para a raiz da sub-árvore esquerda, R é o apontador para a raiz da sub-árvore direita e p_i, q_i, E(i) e D(i)

tem o significado definido anteriormente.

Na implementação do algoritmo a sequência de maior ganho acumulado é sempre guardada até surgir uma sequência cujo ganho acumulado seja maior.

2.3.3. Promoções Redundantes

Dado um último caminho de m+1 nós, associados a numeração 1,2,...,m+1, a partir da raiz, seja uma sequência qualquer $n_1, n_2, \ldots, n_k$, dos nós rebaixados. Associemos a cada nó rebaixado um nó t_i, $i = 1,2,\ldots,k$ onde $n_i \leftarrow t_i$, isto é, t_i foi promovido para o lugar de i, formando a sequência dos nós promovidos para o lugar dos n_i, $t_1, t_2, \ldots, t_k$.

Def.: 2.3 - Dizemos que uma promoção $n_j \leftarrow t_j$ é redundante se $n_j \neq t_{j-1}$ e $n_j > n_{j-1}$, de acordo com a numeração acima citada.

Por exemplo, na fig. 13, a sequência $1 \leftarrow 2$, $3 \leftarrow 4$

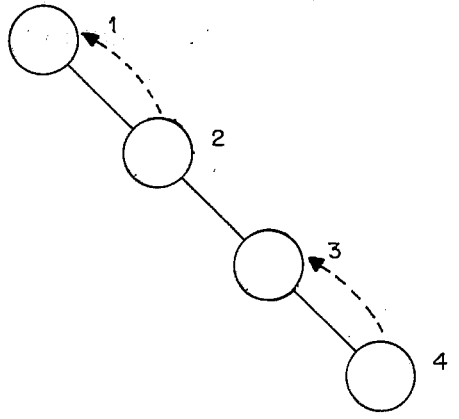


FIG.13

mostra que 3 ← 4 é redundante.

Obviamente, chamaremos sequência redundante qualquer sequência que contenha pelo menos uma promoção redundante.

Observa-se que se uma sequência é redundante, existe uma outra sequência não redundante já examinada em pré-ordem cujo ganho acumulado equivale ao da sequência redundante. No exemplo anterior, F(1,3) = D(2) - E(1) + D(4) - E(3) e F(3,1) = D(4) - E(3) + D(2) - E(1).

É claro que a eliminação destas sequências não impede que sejam examinadas todas as árvores obtidas pelas transformações no último caminho, porque existe outra sequência não-redundante a partir da qual se obtem a mesma árvore binária que seria indicada pela sequência de rotações redundante.

As afirmações acima estabelecem um segundo critério de poda dos ramos da árvore auxiliar, tornando mais rápida a execução deste algoritmo, vez que todos os vértices que emanam de um vértice representativo de uma promoção redundante nunca serão visitados.

2.4. Atualização da Árvore

Obtida a sequência de ganho máximo, faz-se a atualização da árvore, executando-se as rotações indicadas pela sequência e modificando os pesos correspondentes.

Por exemplo, no último caminho abaixo - fig. 14 - T é transformada em $\hat{T}$ após a atualização indicada pela sequência (23,18,11), correspondente a sequência (5,4,2) de acordo com a numeração associada a partir da raiz.

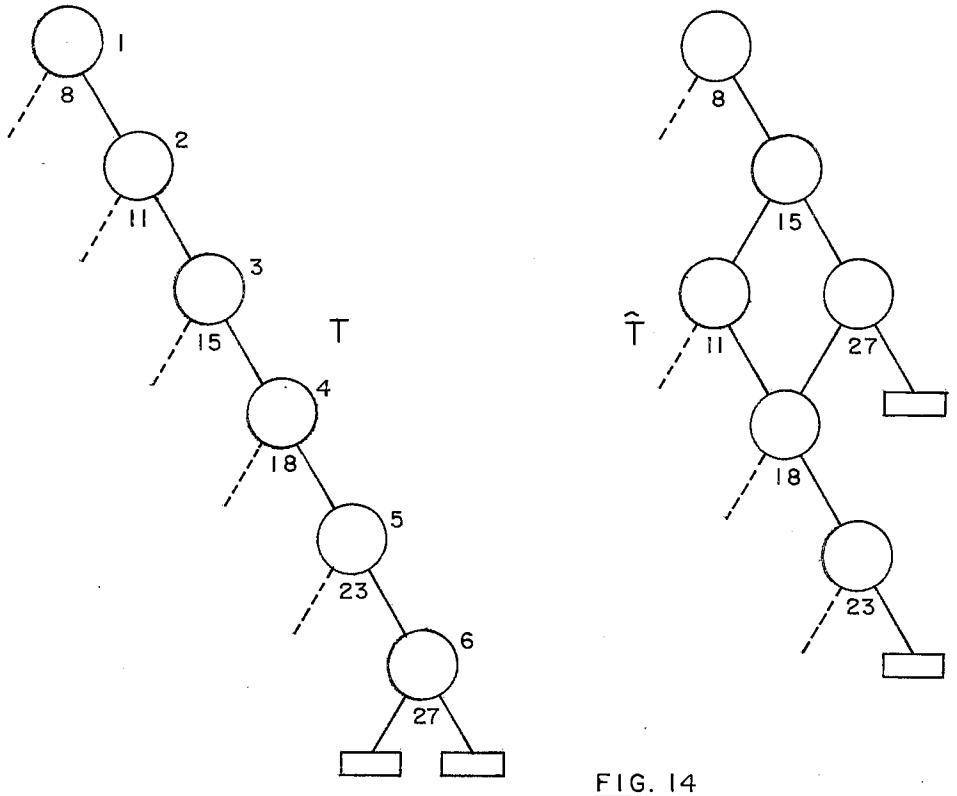


FIG. 14

Surge agora um tipo de rotação não aplicado no último caminho.

Seja a árvore da fig. 15, sem peso nas folhas, onde a ordem lexicográfica é alfabética e seja (G) a sequência ótima do caminho ACGIJ, cujos pesos se encontram na tabela 2.3.

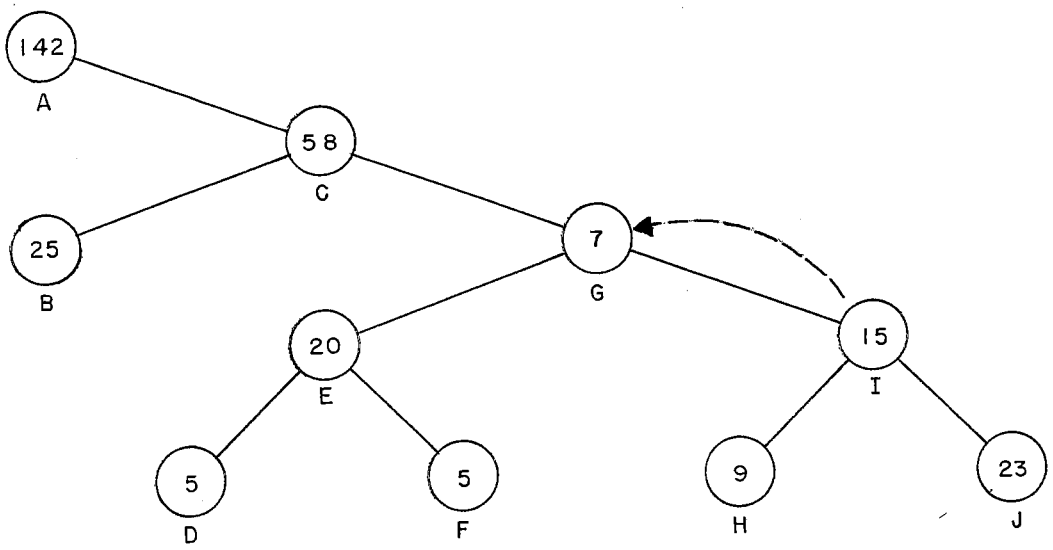


FIG.15

Tabela de Pesos 2.3

Nó	E	D
A	142	319
C	93	142
G	37	54
I	24	38
J	23	23
E	25	25

Quando se faz a atualização indicada pela sequência, através do rebaixamento do nó G, temos:

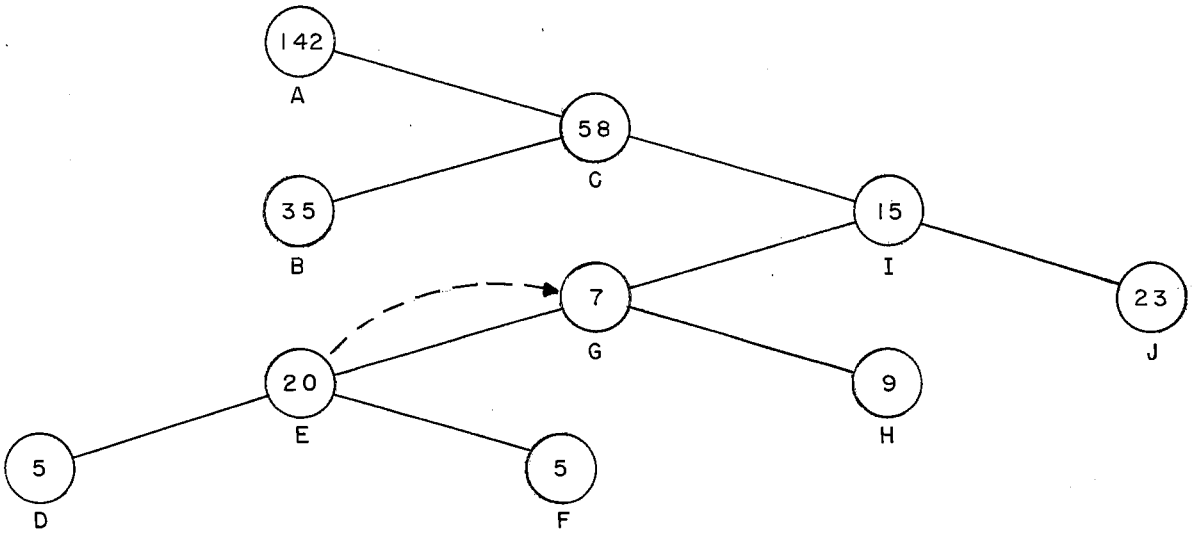


FIG. 16

Tabela de Pesos 2.4

Nó	$\underline{E}$	D	$\bar{E}$	$\bar{D}$
A	142	319		
C	93	142		
I	61	38	$\bar{E} = \underline{E}$	
J	23	23	$\bar{D} = D$	
G	37	16	12	16
E	25	25	25	41

A árvore obtida é:

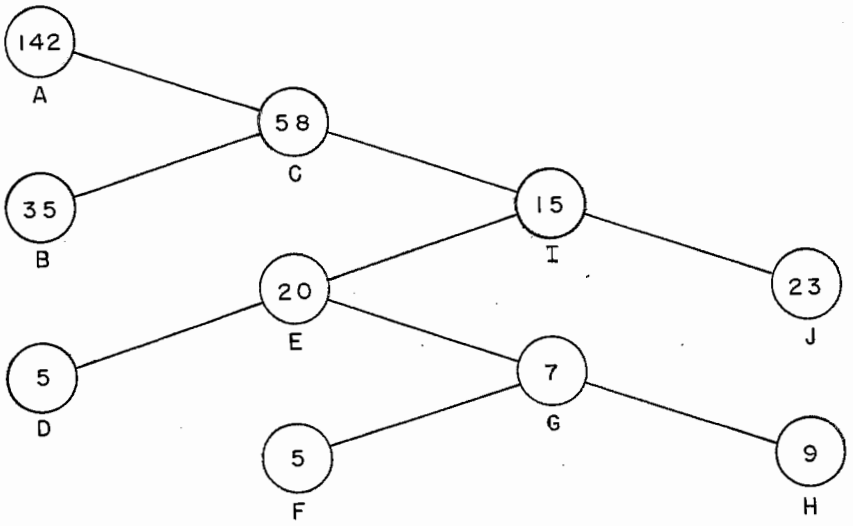


FIG. 17

Trata-se, pois, de uma promoção para a direita que é testada após cada rotação no processamento da sequência de ganho máximo.

No apêndice 4 mostra-se um exemplo completo da execução deste algoritmo.

3. ALGORITMO II - INSERÇÃO EM ORDEM ARBITRÁRIA

Ao contrário do algoritmo anterior, em que a inclusão é feita obedecendo a ordem lexicográfica, os algoritmos que se seguem permitem a inserção em uma ordem arbitrária.

As principais diferenças em relação ao primeiro algoritmo são:

1. inserção dos nós na árvore binária de acordo com 2 critérios heurísticos (peso $q_{i-1}+p_i+q_i$ e centróide), a serem descritos no parágrafo 3.3;
2. aplicação do algoritmo de geração e percurso da árvore auxiliar em qualquer caminho de T.

3.0. Descrição

Seja z o 1º nó a ser incluído na árvore. Faz-se a inserção de z junto com p_z, q_{z-1} e q_z, de acordo com a figura 18.

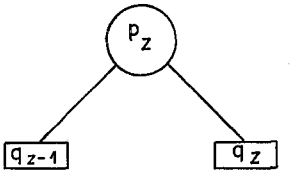


FIG.18

A inserção de outros nós $i \neq z$ é feita através de uma pesquisa binária na árvore, da seguinte forma:

a. se $i < z$, então é feita a inserção de p_i+q_{i-1}

b. se $i > z$, então é feita a inserção de p_i+q_i.

Após a inserção de um nó na árvore binária ocorre um aumento do valor de $W(L(v))$ ou de $W(R(v))$, onde v é um nó pertencente ao caminho onde foi feita a inserção e o nó incluído i, associado ao número m+1. Temos então, os seguintes casos para o cálculo dos novos valores de $W(R(v))$ e $W(L(v))$ e, consequentemente, dos $E(j)$ e $D(j)$:

a. $i < z$

   1. $i < v$

      $\bar{W}(L(v))=W(L(v))+p_i+q_{i-1}$, $\bar{W}(R(v)) = W(R(v))$

   2. $i > v$

      $\bar{W}(L(v)) = W(v)$, $\bar{W}(R(v)) = W(R(v)) + p_i + q_{i-1}$

b. $i > z$

   1. $i < v$

      $\bar{W}(L(v)) = W(L(v)) + p_i + q_i$, $\bar{W}(R(v)) = W(R(v))$

   2. $i > v$

      $\bar{W}(L(v)) = W(L(v))$, $\bar{W}(R(v)) = W(R(v)) + p_i + q_i$.

É óbvio que para todo $j \notin$ ao caminho citado, $\bar{W}(L(j)) = W(L(j))$ e $\bar{W}(R(j)) = W(R(j))$.

A figura a seguir ilustra a inserção do nó I.

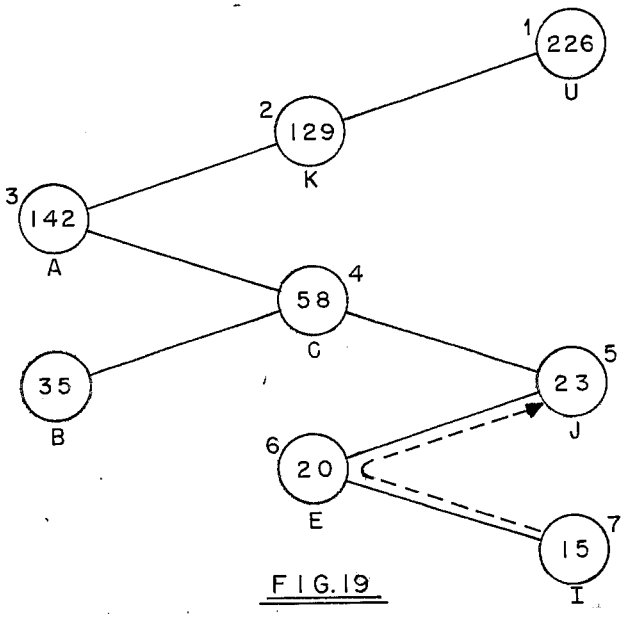


FIG.19

Em consequência, e a exemplo do algoritmo anterior, utilizamos uma tabela de pesos E(j) e D(j), j = 1,...,m+1, que é atualizada de acordo com os casos acima. Os procedimentos de geração e percurso da árvore auxiliar são semelhantes.

3.1. Geração e Percurso da Árvore Auxiliar

O contra-exemplo a seguir nos mostra que não se pode aplicar a poda na árvore auxiliar a partir do primeiro $G_j(j) \leq 0$.

Tabela de Pesos 3.0

NÓ	E(i)	D(i)	$\hat{E}(i)$	$\hat{D}(i)$
u - 1	756	636		
k - 2	422	237		
a - 3	142	293	$\hat{E}=D$	$\hat{D}=\bar{D}$
c - 4	93	116		
j - 5	58	23	58	23
e - 6	20	35	5	35
i - 7	15	15	35	15

$F(6,5) = +7$ porque $G(6) = -5$ e $G(5) = 12$

Encontramos, deste modo, um novo tipo de sequências de rotações viáveis, qual seja, uma sequência com duas rotações em direção diferente, com $G(j) < 0$ na 1a. rotação e ganho acumulado positivo.

Teorema 3.0 - Seja um caminho dado pela fig. 19 e sua tabela de pesos:

Tabela de Pesos 3.1

NÓ	E	D
j	E(j)	D(j)
j+1	E(j+1)	D(j+1)
j+2	E(j+2)	D(j+2)

Se $D(j+2) < E(j+1)$ e $E(j+2) + D(j+2) > D(j)$, então a sequência $j+1 \leftarrow j+2$, $j+2 \rightarrow j$ tem ganho acumulado positivo para $j+2 \leqslant m+1$ e $j \geqslant 1$.

Prova: Basta efetuar a sequência de promoções e verificar a condição sob a qual o ganho acumulado é positivo.

O ganho acumulado $F(j+1, j)$ é:

$$D(j+2) - E(j+1) + E(j+1) + E(j+2) - D(j) = D(j+2) + E(j+2) - D(j).$$

Então, se $D(j+2) + E(j+2) > D(j)$, $F(j+1,j) > 0$.

A condição $D(j+2) < E(j+1)$ serve apenas para mostrar que a sequência de rotações pode conter promoções com $G_i(n_i) \leqslant 0$.

Adotou-se, heuristicamente, com base no teorema anterior, o seguinte critério de poda:

1. se uma promoção para a esquerda não é viável, e a próxima rotação a ser examinada é para a direita, tenta-se a rotação para a direita. Se o ganho acumulado for negativo poda-se o ramo da árvore auxiliar;
2. se uma promoção para a direita não é viável e a próxima rotação a ser examinada é para a esquerda, tenta-se as duas rotações seguidas. Se o ganho acumulado for negativo, poda-se o ramo da árvore auxiliar.

Analogamente, prova-se que se $E(j+2) < D(j+1)$ e $E(j+2) + D(j+2) > E(j)$ então $j+2 \rightarrow j+1$, $j \leftarrow j+2$ dá ganho acumulado positivo, conforme a figura abaixo:

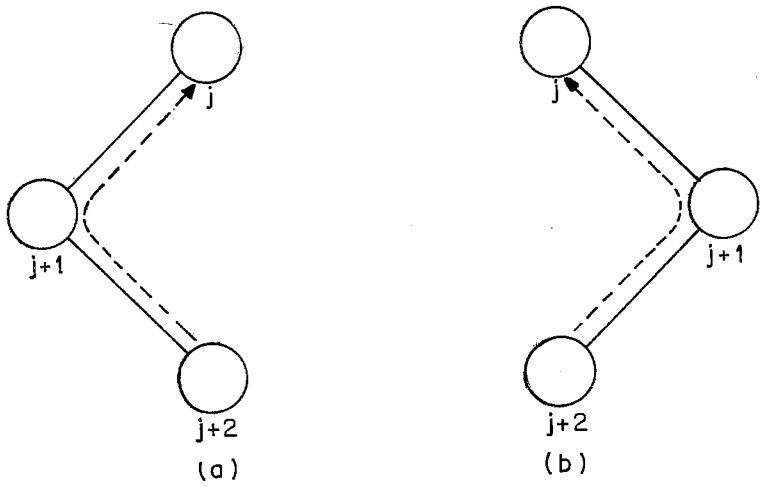


FIG. 20

É claro que se as duas rotações seguidas tem ganho acumulado positivo prossegue-se o exame da sequência, a exemplo do caso em que a rotação única dá ganho. No caso em que a próxima rotação é no mesmo sentido da anterior e esta dá perda, então poda-se a árvore auxiliar naquele ramo.

3.2. Atualização da Árvore Binária

Como as rotações da sequência obtida podem ser tanto para a esquerda como para a direita, torna-se necessário testar esta condição. A cada rotação efetuada, faz-se uma otimização local para realizar as promoções laterais do tipo mostrado na fig. 15. Neste algoritmo, tais promoções poderão ser para a direita ou para a esquerda.

3.3. Descrição e Justificativa da Ordem de Inserção

Observações citadas por Walker e Gotlieb |14| indicam que a árvore binária ótima de busca depende de 2 fatores: colocação próximo da raiz dos nós e folhas com maior probabilidade de consulta e do balanceamento das sub-árvores da esquerda e direita de cada nó.

Por exemplo:

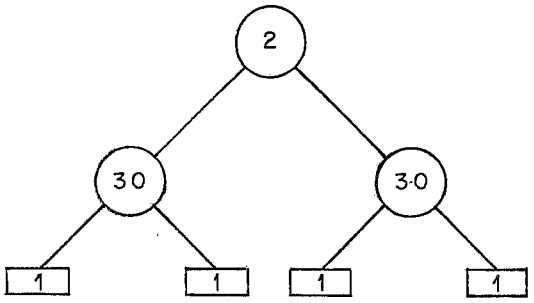


FIG.21

É uma árvore ótima de busca.

