stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE STATISTIEK SW 52/77 OKTOBER
(DEPARTMENT OF MATHEMATICAL STATISTICS)

R. KAAS

INTERPOLATION SEARCH FOR ORDER STATISTICS OF A UNIFORM DISTRIBUTION

Preprint

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 68A20,68A50,62G90

Interpolation search for order statistics of a uniform distribution [*)]

by

R. Kaas

ABSTRACT

This paper contains an analysis of the mean running time of interpolation search in an ordered file, under the assumption that the items to be searched have been randomly obtained from a known continuous distribution. We prove the 0(log log n) average time bound found, but incorrectly proved by ITAI and PERL [3].

KEY WORDS & PHRASES: *interpolation search, complexity.*

*) This report will be submitted for publication elsewhere

1. INTRODUCTION

Searching in an indexed ordered file is a very common problem in data manipulation. The situation considered is that we have an increasing sequence of real numbers $x_1, x_2, \ldots, x_n$ and a number y of which we would like to determine whether and with what index it occurs in the row. This situation might arise when we consider the names of a set of people, punched on cards together with other relevent variables, as 26-ary numbers.

An efficient tool for obtaining good algorithms is the "divide and conquer"-technique as described in [1]. We divide the problem into two subproblems and recursively solve one of these subproblems. In our case this is achieved by choosing a "cutting-index" c, determining whether $x_c = y$ (then we are ready), $x_c > y$ (then we consider $x_1, x_2, \ldots, x_{c-1}$) or $x_c < y$ (then we consider $x_{c+1}, x_{c+2}, \ldots, x_n$). If we do not use the fact that the numbers are ordered and choose for c every time the lowest possible index, i.e. $c = 1,2,3,\ldots.$, we obtain an algorithm that runs in $O(n)$ time[*)]. An $O(\log n)$ (average and worst case) algorithm arises if we choose every time the median index of the set of possible indices, cf. the binary search algorithm in [1] and [4].

Under the extra assumption that the data have been obtained as a random uniform-(0,1) sample the index of y has a binomial-(n,y) distribution. We consider the average running time under this assumption of the algorithm computing the cutting-index every time by linear interpolation, i.e. taking $c = n \cdot y$ rounded off. In section 2 we prove that if $m \geq {}^2\log{}^2\log n$, c_m is the m^{th} cutting-index obtained in this way, and $y = x_i$, $|c_m - i|$ has an expected value of less than 2.636.... ITAI and PERL [3] proved an upper bound of two for this quantity, but their proof contained some unjustified approximations. In section 3 we prove that given x_c the row $x_1, x_2, \ldots, x_{c-1}$ may be treated as an ordered uniform-$(0,x_c)$ sample. So with respect to the interpolation search algorithm the subproblems to be solved are isomorphic to the main problem. Section 4 contains the results of a Monte Carlo investigation of the algorithm, not only with underlying uniform distribution functions (d.f.'s), but also with normal, polygonal and other d.f.'s.

[*)] An algorithm runs in (average) time $O(f(n))$ if as $n \to \infty$ the (average) number of elementary computing steps (such as addition, division) needed by the algorithm to come to a solution is $O(f(n))$.

Recently we learned that YAO and YAO [5] have also proved the same average time bound. We think our construction of the proof: a row of conditionally binomial r.v.'s, each having as expected value the absolute error of the previous one, is more elegant and transparent than theirs. They were able to prove the optimality of the interpolation search algorithm, up to an additive constant. They did not pay attention to the isomorphism of the main problem and the subproblems. This is, however, not completely trivial, since conditional distributions are involved with conditions of zero probability.

Throughout the paper all random variables (r.v.'s) will be underlined and v.v., and all logarithms are to the base 2.

2. THE INTERPOLATION ALGORITHM AND ITS COMPLEXITY

We will start by giving some inequalities necessary to estimate the complexity of the algorithm. The expectation of an r.v. $\underline{x}$ will be denoted by $E\underline{x}$ and its variance $E(\underline{x}-E\underline{x})^2$ by $\mathrm{var}(\underline{x})$ or $\sigma_{\underline{x}}^2$.

LEMMA 2.1. *For every* r.v. $\underline{x}$ *the following inequalities hold:*

(2.1) $\quad E\underline{x} \le \sqrt{E\underline{x}^2}$, *and*

(2.2) $\quad E|\underline{x}-E\underline{x}| \le \sqrt{\mathrm{var}(\underline{x})}$.

