OPTIMAL MULTIWAY SEARCH TREES FOR VARIABLE SIZE KEYS

Jayme Luiz Szwarcfiter

NCE-0282

Outubro 1982

Jayme Luiz Szwarcfiter
Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
Caixa Postal 2324
20.001 Rio de Janeiro RJ

RESUMO

Dada uma sequência de n chaves de tamanho variável, consideram-se algumas árvores ótimas de busca. A construção de árvores multibifurcadas de busca de custo ótimo é NP-difícil, embora o problema seja solúvel em tempo pseudo-polinomial $O(n^3L)$ e espaço $O(n^2L)$, onde L é o limite dado para o tamanho da página. Tais árvores de espaço ótimo podem ser obtidas em tempo $O(n^3)$, e espaço $O(n^2)$, enquanto que aós de altura ótima são encontráveis em tempo $O(n^2 \log n)$ e espaço $O(n \log n)$. O princípio da monotonicidade não se aplica para os casos acima. A determinação de uma árvore B geral de custo ótimo é um problema também NP-difícil. Contudo, uma árvore B geral de altura 2 e tamanho mínimo de raiz pode ser construída em tempo $O(n \log n)$ e espaço $O(n)$. Além disso, caso o número de chaves na raiz seja fixado em M, a complexidade de tempo aumenta para $O(n^2M)$. Isto responde a uma conjectura de McCreight [11].

ABSTRACT

Given a sequence of n keys of variable sizes, some optimal search trees are considered. Constructing optimal cost multiway search trees is NP-hard, although it can be done in pseudo-polynomial time $O(n^3L)$ and space $O(n^2L)$ where L is the page size limit. Optimal space multiway search trees are obtained in $O(n^3)$ time and $O(n^2)$ space, while the optimal height problem is solved in $O(n^2 \log n)$ time and $O(n \log n)$ space. The monotonicity principle does not apply to the above cases. Finding optimal cost general B-trees is NP-hard. But a general B-tree of height 2 and minimal root size can be constructed in $O(n \log n)$ time and $O(n)$ space. In addition, if its root is restricted to contain M keys then the time complexity increases to $O(n^2M)$. This answers a conjecture by McMcCreight [11].

1. Introduction

Optimal cost binary search trees were constructed by Gilbert and Moore [2] in $O(n^3)$ time. Knuth [9] described an algorithm including gap weights and proved a monotonicity principle which decreased the time complexity to $O(n^2)$. Garey [1] and Wessner [13] found the above trees with a height restriction. The space bound for the mentioned algorithms is $O(n^2)$. Hu and Tucker [6] showed that optimal cost binary alphabetic trees can be found more efficiently in $O(n\log n)$ time and $O(n)$ space. Itai [7] presented an algorithm for optimal cost multiway alphabetic trees, while Vaishnavi, Kriegel and Wood [12] and Gotlieb [3] solved the corresponding problem for multiway search trees. The algorithms [3, 7, 12] require $O(n^3t)$ time and $O(n^2t)$ space whene t is the given maximum number of keys per node. Gotlieb [3] and Gotlieb and Wood [4] showed that an extension of the monotonicity principle does not apply to general multiway trees. However it does when the gap weights are absent, which reduces the running time in this case to $O(n^2t)$ [3]. Itai [7] describes a technique for reducing the factor t to logt, in the above time complexity.

Multiway trees for variable size keys were discussed in [5, 10] and with more detail by McCreight [11]. The present paper considers finding some optimal trees of this kind. It is shown that the problem of constructing an optimal cost multiway search tree is NP-hard. However it can be solved in pseudo-polinomial time $O(n^3L)$ and space $O(n^2L)$ where L is the given page size limit. Optimal space multiway search trees are constructed in $O(n^3)$ time and $O(n^2)$ space, while the optimal height problem is solved in time $O(n^2\log n)$ and space $O(n\log n)$. These two algorithms are described by a common formulation. Next it is shown that the problem of constructing an optimal cost general B-tree is NP-hard, although it also admits a pseudo-polynomial time solution. These trees are generalizations of B-trees, in which the lower and upper page size limits may be independent numbers. Following is described an $O(n\log n)$ time algorithm for constructing a general B-tree of height 2 and minimal root size. If additionaly its root is restricted to be formed by a given number M of keys then a different algorithm is applied. The latter has time

complexity $O(n^2M)$, while both require $O(n)$ space. This solves a problem by McCreight. In [11], $O(n\log n)$ time algorithms were described for $M = 2$ or 3 and the case of general M has been proposed as an open problem. Finally simple examples show that the monotonicity principle does not extend to any of the above multiway search tree algorithms. In particular it fails also for the optimal cost criterion with no gap weights. In any of the cases, the rightmost key in the root may move right when a new key is added at the left of the key sequence.