Estas colocações nos induziram a utilizar dois critérios de ordenação para inclusão de nós e folhas na árvore binária:

a. inserção por ordem decrescente da variável

$q_{i-1}+p_i+q_i$, $i=1,\ldots,N$

b. construção de uma árvore binária inicial pelo método dos centróides e ordenação de acordo com a numeração da fig. 22.

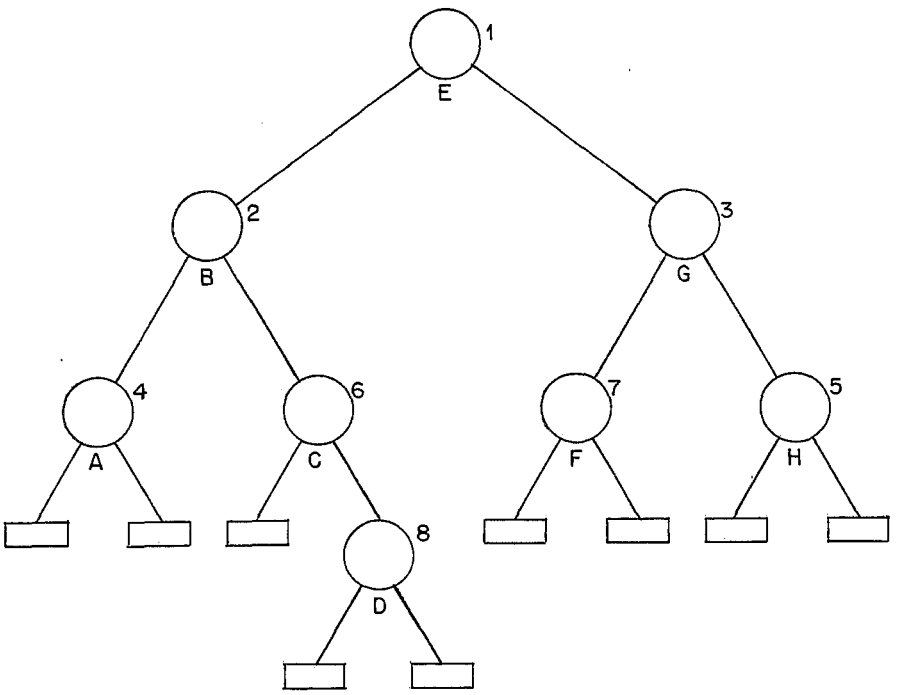


FIG. 22

Observa-se que neste caso se inserem os nós, nível por nível, a partir da raiz. Dentro de cada nível, ordenam-se os nós tomando, alternadamente, um nó da esquerda e um da direita da raiz, até completar o último nó do nível.

É lógico admitir que no 1º método de inserção, a formação dos níveis mais próximos da raiz ocorre logo no início das inserções, contribuindo para a obtenção mais rápida da sequência de ganho máximo.

4. ALGORITMO ADAPTATIVO

4.0. Descrição

Na maioria das situações práticas, não se conhece a distribuição das probabilidades de consulta aos nós e folhas, tornando necessária a construção de um algoritmo em que se realizem as rotações necessárias à medida que as chaves foram consultadas, a fim de manter a otimalidade da árvore.

Este algoritmo adaptativo tem por base o princípio de consulta e inserção utilizado no capítulo 3, e testado apenas para distribuições com peso nulo nas folhas. Seus princípios podem ser enunciados assim:

Dadas N chaves ($K_1 < K_2 < .... < K_N$) na primeira vez que uma chave é consultada faz-se a sua inclusão na árvore binária. Sempre que se faz consulta a uma determinada chave, o seu campo p_i, de frequência de consulta, é incrementado de 1. Em ambos os casos, atualizam-se os pesos dos nós pertencentes ao caminho que liga a raiz ao nó consultado. Executa-se, em seguida, o algoritmo de geração e percurso desde o nó consultado até a raiz, determinando a sequência de ganho máximo. Fazem-se as rotações indicadas pela referida sequência. Feita a última atualização da árvore, uma visita de seus nós permite calcular o custo esperado da

árvore.

4.1. Geração de Consultas

Utilizaram-se geradores randômicos de tal forma que a distribuição de consultas seja exponencial. Aplicamos o método da transformação inversa|3,10|.

4.2. Atualização e Inserção

Dada uma chave K fornecida pelo gerador randômico, este procedimento busca se K pertence a árvore binária (Se a árvore for vazia, insere-se o nó consultado). Durante a busca, atualiza os pesos esquerdo ou direito dos nós do caminho percorrido, conforme o caso. Ao final da busca, se houve sucesso incrementa-se de 1 o contador de consultas ao nó. Se houve insucesso, inclui-se a chave K na árvore, à direita ou à esquerda do último nó visitado, de acordo com a ordem das chaves.

4.3. Árvore Auxiliar

Os algoritmos de geração e percurso da árvore auxiliar e o de atualização da árvore binária funcionam de modo idênti

co ao segundo algoritmo apresentado. A árvore auxiliar é gerada a partir do nó consultado ou incluído até a raiz. Desta forma, se temos um caminho com m nós e foi gerada uma consulta para o nó j, $1 \leqslant j \leqslant m$, então as rotações dos nós j+1,..., m não são examinadas já que se houvesse alguma rotação viável para k = j+1,..., m faria parte de uma sequência já processada.

4.4. Algoritmo Adaptativo Modificado

Observa-se que a visita por enumeração implícita e em pré-ordem dos vértices de promoções entre o nó referenciado e a raiz pode abranger elevado número de vértices degradando o tempo de execução quando o nó referenciado está nos níveis mais baixos da árvore binária. Sabe-se que cada referência aumenta apenas de 1 o peso direito ou esquerdo de cada nó no caminho visitado, indicando a viabilidade de uma maneira mais simples para otimização do caminho referenciado.

Uma política que pode ser examinada seria a seguinte:

Dados 1, 2, ..., m — numeração dos nós desde a raiz 1 até ao nó referenciado m — executar os seguintes passos:

1. se m = 1, parar; caso contrário, tentar rotação simples com o nó de número m. Se houver ganho, realizar a rotação simples e

voltar ao passo 1;

2. tentar rotação dupla de m. Se houver ganho, concretizar a rotação, fazer m = m-2 e voltar ao passo 1. Se não houver ganho, m=m-1 e voltar ao passo 1.

Neste caso, o máximo número de rotações é m-1. O máximo número de nós examinados ocorre quando nenhum ganho é possível no caminho e é 2m - 3, qual seja, o número de rotações duplas possíveis. A subtração de 3 é causada pelo fato de a raiz e seu filho não poderem receber dupla rotação. A redução de tempo provavelmente será considerável, dependendo da distribuição de probabilidade das consultas geradas, sem grande prejuízo no valor esperado do custo.

A implementação deste algoritmo e análise comparativa dos resultados poderá dar origem a uma futura pesquisa.

5. ANÁLISE DE CASOS ESPECIAIS

Existem distribuições de probabilidade especiais que podem tornar mais lentos ou mais rápidos os algoritmos apresentados. Casos típicos são examinados em seguida.

5.0. Algoritmo I

Peso nulo nas folhas

- Caso mais favorável

Seja $p_j > p_{j-1} + p_{j-2} + \ldots + p_2 + p_1$, $j=2, \ldots, N$.
A árvore ótima obtida é uma lista linear com raiz p_N.

Neste caso, produz-se uma lista linear em tempo proporcional a N, porque o último caminho sempre terá apenas 2 nós após cada inclusão e 1 nó após o processamento da sequência ótima. Visita-se apenas 2 vértices da árvore auxiliar para cada nó inserido.

- Caso mais desfavorável

Seja $p_j > p_{j+1} + p_{j+2} + \ldots + p_{N-1} + p_N$, $j=1, \ldots N-1$.

A árvore ótima é obtida e consiste de uma lista linear com raiz p_1.

Sabe-se que todos os nós incluídos permanecem no último caminho e $D(i) < E(i-1)$, $i=2,\ldots, m+1$. Sendo $m+1$ o nº de nós do último caminho, a constatação de que a sequência ótima é vazia consome a visita da raiz e dos m vértices do primeiro nível da árvore auxiliar e mais uma nova visita ao 1º vértice, totalizando $m+2$ visitas. Então, o tempo de execução é da ordem de N^2.

Peso não nulo nas folhas

- Caso mais favorável

$$\text{Seja } p_j + q_j > \sum_{i=j-1}^{0} q_i + \sum_{k=j-1}^{1} p_k,\ j=2,\ \ldots,\ N.$$

É obtida uma lista linear, que é a árvore ótima. A raiz é o N-ésimo nó. Verifica-se que após cada inclusão o último caminho possui apenas 2 nós. Com a visita do único vértice da árvore auxiliar determina-se a sequência ótima. Logo, o tempo de execução é proporcional a N.

- Caso mais desfavorável

$$\text{Seja } p_j + q_j > \sum_{i=j-1}^{1} p_i + \sum_{k=i}^{0} q_k,\ j=1,\ \ldots,\ N$$

A exemplo do caso mais desfavorável, anterior, a árvore ótima é obtida e coincide com uma lista linear. A raiz da árvore é o nó de ordem 1. A estrutura binária ótima é produzida em tempo $O(N^2)$.

5.1. Algoritmo II

- Casos mais desfavoráveis

Seja $p_j > p_{j+1} + p_{j+2} + \ldots + p_N$ $(j=1,\ldots,N-1)$ ou

$p_j > p_{j-1} + p_{j-2} + \ldots + p_1$ $(j=2,\ldots,N)$.

Em ambos os casos o algoritmo consome $O(N^2)$ unidades de tempo e constrói uma árvore ótima, que no caso é uma lista linear. A raiz é p_1 no primeiro caso acima e p_N no segundo. Após a k-ésima inserção, o único caminho existente na árvore binária terá k nós, de forma que a execução consome tempo aproximadamente igual a $O(\sum_{k=1}^{N} (k) = O(N^2)$, em ambos os casos.

6. TESTES E RESULTADOS

Para fins de testes dos algoritmos foram elaborados os programas constantes do Apêndice 6, escritos em ALGOL-W|13| e rodados no computador IBM-/370-65, da PUC/RJ.

Teste 0

Tem como objetivo comparar os resultados obtidos pelo "Algoritmo GW" com o dos algoritmos I e II, de acordo com o quadro 6.0.

Quadro 6.0

Síntese do Teste

ALGORITMO	MÉDIA DA RELAÇÃO $E(C_T)/E(C_{T*})$	DESVIO PADRÃO DA RELAÇÃO $E(C_T)/E(C_{T*})$	COEFICIENTE DO MOMENTO DE ASSIMETRIA
I	1,017	0,008	0,492
GW	1,013	0,012	1,269
II-Centróide	1,012	0,007	- 0,160
II-Peso Decrescente	1,008	0,005	0,681

onde T* é a árvore ótima e o coeficiente de momento de assimetria da relação $R = E(C_T)/E(C_{T*})$ é dado por $(\sum_{i=1}^{n}(R-\bar{R})^3)/\sigma^3$, sendo $\bar{R}$ a média aritmética de R e σ o seu desvio padrão.

Ve-se no quadro anterior que as árvores de melhor qualidade são obtidas pelo Algoritmo II, com inserção pelo valor decrescente de $q_{i-1} + p_i + q_i$, porque a média da relação $E(C_T)/(E(C_{T*})$ é de 1,008. O desvio padrão de 0,005 indica a maior concentração em torno da média em relação aos outros casos, e o coeficiente de momento de assimetria mostra que a dispersão está acima da média.

Na inserção pelo método do centróide foi contado o número máximo de vértices por cada inclusão e o número total de vértices visitados nas árvores auxiliares. Com base nos valores obtidos, pode-se dizer que há indícios de que a sequência ótima encontrava-se, na maioria dos casos, no caminho da raiz ao vértice mais à esquerda da árvore auxiliar indicando adequação na geração e sequência de visita da árvore auxiliar. Deste modo o percurso em pré-ordem por enumeração implícita visitaria apenas o caminho desde à raiz até ao vértice mais à esquerda e os vértices do nível 1 da árvore auxiliar, tornando rápida a obtenção da sequência ótima. A fig. 23 ilustra o percurso.

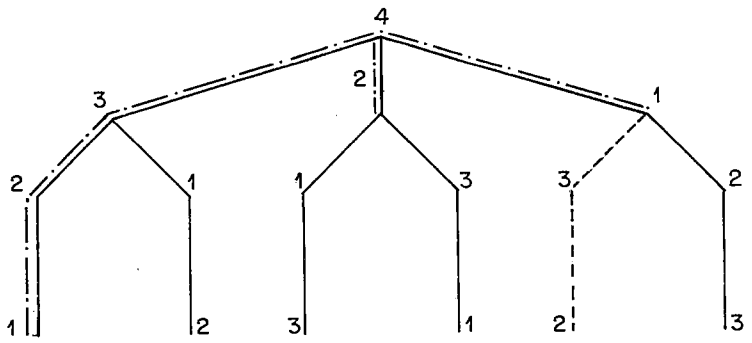


FIG.23

Os apêndices 1, 2 e 3 mostram os dados e resultados utilizados para comparação dos valores obtidos por Walker & Gotlieb |14| e os alcançados pelos algoritmos desenvolvidos neste trabalho.

Em função dos resultados deste teste, passou-se a examinar o comportamento do algoritmo II com inserção em ordem decrescente do valor de $q_{i-1} + p_i + q_i$.

Teste 1

Serve para analisar o comportamento do tempo de execução em função do número de nós.

Quadro 6.1

Síntese do Teste

Nº DE NÓS	TEMPO DE EXECUÇÃO POR NÓ (MILISSEGUNDOS)		$E(C_T)$	
	I	II	I	II
64	1,72	2,34	5,77	5,69
128	1,96	2,26	6,81	6,51
256	1,91	2,50	7,68	7,60
512	2,35	2,57	8,73	8,55
1024	2,30	2,76	9,71	9,56
2048	2,33	2,87	10,68	10,52
4096	2,39	3,06	11,66	11,59
8180	2,42	-	12,69	12,59

As frequências de nós e folhas foram geradas segundo uma distribuição exponencial de média 100, a partir de um valor inicial IX=7.

Verifica-se, observando os gráficos do apêndice 5 e o Quadro 6.1, que:

a. o tempo de execução do primeiro e do segundo algoritmo são da ordem de $N \log_2 N$;
b. pela inclinação das retas do gráfico 5.3 podemos concluir que os algoritmos I e II são assintoticamente mais rápidos que o al

goritmo GW.

Teste 2

Serve para testar o comportamento dos algoritmos para exemplos apresentados por Knuth |8,9|.

Quadro 6.2

Síntese do Teste (*)

Nº DO EXEMPLO	Nº.	$E(C_T^*)$	RELAÇÃO $E(C_T)/E(C_{T^*})$	
			I	II
1	35	2,991	1,001	1,000
2	36	3,238	1,000	1,000
3	35	4,800	-	1,000
4	36	4,722	-	1,017

Nos exemplos 1 e 2, acima foram considerados pesos nas folhas. Nos exemplos 3 e 4, $q_i = 0$, i=0, ...N.

(*) A distribuição de frequências dos exemplos 1 e 2 foram retiradas da referência |8|, pág. 437. A dos exemplos 2 e 4 foram obtidas na referência |9|, pág. 20.

No quadro 6.2, vemos:

a. um exemplo em que uma árvore ótima é produzida pelos algoritmos I e II (Ex. 2);
b. um exemplo no qual a árvore construída por I é semi-ótima e a construída por II é ótima (Ex. 1);
c. um exemplo em que a árvore constituída pelo segundo algoritmo não é ótima (Ex. 4).

Teste 3

Serve para analisar o comportamento do Algoritmo 1 para p_i, i=1, ..., N. escolhidos aleatoriamente segundo uma distribuição exponencial e $q_i = 0$, i=0, ..., N.

Quadro 6.3

Síntese do Teste

Nº DO EXEMPLO	N	$E(C_{T*})$	RELAÇÃO $E(C_T)/E(C_{T*})$	MÉDIA DA EXPONENCIAL
1	50	3,896	1,000	100
2	50	3,897	1,000	80
3	50	3,924	1,003	100
4	60	4,146	1,000	100
5	60	4,145	1,000	80
6	60	4,184	1,007	100

Observa-se que o primeiro algoritmo produz árvores semi-ótimas bastante satisfatórias quando N é pequeno (abaixo de 100 nós) já que foram obtidas árvores ótimas nos exemplos 1, 2, 4 e 5 acima e nos exemplos 3 e 6 a relação $E(C_T)/E(C_T*)$ é de 1,003 e 1,007, respectivamente.

Teste 4

Aplica o algoritmo 1 para as seguintes distribuições de frequência de consulta:

a. Progressão Aritmética, com

$p_1 = 2000$, $p_i = p_{i-1} - x$, $i=2, \ldots, 60$

$x = 20$ $\quad q_i = 0$, $i = 0, \ldots, 60$

b. Progressão Aritmética, com

$p_1 = 2000$ $\quad p_i = p_{i-1} - x$, $i=2, \ldots, 60$

$x = 30$, $\quad q_j = 0$, $j=0, \ldots, 60$

c. Progressão Geométrica, com

$p_1 = 2000$ $p_i = p_{i-1} \cdot r^{i-1}$, $i=2, \ldots, 60$, $r = 0,8$

d. Progressão Geométrica, igual ao caso c, acima, com $r = 0,9$

e. Progressão Geométrica, igual ao caso c, acima, com $p_1 = 8000$

Quadro 6.4

Síntese do Teste

TESTE	$E(C_{T*})$	$E(C_T)/E(C_{T*})$
a	5,026	1,009
b	3,924	1,003
c	3,049	1,000(*)
d	3,067	1,000(*)
e	4,897	1,001

(*) A árvore obtida é semi-ótima; a diferença $E(C_T) - E(C_{T*})$ é insignificante.

Teste 5

Consta dos testes efetivados com o Algoritmo Adaptativo, resumidos no quadro a seguir.

Quadro 6.5

Algoritmo Adaptativo

Síntese do Teste

MÉDIA DA EXPONENCIAL	$E(C_T)$		RELAÇÃO A:B
	ADAPTATIVO (A)	ÓTIMO (B)	
10	4,083	4,018	1,016
20	4,959	4,893	1,014
25	5,263	5,139	1,023

Observa-se no quadro acima, construído para N = 128 e 30001 consultas, que o pior resultado dá uma diferença de 2,3% do custo esperado ótimo. O tempo de execução observado parece ser crescente com a média da distribuição de probabilidade de consulta aos nós.

Um último teste foi efetivado para N = 128 e 4001 consultas, sendo obtidos os custos esperados de 5,386 e 5,624, para números randômicos exponenciais com média 30 e 40, respectiva-

mente.

Os resultados da página anterior indicam que há um aumento do custo esperado em função da elevação da média da exponencial.

7. CONCLUSÕES

Os algoritmos apresentados parecem atrativos para construção de árvores de busca por causa dos seguintes aspectos:

- o primeiro porque é rápido e simples;
- o segundo permite construção e otimização de árvore numa ordem arbitrária de inserção de chaves;
- o adaptativo permite construir árvores semi-ótimas sem conhecimento prévio algum sobre as probabilidades de consulta aos nós e folhas;
- cada um dos algoritmos consome memória proporcional a N e tempo proporcional a $N \log_2 N$;
- os algoritmos implementados apresentam "performance" superior à existente na literatura, tanto com relação ao custo da árvore quanto ao tempo de execução.

Finalmente, registrem-se alguns assuntos correlatos tais como analisar o comportamento do custo esperado quando a inserção é feita em ordem crescente da variável $q_{i-1} + p_i + q_i$, implementação do algoritmo alternativo apresentado no parágrafo 4.4, tentativa de criação de algoritmos $O(N \log_2 N)$ e $O(N)$ de tempo e memória, respectivamente, para construção da árvore ótima, os quais poderão servir de tema de pesquisa no campo das árvores de busca.

8. APÊNDICES

APÊNDICE 0 |4|

Algoritmo K

Árvores Binárias de Busca Ótimas

Inicialização

```
FOR i ← 0 STEP 1 UNTIL N DO
    BEGIN C_{i,i} ←0;  W_{ii} ← q_i
          FOR j ←i+1 STEP 1 UNTIL N DO
                W_{i,j} ← W_{i,j-1} + p_j + q_j;
          END;
```

```
COMMENT: Árvores Iniciais - Encontra todas as árvores ótimas com a
         penas 1 nó;
    FOR j ← 1 STEP 1 UNTIL N DO
    BEGIN C_{j-1,j} ← W_{j-1,j}
          r_{j-1,j} ←j;
    END;
```

```
COMMENT: Obtêm subárvores ótimas com 2,3,...,N nós sucessivamente;
    FOR d ← 2 STEP 1 UNTIL N DO
        BEGIN FOR j ←d STEP 1 UNTIL N DO
          BEGIN i ←j-d; M ← +∞
            FOR K ← r_{i,j-1} STEP 1 UNTIL r_{i+1,j} DO
               BEGIN s ←C_{i,k-1} + C_{k,j}
                 IF  s < M THEN {M ←s, K1 ← K}
               END loopk;
```

```
            C(i,j) ← W(i,j) + M
            r(i,j) ← K1;
        END JLOOP;
    END DLOOP
END ALGK.
```

APÊNDICE 1

Dados para Construção do Apêndice 2 [14]

Classe 1: todos os q_i são iguais

Caso 1: $q_i = 0$, $0 \leqslant i \leqslant N$

Caso 2: $q_i = 10$, $0 \leqslant i \leqslant N$

Classe 2: Os q_i foram calculados como função da vizinhança de p_i, onde $p_i = 0$ para $i = 0$ ou $i=N$.