PROOF. $0 \le E(\underline{x}-E\underline{x})^2 = E(\underline{x}^2-2\underline{x}.E\underline{x}+(E\underline{x})^2) = E\underline{x}^2-(E\underline{x})^2$, so $(E\underline{x})^2 \le E\underline{x}^2$ and $E\underline{x} \le \sqrt{E\underline{x}^2}$. Applying this result to the r.v. $\underline{y} = |\underline{x} - E\underline{x}|$ one obtains:

$$E|\underline{x}-E\underline{x}| = E\underline{y} \le \sqrt{E\underline{y}^2} = \sqrt{E|\underline{x}-E\underline{x}|^2} = \sqrt{\mathrm{var}(x)} . \qquad \Box$$

For convenience we will write $tt(n)$ for 2^{2^n}. Now $n = tt(\log\log n)$ and $\sqrt{n} = tt(\log\log n-1)$. Let $m = \lceil \log\log n \rceil$, $\lceil t \rceil$ being the smallest integer not less than t. In our algorithm we will do m iteration steps each reducing the complexity of the problem considered from x, say, to $\sqrt{x} + \frac{1}{2}$. We want to prove that after taking m steps with original complexity n the resulting problem has a bounded complexity. Let us define f_n by

$$f_0(x) = x,$$

$$f_{n+1}(x) = \sqrt{f_n(x)} + \tfrac{1}{2}, \qquad n = 0,1,2,\dots .$$

The row $(t_n)_{n=1}^{\infty}$ is defined by $t_n = f_n(tt(n))$, $n = 0,1,2,\dots$.

LEMMA 2.2. *For* $n = 0,1,\dots$ *and* $i = 0,1,\dots,n$

(2.3) $\qquad f_{n+1}(x) = f_n(\sqrt{x} + \tfrac{1}{2})$, *and*

(2.4) $\qquad f_i(tt(n)) \geq tt(n-i)$.

PROOF. $f_{n+1}(x) = \sqrt{f_n(x)} + \tfrac{1}{2} = \sqrt{f_{n-1}(\sqrt{x}+\tfrac{1}{2})} + \tfrac{1}{2} = f_n(\sqrt{x}+\tfrac{1}{2})$, so (2.3) follows by induction. (2.4) holds for $i = 0$, and $f_{i_0+1}(tt(n)) = \sqrt{f_{i_0}(tt(n))} + \tfrac{1}{2}$ $\geq \sqrt{tt(n-i_0)} + \tfrac{1}{2} > tt(n-i_0-1)$. □

LEMMA 2.3. $0 < t_{n+1} - t_n \leq 2^{-n}/tt(n)$ *for every* n.

PROOF. $t_{n+1} - t_n = f_{n+1}(tt(n+1)) - f_n(tt(n)) = f_n(tt(n)+\tfrac{1}{2}) - f_n(tt(n))$. Since $f_n'(x) = (\tfrac{1}{2})^n \prod_{i=0}^{n-1} [f_i(x)]^{-\frac{1}{2}}$, we see that $f_n(x)$ increases monotonically with x, but $f_n'(x)$ decreases. Now the mean value theorem directly yields

$$f_n(tt(n)+\tfrac{1}{2}) - f_n(tt(n)) \leq \tfrac{1}{2}\, f_n'(tt(n)) =$$

$$= 2^{-n-1} \prod_{i=0}^{n-1} f_i(tt(n))^{-\frac{1}{2}} \leq 2^{-n-1} \prod_{i=0}^{n-1} tt(n-i)^{-\frac{1}{2}},$$

because of (2.4). This product is equal to 2 to the power

$$-n-1 - \sum_{i=0}^{n-1} 2^{n-i-1} = -n-2^n,$$

so the lemma is proven. □

Now $\lim_{m\to\infty} t_m \leq t_n + \sum_{i=n}^{\infty} 2^{-i}/tt(i) = t_n + 2^{-n}/tt(n) + \sum_{i=n+1}^{\infty} 2^{-i}/tt(i)$ $\leq t_n + 2^{-n}/tt(n) + 2^{-(n+1)}/tt(n+1) \sum_{i=0}^{\infty} 2^{-i} = t_n + 2^{-n}/tt(n) + 2^{-n}/tt(n+1)$.

t_6, computed by a 48 bits real mantissa computer, is 2.636339126878... , and the difference between t_6 and $\lim t_m$ is less than $2^{-70} + 2^{-134}$, so we have the following result:

COROLLARY 2.1. *If a row of positive real numbers* $(r_i)_{i=0}^{\infty}$ *satisfies* $r_{i+1} \leq \sqrt{r_i} + \frac{1}{2}$ *for every* i, *then* $m \geq \lceil \log \log r_0 \rceil$ *implies* $r_m \leq 2.636339126978...$. □

We will now give a formal definition of the algorithm announced in the introduction. It manipulates a random vector $\underline{V}_i$ having as components the ith approximation of the index wanted, the lower and the upper index of the part of the file still under consideration, and the parameters of the conditional binomial distribution given these bounds, of the index looked for.

In the following n is a natural number; $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n$ is a sorted random sample from a uniform -(0,1) distribution. The r.v.'s $\underline{x}_0$ and $\underline{x}_{n+1}$ are defined by $P(\underline{x}_0 = 0,\ \underline{x}_{n+1} = 1) = 1$, the r.v. $\underline{i}$, the index wanted, is defined by:

(2.5) $$\underline{i} = \max\{i \mid \underline{x}_i \leq y\}.$$

Finally m is the smallest integer not less than log log n ($m \coloneqq \lceil \log \log n \rceil$). The formal definition of the algorithm is given by:

DEFINITION 2.1. For $i = 0,1,\ldots,m$ let $\underline{V}_i = (\underline{c}_i, \underline{\ell}_i, \underline{u}_i, \underline{n}_i, \underline{p}_i)$ be a random vector. With probability one $\underline{V}_0 = (0,0,n+1,n,y)$, and for $i = 1,2,\ldots,m$:

(2.6) $\underline{c}_i = \underline{\ell}_{i-1} + \underline{n}_{i-1} \cdot \underline{p}_{i-1}$ rounded off to the nearest integer;

(2.7) if $\underline{x}_{\underline{c}_i} \leq y$, then $\underline{\ell}_i = \underline{c}_i$ and $\underline{u}_i = \underline{u}_{i-1}$,
else $\underline{u}_i = \underline{c}_i$ and $\underline{\ell}_i = \underline{\ell}_{i-1}$;

(2.8) $\underline{n}_i = \underline{u}_i - \underline{\ell}_i - 1$,
$\underline{p}_i = (y - \underline{x}_{\underline{\ell}_i}) / (\underline{x}_{\underline{u}_i} - \underline{x}_{\underline{\ell}_i})$. □

The following lemma concerns the conditional contribution of $\underline{i}$ in successive stages of the execution of the algorithm. In the sequel V will denote a suitable vector, $V = (c,\ell,u,k,p)$, with $0 \leq \ell \leq c \leq u \leq n+1$, $k = u - \ell - 1$, either $c = u$ or $c = \ell$, and

$$p = (y-x_\ell)/(x_u-x_\ell).$$

<u>LEMMA 2.4</u>. *Under the condition* $\underline{V}_i = V$ *the* r.v. $\underline{i} - \ell$ *has a binomial*-(k,p) *distribution, for* $i = 0, 1, \ldots, m$. *For* $i = 0$ *this reduces to:*

$$P(\underline{i} = j) = \binom{n}{j} y^j (1 - y)^{n-j}.$$

<u>PROOF</u>. First assume $i = 0$. Now $\underline{i} = j$ is equivalent to the event that of n independent uniformly-(0,1) distributed r.v.'s exactly j do not exceed y. This amounts to n independent trials, each with probability y of success.

Now assume $i > 0$. In the next section it is proved, that under the condition $\underline{x}_\ell = x_\ell$ and $\underline{x}_u = x_u$ the joint distribution function of $\underline{x}_{\ell+1}$, $\underline{x}_{\ell+2}, \ldots, \underline{x}_{u-1}$ is that of an ordered random sample of size $u - \ell - 1$ from a uniform-(x_ℓ, x_u) distribution. But then we have a copy of the situation with $i = 0$ after performing the following transformations:

$$n := u - \ell - 1;$$

$$\underline{x}_i := (\underline{x}_{\ell+i} - x_\ell)/(x_u - x_\ell), \quad i = 1,2,\ldots,u-\ell-i;$$

$$y := (y - x_\ell)/(x_u - x_\ell). \qquad \square$$

The following two lemma's concern the first reduced absolute moments of $\underline{i}$ under the condition $\underline{V}_i = V = (c, \ell, u, k, p)$.