2. Preliminaries

Let $E = \langle e_1, \ldots, e_n \rangle$ be a sequence of elements called keys, each e_i with a non-negative integer size s_i and an arbitrary value y_i, satisfying $y_i < y_{i+1}$, $1 \le i < n$. Let T be a multiway search tree for the key sequence E and x a node of T. The size of x is the sum of the sizes of the keys in x. Let L be an integer. T has page limit (or limit) L whenever x has size at most L, for all nodes x of T. The level of x is the number of nodes in the path between x and the root of T. The height of the tree is the maximum level among the nodes. The space of T is its number of nodes. Now, suppose associated to E non-negative real key weights $p_1, \ldots, p_n$ and gap weights $q_0, \ldots, q_n$. Let $W = \sum_{1 \le i \le n} p_i + \sum_{0 \le j \le n} q_j$. Define $y_0 = -\infty$ and $y_{n+1} = +\infty$. Then (i) p_i/W and (ii) q_j/W are the probabilities that the search argument has value, respectively (i) y_i and (ii) strictly between y_j and y_{j+1}. The cost of T is the sum

$$\sum_{1 \le i \le n} p_i \, \mathrm{level}(e_i) + \sum_{0 \le j \le n} q_j \, (\mathrm{level}(e^*_j) - 1),$$

where e^*_j denotes the leaf of T representing the missing key range (y_j, y_{j+1}).

Except [6], all known optimal cost algorithms are of dynamic programming. They employ the following decomposition. Let $\langle e_i, \ldots, e_j \rangle$ be a key sequence and T the corresponding (t+1)-ary optimal cost tree. The problem of finding T is decomposed in

those of finding optimal cost trees T_L and T_R for $<e_i,\ldots,e_{k-1}>$ and $<e_{k+1},\ldots, e_j>$, respectively. Suppose the root r of T consists of m keys:

case 1: m = 1. Then e_k is the key in r and T_L , T_R are both t-ary trees.

case 2: m > 1. Then (i) e_k is the rightmost key in r and (ii) T_L, T_R are both t-ary trees, except that the root of T_L is restricted to m-1 keys.

This decomposition was first described in [2] for binary trees (case 1) and generalized in [7].

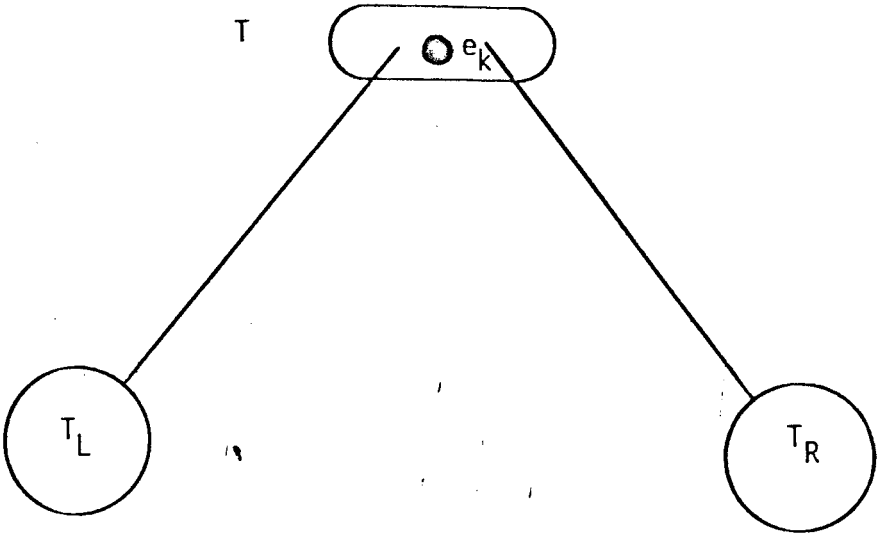
3. Optimal cost multiway search trees

In this section we use optimal tree meaning an optimal cost multiway search tree. Let $E = <e_i>$ be a sequence of keys with sizes s_i, key weights p_i and gap weights q_j, $1 \le i \le n$ and $0 \le j \le n$. Let L and C be positive integers, $L \ge s_i$.