Caso 3: $q_i = (p_i + p_{i+1})/2$, $0 \leqslant i \leqslant N$;

Caso 4: $q_i = (p_{i-1} + p_i + p_{i+1} + p_{i+2})/4$, $0 \leqslant i \leqslant N$;

Caso 5: $q_i = ABS(3p_{i+1} - 2p_{i+3})$, $0 \leqslant i \leqslant N$

Classe 3: Os q_i foram escolhidos de tal modo que a soma dos q_i fosse igual ou menor que a soma dos p_i do apêndice 3

Caso 6: q_i escolhidos dos conjuntos 1, 2 e 3, casos 3 e 4 do apêndice 4

Caso 7: q_i selecionados dos conjuntos 1, 2 e 3, casos 3 e 4 do apêndice 3

Classe 4: Os q_i foram escolhidos de tal modo que a soma dos q_i fosse maior que a soma dos p_i do apêndice 3.

Caso 8: q_i escolhidos dos conjuntos 1,2 e 3, casos 3 e 4, do apêndice 3

Caso 9: q_i selecionados aleatòriamente

Caso 10: $q_i = 1{,}5(p_{i-1}+p_i+p_{i+1}+p_{i+2})$, $0 \leqslant i \leqslant N$, de acordo com o apêndice 3.

APÊNDICE 2

Comparação dos Resultados - $E(C_T)/E(C_T^*)$

DADOS	BUSCA ÓTIMA	Algoritmo G No=15 F=4		Algoritmo I		Algoritmo II	
	(a)	(b)	c=b:a	(d)	e=d:a	(f)	g=f:a
Conjunto 1: 1	4.2944	4.3635	1.016	4.3601	1.015	4.3134	1.004
2	6.6060	6.6809	1.011	6.9022	1.019	6.6858	1.012
3	5.8749	5.9439	1.011	6.1270	1.042	5.9009	1.004
4	6.0650	6.1591	1.015	6.1526	1.014	6.0746	1.001
5	6.6424	6.7192	1.011	6.7426	1.015	6.6807	1.005
6	5.9633	6.0404	1.012	6.0319	1.011	6.0289	1.011
7	5.8250	5.8256	1.000	5.8557	1.005	5.8825	1.009
8	6.2576	6.3669	1.017	6.3070	1.007	6.3123	1.008
9	7.0856	7.2746	1.026	7.2276	1.020	7.1704	1.011
10	7.3376	7.3675	1.004	7.4838	1.019	7.4478	1.015
2: 1	4.6317	4.7003	1.014	4.8012	1.036	4.6558	1.005
2	7.2555	7.2602	1.000	7.4713	1.029	7.2983	1.005
3	6.2534	6.2854	1.005	6.3889	1.021	6.2789	1.004
4	6.3976	6.4308	1.005	6.5469	1.023	6.4303	1.005
5	7.0182	7.0570	1.004	7.1664	1.021	7.1082	1.012
6	6.5382	6.5632	1.003	6.6014	1.009	6.6642	1.019
7	6.4549	6.4638	1.001	6.6429	1.029	6.4838	1.004
8	6.5810	6.8593	1.042	6.6020	1.003	6.6162	1.005
9	7.1216	7.4623	1.047	7.1929	1.010	7.1714	1.006
10	7.6958	7.7384	1.005	7.8551	1.020	7.7518	1.007
3: 1	4.0035	4.0547	1.012	4.0594	1.020	4.0073	1.000
2	6.4230	6.4747	1.008	6.5407	1.018	6.4802	1.008
3	5.5561	5.5814	1.004	5.6037	1.008	5.5593	1.000
4	5.7628	5.8182	1.009	5.8323	1.012	5.8382	1.013
5	6.2211	6.2596	1.006	6.3044	1.013	6.2320	1.002
6	5.6524	5.7149	1.011	5.7428	1.015	5.6709	1.003
7	5.5935	5.7259	1.023	5.7076	1.020	5.6188	1.004
8	6.0850	6.3225	1.039	6.1500	1.010	6.1298	1.007
9	7.0125	7.2805	1.038	7.1133	1.014	7.0583	1.006
10	6.9738	7.0482	1.010	7.0602	1.012	7.0423	1.009
4: 1	5.0362	5.0572	1.004	5.1175	1.016	5.0772	1.008
2	7.4164	7.4609	1.006	7.5424	1.016	7.4975	1.001
3	6.5463	6.5611	1.002	6.7034	1.023	6.6114	1.009
4	6.6505	6.6667	1.002	6.8013	1.022	6.7246	1.010
5	7.2702	7.3371	1.009	7.4051	1.018	7.3869	1.016
6	6.5494	6.5627	1.002	6.6970	1.022	6.6057	1.008
7	6.7508	6.7954	1.006	6.8906	1.020	6.8038	1.007
8	6.7320	6.9473	1.031	6.7797	1.007	6.8286	1.014
9	7.1119	7.3833	1.038	7.2024	1.012	7.1607	1.006
10	7.8730	7.8765	1.000	7.9968	1.015	8.0008	1.016

APÊNDICE 2

Comparação dos Resultados - $E(C_T)/E(C_T^*)$

(Continuação)

DADOS	BUSCA ÓTIMA	Algoritmo G No=15 F=4		Algoritmo I		Algoritmo II	
	(a)	(b)	c=b:a	(d)	e=d:a	(f)	g=f:a
Conjunto 5: 1	4.1379	4.1815	1.010	4.1449	1.001	4.2235	1.020
2	5.9726	6.0523	1.013	6.1240	1.025	5.9968	1.004
3	5.7540	5.7765	1.003	5.8487	1.016	5.7853	1.005
4	5.9979	6.0469	1.008	6.0771	1.013	6.0317	1.005
5	6.6600	6.8071	1.022	6.7745	1.017	6.7698	1.016
6	5.3331	5.3672	1.006	5.4599	1.023	5.3830	1.009
7	5.4117	5.4284	1.003	5.5404	1.023	5.4288	1.003
8	5.8119	5.8935	1.014	5.8346	1.003	5.8545	1.007
9	6.9904	7.2023	1.030	7.1415	1.021	7.1036	1.016
10	7.2693	7.3624	1.012	7.3964	1.017	7.3279	1.008

APÊNDICE 2

Comparação dos Resultados - $E(C_T)/E(C_T^*)$

DADOS	Algoritmo II - Centróide (h)	i=h:a	Nº MÁXIMO DE VÉRTICES VISITADOS POR INCLUSÃO (j)	TOTAL DE VÉRTICES VISITADOS NA ÁRVORE AUXILIAR (k)
Conjunto 1: 1	4.3081	1.003	133	3389
2	6.6973	1.013	109	3497
3	5.9063	1.005	348	3364
4	6.0900	1.004	137	3785
5	6.7195	1.011	343	4372
6	6.0373	1.012	135	3434
7	5.9135	1.015	134	3891
8	6.3661	1.017	235	3682
9	7.2012	1.016	313	4201
10	7.4210	1.011	128	4208
2: 1	4.6558	1.005	180	3281
2	7.3999	1.019	167	3532
3	6.2732	1.003	160	3593
4	6.4206	1.003	106	3345
5	7.1690	1.021	195	4138
6	6.5901	1.007	273	3548
7	6.6219	1.025	167	3959
8	6.6946	1.017	171	3343
9	7.2423	1.016	245	3537
10	7.8870	1.024	149	4159
3: 1	4.0169	1.003	216	3535
2	6.4879	1.010	76	3541
3	5.5672	1.001	366	4664
4	5.8177	1.009	169	4463
5	6.2722	1.008	239	4591
6	5.7861	1.023	186	4562
7	5.6366	1.007	197	3689
8	6.1158	1.005	475	4521
9	7.1852	1.024	233	3808
10	7.0557	1.011	154	4435
4: 1	5.1199	1.016	312	3620
2	7.5196	1.013	76	3298
3	6.5917	1.006	134	3336
4	6.7651	1.017	136	3718
5	7.3010	1.004	604	4408
6	6.6438	1.014	109	3587
7	6.8963	1.021	174	3334
8	6.7821	1.007	137	2950
9	7.1938	1.011	168	3327
10	8.0889	1.027	233	3832

APÊNDICE 2

Comparação dos Resultados - $E(C_T)/E(C_T^*)$

(Continuação)

DADOS	Algoritmo II - Centróide (h)	i=h:a	Nº MÁXIMO DE VÉRTICES VISITADOS POR INCLUSÃO (j)	TOTAL DE VÉRTICES VISITADOS NA ÁRVORE AUXILIAR (k)
Conjunto 5: 1	4.1428	1.001	250	3739
2	6.0927	1.020	245	3440
3	5.8210	1.011	251	3894
4	6.0618	1.010	120	3871
5	6.7077	1.007	804	5009
6	5.3811	1.009	95	3239
7	5.5175	1.019	231	4131
8	5.8329	1.003	299	4417
9	7.0789	1.012	172	3847
10	7.3930	1.017	130	3853

APÊNDICE 3

Dados Para Teste |14|

	Frequências Pi																			
Conjunto 1	4	19	3	3	1	1	28	4	4	5	1	1	1	35	13	27	11	32	1	50
	2	1	1	27	2	31	2	3	1	2	7	59	4	1	1	7	3	2	3	22
	2	1	52	3	1	4	1	1	1	6	8	13	1	4	1	1	1	1	1	26
	33	1	1	8	7	1	2	1	1	2	4	10	5	10	23	1	1	9	6	2
	128	350	1	2	1	87	2	1	8	3	4	1	63	2	3	14	150	1	1	1
	1	2	1	1	8	1	2	1	3	4	21	1	9	1	1	2	10	4	15	1
	1	24	6	2	2	1	1	1	3	1	1	3	14	4	1	2	1	1	2	2
	2	4	43	2	1	4	1	1	18	1	1	7	1	45	15	9	3	1	5	4
	10	8	1	5	10	1	2	10	2	1	24	29	1	16	1	1	4	1	2	1
	1	1	1	1	1	4	2	2	6	4	1	14	5	5	1	53	1	1	1	4
2	7	4	1	1	20	1	2	1	3	1	1	1	30	1	1	1	1	1	3	30
	2	1	1	4	2	2	1	3	1	2	1	7	1	2	4	2	6	6	7	1
	90	3	1	4	1	4	29	1	1	1	1	2	6	2	2	1	5	5	2	40
	1	3	5	4	62	7	1	1	1	3	11	4	2	1	1	2	2	6	1	1
	1	1	1	24	1	1	1	6	100	15	1	2	7	1	1	1	10	3	1	2
	3	1	3	2	1	1	29	2	13	3	1	1	2	1	2	8	1	10	1	9
	1	6	19	2	1	3	1	1	1	4	1	20	1	1	3	3	6	2	1	3
	2	4	1	1	1	1	1	1	7	1	3	39	1	1	15	11	2	1	4	6
	2	1	1	1	1	2	29	3	10	24	2	1	1	1	3	1	1	1	5	1
	3	6	2	2	1	23	1	9	1	1	8	1	6	1	1	30	2	1	14	90
3	8	5	306	1	12	1	13	1	1	10	1	2	2	2	3	2	2	1	2	1
	1	1	1	2	1	5	1	1	1	1	4	2	12	3	1	1	1	1	1	1
	2	1	7	24	1	1	1	5	2	42	1	1	2	1	4	2	1	1	5	1
	1	1	10	1	1	1	3	2	1	1	2	1	1	4	1	7	1	7	12	2
	4	10	1	2	1	1	11	40	2	51	2	1	11	54	227	227	1	1	9	4
	3	1	24	2	2	2	1	2	1	7	1	1	3	1	5	1	1	3	1	1
	1	1	78	4	3	1	40	3	1	1	2	3	1	84	7	37	30	1	5	12
	1	2	1	1	33	2	183	2	7	2	1	4	1	15	173	1	1	2	4	4
	4	1	1	6	17	12	3	2	1	1	4	3	2	1	10	2	107	7	1	4
	1	1	1	3	2	2	2	6	1	1	1	2	2	5	2	1	11	1	1	8
4	7	4	1	1	20	1	2	1	3	1	1	1	30	1	1	1	1	1	3	30
	2	1	1	4	2	2	1	3	1	2	1	7	1	2	4	2	6	6	7	1
	3	1	3	2	1	1	29	2	13	3	1	1	2	1	2	8	1	10	1	9
	1	6	19	2	1	3	1	1	1	4	1	20	1	1	3	3	6	2	1	3
	2	1	1	1	1	2	29	3	10	24	2	1	1	1	3	1	1	1	5	1
	4	19	3	3	1	1	28	4	4	5	1	1	1	35	13	27	11	32	1	50
	2	1	1	27	2	31	2	3	1	2	7	59	4	1	1	7	3	2	3	22
	33	1	1	8	7	1	2	1	1	2	4	10	5	10	23	1	1	9	6	2
	1	2	1	1	8	1	2	1	3	4	21	1	9	1	1	2	10	4	15	1
	2	4	43	2	1	4	1	1	18	1	1	7	1	45	15	9	3	1	5	4

APÊNDICE 3

Dados Para Teste |14|

(Continuação)

	Frequências Pi																			
Conjunto 5	90	3	1	4	1	4	29	1	1	1	1	2	6	2	2	1	5	5	2	40
	1	2	1	1	33	2	183	2	7	2	1	4	1	15	173	1	1	2	4	4
	1	1	1	24	1	1	1	6	100	15	1	2	7	1	1	1	10	3	1	2
	1	1	78	4	3	1	40	3	1	1	2	3	1	84	7	37	30	1	5	12
	2	1	52	3	1	4	1	1	1	6	8	13	1	4	1	1	1	1	1	26
	2	4	43	2	1	4	1	1	18	1	1	7	1	45	15	9	3	1	5	4
	8	5	306	1	12	1	13	1	1	10	1	2	2	2	3	2	2	1	2	1
	4	10	1	2	1	1	11	40	2	51	2	1	11	54	227	227	1	1	9	4
	10	8	1	5	10	1	2	10	2	1	24	29	1	16	1	1	4	1	2	1
	128	350	1	2	1	87	2	1	8	3	4	1	63	2	3	14	150	1	1	1

APÊNDICE 3

Dados Para Teste |14|

Frequências Qi

Conjunto 6

3	1	3	2	1	1	29	2	13	3	1	1	2	1	2	8	1	10	1	9
1	6	19	2	1	3	1	1	1	4	1	20	1	1	3	3	6	2	1	3
2	1	1	1	1	2	29	3	10	24	2	1	1	1	3	1	1	1	5	1
2	1	1	4	2	2	1	3	1	2	1	7	1	2	4	2	6	6	7	1
7	4	1	1	20	1	2	1	3	1	1	1	30	1	1	1	1	1	3	30
4	19	3	3	1	1	28	4	4	5	1	1	1	35	13	27	11	32	1	50
2	1	1	27	2	31	2	3	1	2	7	59	4	1	1	7	3	2	3	22
33	1	1	8	7	1	2	1	1	2	4	10	5	10	23	1	1	9	6	2
1	2	1	1	8	1	2	1	3	4	21	1	9	1	1	2	10	4	15	1
2	4	43	2	1	4	1	1	18	1	1	7	1	45	15	9	3	1	5	4
4																			

7

2	4	43	2	1	4	1	1	18	1	1	7	1	45	15	9	3	1	5	4
1	2	1	1	8	1	2	1	3	4	21	1	9	1	1	2	10	4	15	1
33	1	1	8	7	1	2	1	1	2	4	10	5	10	23	1	1	9	6	2
2	1	1	27	2	31	2	3	1	2	7	59	4	1	1	7	3	2	3	22
4	19	3	3	1	1	28	4	4	5	1	1	1	35	13	27	11	32	1	50
3	1	3	2	1	1	29	2	13	3	1	1	2	1	2	8	1	10	1	9
1	6	19	2	1	3	1	1	1	4	1	20	1	1	3	3	6	2	1	3
2	1	1	1	1	2	29	3	10	24	2	1	1	1	3	1	1	1	5	1
2	1	1	4	2	2	1	3	1	2	1	7	1	2	4	2	6	6	7	1
7	4	1	1	20	1	2	1	3	1	1	1	30	1	1	1	1	1	3	30
4																			

8

2	1	52	3	1	4	1	1	1	6	8	13	1	4	1	1	1	1	1	26
128	350	1	2	1	87	2	1	8	3	4	1	63	2	3	14	150	1	1	1
2	4	43	2	1	4	1	1	18	1	1	7	1	45	15	9	3	1	5	4
10	8	1	5	10	1	2	10	2	1	24	29	1	16	1	1	4	1	2	1
90	3	1	4	1	4	29	1	1	1	1	2	6	2	2	1	5	5	2	40
1	1	1	24	1	1	1	6	100	15	1	2	7	1	1	1	10	3	1	2
8	5	306	1	12	1	13	1	1	10	1	2	2	2	3	2	2	1	2	1
4	10	1	2	1	1	11	40	2	51	2	1	11	54	227	227	1	1	9	4
1	1	78	4	3	1	40	3	1	1	2	3	1	84	7	37	30	1	5	12
1	2	1	1	33	2	183	2	7	2	1	4	1	15	173	1	1	2	4	4
4																			

APÊNDICE 3

Dados Para Teste |14|

(Continuação)

	Frequências Qi																			
Conjunto 9	12	8	315	18	7	24	6	5	6	37	49	78	6	23	6	8	10	6	156	720
	900	6	14	12	623	12	8	48	19	25	6	479	12	19	84	900	7	6	6	18
	24	248	13	7	28	6	8	129	6	8	42	6	570	94	56	18	13	30	25	160
	49	24	30	68	6	12	60	12	9	145	183	6	97	8	9	27	7	13	5	840
	6	14	37	11	9	4	34	34	14	240	6	6	5	145	8	8	7	43	907	110
	7	11	41	8	7	7	211	60	18	7	14	49	31	930	7	75	7	98	7	6
	70	7	15	15	121	23	18	14	7	12	6	25	63	7	315	7	7	66	350	14
	307	14	6	66	323	999	960	6	6	54	4	1	1	468	24	18	6	540	18	31
	40	73	7	12	6	6	195	12	970	12	43	2	7	25	1	90	970	6	5	12
	25	22	11	18	7	497	45	221	190	6	30	72	6	5	6	37	49	78	6	8
	10																			

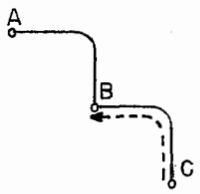
Os pi e qi estão arranjados de acordo com a seguinte matriz:

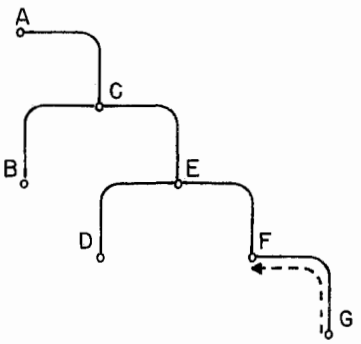
P1, P2, P3,..., P19, P20
P21, P22, ..., P39, P40
P181, P182, ..., P199, P200

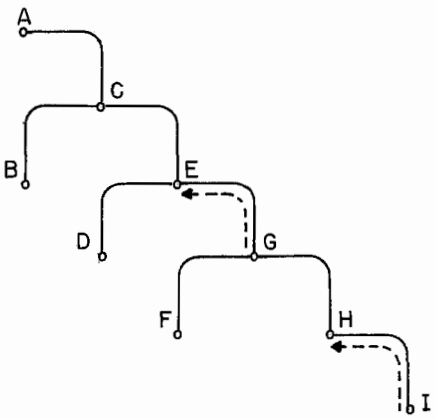
APÊNDICE 4

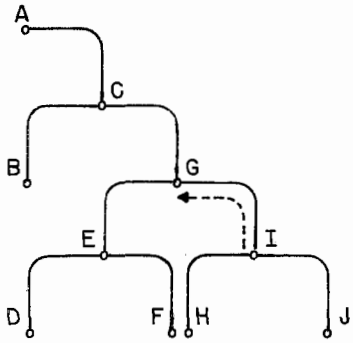
EXECUÇÃO DO ALGORITIMO 1 PARA OS DADOS DO APÊNDICE 4.1

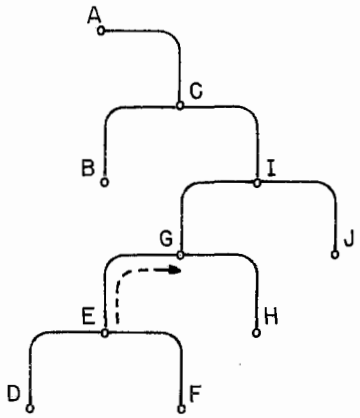
	E	D	E	D	E	D	E	D	E	D	E	D	E	D	E	D	E	D	E	D	E	D	E	D	E	D
A	142	142	142	177	142	235	142	235	142	240	142	260	142	260	142	265	142	272	142	272	142	281	142	296	142	296	142	319
B			35	35	35	93	35	35																				
C					58	58	93	58	93	63	93	83	93	83	93	88	93	95	93	95	93	104	93	119	93	119	93	142
D									5	5	5	25	5	5														
E											20	20	25	20	25	25	25	32	25	32	25	41	25	56	25	25		
F															5	5	5	12	5	5								
G																	7	7	12	7	12	16	12	31	37	31	37	54
H																					9	9	9	24	9	9		
I																							15	15	24	15	24	38
J																											23	23