<u>LEMMA 2.5</u>. $E(|\underline{i} - \underline{c}_i| \mid \underline{V}_i = V) \geq k.p.(1-p)$.

<u>PROOF</u>. Because of (2.7) either $\underline{c}_i = \ell$ or $\underline{c}_i = u$. Now

$$E(|\underline{i} - \ell| \mid \underline{V}_i = V) = k.p \geq k.p.(1-p), \text{ but also}$$

$$E(|\underline{i} - u| \mid \underline{V}_i = V) = k + 1 - E(|\underline{i} - \ell| \mid \underline{V}_i = V) =$$

$$= k + 1 - k.p > k.p.(1-p), \text{ so the lemma is proved.} \qquad \square$$

<u>LEMMA 2.6</u>. $E(|\underline{i} - \underline{c}_{i+1}| \mid \underline{V}_i = V) \leq \sqrt{k.p.(1-p)} + \frac{1}{2}$.

<u>PROOF</u>. Because of lemma 2.4 $\text{var}(\underline{i} \mid \underline{V}_i = V) = k.p.(1-p)$. Define s by

$s = \ell + k.p$, then $s = E(\underline{i} \mid \underline{V}_i = V)$, and

$$E(|\underline{i} - \underline{c}_{i+1}| \mid \underline{V}_i = V) = E(|\underline{i} - s + s - \underline{c}_{i+1}| \mid \underline{V}_i = V) \le$$

$$\le E(|\underline{i} - s| \mid \underline{V}_i = V) + E(|s - \underline{c}_{i+1}| \mid \underline{V}_i = V) \le$$

$$\le \sqrt{k.p.(1-p)} + \tfrac{1}{2}, \text{ because of (2.4) and (2.6).} \qquad \square$$

<u>COROLLARY 2.2</u>. $E(|\underline{i} - \underline{c}_{i+1}|) \le \sqrt{E(|\underline{i} - \underline{c}_i|)} + \frac{1}{2}$, *for* $i < m$.

<u>PROOF</u>. Because of the two preceding lemma's

$$E_{\underline{V}_i}\left[E(|\underline{i}-\underline{c}_{i+1}| \mid \underline{V}_i) - \sqrt{E(|\underline{i}-\underline{c}_i| \mid \underline{V}_i)} - \tfrac{1}{2}\right] \le 0.$$

So

$$E(|\underline{i}-\underline{c}_{i+1}|) \le E_{\underline{V}_i}\sqrt{E(|\underline{i}-\underline{c}_i| \mid \underline{V}_i)} + \tfrac{1}{2} \text{ (using (2.1))}$$

$$\le \sqrt{E(|\underline{i}-\underline{c}_i|)} + \tfrac{1}{2}. \qquad \square$$

<u>THEOREM 2.1</u>. $E(|\underline{i}-\underline{c}_m|) < 2.636339126878\ldots$.

<u>PROOF</u>. Apply corollary 2.1 with $r_i = E(|\underline{i} - \underline{c}_i|)$. $r_0 = E\underline{i} = n.y$, so certainly $m \ge {}^2\log{}^2\log r_0$, and the relation between successive r_i is satisfied because of corollary 2.2. $\square$

The theorem states, that if after m interpolations we proceed with cutting-indices $\underline{c}_m+1$, $\underline{c}_m+2$, ... (if $x_{\underline{c}_m} < y$) or with $\underline{c}_m-1$, $\underline{c}_m-2$, ... (if $x_{\underline{c}_m} > y$), 2.636.... steps on an average are sufficient to reach $\underline{i}$. If $x_{\underline{c}_m} < y$ and no $\underline{x}_i$ equals y we must do one more step, since $\underline{i}$ must satisfy $\underline{x}_i < y < \underline{x}_{i+1}$, so it is necessary to know the value of $\underline{x}_{i+1}$.