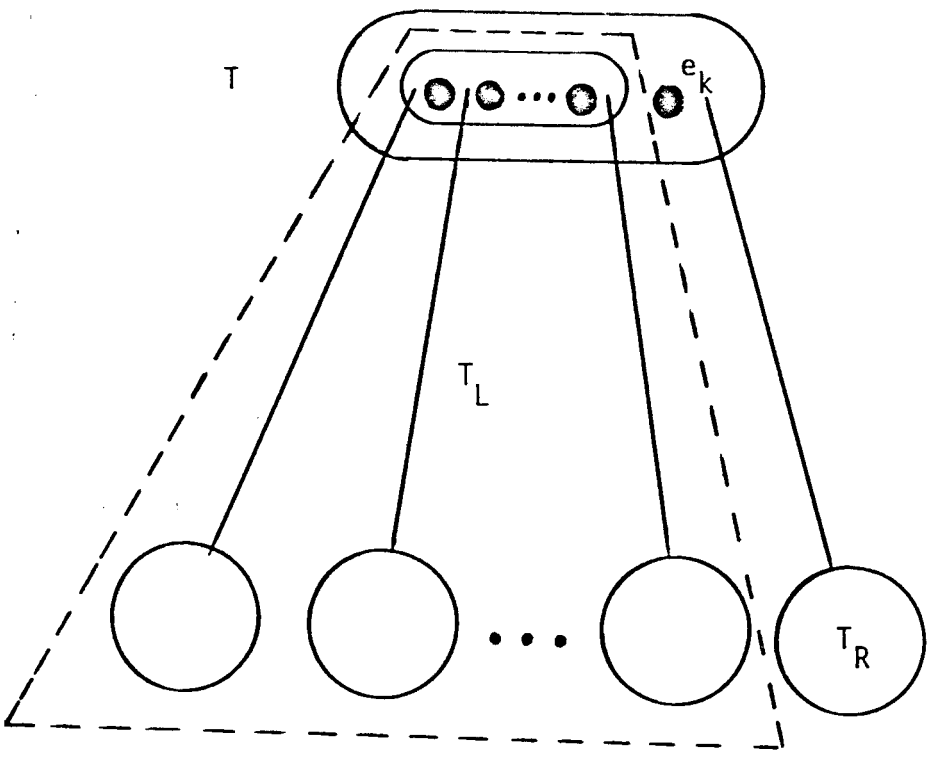
Lemma 1: Deciding whether there exists a multiway search tree for E having page limit L and cost $\le C$ is NP-complete.

Proof: Transformation from partition [8]. Its instance is a set $A = \{a_1,\ldots, a_n\}$, each a_i with a non-negative integer value v_i. Denote $b = \frac{1}{2} \sum_{a_j \in A} v_j$. Define the key sequence $<e_1,\ldots, e_n>$ such that (i) $s_i = p_i = v_i$, $1 \le i \le n$ and (ii) $q_j = 0$, $0 \le j \le n$. It follows that there exists a subset $A' \subseteq A$ satisfying $\sum_{a_j \in A'} v_j = b$ iff there exists a multiway search tree for E, having page limit b and cost $\le 3b$. Such a tree would have height 2 and the subset of keys in the root is in one-to-one correspondence with A'. ▲

Observe that the NP-completeness remains even if all gap weights are zero, each key size equals the corresponding key weight and the height h of the tree is fixed, h > 1.



case 1



case 2

Fig. 1: The decomposition rule

However an optimal tree can be constructed in pseudo-polynomial time. Let $E = \langle e_1, \ldots, e_n \rangle$ be a key sequence as above and L the limit of the desired optimal tree T for E. For $0 \le i < j \le n$ and $0 \le m \le L$ define:

$$w(i, j) = \sum_{1 < k \le j} p_k + \sum_{i \le k \le j} q_k$$

$$\alpha(i, j, m) = \begin{cases} \infty \text{ , when } s_k > m \text{ for all } k,\ i < k \le j. \text{ Otherwise} \\ \text{cost of the optimal tree of limit L for } \langle e_{i+1}, \ldots, e_j \rangle \\ \text{such that the root has size } \le m. \end{cases}$$

For $0 \le i \le n$ and $0 \le m \le L$ define $\alpha(i, i, m) = q_i$ and $w(i, i, m) = 0$. Clearly, $\alpha(0, n, L)$ is the cost of the desired optimal tree. Applying the decomposition of § 2, let e_k be the rightmost key in the root of T. Then T_R is an optimal tree of limit L. So is T_L, except that in case 2 its root is restricted to size at most $m-s_k$. Therefore