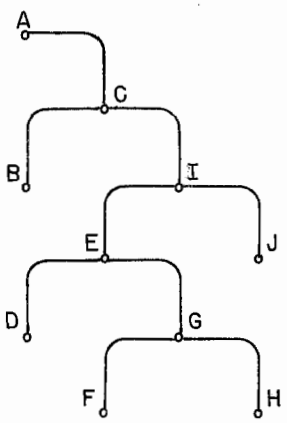












PROMOCÃO E → G				
E	25	25	25	31
G	37	16	12	16
I	61	38		

APÊNDICE 4.1

DISTRIBUIÇÃO DE FREQUÊNCIAS

A	142
B	35
C	58
D	5
E	20
F	5
G	7
H	9
I	15
J	23

GRÁFICO 5.0
ALGORITMOS I e II
TEMPO MÉDIO DE EXECUÇÃO POR NÓ

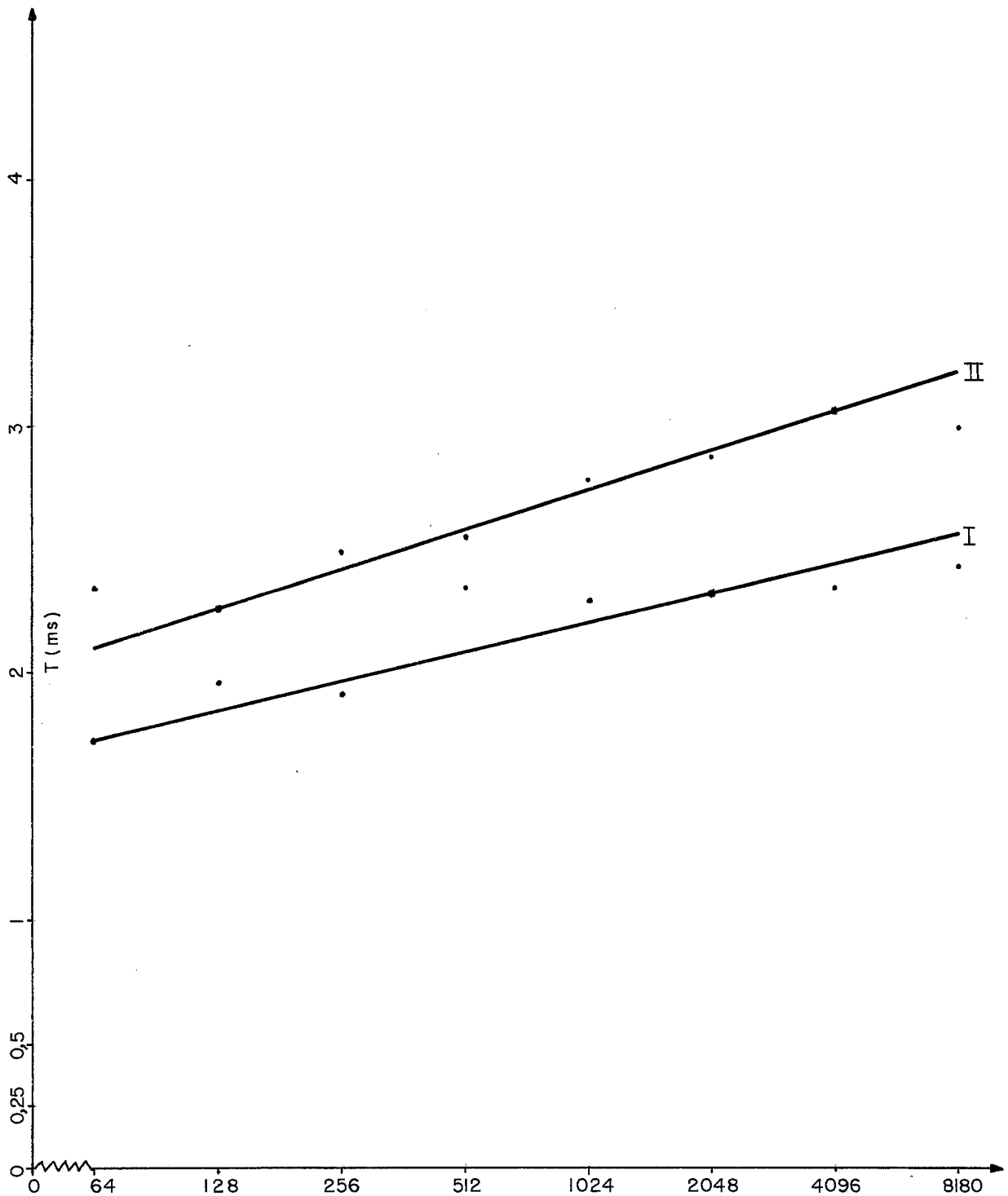


GRÁFICO 5.1
ALGORITMO I - DE INSERÇÃO À DIREITA
TEMPO DE EXECUÇÃO

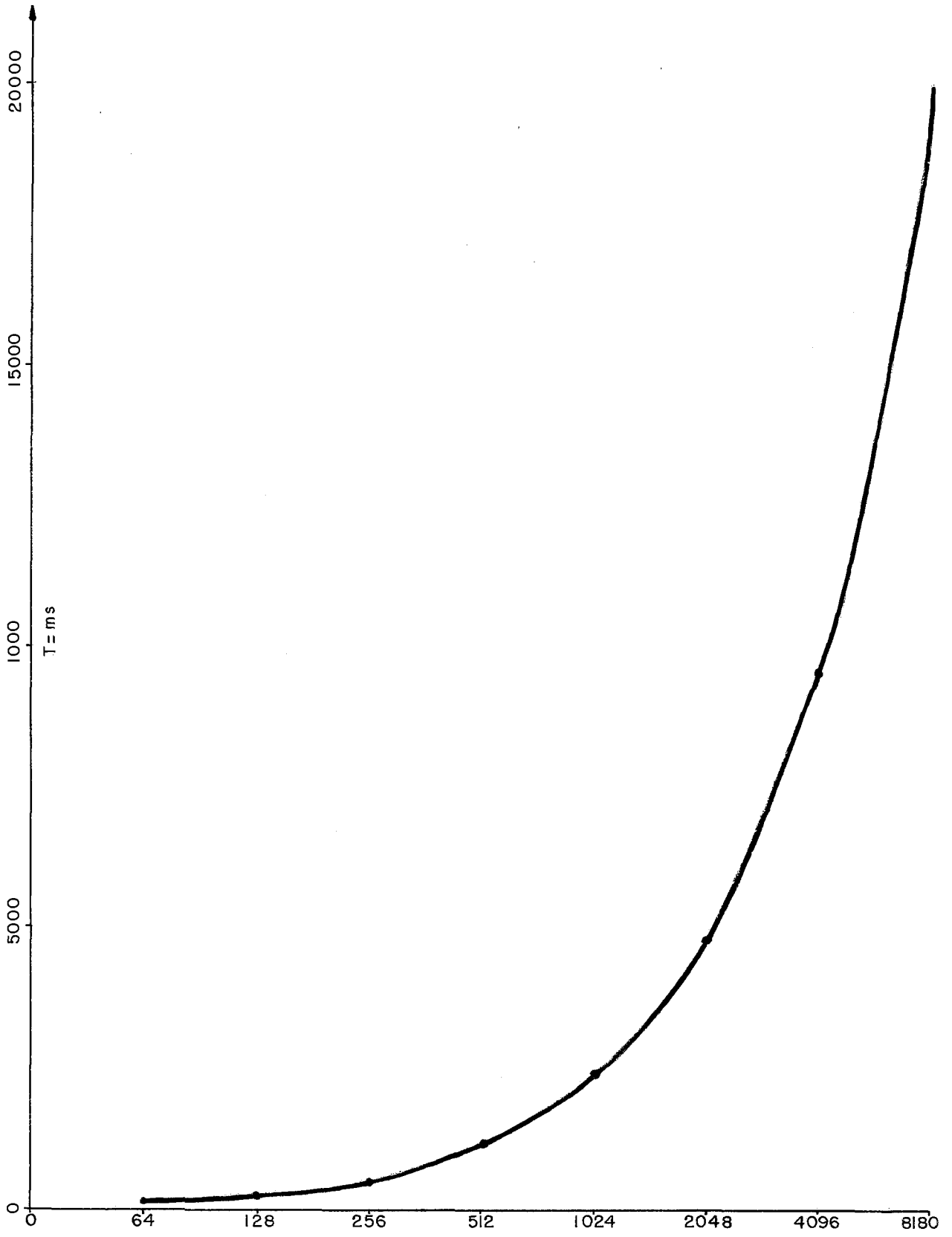


GRÁFICO 5.2
ALGORITMO II- INSERÇÃO A PESO DECRESCENTE
TEMPO DE EXECUÇÃO

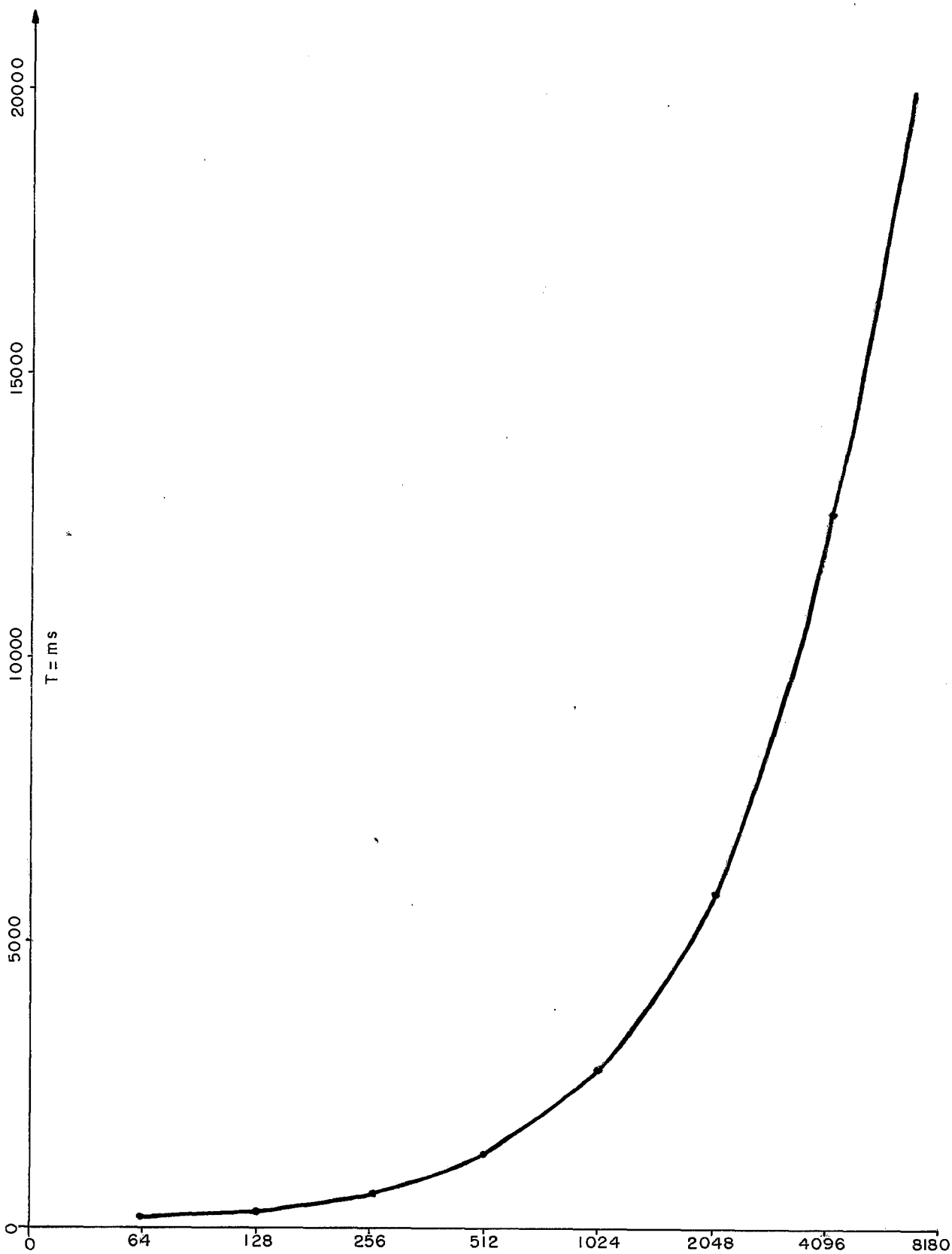
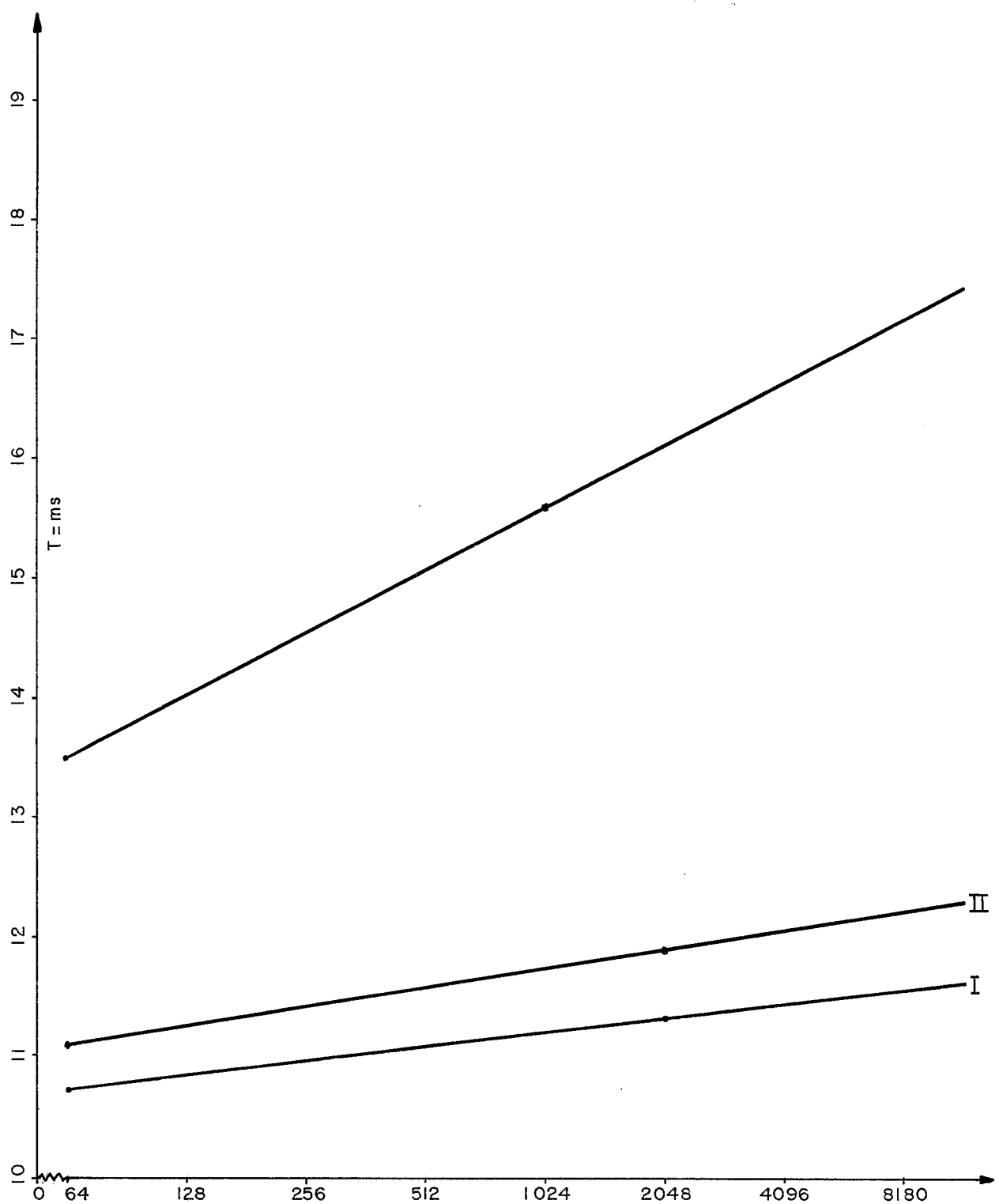