We conclude this section with two final remarks about the algorithm. In the first case its worst case time behaviour is O(n): suppose $n \cdot y < \frac{1}{2}$ and consider all samples with $\underline{x}_n < y$, having total probability y^n. Now all cutting-indices will have the value zero, so n sequential tail steps are necessary. This should come as no surprise to us, since our extra assumption about the underlying distribution is then useless. In the second place it is not necessary that the underlying distribution is uniform-(0,1). If it is continuous with known d.f. F then $F(\underline{x}_1)$, $F(\underline{x}_2),\ldots,F(\underline{x}_n)$ are a random sample from a uniform distribution, since (defining $F^{-1}(y) = \inf\{z \mid F(z) \geq y\}$)

$$P(F(\underline{x}) \leq x) = P(\underline{x} \leq F^{-1}(x)) = F(F^{-1}(x)) = x.$$

3. THE ISOMORPHISM OF SUBPROBLEMS

In this section $\underline{y}_1,\underline{y}_2,\ldots,\underline{y}_n$ will be a sequence of independent r.v.'s each with d.f. F and density function f. The i^{th} smallest of the $\underline{y}_i$ is denoted by $\underline{x}_i$. We first give (without proof) a basic theorem on order sattistics. More details can be found in DAVID [2].

THEOREM 3.1. *Suppose we have an increasing row of natural numbers* $r_1,r_2,\ldots,r_k$ *with* $1 \leq r_1$ *and* $r_k \leq n$, *and a non-decreasing row of real numbers* $x_1,x_2,\ldots,x_k$ *with* $0 \leq x_1$ *and* $x_k \leq 1$. *Furthermore let* $r_0 = 0$, $r_{k+1} = n+1$, $x_0 = -\infty$ *and* $x_{k+1} = +\infty$. *Then the joint density of* $\underline{x}_{r_1},\underline{x}_{r_2},\ldots,\underline{x}_{r_k}$ *in* $(x_1,\ldots,x_k)$ *is given by:*

$$(3.1)\quad \begin{aligned} &f_{r_1,\ldots,r_k}(x_1,\ldots,x_k) = \\ &= n! \prod_{i=1}^{k} f(x_i) \prod_{i=0}^{k} \frac{(F(x_{i+1})-F(x_i))^{r_{i+1}-r_i-1}}{(r_{i+1}-r_i-1)!} \end{aligned}$$

THEOREM 3.2. *The joint density of* $\underline{x}_1,\underline{x}_2,\ldots,\underline{x}_{j-1}$ *under the condition* $\underline{x}_j = b$ *in the point* $(x_1,x_2,\ldots,x_{j-1})$ *with* $0 < F(b)$ *and* $0 \leq x_1 \leq x_2 \leq \ldots \leq x_{j-1} \leq b$ *is given by:*

$$(3.2)\quad \begin{aligned} &f_{1,2,\ldots,j-1|\underline{x}_j=b}(x_1,\ldots,x_{j-1}) = \\ &= (j-1)!\,\frac{f(x_1)f(x_2)\ldots f(x_{j-1})}{F(b)^{j-1}}\,. \end{aligned}$$

PROOF. By definition this density is equal to

$$\frac{f_{1,2,\dots,j-1,j}(x_1,x_2,\dots,x_{j-1},b)}{f_j(b)} .$$

The numerator equals

$$n!\ f(x_1)\ f(x_2)\ \dots\ f(x_{j-1})\ f(b)\ \frac{(1-F(b))^{n-j}}{(n-j)!}$$

and the denominator equals

$$n!\ f(b)\ \frac{F(b)^{j-1}}{(j-1)!} \cdot \frac{[1-F(b)]^{n-j}}{(n-j)!} ,$$

and the quotient is

$$\frac{(j-1)!f(x_1)f(x_2)\dots\dots f(x_{j-1})}{F(b)^{j-1}} . \qquad \square$$

Because of (3.1) the joint density of all order statistics in $(x_1,x_2,\dots\dots,x_n)$ with $x_1 \le x_2 \le \dots \le x_n$ is given by:

(3.3) $$f_{1,2,\dots,n}(x_1,\dots,x_n) = n!\ f(x_1)\ f(x_2)\ \dots\ f(x_n).$$

If $\underline{u}$ has a uniform-(0,1) distribution and $0 \le t \le b \le 1$, then

(3.4) $$P(\underline{u} \le t |\ \underline{u} \le b) = \frac{P(\underline{u} \le t\ \&\ \underline{u} \le b)}{P(\underline{u} \le b)} = \frac{t}{b} ,$$

so under the condition $\underline{u} \le b$ the r.v. $\underline{u}$ has a uniform-(0,b) distribution. Now comparing (3.3) and (3.2) for the case, that the $\underline{y}_j$ have a uniform-(0,1) distribution we find, that under the condition $\underline{x}_j = b$ the r.v.'s $\underline{x}_1$, $\underline{x}_2,\dots\dots,\underline{x}_{j-1}$ have the joint d.f. of an ordered random sample of size j-1 from a uniform-(0,b) distribution.