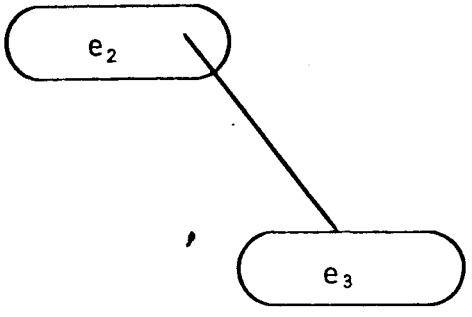
$$\alpha(i, j, m) = \begin{cases} \infty \text{ , when } s_k > m \text{ for all } i < k \le j. \text{ Otherwise} \\ \min\limits_{\substack{1 < k \le j \\ \text{such that} \\ s_k \le m}} \{\min[\alpha(i, k-1, m-s_k), \alpha(i, k-1, L) + w(i, k-1)] + p_k + \\ \qquad + \alpha(k, j, L) + w(k, j)\} \end{cases}$$

$$\text{for } 0 \le m \le L$$
$$0 \le i < j \le n.$$

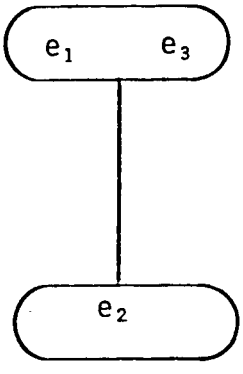
This algorithm requires $O(n^3L)$ time and $O(n^2L)$ space. The time bound can not be reduced by the application of the monotonicity principle. Unlike the fixed size key case [3], it does not apply even if all gap weights are zero. See figure 2.

4. Optimal height or space multiway search trees

Through this section, an optimal tree means an optimal height or space multiway search tree. Below is described an algorithm for finding an optimal tree for a given key sequence and limit. It uses the decomposition of section 2, as follows.



Optimal cost tree of page limit 2 for $<e_2, e_3>$



Optimal cost tree of page limit 2 for $<e_1, e_2, e_3>$

i	0	1	2	3
s_i	-	1	2	1
p_i	-	2	2	1
q_i	0	0	0	0

key sequence

Fig. 2: Failure of the monotonicity principle for the cost criterion

Let T be an optimal tree of limit L, having space S and height H. Suppose T is space optimal. Then T_R is space optimal and of limit L. Let S' be the space of T_R. In case 1, T_L is space optimal, has limit L and space S - S' - 1. So in case 2, except that its root is restricted to size at most L - s_k and its space is S - S'. Suppose now T is height optimal. Then T_L has height at most H - 1 in case 1 and at most H otherwise. The height of T_R is no more then H - 1. Clearly, T_L or T_R is height optimal, but we can restrict the search to the case in which both are.

A quasi-multiway search tree of limit L is a multiway search tree of limit L, except for the root whose size is unbounded. A multiway (or quasi multiway) search tree has parameter z when its height or space is z. Let E = $\langle e_1, \ldots, e_n \rangle$, each key e_i with size s_i. For given L and z > 0 define:

$$\alpha(i, j, z) = \begin{cases} 0, \text{ when } i > j. \text{ Otherwise} \\ \text{size of the root of the quasi-multiway search tree of limit L and parameter z for the key sequence } \langle e_i, \ldots, e_j \rangle \text{ and such that the root has minimal size.} \end{cases}$$

When $z \leq 0$, define $\alpha(i, j, z) = \infty$ for all i, j. Now let

$$\delta(i, j, z) = \begin{cases} \infty, \text{ when } \alpha(i, j, z') > L \text{ for all } z', 1 \leq z' \leq z. \text{ Otherwise} \\ \min_{1 \leq z' \leq z} \{z' \mid \alpha(i, j, z') \leq L\}. \end{cases}$$

$$\beta(i, j, z) = \begin{cases} 0, \text{ when } \delta(i, j, z) \leq z. \text{ Otherwise} \\ \infty. \end{cases}$$

In other words, $\delta(i, j, z)$ equals the minimal parameter z' of the optimal tree of limit L for $\langle e_i, \ldots, e_j \rangle$, such that $z' \leq z$. If no such tree exists then $\delta(i, j, z) = \infty$. Now, α can be computed by

$$\alpha(i, j, z) = \min_{i \leq k \leq j} \{\min[\alpha(i, k-1, f(k, j, z)), \beta(i, k-1, f(k, j, z)-1)] + s_k\}$$

$$\text{for } z = 2, 3, \ldots$$

$$1 \leq i \leq j \leq n,$$

where $f(k, j, z) = \begin{cases} z - \delta(k+1, j, z-1) & \text{for space minimization} \\ z - \beta(k+1, j, z-1) & \text{for height minimization.} \end{cases}$

The process is initiated by

$$\alpha(i, j, 1) = \sum_{1 \le k \le j} s_k, \quad 1 \le i \le j \le n .$$

After calculating each α the corresponding δ and β are evaluated. All computations are common for height or space minimization, except f. The process stops at the least z such that $\alpha(1, n, z) < \infty$. The following is clear.