GRÁFICO 5.3
ALGORITMOS GW, I e II
COMPARAÇÃO DOS TEMPOS DE EXECUÇÃO POR NÓ



```
1-      BEGIN COMMENT: CONSTRUCAO DINAMICA DE UMA ARVORE BINARIA LEXICOGRAFICA
--                     SEMI-OTIMA A PARTIR DAS FREQUENCIAS DE CONSULTA AOS NOHS
--                     E DAS FREQUENCIAS DE CONSULTA ENTRE NOHS CONSECUTIVOS;
--
--      INTEGER ARRAY PESODIREITO,PESOESQUERDO,FATHER,
--                    LEFT,GANHOPILHA,FILA,PILHA
--                    (1::200); INTEGER ARRAY GANHOACUMULADO(0::200);
--      INTEGER ARRAY FILHOANTERIOR (0::200);
--      INTEGER ARRAY RIGHT(0::200);
--      INTEGER RAIZ,FRENTE,RETAGUARDA,ULTIMO,N;
--      INTEGER PAI,FILHO,STACK,GANHOMAXIMO;
--      INTEGER K;
--      INTEGER IX,IY;      REAL YFL,X,AVR;
--      STRING (10) ARRAY WORD(1::201); INTEGER ARRAY ALFA,BETA (0::201);
--
--      PROCEDURE EXPONENCIAL;
2-      BEGIN IY:=IX*1220703125;
--        IF IY<0 THEN IY:=IY+2147483647+1;
--        YFL:=IX:=IY; YFL:=YFL*0.4656613'-9;
--        X:=-1*AVR*LN(YFL); K:=ROUND(X)
-2      END EXPONENCIAL;
--
--
--      PROCEDURE INCLUSAO (INTEGER VALUE I);
2-      BEGIN COMMENT: ACRESCENTA NOH AO RAMO DIREITO DA ARVORE;
--        FATHER(I):=ULTIMO; RIGHT(ULTIMO):=I;
--        RIGHT(I):=LEFT(I):=0; ULTIMO:=I
-2      END INCLUSAO;
--
--
--      PROCEDURE ATUALIZA (INTEGER VALUE I);
2-      BEGIN
--      INTEGER POINT;
--      PESOESQUERDO(I):=ALFA(I-1)+BETA(I);
--      POINT:=ULTIMO; PESODIREITO(POINT):=0;
--        GANHOPILHA(POINT):=GANHOACUMULADO(POINT):=FILA(POINT):=PILHA(POINT):=0
--          ;
--      WHILE POINT ¬= 0 DO
3-        BEGIN
--        PESODIREITO(POINT):=PESODIREITO(POINT)+ALFA(I)+BETA(I);
--        POINT:=FATHER(POINT)
-3        END
-2      END ATUALIZA;
--
--
--      PROCEDURE ARVOREAUXILIAR;
2-      BEGIN COMMENT *** PRODUZ UMA FILA DE NOHS QUE FORNECE O GANHO MAXIMO  NO
--                        RAMO DIREITO DA ARVORE BINARIA;
--      INTEGER PAI,FILHO,STACK,GANHOMAXIMO;
--
--      STACK:=FRENTE:=RETAGUARDA:=GANHOMAXIMO:=0;
--      FATHER(RAIZ):=ULTIMO;
--      FILHO:=ULTIMO; PAI:=FATHER(ULTIMO);
--      IF RAIZ ¬= ULTIMO THEN
3-      BEGIN
--
--      PROMOCAO: IF PESODIREITO(FILHO)>PESOESQUERDO(PAI)
4-                        THEN BEGIN
```

```
--
--              COMMENT: *** A PROMOCAO ATUAL EH VIAVEL;
--
--               GANHOPILHA(PAI):=PESODIREITO(FILHO)-PESOESQUERDO(PAI);
--               GANHOACUMULADO(PAI) :=GANHOPILHA(PAI)+GANHOACUMULADO(STACK);
--               PILHA(PAI):=STACK;
--               FILHOANTERIOR(PAI):=FILHO;
--               STACK:=PAI;
--               PESOESQUERDO(FILHO):=PESOESQUERDO(FILHO)+PESOESQUERDO(PAI);
--               FATHER(FILHO):=PAI:=FATHER(PAI);
--               IF GANHOACUMULADO(STACK) > GANHOMAXIMO
5-                  THEN BEGIN
--
--       COMMENT: PROMOCAO VANTAJOSA, COPIA PILHA NA FILA;
--               GANHOMAXIMO:=GANHOACUMULADO(STACK);
--               RETAGUARDA:=STACK;
--               WHILE PILHA(RETAGUARDA) ¬= 0 DO
6-                BEGIN FILA(PILHA(RETAGUARDA)):=RETAGUARDA;
--                      RETAGUARDA:=PILHA(RETAGUARDA)
-6                END;
--               FRENTE:=RETAGUARDA; RETAGUARDA:=STACK;  FILA(RETAGUARDA):=0;
--               GOTO TESTAFIM
-5             END
-4              END;
--       RECUPERANOHS: WHILE (STACK<PAI) AND (STACK¬=0)  DO
4-                      BEGIN
--                      K:=FILHOANTERIOR(STACK);
--                      PESOESQUERDO(K):=PESOESQUERDO(K)-PESOESQUERDO(STACK);
--                      GANHOACUMULADO(STACK):=GANHOACUMULADO(STACK)
--                                          -  GANHOPILHA(STACK);
--                      FATHER(K):=STACK;
--                     PAI:=STACK;
--                      STACK:=PILHA(STACK)
-4                     END;
--                     FILHO:=PAI;
--                     PAI:=FATHER(FILHO);
--       TESTAFIM:
--                IF STACK¬=0 THEN
4-              BEGIN
--              WHILE (PAI>STACK) AND (PAI¬=FILHOANTERIOR(STACK)) DO
55                    BEGIN FILHO:=PAI; PAI:=FATHER(FILHO) END;
--                          IF PAI=ULTIMO THEN GOTO RECUPERANOHS
-4                                        ELSE GOTO PROMOCAO END
4-                 ELSE BEGIN
--                IF FILHO¬=ULTIMO THEN IF PAI<FILHO THEN GOTO PROMOCAO
-4                 END
-3      END
-2      END ARVOREAUXILIAR ;
--
--
--      PROCEDURE FILAGANHOMAXIMO;
2-      BEGIN
--      ICCONTROL(2);
--      FATHER(RAIZ):=0;
--      WHILE FRENTE ¬= 0 DO
3-      BEGIN
--      IF FRENTE = RAIZ
--          THEN RAIZ:=RIGHT(FRENTE)
```

```
--          ELSE RIGHT(FATHER(FRENTE)):=RIGHT(FRENTE);
--       K:=RIGHT(FRENTE);
--       PESOESQUERDO(K):=PESOESQUERDO(K)+PESOESQUERDO(FRENTE);
--       PESODIREITO(FRENTE):=PESODIREITO(FRENTE)-PESODIREITO(K);
--       FATHER(K):=FATHER(FRENTE);
--       FATHER(FRENTE):=K;
--       RIGHT(FRENTE):=LEFT(K);
--       LEFT(K):=FRENTE;
--       IF (LEFT(FRENTE)¬=0) AND
--             (PESOESQUERDO(LEFT(FRENTE))>PESODIREITO(FRENTE))
4-        THEN BEGIN
--             K:=RIGHT(LEFT(FRENTE));
--             PESODIREITO(LEFT(FRENTE)):=PESODIREITO(LEFT(FRENTE))
--                                        + PESODIREITO(FRENTE);
--             LEFT(FATHER(FRENTE)):=LEFT(FRENTE);
--             RIGHT(LEFT(FRENTE)):=FRENTE;
--             FATHER(LEFT(FRENTE)):=FATHER(FRENTE);
--             LEFT(FRENTE):=K
-4              END;
--       FRENTE:=FILA(FRENTE)
-3       END
-2       END FILAGANHOMAXIMO;
--
--
--       PROCEDURE ORDEMFINAL(INTEGER VALUE N);
2-       BEGIN COMMENT *** PERCORRE RECURSIVEMENTE A SUBARVORE ESQUERDA,SUBARVO-
--                      RE DIREITA E RAIZ. CALCULA TEMPO MEDIO DE PESQUISA;
--
--       INTEGER P,DE; INTEGER ARRAY W,D,A (1::N+1); REAL EDT; INTEGER IEDT;
--       INTEGER STAQUE,J, IALFANT,IALFA;
--           J:=1; STAQUE:=0; IEDT:=0;
3-       T1: WHILE J<=N   DO BEGIN
--           WRITE("LEFT",LEFT(J),"NOH",J,WORD(J),"RIGHT",RIGHT(J));
--                    W(J):=0; J:=J+1
-3                        END;
--           P:= RAIZ;
--           WRITE("LEFT",LEFT(RAIZ),"RAIZ",RAIZ,"RIGHT",RIGHT(RAIZ));
--
3-       T2: WHILE P ¬= 0 DO BEGIN
--                        STAQUE:=STAQUE+1;
--                        D(STAQUE):=C;
--                        A(STAQUE):=P;
--                        P:=LEFT(P)
-3                        END;
--
3-       T4: IF STAQUE=0 THEN BEGIN
--                        WRITE("PESO PONDERADO",IEDT,"SOMA DOS PESOS",
--                        W(RAIZ));
--                WRITE("O TEMPO MEDIO DE PESQUISA EH:  ",IEDT/W(RAIZ));
--                        GOTO SAIDA
-3                        END
3-                    ELSE BEGIN
--
--                        P:=A(STAQUE);
--                        DE:=D(STAQUE);
--                        STAQUE:=STAQUE-1
-3                        END;
3-        IF DE = 0 THEN BEGIN
```

```
--                    STAQUE:=STAQUE+1;
--                    D(STAQUE):=1;
--                    A(STAQUE):=P;
--                    P:=RIGHT(P);
--                    GOTO T2
-3                    END
3-                 ELSE BEGIN
--                      IF LEFT(P)=0 THEN IALFANT:=ALFA(P-1) ELSE IALFANT:=0;
--                      IF RIGHT(P)=0 THEN IALFA:=ALFA(P) ELSE IALFA:=0;
--                    FOR J:=1 STEP 1 UNTIL STAQUE DO
--                            W(A(J)):=W(A(J))+BETA(P)+IALFANT+IALFA;
--                    W(P):=W(P)+BETA(P)+IALFANT+IALFA;
--                         IEDT:=IEDT+W(P);
--                         GOTO T4
-3                    END;
--       SAIDA:
-2       END ORDEMFINAL;
--
--
--
--
--       GANHOACUMULADO(0):=0; RAIZ:=1; ULTIMO:=0;
--       FILACANTERIOR(0):=0;
--       N:=0; INTFIELDSIZE:=5;
--       INTOVFL:=NULL;
--       WRITE("AS FREQUENCIAS DADAS SAO:");
--
--       READ(IX,AVR);
--       LEITURA:-READCN(WORD(N+1)); ALFA(N):=0;
--                    WRITE("          ",ALFA(N));
--                    EXPONENCIAL;
--                    BETA(N+1):=K;
2-       IF WORD(N+1)(0|1)¬="." THEN BEGIN N:=N+1;
--                                   WRITE("      ",WORD(N),BETA(N));
--                                   GOTO LEITURA
-2                   END;
--
2-         BEGIN COMMENT *** PARA CADA NOH EXECUTA AS PROCEDURES DE INCLUSAO,ATUA
--                       *** LIZACAO DA TABELA DE PESOS DO RAMO DA DIREITA,PERCUR
--                       *** SO DA ARVORE AUXILIAR PARA OBTENCAO DA FILA DE GANHO
--                       *** MAXIMO;
--
--         FOR I:=1 STEP 1 UNTIL N DO
3-            BEGIN INCLUSAO(I);
--                ATUALIZA(I);
--                ARVOREAUXILIAR;
--                FILAGANHOMAXIMO
-3            END
-2        END;
--        ORDEMFINAL(N)
-1       END.

71 SECONDS IN COMPILATION, 07032 BYTES OF CODE GENERATED
```

```
ANFORD ALGOL W  (VERSION 21NOV69)                                    23 NOVEM

01 1-       BEGIN COMMENT: *** CONSTRUCAO HEURISTICA DE UMA ARVORE BINARIA LEXICOGRA
02 --       * FICAMENTE ORDENADA, POR INSERCAO DOS NOS EM ORDEM DECRESCENTE DO VALOR
03 --       * DA VARIAVEL Q(I-1)+P(I)+Q(I) ***;
04 --       INTEGER ARRAY PESODIREITO,PESOESQUERDO,FATHER,RIGHT,LEFT,GANHOPILHA,FILA
05 --       ,PILHA,GANHOACUMULADO,FILHOFILA,FILHOANTERIOR,P,Q(0::1024);
06 --       INTEGER ARRAY NIVEL (0::1035);
07 --       INTEGER RAIZ,R1,FRENTE,RETAGUARDA,N,K,N1,IX
08 --       ,IY,NOH,LEVEL,DELTA,PEE,STACK,AVO;
09 --       REAL YFL,X,AVR;
10 --
11 --       PROCEDURE ENDORDER(INTEGER VALUE N);
12 2-       BEGIN COMMENT *** PERCORRE RECURSIVAMENTE A SUBARVORE ESQUERDA,SUBARVORE
13 --                        DIREITA E RAIZ. CALCULA TEMPO MEDIO DE PESQUISA;
14 --       INTEGER ARRAY W,D,A(1::N); INTEGER IEDT,J,IALFANT,IALFA,DE;
15 --       J:=1; STACK:=0; IEDT:=0; IOCONTROL(2);
16 --       WRITE("     LEFT","      NOH""     RIGHT");
17 --       T1: WHILE J<=N DO
18 3-         BEGIN WRITE(LEFT(J),J,RIGHT(J),         PESOESQUERDO(J),
19 -3                  PESODIREITO(J)); W(J):=0; J:=J+1 END;
20 --        PEE:=RAIZ; WRITE("LEFT",LEFT(RAIZ),"RAIZ",RAIZ,"RIGHT",RIGHT(RAIZ));
21 --       T2: WHILE PEE ¬=0 DO
22 3-         BEGIN STACK:=STACK+1; D(STACK):=0; A(STACK):=PEE; PEE:=LEFT(PEE)
23 -3         END;
24 --       T4: IF STACK=0 THEN
25 3-         BEGIN WRITE("CUSTO DA ARVORE",IEDT,"SOMA DOS PESOS",W(RAIZ));
26 --          WRITE("C OMPRIMENTO MEDIO DE PESQUISA EH:     ",IEDT/W(RAIZ));
27 -3          GOTO SAIDA END
28 33        ELSE BEGIN PEE:=A(STACK); DE:=D(STACK); STACK:=STACK-1 END;
29 3-       IF DE=0 THEN BEGIN STACK:=STACK+1; D(STACK):=1; A(STACK):=PEE;
30 -3                    PEE:=RIGHT(PEE);GOTO T2 END
31 3-            ELSE BEGIN
32 --                 IF LEFT(PEE)=0  THEN IALFANT:=Q(PEE-1) ELSE IALFANT:=0;
33 --                IF RIGHT(PEE)=0 THEN IALFA:=Q(PEE) ELSE IALFA:=0;
34 --                 FOR J:=1 STEP 1 UNTIL STACK DO
35 --                  W(A(J)):=W(A(J))+P(PEE)+IALFANT+IALFA;
36 --                 W(PEE):=W(PEE)+P(PEE)+IALFANT+IALFA;
37 -3                 IEDT:=IEDT+W(PEE)+IALFANT+IALFA; GOTO T4 END;
38 --       SAIDA:
39 -2       END ENDORDER;
40 --
41 --       PROCEDURE RADIXLISTSORT;
42 2-       BEGIN INTEGER KP,I,K1,N1,CASAS; N1:=N+1; READ(CASAS);
43 --       FOR J:=N STEP -1 UNTIL 2 DO NIVEL(J):=J-1; NIVEL(1):=0; PEE:=N;
44 --       FOR K:=1 STEP 1 UNTIL CASAS DO
45 3-       BEGIN
46 --        FOR I:=0 STEP 1 UNTIL 9 DO
47 44        BEGIN FATHER(I):=N1+I; NIVEL(N1+I):=0 END;
48 --       NEXTREG: KP:=GANHOACUMULADO(PEE);
49 --        FOR J:=1 STEP 1 UNTIL K DO
50 44        BEGIN I:=KP REM 10; KP:=TRUNCATE((KP-I)/10) END;
51 --        FATHER(I):=NIVEL(FATHER(I)):=PEE; PEE:=NIVEL(PEE);
52 --        IF PEE ¬=0 THEN GOTO NEXTREG;
53 --       I:=0;
54 --       LINKPILE: PEE:=FATHER(I);
55 --       NEXTPILE: I:=I+1;
56 --        IF I=10 THEN NIVEL(PEE):=0
57 4-        ELSE BEGIN IF NIVEL(N1+I)=0 THEN GOTO NEXTPILE
58 55                  ELSE BEGIN NIVEL(PEE):=NIVEL(N1+I); GOTO LINKPILE END
```

```
FORD ALGOL W  (VERSION 21NOV69)                                    23 NOVE
 -4                END;
 --          PEE:=NIVEL(N1); I:=PEE
 -3        END LOOPK;
 --        N1:=NIVEL(I); NIVEL(I):=0;
 3-          FOR M:=2 STEP 1 UNTIL N DO BEGIN IX:=NIVEL(N1); NIVEL(N1):=I; I:=N1;
 -3         N1:=IX END; PEE:=I
 -2        END RADIXLISTSORT;
 --
 --        PROCEDURE IESQUERDA;
 2-        BEGIN COMMENT *** INCLUI NOH A ESQUERDA DE R1. ATUALIZA PESOS E LIGACOES
 --        ; K:=RAIZ; LEVEL:=1; DELTA:=P(NOH)+Q(NOH-1);
 --        T3: WHILE NOH < K DO
 3-              BEGIN NIVEL(K):=LEVEL; LEVEL:=LEVEL+1;
 --              PESOESQUERDO(K):=PESOESQUERDO(K)+DELTA;
 --              IF LEFT(K) ¬=0 THEN K:=LEFT(K)
 4-               ELSE BEGIN LEFT(K):=NOH; NIVEL(NOH):=LEVEL;
 --                     PESOESQUERDO(NOH):=DELTA;
 --                     PESODIREITO(NOH):=P(NOH)+PESOESQUERDO(K)-P(K)
 --                                       -PESOESQUERDO(NOH);
 -4                    GOTO RETORNO END
 -3              END;
 --
 --              WHILE (NOH>K) AND (K¬=0) DO
 3-               BEGIN NIVEL(K):=LEVEL; LEVEL:=LEVEL+1;
 --               PESODIREITO(K):=PESODIREITO(K)+DELTA;
 --                 IF RIGHT(K) ¬=0 THEN K:=RIGHT(K)
 4-                 ELSE BEGIN RIGHT(K):=NOH; NIVEL(NOH):=LEVEL;
 --                       PESOESQUERDO(NOH):=DELTA;
 --                       PESODIREITO(NOH):=P(NOH)+PESODIREITO(K)-P(K)
 --                                         -PESOESQUERDO(NOH);
 -4                      GOTO RETORNO END
 -3               END;
 --        IF K ¬=0 THEN GOTO T3;
 --        RETORNO: LEFT(NOH):=RIGHT(NOH):=0; FATHER(NOH):=K
 -2        END IESQUERDA;
 --
 --        PROCEDURE IDIREITA;
 2-        BEGIN COMMENT INCLUI NOH A DIREITA  DE R1. ATUALIZA PESOS E LIGACOES;
 --         K:=RAIZ; LEVEL:=1; DELTA:=P(NOH)+Q(NOH);
 --        T3: WHILE NOH<K DO
 3-              BEGIN NIVEL(K):=LEVEL; LEVEL:=LEVEL+1;
 --              PESOESQUERDO(K):=PESOESQUERDO(K)+DELTA;
 --              IF LEFT(K) ¬=0 THEN K:=LEFT(K)
 4-               ELSE BEGIN LEFT(K):=NOH; NIVEL(NOH):=LEVEL;
 --                     PESODIREITO(NOH):=DELTA;
 --                     PESOESQUERDO(NOH):=P(NOH)+PESOESQUERDO(K)-P(K)
 --                                       -PESODIREITO(NOH);
 -4                    GOTO RETORNO END
 -3              END;
 --             WHILE (NOH>K) AND (K¬=0) DO
 3-              BEGIN NIVEL(K):=LEVEL; LEVEL:=LEVEL+1;
 --              PESODIREITO(K):=PESODIREITO(K)+DELTA;
 --              IF RIGHT(K) ¬=0 THEN K:=RIGHT(K)
 4-               ELSE BEGIN RIGHT(K):=NOH; NIVEL(NOH):=LEVEL;
 --                     PESODIREITO(NOH):=DELTA;
 --                     PESOESQUERDO(NOH):=P(NOH)+PESODIREITO(K)-P(K)
 --                                       -PESODIREITO(NOH);
 -4                    GOTO RETORNO END
```

```
17 -3          END;
18 --       IF K¬=0 THEN GOTO T3;
19 --       RETORNO: LEFT(NOH):=RIGHT(NOH):=0; FATHER(NOH):=K
20 -2       END IDIREITA;
21 --
22 --       PROCEDURE ARVOREAUXILIAR;
23 2-       BEGIN COMMENT *** PRODUZ A FILA DE NOHS QUE MAXIMIZA A FUNCAO GANHO;
24 --       INTEGER PAI,FILHO,GANHOMAXIMO;
25 --       PROCEDURE ESQUERDA;
26 3-       BEGIN GANHOPILHA(PAI):=PESODIREITO(FILHO)-PESOESQUERDO(PAI);
27 --        GANHOACUMULADO(PAI):=GANHOPILHA(PAI)+GANHOACUMULADO(STACK);
28 --        PILHA(PAI):=STACK; FILHOANTERIOR(PAI):=FILHO; STACK:=PAI;
29 --        PESOESQUERDO(FILHO):=PESOESQUERDO(FILHO)+PESOESQUERDO(PAI);
30 --       FATHER(FILHO):=PAI:=FATHER(PAI); AVO:=FATHER(PAI)
31 -3       END ESQUERDA;
32 --
33 --       PROCEDURE DIREITA;
34 3-       BEGIN COMMENT *** PROMOVE NOH DA ESQUERDA PARA A DIREITA;
35 --        GANHOPILHA(PAI):=PESOESQUERDO(FILHO)-PESODIREITO(PAI);
36 --        GANHOACUMULADO(PAI):=GANHOPILHA(PAI)+GANHOACUMULADO(STACK);
37 --        PILHA(PAI):=STACK; FILHOANTERIOR(PAI):=FILHO; STACK:=PAI;
38 --        PESODIREITO(FILHO):=PESODIREITO(FILHO)+PESODIREITO(PAI);
39 --        FATHER(FILHO):=PAI:=FATHER(PAI); AVO:=FATHER(PAI)
40 -3       END DIREITA;
41 --       PROCEDURE COPYSTACK;
42 3-       BEGIN GANHOMAXIMO:=GANHOACUMULADO(STACK); RETAGUARDA:=STACK;
43 --        WHILE PILHA(RETAGUARDA) ¬=0 DO
44 4-         BEGIN FILA(PILHA(RETAGUARDA)):=RETAGUARDA;
45 --          FILHOFILA(PILHA(RETAGUARDA)):=FILHOANTERIOR(PILHA(RETAGUARDA));
46 -4         RETAGUARDA:=PILHA(RETAGUARDA).END;
47 --        FRENTE:=RETAGUARDA; RETAGUARDA:=STACK;
48 --        FILHOFILA(STACK):=FILHOANTERIOR(STACK); FILA(RETAGUARDA):=0
49 -3       END COPYSTACK;
50 --
51 --        STACK:=RETAGUARDA:=FRENTE:=GANHOMAXIMO:=0;
52 --       FILHO:=FATHER(RAIZ):=NOH; PAI:=FATHER(FILHO); AVO:=FATHER(PAI);
53 --       PROMOCAO:  IF PAI<FILHO THEN
54 3-         BEGIN COMMENT PROMOCAO PARA A ESQUERDA;
55 --          IF PESODIREITO(FILHO)>PESOESQUERDO(PAI) THEN
56 4-           BEGIN ESQUERDA;
57 --            IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
58 54             BEGIN COPYSTACK; GOTO TESTAFIM  END END
59 4-           ELSE BEGIN IF (AVO>PAI) AND (AVO ¬= NOH)
60 5-                  THEN BEGIN
61 --                      IF PESODIREITO(FILHO)+PESOESQUERDO(FILHO)
62 --                         >PESODIREITO(AVO) THEN
63 6-                       BEGIN ESQUERDA; DIREITA;
64 --                        IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
65 77                         BEGIN COPYSTACK; GOTO TESTAFIM END
66 -6                       END
67 -5                    END
68 -4             END
69 -3          END
70 3-          ELSE BEGIN COMMENT PROMOCAO PARA A DIREITA;
71 --           IF PESOESQUERDO(FILHO)>PESODIREITO(PAI) THEN
72 4-            BEGIN DIREITA;
73 --             IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
74 54              BEGIN COPYSTACK; GOTO TESTAFIM END END
```

```
FORD ALGOL W  (VERSION 21NOV69)                                        23 NOVEM

 4-              ELSE BEGIN IF (AVO<PAI) AND (AVO¬= NOH)
 5-                         THEN BEGIN
 --                          IF PESOESQUERDO(FILHO)+PESODIREITO(FILHO)
 --                             > PESOESQUERDO(AVO) THEN
 6-                           BEGIN DIREITA; ESQUERDA;
 --                            IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
 77                             BEGIN COPYSTACK; GOTO TESTAFIM END
 -5                            END END
 -4                    END
 -3              END;
 --       RECUPERANOHS:
 --       WHILE (NIVEL(STACK)<NIVEL(PAI)) AND (STACK¬=0) DO
 3-        BEGIN K:=FILHOANTERIOR(STACK);
 --         IF STACK<K THEN PESOESQUERDO(K):=PESOESQUERDO(K)-PESOESQUERDO(STACK)
 --          ELSE PESODIREITO(K):=PESODIREITO(K)-PESODIREITO(STACK);
 --         GANHOACUMULADO(STACK):=GANHOACUMULADO(STACK)-GANHOPILHA(STACK);
 --         PAI:=FATHER(K):=STACK; STACK:=PILHA(STACK)
 -3         END;
 --       FILHO:=PAI; PAI:=FATHER(FILHO);  AVO:=FATHER(PAI);
 --       TESTAFIM: IF STACK ¬=0 THEN
 3-        BEGIN WHILE (NIVEL(PAI)>NIVEL(STACK)) AND (PAI¬=FILHOANTERIOR(STACK))
 44              DO BEGIN FILHO:=PAI; PAI:=FATHER(FILHO); AVO:=FATHER(PAI) END;
 --         IF PAI=NOH THEN GOTO RECUPERANOHS ELSE GOTO PROMOCAO
 -3        END
 --       ELSE
 3-        BEGIN IF FILHO¬=NOH THEN IF NIVEL(PAI)<NIVEL(FILHO) THEN GOTO PROMOCAO
 -3        END
 -2       END ARVOREAUXILIAR;
 --
 --       PROCEDURE FILAGANHOMAXIMO;
 2-       BEGIN FATHER(RAIZ):=0;
 --        WHILE FRENTE ¬=0 DO
 3-       BEGIN
 --          IF FRENTE<FILHOFILA(FRENTE) THEN
 4-           BEGIN COMMENT: EFETIVA PROMOCAO PARA ESQUERDA;
 --            K:=RIGHT(FRENTE);
 --            IF FRENTE=RAIZ THEN RAIZ:=K
 5-             ELSE BEGIN
 --                 IF FATHER(FRENTE)<FRENTE THEN RIGHT(FATHER(FRENTE)):=K
 -5                  ELSE LEFT(FATHER(FRENTE)):=K END;
 --            PESOESQUERDO(K):=PESOESQUERDO(K)+PESOESQUERDO(FRENTE);
 --            PESODIREITO(FRENTE):=PESODIREITO(FRENTE)-PESODIREITO(K);
 --            FATHER(K):=FATHER(FRENTE); FATHER(FRENTE):=K;
 --            RIGHT(FRENTE):=LEFT(K); FATHER(LEFT(K)):=FRENTE; LEFT(K):=FRENTE;
 --            IF (LEFT(FRENTE)¬=0)
 --               AND (PESOESQUERDO(LEFT(FRENTE))>PESODIREITO(FRENTE)) THEN
 5-             BEGIN K:=RIGHT(LEFT(FRENTE));
 --              PESODIREITO(LEFT(FRENTE)):=PESODIREITO(LEFT(FRENTE))
 --                                        +PESODIREITO(FRENTE);
 --              PESOESQUERDO(FRENTE):=PESOESQUERDO(FRENTE)
 --                                   -PESOESQUERDO(LEFT(FRENTE));
 --              LEFT(FATHER(FRENTE)):=LEFT(FRENTE); RIGHT(LEFT(FRENTE)):=FRENTE;
 --              FATHER(LEFT(FRENTE)):=FATHER(FRENTE);
 --              FATHER(FRENTE):=LEFT(FRENTE); FATHER(K):=FRENTE;
 --              LEFT(FRENTE):=K
 -5             END
 -4           END
 --           ELSE
```