4. NON-UNIFORM UNDERLYING D.F.'S

In this section we give some observations of the behaviour of the algorithm under non-ideal circumtances. We generated $n = 16384 = 2^{14}$ pseudorandom numbers in the interval (0,1). Having sorted these into ascending order we simulated several distinct underlying d.f.'s by looking at $F^{-1}(x_1)$, $F^{-1}(x_2),\ldots.,$ $F^{-1}(x_n)$. In most cases we performed the interpolation search on the centiles (the real numbers $F^{-1}(i/100)$, $i = 1,2,\ldots.,99$; these numbers did not occur in the file) and also for 100 random elements occurring in the row.

Taking for F the uniform-(0,1) distribution resulted in an average of 4.87 interpolation steps for the centiles and 3.98 for the numbers randomly taken from the file. We actually did interpolation steps only as long as these resulted in a cutting-index different from the previous one, after which we did sequential steps until the required index was determined. The binary search algorithm mentioned in the introduction would have required 15 and 13 steps respectively. Since a lucky hit is possible when searching numbers in the file in general less searching steps are necessary in this case.

A distribution that is often treated as being "even", i.e. resembling the continuous uniform d.f., is the distribution of the family names, considered as 26-ary numbers, in a telephone directory. We approximated the actual distribution of the names by taking a polygone having constant densities over the first letters of the names. We estimated the densities by counting the number of entries with the corresponding first letter in the Amsterdam telephone book. The average search-length was about 10.8, not too much better than the 14 that would have been obtained by binary search.

A situation frequently encountered in practice when dealing with large files of numbers is a paging environment: by accessing a number we simultaneously gain access to many other numbers on the same "page". This situation arises in the telephone-directories already mentioned, in dictionaries, but also when a computer buffers its input. We simulated this situation by taking for F the discrete uniform distribution over the integers 1,2,....,k for k = 100, 1000, and 10000. This was achieved by multiplying every number in the file by k, and rounding it to the nearest integer. On the average there were about 164, 16 and 1.6 numbers on every page. The average number of pages visited was 1.4, 2.2 and 3.7 respectively. The binary search algorithm would have visited about $\log 100 \approx 6.6$, $\log 1000 = 10.0$

and log 10000 = 13.3 pages respectively.

We tried some continuous d.f.'s with non-zero tails, such as the normal, exponential and logistic d.f., resulting in about 40 steps on an average. A distribution having thick tails, such as the Cauchy-distribution, has all quantiles "concentrated in the middle". The first p-value is computed as

$$p = \frac{F^{-1}(y)-F^{-1}(x_1)}{F^{-1}(x_n)-F^{-1}(x_1)},$$

and for moderate y-values $F^{-1}(y) = \mathrm{tg}(\pi(y-\frac{1}{2}))$ is very small compared to the other two components. So it is not surprising that for y between 0.01 and 0.99 the minimal and maximal p did not differ by more than 0.001. Because x_1 had the value $5.7*10^{-6}$ and x_n was $1 - 4.2*10^{-5}$, the first distribution considered was truncated to the interval $[-5.6*10^{+4}, 7.6*10^{+3}]$. Because of this asymmetry the p-values ranged in [0.880, 0.881] for $y \in [0.01, 0.99]$, so the first cutting index was almost the same: 14426, irrespective of the index looked for. The situation resembles a telephone book with one name aaaa.... , one name zzzz... , and all the other names starting with vul.... .

The conclusion seems to be that our algorithm is not suited for all underlying distributions, since for the Cauchy-distribution a number of search steps of even 8000 was by no means exceptional.

REFERENCES

[1] AHO, A.V., J.E. HOPCROPT & J.D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison Wesley, Reading Mass. (1974).

[2] DAVID, H.A., *Order Statistics*, John Wiley & Sons Inc., New York (1970).

[3] ITAI, A. & Y. PERL, *Interpolation Search: a* log log N *Search*, Preprint from the Weizmann Institute, Israel (1975).

[4] KNUTH, D.E., *The Art of Computer Programming*, Vol. 3: *Sorting and Searching*, Addison Wesley, Reading, Mass. (1973).

[5] YAO, A.C. & F.F. YAO, *The Complexity of Searching an Ordered Random Table*, IEEE, FOCS 17 symposium, Houston (1976) p. 173-177.