Lemma 2: $\beta(i, j, z) = \infty \Rightarrow \beta(i, j+1, z) = \infty$ and
$\beta(i, j, z) = 0 \Rightarrow \delta(i, j, z+1) = \delta(i, j, z)$.

A straightforward implementation of this algorithm requires $O(n^3Z)$ time and $O(n^2Z)$ space, where Z is the parameter of the optimal tree. By using lemma 2 above it is possible to modify it in such a way that $\alpha(i, j, z)$ is computed at most twice for each pair i, j (at most once obtaining a value $\alpha > L$ and exactly once $\alpha \le L$). The new formulation runs in $O(n^3)$ time and $O(n^2)$ space.

Now, restrict attention to height minimization. Let $Q(i, j, z)$ denote the multiset of the finite operands in which the minimization of the above expression $\alpha(i, j, z)$ is carried out. Denote By Q_k the operand of Q corresponding to k, i.e. $Q_k = \min [\alpha(i, k-1, f(k, j, z)), \beta(i, k-1, f(k, j, z) - 1)] + s_k$. Suppose $\beta(i, j, z) = 0$. Then $Q(i, j+1, z)$ can be constructed from $Q(i, j, z)$ by the inclusion of the value $\min[\alpha(i, j, z), \beta(i, j, z-1)]+s_{j+1}$ and the exclusion of each Q_k for which $\beta(k+1, j+1, z-1) = \infty$. This allows the computation of $\alpha(i, j+1, z)$. Now suppose $\beta(i, j+1, z) = \infty$. In this case, call j+1 a i-starting point . Then construct $Q(i, j+1, z+1)$ by including in $Q(i, t, z)$ the operands $s_{t+1}, s_{t+2}, \ldots, s_{j+1}$, where t is the largest i-starting point $<j+1$. This allows the computation of $\alpha(i, j+1, z+1)$. Is this scheme, the progress of the computation is i = n, n-1,..., 1. Only one of the α values need to be kept, while redundant β's may be avoided.

The above observations and the use of priority queues reduce the time bound to $O(n^2 \log n)$ and space to $o(nZ)$. Clearly, the minimal height Z is $O(\log n)$.

The monotonicity principle can not be applied to any of the above algorithms. See figure 3.

5. Optimal cost general B-trees

Let L_1 and L_2 be integers, $0 < L_1 \leq L_2$. A general B-tree is a multiway search tree T satisfying (i) $1 \leq size(r) \leq L_2$, where r is the root of T, (ii) $L_1 \leq size(x) \leq L_2$, for all nodes $x \neq r$ of T and (iii) level(x) = height(T), for all leaves x of T. L_1 and L_2 are respectively the lower and upper page limits of T or simply limits and denoted as (L_1, L_2). Clearly, a B-tree is a general B-tree with $L_1 = \lceil L_2/2 \rceil$.

Lemma 3: Let E be a key sequence, L_1, L_2 and C positive integers. Deciding whether there exists a general B-tree for E of limits (L_1, L_2) and cost $\leq C$ is NP-complete.

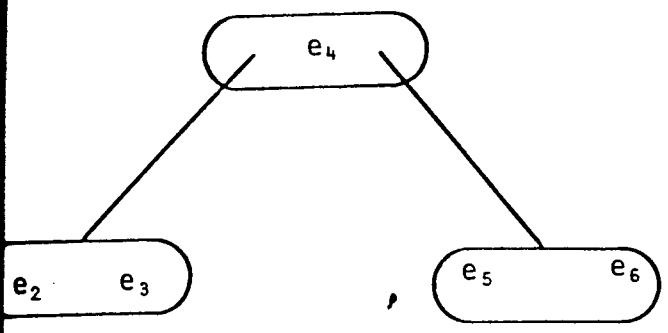
Proof: Transformation from partition. Its instance is a set $A = \{a_1, \ldots, a_n\}$ each a_i having a non-negative integer value v_i such that $b = \frac{1}{2} \Sigma v_i$ is an integer. Construct a key sequence $E = \langle e_1, \ldots, e_{2n+3} \rangle$ as formed by 4 types of keys.