ANFORD ALGOL W (VERSION 21NOV69) 23 NOVEM

```
33 4-            BEGIN COMMENT EFETIVA PROMOCAO PARA A DIREITA;
34 --             K:=LEFT(FRENTE);
35 --             IF FRENTE =RAIZ THEN RAIZ:=K
36 5--             ELSE BEGIN
37 --                    IF FATHER(FRENTE)<FRENTE THEN RIGHT(FATHER(FRENTE)):=K
38 -5                   ELSE LEFT(FATHER(FRENTE)):=K END;
39 --             PESODIREITO(K):=PESODIREITO(K)+PESODIREITO(FRENTE);
40 --             PESOESQUERDO(FRENTE):=PESOESQUERDO(FRENTE)-PESOESQUERDO(K);
41 --             FATHER(K):=FATHER(FRENTE); FATHER(FRENTE):=K;
42 --             LEFT(FRENTE):=RIGHT(K); FATHER(RIGHT(K)):=FRENTE; RIGHT(K):=FRENTE;
43 --             IF (RIGHT(FRENTE)¬=0)
44 --                AND (PESODIREITO(RIGHT(FRENTE))>PESOESQUERDO(FRENTE)) THEN
45 5-             BEGIN K:=LEFT(RIGHT(FRENTE));
46 --              PESOESQUERDO(RIGHT(FRENTE)):=PESOESQUERDO(RIGHT(FRENTE))
47 --                                          +PESOESQUERDO(FRENTE);
48 --              PESODIREITO(FRENTE):=PESODIREITO(FRENTE)
49 --                                  -PESODIREITO(RIGHT(FRENTE));
50 --              RIGHT(FATHER(FRENTE)):=RIGHT(FRENTE); LEFT(RIGHT(FRENTE)):=FRENTE
51 --             ;FATHER(RIGHT(FRENTE)):=FATHER(FRENTE);
52 --              FATHER(FRENTE):=RIGHT(FRENTE); FATHER(K):=FRENTE;
53 --              RIGHT(FRENTE):=K
54 -5             END
55 -4           END;
56 --          FRENTE:=FILA(FRENTE)
57 -3        END
58 -2     END FILAGANHOMAXIMO;
59 --
60 --     PROCEDURE LER;
61 2-     BEGIN
62 --     LEITURA: READON(P(N+1));
63 --              IF (N+1) REM 10 = 1 THEN IOCONTROL(2); WRITEON(P(N+1));
64 33             IF P(N+1)¬=-1 THEN BEGIN N:=N+1; GOTO LEITURA END
65 -2     END LER;
66 --     PROCEDURE CASO3;
67 2-     BEGIN COMMENT: Q(I)=(P(I)+P(I+1)):2;
68 3-     Q(N+1):=0; FOR L:=0 STEP 1 UNTIL N DO BEGIN
69 --                  Q(L):=ENTIER((P(L)+P(L+1))*0.5);
70 -3                 IF L REM 10=0 THEN IOCONTROL(2); WRITEON(Q(L)) END
71 -2     END CASO3;
72 --
73 --     N:=0;INTOVFL:=NULL; INTFIELDSIZE:=10; FATHER(0):=0;
74 --     PESOESQUERDO(0):=PESODIREITO(0):=MAXINTEGER;
75 --     GANHOACUMULADO(0):=FILHOANTERIOR(0):=P(0):=NIVEL(0):=FILA(0):=0;
76 --     WRITE("AS FREQUENCIAS DADAS SAO:"); IOCONTROL(2);
77 --     LER; CASO3;
78 --     FOR J:=1 STEP 1 UNTIL N DO GANHOACUMULADO(J):=Q(J-1)+P(J)+Q(J);
79 --     RADIXLISTSORT;
80 --     R1:=RAIZ:=PEE; FATHER(RAIZ):=RIGHT(RAIZ):=LEFT(RAIZ):=0;
81 --     PESOESQUERDO(RAIZ):=P(RAIZ)+Q(RAIZ-1); PESODIREITO(RAIZ):=P(R1)+Q(R1);
82 --     PEE:=NIVEL(PEE);
83 --     FOR J:=2 STEP 1 UNTIL N DO
84 2-      BEGIN NOH:=PEE; PEE:=NIVEL(PEE);
85 --       IF NOH<R1 THEN IESQUERDA ELSE IDIREITA;
86 -2       ARVOREAUXILIAR; FILAGANHOMAXIMO END; ENDORDER(N)
87 -1     END.

.19 SECONDS IN COMPILATION, 13792 BYTES OF CODE GENERATED
```

```
1-      BEGIN COMMENT: *** CONSTRUCAO HEURISTICA DE UMA ARVORE BINARIA LEXICOGRA
--      * FICAMENTE ORDENADA, POR INSERCAO DOS NOS EM ORDEM DECRESCENTE DO VALOR
--      * DA VARIAVEL Q(I-1)+P(I)+Q(I) ***;
--      INTEGER ARRAY PESODIREITO,PESOESQUERDO,FATHER,RIGHT,LEFT,GANHOPILHA,FILA
--      ,PILHA,GANHOACUMULADO,FILHOFILA,FILHOANTERIOR,P,Q(0::1025);
--      INTEGER ARRAY NIVEL (0::1035);
--      INTEGER RAIZ,R1,FRENTE,RETAGUARDA,N,K,N1,IX,NUMAX,NUMTOT
--      ,IY,NOH,LEVEL,DELTA,PEE,STACK,AVO;
--      REAL YFL,X,AVR;
--
--
--      PROCEDURE IESQUERDA;
2-      BEGIN COMMENT *** INCLUI NOH A ESQUERDA DE R1. ATUALIZA PESOS E LIGACOES
--      ; K:=RAIZ; LEVEL:=1; DELTA:=P(NOH)+Q(NOH-1);
--      T3: WHILE NOH < K DO
3-        BEGIN NIVEL(K):=LEVEL; LEVEL:=LEVEL+1;
--        PESOESQUERDO(K):=PESOESQUERDO(K)+DELTA;
--        IF LEFT(K) ¬=0 THEN K:=LEFT(K)
4-          ELSE BEGIN LEFT(K):=NOH; NIVEL(NOH):=LEVEL;
--                 PESOESQUERDO(NOH):=DELTA;
--                 PESODIREITO(NOH):=P(NOH)+PESOESQUERDO(K)-P(K)
--                                   -PESOESQUERDO(NOH);
-4                 GOTO RETORNO END
-3        END;
--
--        WHILE (NOH>K) AND (K¬=0) DO
3-        BEGIN NIVEL(K):=LEVEL; LEVEL:=LEVEL+1;
--        PESODIREITO(K):=PESODIREITO(K)+DELTA;
--          IF RIGHT(K) ¬=0 THEN K:=RIGHT(K)
4-          ELSE BEGIN RIGHT(K):=NOH; NIVEL(NOH):=LEVEL;
--                 PESOESQUERDO(NOH):=DELTA;
--                 PESODIREITO(NOH):=P(NOH)+PESODIREITO(K)-P(K)
--                                   -PESOESQUERDO(NOH);
-4                 GOTO RETORNO END
-3        END;
--      IF K ¬=0 THEN GOTO T3;
--      RETORNO: LEFT(NOH):=RIGHT(NOH):=0; FATHER(NOH):=K
-2      END IESQUERDA;
--
--      PROCEDURE IDIREITA;
2-      BEGIN COMMENT INCLUI NOH A DIREITA DE R1. ATUALIZA PESOS E LIGACOES;
--        K:=RAIZ; LEVEL:=1; DELTA:=P(NOH)+Q(NOH);
--      T3: WHILE NOH<K DO
3-        BEGIN NIVEL(K):=LEVEL; LEVEL:=LEVEL+1;
--        PESOESQUERDO(K):=PESOESQUERDO(K)+DELTA;
--        IF LEFT(K) ¬=0 THEN K:=LEFT(K)
4-          ELSE BEGIN LEFT(K):=NOH; NIVEL(NOH):=LEVEL;
--                 PESODIREITO(NOH):=DELTA;
--                 PESOESQUERDO(NOH):=P(NOH)+PESOESQUERDO(K)-P(K)
--                                   -PESODIREITO(NOH);
-4                 GOTO RETORNO END
-3        END;
--      WHILE (NOH>K) AND (K¬=0) DO
3-        BEGIN NIVEL(K):=LEVEL; LEVEL:=LEVEL+1;
--        PESODIREITO(K):=PESODIREITO(K)+DELTA;
--        IF RIGHT(K) ¬=0 THEN K:=RIGHT(K)
4-          ELSE BEGIN RIGHT(K):=NOH; NIVEL(NOH):=LEVEL;
--                 PESODIREITO(NOH):=DELTA;
```

```
FORC ALGOL W  (VERSION 21NOV69)                                  17 DECEM
--                    PESOESQUERDO(NCH):=P(NCH)+PESODIREITO(K)-P(K)
--                                       -PESODIREITO(NCH);
-4                    GOTO RETORNO END
-3          END;
--        IF K¬=0 THEN GOTO L3;
--        RETORNO: LEFT(NCH):=RIGHT(NOH):=0; FATHER(NOH):=K
-2        END DIREITA;
--
--        PROCEDURE ARVOREAUXILIAR;
2-        BEGIN COMMENT *** PRODUZ A FILA DE NOHS QUE MAXIMIZA A FUNCAO GANHO;
--        INTEGER PAI,FILHO,GANHOMAXIMO,NUMATL;
--        PROCEDURE ESQUERDA;
3-        BEGIN GANHOPILHA(PAI):=PESODIREITO(FILHO)-PESOESQUERDO(PAI);
--         GANHOACUMULADO(PAI):=GANHOPILHA(PAI)+GANHOACUMULADO(STACK);
--         PILHA(PAI):=STACK; FILHOANTERIOR(PAI):=FILHO; STACK:=PAI;
--         PESOESQUERDO(FILHO):=PESOESQUERDO(FILHO)+PESOESQUERDO(PAI);
--        FATHER(FILHO):=PAI:=FATHER(PAI); AVO:=FATHER(PAI)
-3        END ESQUERDA;
--
--        PROCEDURE DIREITA;
3-        BEGIN COMMENT *** PROMOVE NOH DA ESQUERDA PARA A DIREITA;
--         GANHOPILHA(PAI):=PESOESQUERDO(FILHO)-PESODIREITO(PAI);
--         GANHOACUMULADO(PAI):=GANHOPILHA(PAI)+GANHOACUMULADO(STACK);
--         PILHA(PAI):=STACK; FILHOANTERIOR(PAI):=FILHO; STACK:=PAI;
--         PESODIREITO(FILHO):=PESODIREITO(FILHO)+PESODIREITO(PAI);
--         FATHER(FILHO):=PAI:=FATHER(PAI); AVO:=FATHER(PAI)
-3        END DIREITA;
--        PROCEDURE COPYSTACK;
3-        BEGIN GANHOMAXIMO:=GANHOACUMULADO(STACK); RETAGUARDA:=STACK;
--         WHILE PILHA(RETAGUARDA) ¬=0 DO
4-          BEGIN FILA(PILHA(RETAGUARDA)):=RETAGUARDA;
--          FILHOFILA(PILHA(RETAGUARDA)):=FILHOANTERIOR(PILHA(RETAGUARDA));
-4          RETAGUARDA:=PILHA(RETAGUARDA) END;
--         FRENTE:=RETAGUARDA; RETAGUARDA:=STACK;
--         FILHOFILA(STACK):=FILHOANTERIOR(STACK); FILA(RETAGUARDA):=0
-3        END COPYSTACK;
--
--         STACK:=RETAGUARDA:=FRENTE:=GANHOMAXIMO:=NUMATL:=0;
--        FILHO:=FATHER(RAIZ):=NOH; PAI:=FATHER(FILHO); AVO:=FATHER(PAI);
--        PROMOCAO: NUMATL:=NUMATL+1; IF PAI<FILHO THEN
3-          BEGIN COMMENT PROMOCAO PARA A ESQUERDA;
--           IF PESODIREITO(FILHO)>PESOESQUERDO(PAI) THEN
4-            BEGIN ESQUERDA;
--             IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
54              BEGIN COPYSTACK; GOTO TESTAFIM  END END
4-           ELSE BEGIN IF  (AVO>PAI) AND (AVO ¬= NOH)
5-                  THEN BEGIN
--                       IF PESODIREITO(FILHO)+PESOESQUERDO(FILHO)
--                          >PESODIREITO(AVO) THEN
6-                        BEGIN ESQUERDA; DIREITA;
--                         IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
77                          BEGIN COPYSTACK; GOTO TESTAFIM END
-6                        END
-5                      END
-4             END
-3          END
3-          ELSE BEGIN COMMENT PROMOCAO PARA A DIREITA;
--           IF PESOESQUERDO(FILHO)>PESODIREITO(PAI) THEN
```

```
4-            BEGIN DIREITA;
--             IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
54              BEGIN COPYSTACK; GOTO TESTAFIM END END
4-              ELSE BEGIN IF (AVO<PAI) AND (AVO¬= NOH)
5-                           THEN BEGIN
--                            IF PESOESQUERDO(FILHO)+PESODIREITO(FILHO)
--                               > PESOESQUERDO(AVO) THEN
6-                             BEGIN DIREITA; ESQUERDA;
--                              IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
77                               BEGIN COPYSTACK; GOTO TESTAFIM END
-5                              END END
-4                  END
-3              END;
--        RECUPERANOHS:
--        WHILE (NIVEL(STACK)<NIVEL(PAI)) AND (STACK¬=0) DO
3-         BEGIN K:=FILHOANTERIOR(STACK);
--          IF STACK<K THEN PESOESQUERDO(K):=PESOESQUERDO(K)-PESOESQUERDO(STACK)
--           ELSE PESODIREITO(K):=PESODIREITO(K)-PESODIREITO(STACK);
--          GANHOACUMULADO(STACK):=GANHOACUMULADO(STACK)-GANHOPILHA(STACK);
--          PAI:=FATHER(K):=STACK; STACK:=PILHA(STACK)
-3         END;
--        FILHO:=PAI; PAI:=FATHER(FILHO);  AVO:=FATHER(PAI);
--        TESTAFIM: IF STACK ¬=0 THEN
3-         BEGIN WHILE (NIVEL(PAI)>NIVEL(STACK)) AND (PAI¬=FILHOANTERIOR(STACK))
44               DO BEGIN FILHO:=PAI; PAI:=FATHER(FILHO); AVO:=FATHER(PAI) END;
--          IF PAI=NOH THEN GOTO RECUPERANOHS ELSE GOTO PROMOCAO
-3         END
--        ELSE
3-         BEGIN IF FILHO¬=NOH THEN IF NIVEL(PAI)<NIVEL(FILHO) THEN GOTO PROMOCAO;
--          NUMTOT:=NUMTOT+NUMATL; IF NUMATL>NUMAX THEN NUMAX:=NUMATL
-3         END
-2        END ARVOREAUXILIAR;
--
--        PROCEDURE FILAGANHOMAXIMO;
2-        BEGIN FATHER(RAIZ):=0;
--         WHILE FRENTE ¬=0 DO
3-        BEGIN
--          IF FRENTE<FILHOFILA(FRENTE) THEN
4-           BEGIN COMMENT: EFETIVA PROMOCAO PARA ESQUERDA;
--           K:=RIGHT(FRENTE);
--           IF FRENTE=RAIZ THEN RAIZ:=K
5-            ELSE BEGIN
--                IF FATHER(FRENTE)<FRENTE THEN RIGHT(FATHER(FRENTE)):=K
-5               ELSE LEFT(FATHER(FRENTE)):=K END;
--           PESOESQUERDO(K):=PESOESQUERDO(K)+PESOESQUERDO(FRENTE);
--           PESODIREITO(FRENTE):=PESODIREITO(FRENTE)-PESODIREITO(K);
--           FATHER(K):=FATHER(FRENTE); FATHER(FRENTE):=K;
--           RIGHT(FRENTE):=LEFT(K); FATHER(LEFT(K)):=FRENTE; LEFT(K):=FRENTE;
--           IF (LEFT(FRENTE)¬=0)
--              AND (PESOESQUERDO(LEFT(FRENTE))>PESODIREITO(FRENTE)) THEN
5-            BEGIN K:=RIGHT(LEFT(FRENTE));
--             PESODIREITO(LEFT(FRENTE)):=PESODIREITO(LEFT(FRENTE))
--                                        +PESODIREITO(FRENTE);
--             PESOESQUERDO(FRENTE):=PESOESQUERDO(FRENTE)
--                                  -PESOESQUERDO(LEFT(FRENTE));
--             LEFT(FATHER(FRENTE)):=LEFT(FRENTE); RIGHT(LEFT(FRENTE)):=FRENTE;
--             FATHER(LEFT(FRENTE)):=FATHER(FRENTE);
--             FATHER(FRENTE):=LEFT(FRENTE); FATHER(K):=FRENTE;
```

```
FORC ALGOL W  (VERSION 21NOV69)                                    17 DECEM
--                LEFT(FRENTE):=K
-5                END
-4              END
--              ELSE
4-              BEGIN COMMENT EFETIVA PROMOCAO PARA A DIREITA;
--               K:=LEFT(FRENTE);
--               IF FRENTE =RAIZ THEN RAIZ:=K
5-                ELSE BEGIN
--                     IF FATHER(FRENTE)<FRENTE THEN RIGHT(FATHER(FRENTE)):=K
-5                    ELSE LEFT(FATHER(FRENTE)):=K END;
--               PESODIREITO(K):=PESODIREITO(K)+PESODIREITO(FRENTE);
--               PESOESQUERDO(FRENTE):=PESOESQUERDO(FRENTE)-PESOESQUERDO(K);
--               FATHER(K):=FATHER(FRENTE); FATHER(FRENTE):=K;
--               LEFT(FRENTE):=RIGHT(K); FATHER(RIGHT(K)):=FRENTE; RIGHT(K):=FRENTE;
--               IF (RIGHT(FRENTE)¬=0)
--                  AND (PESODIREITO(RIGHT(FRENTE))>PESOESQUERDO(FRENTE)) THEN
5-                BEGIN K:=LEFT(RIGHT(FRENTE));
--                 PESOESQUERDO(RIGHT(FRENTE)):=PESOESQUERDO(RIGHT(FRENTE))
--                                              +PESOESQUERDO(FRENTE);
--                 PESODIREITO(FRENTE):=PESODIREITO(FRENTE)
--                                      -PESODIREITO(RIGHT(FRENTE));
--                 RIGHT(FATHER(FRENTE)):=RIGHT(FRENTE); LEFT(RIGHT(FRENTE)):=FRENTE
--                ;FATHER(RIGHT(FRENTE)):=FATHER(FRENTE);
--                 FATHER(FRENTE):=RIGHT(FRENTE); FATHER(K):=FRENTE;
--                 RIGHT(FRENTE):=K
-5                END
-4              END;
--            FRENTE:=FILA(FRENTE)
-3            END
-2          END FILAGANHOMAXIMO;
--
--
--          N:=0;INTOVFL:=NULL; INTFIELDSIZE:=10; FATHER(0):=NUMAX:=NUMTOT:=0;
--          PESOESQUERDO(0):=PESODIREITO(0):=MAXINTEGER;
--          GANHOACUMULADO(0):=FILHOANTERIOR(0):=P(0):=NIVEL(0):=FILA(0):=0;
--          GANHOACUMULADO(0):=MAXINTEGER;
2-          BEGIN
--          PROCEDURE EXPONENCIAL;
3-          BEGIN AVR:=100; IX:=7;
--          N:=128;
4-          FOR L:=0 STEP 1 UNTIL N DO BEGIN IY:=IX*1220703125;
--              IF IY<0 THEN IY:=IY+2147483647+1; YFL:=IX:=IY; YFL:=YFL*0.4656613'-9
--          ;   X:=-1*AVR*LN(YFL); P(L+1):=ROUND(X); IY:=IX*1220703125;
--              IF IY<0 THEN IY:=IY+2147483647+1; YFL:=IX:=IY; YFL:=YFL*0.4656613'-9
-4          ;   X:=-1*AVR*LN(YFL); Q(L):=ROUND(X) END;
4-          FOR L:=1 STEP 1 UNTIL N DO BEGIN
-4          IF L REM 10 = 1 THEN IOCONTROL(2); WRITEON(P(L)) END; IOCONTROL(2);
4-          FOR L:=0 STEP 1 UNTIL N DO BEGIN
-4          IF L REM 10 = 1 THEN IOCONTROL(2); WRITEON(Q(L)) END
-3          END EXPONENCIAL;
--          PROCEDURE CENTER(INTEGER VALUE N);
3-          BEGIN COMMENT: NUMERA A ARVORE COMPLETA DOS CENTROIDES;
--          INTEGER M1,INFER,SUPER,ACFINAL; LOGICAL DIREC;
--          LOGICAL ARRAY ZEROUM (1::N);
--
4-          PROCEDURE CENTROIDE; BEGIN
--          COMMENT: DEVOLVE O NUMERO DA ARVORE COMPLETA CORRESPONDENTE AO NOH CEN
--                   TROIDE ENTRE INFER E SUPER;
```

```
ORC ALGOL W  (VERSION 21NOV69)                                  17 DECEM
--      INTEGER AMP,WINF,WSUP,DIFFX,WK1,WK2,DIFFSUP,DIFFINF;
--
--      IF INFER=SUPER THEN ACFINAL:=INFER
5-      ELSE BEGIN AMP:=(SUPER-INFER) DIV 2; AMP:=AMP+INFER; WINF:=Q(INFER-1);
--              WSUP:=Q(SUPER);
--      FOR L:=INFER STEP 1 UNTIL AMP-1 DO WINF:=WINF+P(L)+Q(L);
--      FOR L:=AMP+1 STEP 1 UNTIL SUPER DO WSUP:=WSUP+P(L)+Q(L-1);
--      DIFFX:=ABS(WSUP-WINF);
6-      IF AMP¬=SUPER THEN BEGIN WK1:=WINF+P(AMP)+Q(AMP);
-6         WK2:=WSUP-P(AMP+1)-Q(AMP); DIFFSUP:=ABS(WK1-WK2) END
--      ELSE DIFFSUP:=Q(SUPER);
6-      WHILE DIFFX>DIFFSUP DO BEGIN DIFFX:=DIFFSUP; AMP:=AMP+1; WINF:=WK1;
--       WSUP:=WK2;
7-       IF AMP+1<=SUPER THEN BEGIN WK1:=WK1+P(AMP)+Q(AMP);
-7          WK2:=WK2-P(AMP+1)-Q(AMP); DIFFSUP:=ABS(WK1-WK2) END
-6       ELSE DIFFSUP:=MAXINTEGER END;
--      WK1:=WINF-P(AMP-1)-Q(AMP-1); WK2:=WSUP+P(AMP)+Q(AMP-1);
--      DIFFINF:=ABS(WK1-WK2);
6-      WHILE DIFFINF<DIFFX DO BEGIN AMP:=AMP-1; DIFFX:=DIFFINF;
--       WK1:=WK1-P(AMP-1)-Q(AMP-1); WK2:=WK2+P(AMP)+Q(AMP-1);
-6       DIFFINF:=ABS(WK1-WK2) END;ACFINAL:=AMP
-5      END
-4      END CENTROIDE;
--      M1:=INFER:=1; SUPER:=N; STACK:=0;
4-      T1: WHILE SUPER>=INFER DO BEGIN CENTROIDE; GANHOACUMULADO(ACFINAL):=M1;
--           STACK:=STACK+1; LEFT(STACK):=INFER; RIGHT(STACK):=SUPER;
--           FILHOFILA(STACK):=M1; FILHOANTERIOR(STACK):=ACFINAL;
-4          ZEROUM(STACK):=FALSE; SUPER:=ACFINAL-1; M1:=2*M1 END;
--
4-      T2: IF STACK¬=0 THEN BEGIN M1:=FILHOFILA(STACK); DIREC:=ZEROUM(STACK);
--          INFER:=LEFT(STACK); SUPER:=RIGHT(STACK); STACK:=STACK-1;
5-          IF DIREC=FALSE THEN BEGIN STACK:=STACK+1; ZEROUM(STACK):=TRUE;
--             SUPER:=RIGHT(STACK); INFER:=FILHOANTERIOR(STACK)+1;
-5             M1:=2*M1+1; GOTO T1 END
-4        ELSE GOTO T2 END
-3      END CENTER;
--      PROCEDURE RADIXLISTSORT;
3-      BEGIN INTEGER KP,I,K1,N1,CASAS; N1:=N+1; READ(CASAS);
--      FOR J:=N STEP -1 UNTIL 2 DO NIVEL(J):=J-1; NIVEL(1):=0; PEE:=N;
--      FOR K:=1 STEP 1 UNTIL CASAS DO
4-      BEGIN
--       FOR I:=0 STEP 1 UNTIL 9 DO
55        BEGIN FATHER(I):=N1+I; NIVEL(N1+I):=0 END;
--       NEXTREG: KP:=GANHOACUMULADO(PEE);
--        FOR J:=1 STEP 1 UNTIL K DO
55        BEGIN I:=KP REM 10; KP:=TRUNCATE((KP-I)/10) END;
--        FATHER(I):=NIVEL(FATHER(I)):=PEE; PEE:=NIVEL(PEE);
--        IF PEE ¬=0 THEN GOTO NEXTREG;
--      I:=0;
--      LINKPILE: PEE:=FATHER(I);
--      NEXTPILE: I:=I+1;
--       IF I=10 THEN NIVEL(PEE):=0
5-       ELSE BEGIN IF NIVEL(N1+I)=0 THEN GOTO NEXTPILE
66                  ELSE BEGIN NIVEL(PEE):=NIVEL(N1+I); GOTO LINKPILE END
-5            END;
--       PEE:=NIVEL(N1); I:=PEE
-4      END LOOPK
-3      END RADIXLISTSORT;
```

```
FCRD ALGCL W  (VERSICN 21NOV69)                                    17 DECEM
--       PRCCEDURE DEQUE;
3-       BEGIN CCMMENT: CRIA UM DEQUE PARA CADA NIVEL DA ARVORE CENTROIDE E ORDE
--       NA OS NOHS DO NIVEL,RETIRANDO ALTERNADAMENTE UM NOH DE CADA EXTREMO DO
--       DEQUE;
--       INTEGER IP,ULTIMO,SUP,PROX,PTR1,PTR2,ANT;
--       IP:=FATHER(PEE):=NIVEL(PEE); ULTIMO:=PEE; SUP:=3; PROX:=NIVEL(IP);
--       NEWDEQUE: ANT:=0; PTR1:=IP;
4-         WHILE GANHOACUMULADO(IP)<=SUP DO BEGIN LEFT(IP):=ANT; RIGHT(IP):=PROX;
-4         PTR2:=ANT:=IP; IP:=NIVEL(IP); PROX:=NIVEL(IP) END; RIGHT(ANT):=0;
--
4-       WHILE (PTR1<=PTR2) AND (PTR1¬=0) DO BEGIN
--         IF GANHOACUMULADO(PTR1)<=GANHOACUMULADO(PTR2) THEN
5-          BEGIN ULTIMO:=FATHER(ULTIMO):=PTR1;
-5       PTR1:=RIGHT(PTR1) END;
5-       IF PTR2¬=0 THEN  BEGIN
--         IF GANHOACUMULADO(PTR1)<=GANHOACUMULADO(PTR2) THEN
6-          BEGIN ULTIMO:=FATHER(ULTIMO):=PTR2;
-6       PTR2:=LEFT(PTR2) END
-5                         END
-4                           END;
--         SUP:=2*SUP+1; IF IP¬=0 THEN GOTO NEWDEQUE;
--       FOR L:=1 UNTIL N DO NIVEL(L):=FATHER(L)
-3       END DEQUE;
--       WRITE("AS FREQUENCIAS CADAS SAO:"); ICCONTROL(2);
--       EXPONENCIAL;
--       CENTER(N);
--       RADIXLISTSORT;
--       DEQUE
-2       END;
--       GANHOACUMULADO(0):=0;
--       R1:=RAIZ:=PEE; FATHER(RAIZ):=RIGHT(RAIZ):=LEFT(RAIZ):=0;
--       PESOESQUERDO(RAIZ):=P(RAIZ)+Q(RAIZ-1); PESODIREITO(RAIZ):=P(R1)+Q(R1);
--       PEE:=NIVEL(PEE);
--       FOR J:=2 STEP 1 UNTIL N DO
2-       BEGIN NOH:=PEE; PEE:=NIVEL(PEE);
--         IF NOH<R1 THEN IESQUERDA ELSE IDIREITA;
-2       ARVOREAUXILIAR; FILAGANHOMAXIMO END;
2-       BEGIN
--       PROCEDURE ENDORDER(INTEGER VALUE N);
3-       BEGIN COMMENT *** PERCORRE RECURSIVAMENTE A SUBARVORE ESQUERDA,SUBARVORE
--                         DIREITA E RAIZ. CALCULA TEMPO MEDIO DE PESQUISA;
--                           INTEGER IECT,J,IALFANT,IALFA,DE;
--       J:=1; STACK:=0; IECT:=0; ICCONTROL(2);
--       WRITE("     LEFT","       NOH ",   "RIGHT");
--       T1: WHILE J<=N DO
4-           BEGIN WRITE(LEFT(J),J,RIGHT(J),        PESOESQUERDO(J),
--           PESODIREITO(J));
-4       FATHER(J):=FILHOFILA(J):=FILHOANTERIOR(J):=0; J:=J+1 END;
--         PEE:=RAIZ; WRITE("LEFT",LEFT(RAIZ),"RAIZ",RAIZ,"RIGHT",RIGHT(RAIZ));
--       T2: WHILE PEE ¬=0 DO
4-           BEGIN STACK:=STACK+1; FILHOFILA(STACK):=0; FILHOANTERIOR(STACK):=
-4           PEE; PEE:=LEFT(PEE)  END;
--       T4: IF STACK=0 THEN
4-           BEGIN WRITE("CUSTO DA ARVORE",IECT,"SOMA DOS PESOS",
--       FATHER(RAIZ));
--           WRITE("O COMPRIMENTO MEDIO DE PESQUISA EH:     ",
--       IECT/FATHER(RAIZ));
--           WRITE("O NUMERO MAXIMO DE NOS VISITADOS  NA ARVORE AUXILIAR EH:",
```

```
FCRC ALGCL W  (VERSICN 21NOV69)                                          17 DECEM
--       NUMAX,"C NUM.TOTAL DE NOS VISITADOS NA ARVORE AUXILIAR EH: ",NUMTOT);
-4            GCTC SAICA END
4-           ELSE BEGIN PEE:=FILHCANTERIOR(STACK); DE:=FILHCFILA(STACK);
-4           STACK:=STACK-1 END;
4-       IF DE=0 THEN BEGIN STACK:=STACK+1; FILHOFILA(STACK):=1; FILHOANTERIOR
--            (STACK):=PEE;
-4                   PEE:=RIGHT(PEE);GCTO T2 END
4-               ELSE BEGIN
--                    IF LEFT(PEE)=0  THEN IALFANT:=Q(PEE-1) ELSE IALFANT:=0;
--                  IF RIGHT(PEE)=0 THEN IALFA:=Q(PEE) ELSE IALFA:=0;
--                   FOR J:=1 STEP 1 UNTIL STACK DO
--                    FATHER(FILHCANTERIOR(J)):=FATHER(FILHCANTERIOR(J))+P(PEE)
--                    +IALFANT+IALFA;
--                    FATHER(PEE):=FATHER(PEE)+P(PEE)+IALFANT+IALFA;
-4                  IEDT:=IEDT+FATHER(PEE)+IALFANT+IALFA; GOTO T4 END;
--       SAICA:
-3       END ENDCRDER;
-2       ENDCRDER(N) END
-1       END.

E1 SECCNDS IN COMPILATICN, 17016 BYTES OF CODE GENERATED
```

H A S P S Y S T E M L O G

```
.24.55 JOB 800 -- ALGCLW10 -- BEGINNING EXEC - INIT 1 - CLASS L
.25.00 JOB 800 END EXECUTION.
```

```
P-II JCB STATISTICS --    375 CARDS READ --      543 LINES PRINTED --      0 CARDS PU
```

```
      BEGIN COMMENT: PRODUZ UMA ARVORE LEXICOGRAFICAMENTE ORDENADA, ADAPTATIVA
            MENTE;
      INTEGER ARRAY PESODIREITO,PESOESQUERDO,FATHER,RIGHT,LEFT,GANHOPILHA,FILA
      ,PILHA,GANHOACUMULADO,FILHOFILA,FILHOANTERIOR,P,Q(0::257);
      INTEGER ARRAY NIVEL (0::266);
      INTEGER RAIZ,R1,FRENTE,RETAGUARDA,N,K,N1,IX
      ,IY,NOH,LEVEL,DELTA,PEE,STACK,AVO;
      REAL YFL,X,AVR;
      PROCEDURE ENDORDER(INTEGER VALUE N);
      BEGIN COMMENT *** PERCORRE RECURSIVAMENTE A SUBARVORE ESQUERDA,SUBARVORE
                        DIREITA E RAIZ. CALCULA TEMPO MEDIO DE PESQUISA;
      INTEGER ARRAY W,D,A(1::N); INTEGER IEDT,J,IALFANT,IALFA,DE;
      J:=1; STACK:=0; IEDT:=0; IOCONTROL(2);
      WRITE("     LEFT","      NOH""     RIGHT","PESOESQUERDO","PESO DIREITO",
      "FREQUENCIA DE CONSULTA");
      T1: WHILE J<=N DO
           BEGIN WRITE(LEFT(J),J,RIGHT(J),PESOESQUERDO(J),PESODIREITO(J),
                 P(J));  W(J):=0; J:=J+1 END;
          PEE:=RAIZ; WRITE("LEFT",LEFT(RAIZ),"RAIZ",RAIZ,"RIGHT",RIGHT(RAIZ));
      T2: WHILE PEE ¬=0 DO
           BEGIN STACK:=STACK+1; D(STACK):=0; A(STACK):=PEE; PEE:=LEFT(PEE)
           END;
      T4: IF STACK=0 THEN
           BEGIN WRITE("CUSTO DA ARVORE",IEDT,"SOMA DOS PESOS",W(RAIZ));
            WRITE("O COMPRIMENTO MEDIO DE PESQUISA EH:     ",IEDT/W(RAIZ));
            GOTO SAIDA END
           ELSE BEGIN PEE:=A(STACK); DE:=D(STACK); STACK:=STACK-1 END;
      IF DE=0 THEN BEGIN STACK:=STACK+1; D(STACK):=1; A(STACK):=PEE;
                    PEE:=RIGHT(PEE);GOTO T2 END
              ELSE BEGIN
                    IF LEFT(PEE)=0  THEN IALFANT:=Q(PEE-1) ELSE IALFANT:=0;
                   IF RIGHT(PEE)=0 THEN IALFA:=Q(PEE) ELSE IALFA:=0;
                    FOR J:=1 STEP 1 UNTIL STACK DO
                     W(A(J)):=W(A(J))+P(PEE)+IALFANT+IALFA;
                    W(PEE):=W(PEE)+P(PEE)+IALFANT+IALFA;
                    IEDT:=IEDT+W(PEE)+IALFANT+IALFA; GOTO T4 END;
      SAIDA:
      END ENDORDER;
      PROCEDURE ALEATOR;
      BEGIN COMMENT: PRODUZ UM VALOR DE UMA VARIAVEL ALEATORIA CUJA DISTRIBUI
                     CAO DE FREQUENCIAS EH EXPONENCIAL;
      INICIAL:
      IY:=IX*1220703125; IF IY<0 THEN IY:=IY+2147483647+1; YFL:=IX:=IY;
      YFL:=YFL*0.4656612874'-9;  X:=1.0-YFL; X:=- 25*LN(X);
      NOH:=ENTIER(X)+1;
      IF NOH>N THEN GOTO INICIAL
      END ALEATOR;
      PROCEDURE PESQUISA;
      BEGIN COMMENT: PESQUISA A ARVORE BINARIA PARA O ARGUMENTO NOH E INCREMEN
                     TA DE 1 A FREQUENCIA DOS PESOS ESQUERDO OU DIREITO DO CA
                     MINHO PERCORRIDO. SE NOH ESTA PRESENTE SUA FREQUENCIA EH
                     ACRESCIDA DE 1. CASO CONTRARIO, O NOH EH INCLUIDO NA  AR
                     VORE, COM P=1 ***;
      K:=RAIZ; LEVEL:=1;
      T2: WHILE NOH<K DO BEGIN NIVEL(K):=LEVEL; LEVEL:=LEVEL+1;
           PESOESQUERDO(K):=PESOESQUERDO(K)+1;
           IF LEFT(K)¬=0 THEN K:=LEFT(K) ELSE GO TO T5 END;
          IF NOH>K THEN BEGIN NIVEL(K):=LEVEL; LEVEL:=LEVEL+1;
```

```
        PESODIREITO(K):=PESODIREITO(K)+1;
        IF RIGHT(K)¬=0 THEN BEGIN K:=RIGHT(K); GOTO T2 END
                       ELSE GOTO T5 END;
    P(K):=P(K)+1; PESOESQUERDO(K):=PESOESQUERDO(K)+1;
    PESODIREITO(K):=PESODIREITO(K)+1; NIVEL(NOH):=LEVEL; GOTO FIM;
    T5: LEFT(NOH):=RIGHT(NOH):=0; FATHER(NOH):=K;
        PESOESQUERDO(NOH):=PESODIREITO(NOH):=P(NOH):=1; NIVEL(NOH):=LEVEL;
        IF NOH<K THEN LEFT(K):=NOH ELSE RIGHT(K):=NOH;
    FIM:
    END PESQUISA;
    PROCEDURE ARVOREAUXILIAR;
    BEGIN COMMENT *** PRODUZ A FILA DE NOHS QUE MAXIMIZA A FUNCAO GANHO;
    INTEGER PAI,FILHO,GANHOMAXIMO;
    PROCEDURE ESQUERDA;
    BEGIN GANHOPILHA(PAI):=PESODIREITO(FILHO)-PESOESQUERDO(PAI);
     GANHOACUMULADO(PAI):=GANHOPILHA(PAI)+GANHOACUMULADO(STACK);
     PILHA(PAI):=STACK; FILHOANTERIOR(PAI):=FILHO; STACK:=PAI;
     PESOESQUERDO(FILHO):=PESOESQUERDO(FILHO)+PESOESQUERDO(PAI);
    FATHER(FILHO):=PAI:=FATHER(PAI); AVO:=FATHER(PAI)
    END ESQUERDA;
    PROCEDURE DIREITA;
    BEGIN COMMENT *** PROMOVE NOH DA ESQUERDA PARA A DIREITA;
     GANHOPILHA(PAI):=PESOESQUERDO(FILHO)-PESODIREITO(PAI);
     GANHOACUMULADO(PAI):=GANHOPILHA(PAI)+GANHOACUMULADO(STACK);
     PILHA(PAI):=STACK; FILHOANTERIOR(PAI):=FILHO; STACK:=PAI;
     PESODIREITO(FILHO):=PESODIREITO(FILHO)+PESODIREITO(PAI);
     FATHER(FILHO):=PAI:=FATHER(PAI); AVO:=FATHER(PAI)
    END DIREITA;
    PROCEDURE COPYSTACK;
    BEGIN GANHOMAXIMO:=GANHOACUMULADO(STACK); RETAGUARDA:=STACK;
     WHILE PILHA(RETAGUARDA) ¬=0 DO
      BEGIN FILA(PILHA(RETAGUARDA)):=RETAGUARDA;
       FILHOFILA(PILHA(RETAGUARDA)):=FILHOANTERIOR(PILHA(RETAGUARDA));
       RETAGUARDA:=PILHA(RETAGUARDA) END;
     FRENTE:=RETAGUARDA; RETAGUARDA:=STACK;
     FILHOFILA(STACK):=FILHOANTERIOR(STACK); FILA(RETAGUARDA):=0
    END COPYSTACK;
     STACK:=RETAGUARDA:=FRENTE:=GANHOMAXIMO:=0;
    FILHO:=FATHER(RAIZ):=NOH; PAI:=FATHER(FILHO); AVO:=FATHER(PAI);
    PROMOCAO:  IF PAI<FILHO THEN
      BEGIN COMMENT PROMOCAO PARA A ESQUERDA;
       IF PESODIREITO(FILHO)>PESOESQUERDO(PAI) THEN
        BEGIN ESQUERDA;
          IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
           BEGIN COPYSTACK; GOTO TESTAFIM  END END
        ELSE BEGIN IF (AVO>PAI) AND (AVO ¬= NOH)
                THEN BEGIN
                       IF PESODIREITO(FILHO)+PESOESQUERDO(FILHO)
                          >PESODIREITO(AVO) THEN
                        BEGIN ESQUERDA; DIREITA;
                         IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
                           BEGIN COPYSTACK; GOTO TESTAFIM END
                        END
                     END
             END
      END
      ELSE BEGIN COMMENT PROMOCAO PARA A DIREITA;
      IF FILHO¬=PAI THEN BEGIN
```

```
        IF PESOESQUERDO(FILHO)>PESODIREITO(PAI) THEN
         BEGIN DIREITA;
          IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
           BEGIN COPYSTACK; GOTO TESTAFIM END END
           ELSE BEGIN IF (AVO<PAI) AND (AVO¬= NOH)
                      THEN BEGIN
                       IF PESOESQUERDO(FILHO)+PESODIREITO(FILHO)
                          > PESOESQUERDO(AVO) THEN
                        BEGIN DIREITA; ESQUERDA;
                         IF GANHOMAXIMO<GANHOACUMULADO(STACK) THEN
                          BEGIN COPYSTACK; GOTO TESTAFIM END
                        END END
                END
     END
          END;
    RECUPERANOHS:
    WHILE (NIVEL(STACK)<NIVEL(PAI)) AND (STACK¬=0) DO
     BEGIN K:=FILHOANTERIOR(STACK);
      IF STACK<K THEN PESOESQUERDO(K):=PESOESQUERDO(K)⇨PESOESQUERDO(STACK)
       ELSE PESODIREITO(K):=PESODIREITO(K)⇨PESODIREITO(STACK);
      GANHOACUMULADO(STACK):=GANHOACUMULADO(STACK)⇨GANHOPILHA(STACK);
      PAI:=FATHER(K):=STACK; STACK:=PILHA(STACK)
     END;
    FILHO:=PAI; PAI:=FATHER(FILHO);  AVO:=FATHER(PAI);
    TESTAFIM: IF STACK ¬=0 THEN
     BEGIN WHILE (NIVEL(PAI)>NIVEL(STACK)) AND (PAI¬=FILHOANTERIOR(STACK))
           DO BEGIN FILHO:=PAI; PAI:=FATHER(FILHO); AVO:=FATHER(PAI) END;
      IF PAI=NOH THEN GOTO RECUPERANOHS ELSE GOTO PROMOCAO
     END
    ELSE
     BEGIN IF FILHO¬=NOH THEN IF NIVEL(PAI)<NIVEL(FILHO) THEN GOTO PROMOCAO
     END
    END ARVOREAUXILIAR;
    PROCEDURE FILAGANHOMAXIMO;
    BEGIN FATHER(RAIZ):=0;
     WHILE FRENTE ¬=0 DO
    BEGIN
        IF FRENTE<FILHOFILA(FRENTE) THEN
         BEGIN COMMENT: EFETIVA PROMOCAO PARA ESQUERDA;
          K:=RIGHT(FRENTE);
          IF FRENTE=RAIZ THEN RAIZ:=K
           ELSE BEGIN
                 IF FATHER(FRENTE)<FRENTE THEN RIGHT(FATHER(FRENTE)):=K
                  ELSE LEFT(FATHER(FRENTE)):=K END;
          PESOESQUERDO(K):=PESOESQUERDO(K)+PESOESQUERDO(FRENTE);
          PESODIREITO(FRENTE):=PESODIREITO(FRENTE)⇨PESODIREITO(K);
          FATHER(K):=FATHER(FRENTE); FATHER(FRENTE):=K;
          RIGHT(FRENTE):=LEFT(K); FATHER(LEFT(K)):=FRENTE; LEFT(K):=FRENTE;
          IF (LEFT(FRENTE)¬=0)
             AND (PESOESQUERDO(LEFT(FRENTE))>PESODIREITO(FRENTE)) THEN
           BEGIN K:=RIGHT(LEFT(FRENTE));
            PESODIREITO(LEFT(FRENTE)):=PESODIREITO(LEFT(FRENTE))
                                       +PESODIREITO(FRENTE);
            PESOESQUERDO(FRENTE):=PESOESQUERDO(FRENTE)
                                  ⇨PESOESQUERDO(LEFT(FRENTE));
            LEFT(FATHER(FRENTE)):=LEFT(FRENTE); RIGHT(LEFT(FRENTE)):=FRENTE;
            FATHER(LEFT(FRENTE)):=FATHER(FRENTE);
            FATHER(FRENTE):=LEFT(FRENTE); FATHER(K):=FRENTE;
```

```
            LEFT(FRENTE):=K
           END
         END
         ELSE
         BEGIN COMMENT EFETIVA PROMOCAO PARA A DIREITA;
          K:=LEFT(FRENTE);
          IF FRENTE =RAIZ THEN RAIZ:=K
           ELSE BEGIN
                 IF FATHER(FRENTE)<FRENTE THEN RIGHT(FATHER(FRENTE)):=K
                  ELSE LEFT(FATHER(FRENTE)):=K END;
          PESODIREITO(K):=PESODIREITO(K)+PESODIREITO(FRENTE);
          PESOESQUERDO(FRENTE):=PESOESQUERDO(FRENTE)-PESOESQUERDO(K);
          FATHER(K):=FATHER(FRENTE); FATHER(FRENTE):=K;
          LEFT(FRENTE):=RIGHT(K); FATHER(RIGHT(K)):=FRENTE; RIGHT(K):=FRENTE;
          IF (RIGHT(FRENTE)¬=0)
             AND (PESODIREITO(RIGHT(FRENTE))>PESOESQUERDO(FRENTE)) THEN
           BEGIN K:=LEFT(RIGHT(FRENTE));
            PESOESQUERDO(RIGHT(FRENTE)):=PESOESQUERDO(RIGHT(FRENTE))
                                        +PESOESQUERDO(FRENTE);
            PESODIREITO(FRENTE):=PESODIREITO(FRENTE)
                               -PESODIREITO(RIGHT(FRENTE));
            RIGHT(FATHER(FRENTE)):=RIGHT(FRENTE); LEFT(RIGHT(FRENTE)):=FRENTE
           ;FATHER(RIGHT(FRENTE)):=FATHER(FRENTE);
            FATHER(FRENTE):=RIGHT(FRENTE); FATHER(K):=FRENTE;
            RIGHT(FRENTE):=K
           END
         END;
        FRENTE:=FILA(FRENTE)
       END
      END FILAGANHOMAXIMO;
          INTOVFL:=NULL; INTFIELDSIZE:=10; FATHER(0):=0;
      PESOESQUERDO(0):=PESODIREITO(0):=MAXINTEGER;
      N:=128;
      FOR L:=1 STEP 1 UNTIL N DO PESOESQUERDO(L):=PESODIREITO(L):=P(L):=
      LEFT(L):=RIGHT(L):=Q(L):=0;
      GANHOACUMULADO(0):=FILHOANTERIOR(0):=P(0):=NIVEL(0):=FILA(0):=Q(0):=0;
      ICCONTROL(2);
      IX:=7;
      N1:=30000;
      ALEATOR;
      P(NOH):=PESOESQUERDO(NOH):=PESODIREITO(NOH):=1;
      LEFT(NOH):=RIGHT(NOH):=FATHER(NOH):=0; RAIZ:=NOH;
      FOR L:=1 STEP 1 UNTIL N1 DO BEGIN ALEATOR; PESQUISA; ARVOREAUXILIAR;
          FILAGANHOMAXIMO END; ENDORDER(N);
      BEGIN
      PROCEDURE OPTREE (INTEGER VALUE N);
      BEGIN COMMENT *** ENCONTRA UMA ARVORE BINARIA OTIMA ***;
        RECORD NODE(                                  REFERENCE(NODE) LEFT,RIGHT);
      REFERENCE (NODE) P1;
      PROCEDURE POSORDEM(INTEGER VALUE N; REFERENCE (NODE) VALUE ROOT);
      BEGIN REFERENCE (NODE) P1; P1:=ROOT END POSORDEM;
          BEGIN COMMENT
                      *** ENCONTRA UMA ARVORE OTIMA DE N=NOHS, DADA A FREQUEN=
                      *** CIA RELATIVA P(I) DE ENCONTRAR WD(I) E A FREQUENCIA
                      *** Q(I) DE CONSULTAS ENTRE WD(I) E WD(I+1);
           INTEGER ARRAY PP,W,R(0::N,0::N); COMMENT
                      ***PP(I,J), W(I,J), R(I,J) SIGNIFICAM RESPECTIVAMENTE  O
                      *** PESO PONDERADO, O PESO TOTAL E A RAIZ DA ARVORE OTI=
```

```
RD ALGOL W  (VERSION 21NOV69)                                   13 DECEMBE
                    *** MA PARA AS INFORMACOES ENTRE WD(I) E WD(J+1), QUANDO
                    *** I < J+1. O TEMPO MEDIO DE PESQUISA NESTA ARVORE   EH
                    ***PP(I,J)/W(I,J);
            REFERENCE (NODE) PROCEDURE CREATETREE(INTEGER VALUE I,J);
             IF I¬=J THEN NODE(
                CREATETREE (I,R(I,J)-1),CREATETREE(R(I,J),J)) ELSE NULL;
            FOR I:=0 UNTIL N DO PP(I,I):=W(I,I):=Q(I);
             FOR I:=0 UNTIL N DO FOR J:=I+1 UNTIL N DO
               W(I,J):=W(I,J-1)+P(J)+Q(J);
             FOR K:=1 UNTIL N DO FOR I:=0 UNTIL N-K DO
             BEGIN INTEGER IK,MN,MX;IK:=I+K;
               MX:=IF K=1 THEN IK ELSE R(I,IK-1);MN:=PP(I,MX-1)+PP(MX,IK);
               IF K>1 THEN FOR J:=MX+1 UNTIL R(I+1,IK) DO
                  IF PP(I,J-1) + PP(J,IK) < MN THEN
                    BEGIN MN:=PP(I,J-1) + PP(J,IK); MX:=J END;
                PP(I,IK):=MN+W(I,IK);R(I,IK):=MX
             END;
            WRITE("PESO PONDERADO",PP(0,N),"SOMA DOS PESOS",W(0,N));
            WRITE("O TEMPO MEDIO DE PESQUISA EH",PP(0,N)/W(0,N));
        IOCONTROL(3);
        POSORDEM(N,CREATETREE(0,N));
        IOCONTROL(3);INTFIELDSIZE:=3;
        FOR I:=0 UNTIL N DO
        BEGIN IOCONTROL(2);
          FOR J:=0 UNTIL N DO WRITEON(IF I<J THEN R(I,J) ELSE 0)
        END;
      END
    END OPTREE;
    OPTREE(N)
    END
    END.
 SECONDS IN COMPILATION, 14252 BYTES OF CODE GENERATED
```