Type K_1 : key e_1, with $s_1 = p_1 = n(b+1)$

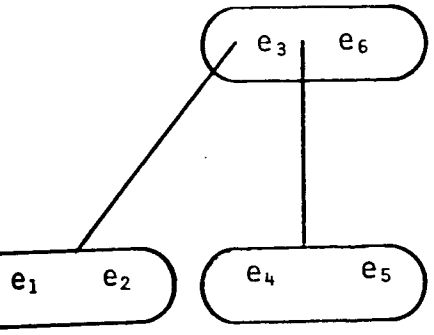
K_2 : key e_2, with $s_2 = p_2 = n$

K_3 : keys $e_3, e_5, \ldots, e_{2n+3}$, with $s_k = p_k = 1$, $k = 3, 5, \ldots, 2n+3$

K_4 : keys $e_4, e_6, \ldots, e_{2n+2}$, with $s_k = p_k = nv_{(k-2)/2}$, $k = 4, 6, \ldots, 2n+2$



Optimal height and space tree of page limit 4 for $\langle e_2, e_3, e_4, e_5, e_6 \rangle$, having minimal root size.



Optimal height and space tree of page limit 4 for $\langle e_1, e_2, e_3, e_4, e_5, e_6 \rangle$, having minimal root size.

i	1	2	3	4	5	6
s_i	2	2	2	2	2	1

key sequence

Fig. 3: Failure of the monotonicity principle for either height or space criterion.

All $q_i = 0$, $0 \le i \le 2n+3$. See figure 4. It follows that there exists a subset $A' \subseteq A$ such that $\sum_{a_i \in A'} v_i = b$ if and only if there exists a general B-tree for E with limits (1, nb+n) and cost $\le 5nb + 5n + 2$. Such a tree would have height 2 and the subset of keys in the root is in one-to-one correspondence with A'.

Observe that the problem remains NP-complete even if all $q_j = 0$, all $s_i' = p_i$ and the height h of the tree is fixed, $h > 1$. Again the NP-completeness is not strong. An optimal cost general B-tree can be found in pseudo-polynomial time by conveniently extending the algorithm of section 3.

6. General B-trees of height 2

Given a key sequence $E = \langle e_1, \ldots, e_n \rangle$, each e_i with size s_i and given integers L_1, L_2 with $0 < L_1 \le L_2$, find a general B-tree T for E having limits (L_1, L_2), height 2 and minimal root size. We assume $\Sigma s_i > L_2$, otherwise there is no reason for a tree of height 2.

Clearly, T can be determined just by finding the subset of keys which forms the root. We construct an acyclic digraph with vertex set $\{u_0, u_1, \ldots, u_{n+1}\}$. There is one directed edge (u_i, u_j) and a distance d_{ij} for $0 \le i < j \le n+1$, as follows:

$$d_{ij} = \begin{cases} s_j, \text{ when } L_1 \le \sum_{i<k<j} s_k \le L_2. \text{ Otherwise} \\ \infty, \end{cases}$$

where $s_{n+1} = 0$. Let D_i be the length $d_{ij} + \ldots + d_{k,n+1}$ of the shortest path P_i from u_i to u_{n+1}. If $D_0 > L_2$ then T does not exist. Otherwise the root of T is formed by the keys $\{e_j \mid u_j \in P_0 - \{u_0, u_{n+1}\}\}$. A straightforward implementation of this process gives an $O(n^2)$ time and space algorithm. However it can be improved by taking advantage of the special distribution of the edge distances. For $0 \le i \le n$, define:

$$\delta_1(i) = \begin{cases} n+2, \text{ when } \sum_{i<k\le n+1} s_k < L_1. \text{ Otherwise} \\ \min \{j \mid \sum_{i<k<j} s_k \ge L_1,\ i < j \le n+1\} \end{cases}$$