EFT	NOH"	RIGHT	PESOESQUERDO	PESO DIREITO	FREQUENCIA DE CONSULTA
0	1	0	1193	1193	1193
1	2	3	2354	3269	1161
0	3	4	1077	2108	1077
0	4	0	1031	1031	1031
2	5	8	5444	5543	982
0	6	7	993	1885	993
0	7	0	892	892	892
6	8	9	2854	2676	969
0	9	10	867	1707	867
0	10	0	840	840	840
5	11	17	10773	8203	768
0	12	13	794	1514	794
0	13	0	720	720	720
12	14	16	2218	2048	704
0	15	0	670	670	670
15	16	0	1344	674	674
14	17	21	4181	3873	619
0	18	0	616	616	616
18	19	20	1201	1132	585
0	20	0	547	547	547
19	21	22	2293	1506	545
0	22	23	489	961	489
0	23	0	472	472	472
11	24	42	18719	11793	511
0	25	26	451	898	451
0	26	0	447	447	447
25	27	28	1310	1168	412
0	28	29	380	756	380
0	29	0	376	376	376
27	30	34	2469	3680	403
0	31	0	362	362	362
31	32	33	732	700	370
0	33	0	330	330	330
32	34	37	1393	2215	331
0	35	36	317	593	317
0	36	0	276	276	276
35	37	39	866	1291	273
0	38	0	260	260	260
38	39	41	533	758	273
0	40	0	230	230	230

40	41	0	485	255	255
30	42	55	5985	5536	239
0	43	44	233	457	233
0	44	0	224	224	224
43	45	47	647	565	190
0	46	0	184	184	184
46	47	0	375	191	191
45	48	52	1202	1122	180
0	49	0	169	169	169
49	50	51	354	352	185
0	51	0	167	167	167
50	52	53	666	421	145
0	53	54	146	276	146
0	54	0	130	130	130
48	55	77	2289	3153	145
0	56	0	129	129	129
56	57	58	255	238	126
0	58	0	112	112	112
57	59	61	495	451	128

0	60	0	112	112	112
60	61	62	217	211	105
0	62	0	106	106	106
59	63	69	924	1016	106
0	64	0	76	76	76
64	65	0	169	93	93
65	66	67	241	228	72
0	67	68	84	156	84
0	68	0	72	72	72
66	69	73	459	513	62
0	70	0	73	73	73
70	71	72	133	118	60
0	72	0	58	58	58
71	73	75	259	260	68
0	74	0	61	61	61
74	75	76	128	131	67
0	76	0	64	64	64
63	77	91	1886	1174	52
0	78	0	50	50	50
78	79	81	95	165	45
0	80	0	40	40	40
80	81	82	76	80	36
0	82	0	44	44	44
79	83	87	264	283	49
0	84	0	46	46	46
84	85	86	77	60	31
0	86	0	29	29	29
85	87	89	142	128	36
0	88	0	35	35	35
88	89	90	66	57	31
0	90	0	26	26	26
83	91	100	534	624	36
0	92	93	29	56	29
0	93	0	27	27	27
92	94	95	81	76	25
0	95	96	30	51	30
0	96	0	21	21	21
94	97	98	167	92	35
0	98	99	30	57	30
0	99	0	27	27	27
97	100	113	248	364	24

0	101	0	22	22	22
101	102	103	41	57	19
0	103	104	22	38	22
0	104	0	16	16	16
102	105	109	93	120	14
0	106	0	20	20	20
106	107	108	32	26	12
0	108	0	14	14	14
107	109	111	61	60	15
0	110	0	18	18	18
110	111	112	30	27	12
0	112	0	15	15	15
105	113	117	208	141	9
0	114	0	10	10	10
114	115	116	27	28	17
0	116	0	11	11	11
115	117	123	51	94	13
0	118	0	10	10	10
118	119	121	20	28	10

0	120	0	3	3	3
120	121	122	11	15	8
0	122	0	7	7	7
119	123	125	47	43	9
0	124	0	11	11	11
124	125	127	20	23	9
0	126	0	4	4	4
126	127	128	9	10	5
0	128	0	5	5	5

11 RAIZ 24 RIGHT 42

DA ARVORE 157885 SOMA DOS PESOS 30001
RIMENTO MEDIO DE PESQUISA EH: 5.26265791140295
ONDERADO 154185 SOMA DOS PESOS 30001
O MEDIO DE PESQUISA EH 5.13932868904370

FORD ALGOL W (VERSION 21NOV69) 29 NOVEI

```
 1=      BEGIN COMMENT *** ENCONTRA UMA ARVORE BINARIA OTIMA ***;
           INTEGER N;
         INTEGER IX,IY,K;     REAL YFL,X,AVR;
                                    INTEGER ARRAY A,B(0::257);
           RECORD NODE(                          REFERENCE(NODE) LEFT,RIGHT);
         REFERENCE (NODE) P;
         PROCEDURE POSORDEM(INTEGER VALUE N; REFERENCE (NODE) VALUE ROOT);
         BEGIN REFERENCE (NODE) P; P:=ROOT END POSORDEM ;

         INTFIELDSIZE:=10;
         INTOVFL:=NULL;
         N:=128;
           WRITE("AS FREQUENCIAS DADAS SAO:");
         FOR L:=1 STEP 1 UNTIL N DO READON(B(L));
         FOR L:=0 STEP 1 UNTIL N DO A(L):=0;
         FOR L:=1 STEP 1 UNTIL N DO BEGIN
         IF L REM 10 = 1 THEN IOCONTROL(2); WRITEON(B(L)) END;

               BEGIN COMMENT
                         *** ENCONTRA UMA ARVORE OTIMA DE N=NOHS, DADA A FREQUEN=
                         *** CIA RELATIVA B(I) DE ENCONTRAR WD(I) E A FREQUENCIA
                         *** A(I) DE CONSULTAS ENTRE WD(I) E WD(I+1);
                 INTEGER ARRAY P,W,R(0::N,0::N); COMMENT
                         *** P(I,J), W(I,J), R(I,J) SIGNIFICAM RESPECTIVAMENTE  O
                         *** PESO PONDERADO, O PESO TOTAL E A RAIZ DA ARVORE OTI=
                         *** MA PARA AS INFORMACOES ENTRE WD(I) E WD(J+1), QUANDO
                         *** I < J+1. O TEMPO MEDIO DE PESQUISA NESTA ARVORE   EH
                         *** P(I,J)/W(I,J);


                REFERENCE (NODE) PROCEDURE CREATETREE(INTEGER VALUE I,J);
                 IF I=J THEN NODE(
                     CREATETREE (I,R(I,J)=1),CREATETREE(R(I,J),J)) ELSE NULL;
                 FOR I:=0 UNTIL N DO P(I,I):=W(I,1):=A(I);
                 FOR I:=0 UNTIL N DO FOR J:=I+1 UNTIL N DO
                   W(I,J):=W(I,J=1)+B(J)+A(J);
                 FOR K:=1 UNTIL N DO FOR I:=0 UNTIL N=K DO
 3=              BEGIN INTEGER IK,MN,MX;IK:=I+K;
                   MX:=IF K=1 THEN IK ELSE R(I,IK=1);MN:=P(I,MX=1)+P(MX,IK);
                   IF K>1 THEN FOR J:=MX+1 UNTIL R(I+1,IK) DO
                      IF P(I,J=1) + P(J,IK) < MN THEN
 44                      BEGIN MN:=P(I,J=1) + P(J,IK);MX:=J END;
                     P(I,IK):=MN+W(I,IK);R(I,IK):=MX
 =3              END;
                 WRITE("PESO PONDERADO",P(0,N),"SOMA DOS PESOS",W(0,N));
                 WRITE("O TEMPO MEDIO DE PESQUISA EH",P(0,N)/W(0,N));
           IOCONTROL(3);
           POSORDEM(N,CREATETREE(0,N));
           IOCONTROL(3);INTFIELDSIZE:=2;
           FOR I:=0 UNTIL N DO
 3=        BEGIN IOCONTROL(2);
             FOR J:=0 UNTIL N DO WRITEON(IF I<J THEN R(I,J) ELSE 0)
 =3        END;
 =2      END
 =1    END.
```

24 SECONDS IN COMPILATION, 03668 BYTES OF CODE GENERATED

```
REQUENCIAS DADAS SAO:
     28          30          22          31          21          25          34
     26          21          27          20          25          21          23
     20          23          28          20          16          17          25
     23          22          22          19          18          20          22
     14          20          18          21          18          17          18
     20          14          14          15          21          20          18
     15          12          17          17          13          10           8
     24          10          15          13          18          18          21
     16          13          15          11           6          15           6
      9          12          12           8          13          12           5
     15          11           8           8          11           8           9
      8           8           3          11           6           8           8
      6           6           6           7          14           9          12
 PONDERADO      11749  SOMA DOS PESOS       2001
MPO MEDIC DE PESQUISA EH       5.87156421789105
```

```
LGOLW1 JOB (4183,TJEK,,2),MSGLEVEL=(2,0),CLASS=M,TIME=(,30),          JOB    6
 REGION=325K
       EXEC PGM=ALGOLX
SPRINT DD SYSOUT=A,DCB=BLKSIZE=133
SIN DD *
42I = STEP WAS EXECUTED = COND CODE 0000
73I STEP /W        / START 74333.1810
74I STEP /W        / STOP  74333.1810 CPU    OMIN 01.93SEC MAIN 326K LCS    OK
75I  JOB /AALGOLW1/ START 74333.1810
76I  JOB /AALGOLW1/ STOP  74333.1810 CPU    OMIN 01.93SEC
```

9. BIBLIOGRAFIA

1. BAYER, R. & McCreight, E.: "Organization and Maintenance of Large Ordered Indexes", Acta Informatica (1972), 173-189

2. BRUNO, J. & Coffman Jr., E.G.: "Nearly Optimal Binary Search Trees", Information Processing 71, 99-103 - North-Holland Publishing Co., 1972

3. GORDON, Geoffrey: System Simulation, Prentice-Hall, 1969

4. GUIMARÃES, Célio: Notas de Aula sobre Classificação e Busca em Arquivos, COPPE/UFRJ - 1974

5. HOEL, Paul G. et alii: Introdution to Probability Theory, Houghton Mifflin Co., 1971

6. HU, T. C. & Tucker, A. C.: SIAM Journal of Applied Mathematics, 21 (1971), 514-532

7. KNUTH, D. E.: The Art of Computer Programming, vol. I, Fundamental Algorithms, Addison-Wesley, 1968

8. KNUTH, D. E.: The Art of Computer Programming, vol. 3, Sorting and Searching, Addison-Wesley, 1973

9. KNUTH, D. E.: "Optimum Binary Search Trees", Acta Informatica I, 14-25 (1971), 270, Springer-Verlag 1971

10. NAYLOR, Thomas H. et alii: Técnicas de Simulação em Computadores, Editora Vozes, 1971

11. NIEVERGELT, J.: "Binary Search Trees and File Organization", Computing Surveys 6, 3 (Set 1974)

12. NIEVERGELT, J. & Wong, C. K.: "On Binary Search Trees", Information Processing 71, 91-98 - North-Holland Publishing Co. 1972

13. SITES, Richard L.: Algol W Reference Manual, Feb. 1972, Stanford University (Biblioteca NCE/UFRJ: 6.000.015.333 S623A)

14. WALKER, W. A. & Gotlieb, C. C.: "A Top-down Algorithm for Constructing Nearly Optimal Lexicografic Trees", in Graph Theory and Computing, 303-323 - Academic Press, 1972

15. WIRTH, Niklaus & Hoare, C. A. R.: "A Contribution to Development of ALGOL", Communications of the ACM 9, 6(Junho 1966) pp. 413-431.