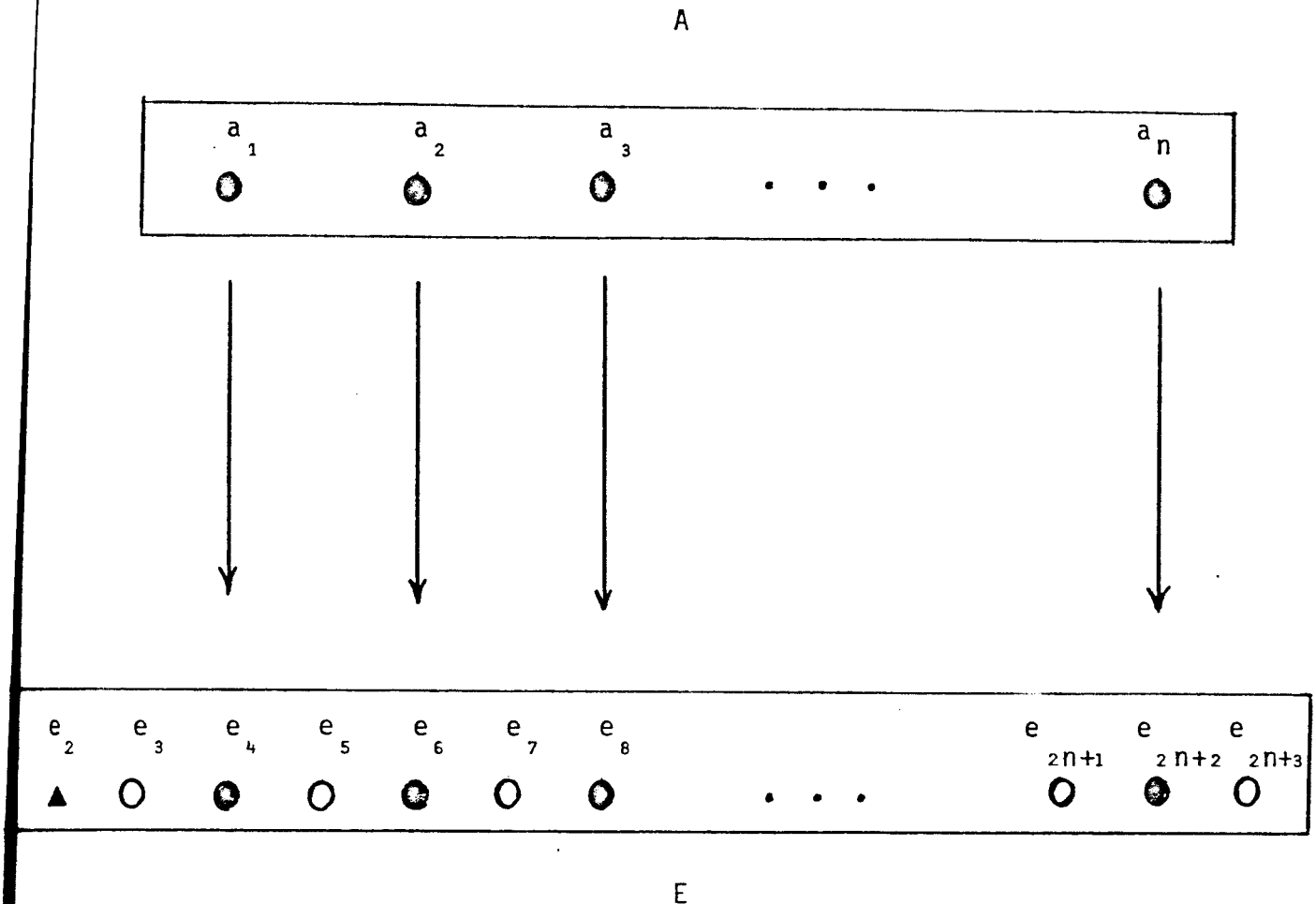


Fig. 4: Construction of the key sequence (lemma 3).

$$\delta_2(i) = \begin{cases} n+1, & \text{when } i = n. \text{ Otherwise} \\ \max \{j \mid \sum_{i< k< j} s_k \leq L_2,\ i< j\leq n+1\}. \end{cases}$$

In other words, if u_j is the vertex that follows u_i in P_i then j is in the interval $[\delta_1(i), \delta_2(i)]$. For $0\leq i\leq n$, define the multiset

$$Q_i = \{d_{ik} + D_k \mid i< k\leq n+1\}.$$

Then D_0 can be computed by

$$D_{n+1} = 0$$

$$D_i = \min Q_i$$

for i = n, n-1,..., 0. After each iteration $i>0$, Q_{i-1} is constructed from Q_i by inserting $\{s_k + D_k \mid \delta_1(i-1)\leq k< \delta_1(i)\}$ and deleting $\{s_k + D_k \mid \delta_2(i-1)< k\leq \delta_2(i)\}$. Using a priority queue to represent the Q_i's is possible to implement the algorithm in O(nlogn) time and O(n) space. Observe that the digraph is not explicitly constructed in this scheme. The algorithm would find a multiway search tree of limit L_2, height 2 and minimal root size by setting $L_1 = 0$ and $\delta_1(i) = i+1$, $0\leq i\leq n$.

Consider now the following variation of the above problem. In addition to the stated conditions, the root of T is required to be formed by a given number M of keys [11]. A dynamic programming algorithm can be described for this variation based again in the decomposition of § 2. Let e_k be the righmost key in the root of T. Then $L_1\leq \sum_{k< i\leq n} s_i \leq L_2$. In case 1, $L_1\leq \sum_{1\leq i< k} s_i\leq L_2$. In case 2, T_L is a general B-tree for $<e_1,\ldots, e_{k-1}>$ of limits (L_1, L_2), height 2, minimal root size and having M-1 keys in the root. For $1\leq i\leq j\leq n$ and $m>0$, define:

$$\alpha(i,j,m) = \begin{cases} \text{size of the root of the general B-tree for } <e_i,\ldots, e_j> \text{ of limits } (L_1, L_2), \text{ height 2, minimal root size and having } m \text{ keys in the root.} \\ \infty \text{ , whenever the above tree does not exist.} \end{cases}$$

For $1 \le i \le j \le n$, define:

$$\alpha(i,j,0) = \begin{cases} 0, & \text{when } L_1 \le \sum_{i \le k \le j} s_k \le L_2. \text{ Otherwise} \\ \infty. \end{cases}$$

The solution is clearly $\alpha(1, n, M)$ which can be computed by the following equation:

$$\alpha(1,j,m) = \min_{2m \le k \le j-1} \{\alpha(1, k-1, m-1) + s_k + \alpha(k+1, j, 0)\},$$

$$\text{for } m = 1, 2, \ldots, M$$
$$j = 2m+1, 2m+2, \ldots, n.$$

Note that if $i_2 - i_1 < 2i_3$ then $\alpha(i_1, i_2, i_3) = \infty$. Using the functions $\delta_1(i)$ and $\delta_2(i)$ as defined it is possible to evaluate $\alpha(i, j, 0)$ in constant time and compute $\alpha(1, n, M)$ in $O(n^2 M)$ time and $O(n)$ space.

7. Conclusions

Problems of finding some optimal multiway search trees and general B-trees for variable size keys have been considered. In special, finding optimal cost trees is NP-hard for both cases, although admit pseudo-polynomial time algorithms. But in some applications the limits of the trees have order of magnitude not greater then the number of keys. Clearly, the algorithms are polynomial for this class of problems.

It might be interesting to consider an alternative strategy for B-trees of variable size keys. Namely, to adopt as lower limit a given number of keys, while maintaining the size as upper limit. The optimal cost problem remains of course NP-hard. But the trees may become easier to manipulate. For instance, an optimal height B-tree can be found in polynomial time if this alternative is adopted. When controling nodes only by sizes, the construction of a B-tree of height 2 can be done in polynomial time (section 6). It would be worth examining the case height ≥ 3.

ACKNOWLEDGEMENTS: To Ysmar V. Silva FQ for all the discussions and insightful remarks.

REFERENCES

1. GAREY, M. R. Optimal binary search trees with restricted maximal depth. SIAM J. COMP. 2 (1974), 101-110.

2. GILBERT, E. N., and MOORE, E. F. Variable-length binary encodings . Bell System Tech. J. 38 (1959), 933-968.

3. GOTLIEB, L. Optimal multiway search trees. SIAM J. COMP. 10 (1981), 422-433.

4. GOTLIEB, L., and WOOD, D. The construction of optimal multiway search trees and the monotonicity principle, Intern. J. Comp. Math., Sc. A 9 (1981), 17-24.

5. HOROWITZ, E., and SAHNI, S. Fundamentals of Data Structures. Computer Science Press, Potomac, 1976.

6. HU, T.C., and TUCKER, A.C. Optimum computer search trees. SIAM J. Appl. Math. 21 (1971), 514-532.

7. ITAI, A. Optimal Alphabetic Trees. SIAM J. COMP. 5 (1976), 9-18.

8. KARP, R. M. Reducibility among combinatorial problems. In Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, New York, 1972, 85-103.

9. KNUTH, D. E. Optimum binary search trees. Acta Informat. 1 (1971), 14-25.

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

11. McCREIGHT, E.M. Pagination of B*-trees with variable length records. Comm. ACM 20 (1977), 670-674.

12. VAISHNAVI, V. K., KRIEGEL, H. P. and WOOD, D. Optimum multiway search trees. Acta Informat. 14 (1980), 119-133.

13. WESSNER, R. L. Optimum alphabetic search trees with restricted maximal height. Inform. Proc. Lett. 4 (1976), 90-94.