ON QUICKSELECT, PARTIAL SORTING AND MULTIPLE QUICKSELECT.

MARKUS KUBA

Abstract. We present explicit solutions of a class of recurrences related to the Quickselect algorithm. Thus we are immediately able to solve recurrences arising at the partial sorting problem, which are contained in this class. We show how the partial sorting problem is connected to the Multiple Quickselect algorithm and present a method for the calculation of solutions for a class of recurrences related to the Multiple Quickselect algorithm. Further an analysis of the partial sorting problem for the ranks $r, \dots, r+p-1$ given the array $A[1, \dots, n]$ is provided.

1. Introduction

Quickselect (Hoare's FIND algorithm) solves the problem of finding the jth smallest element in an array of n given elements $A[1 \dots n]$ by using the so called divide and conquer strategy (see [2], [6], [7]). We consider the task of deriving an explicit solution of a class of recurrences related to Quickselect without using generating function. The method applied here is based on a difference argument of Knuth. It seems to be known that many recurrences contained in this class could be solved in an elementary way (e.g. see [10]), although in almost every work a generating function is applied. Best to our knowledge this is the first time a general solution is stated.

Partial Quicksort is a modification of Quickselect. It finds the p smallest elements of a given array $A[1 \dots n]$ containing n elements and sorts them. We will show that recurrences arising in the analysis of Partial Quicksort are contained in the class of Quickselect type recurrences. Further we provide a generalization of Partial Quicksort capable of sorting the subarray $A[r \dots r+p-1]$ of a given array $A[1 \dots n]$.

We also show how the partial sorting is connected to the Multiple Quickselect algorithm and present a method for the calculation of solutions for a class of recurrences related to the Multiple Quickselect algorithm, which is again based on a difference argument of Knuth.

We assume throughout this work that the input array $A[1 \dots n]$ consist of n distinct elements ordered randomly.

2. The Quickselect recursion

We denote with $\xi_{n,j}$ and $\vartheta_{n,j}$, respectively, the random variable, that counts the number of passes and key comparisons of Quickselect. Further we assume the the first element is chosen as the pivot element. Let $\pi_{n,k}$ denote the probability that the chosen pivot element has the rank k. Under the assumption the input array $A[1 \dots n]$ consist of n distinct elements ordered randomly, we get the splitting probability $\pi_{n,k} = 1/n$ for the analysis. Further we denote with $P_{n,j}^{[l]} = \mathbb{E}(\xi_{n,j}^{\underline{l}})$, $C_{n,j}^{[l]} = \mathbb{E}(\vartheta_{n,j}^{\underline{l}})$ the lst factorial moment of the considered random

Key words and phrases. Analysis of algorithms, Quickselect, Multiple Quickselect, Partial Sorting.

variable, where $x^{\underline{l}} := x(x-1)\dots(x-l+1)$. The recurrences for the expectation are well known.

$$P_{n,j}^{[1]} = 1 + \frac{1}{n} \cdot \sum_{k=1}^{j-1} P_{n-k,j-k}^{[1]} + \frac{1}{n} \cdot \sum_{k=j+1}^{n} P_{k-1,j}^{[1]}. \tag{1}$$

$$C_{n,j}^{[1]} = n - 1 + \frac{1}{n} \cdot \sum_{k=1}^{j-1} C_{n-k,j-k}^{[1]} + \frac{1}{n} \cdot \sum_{k=j+1}^{n} C_{k-1,j}^{[1]}. \tag{2}$$

The solutions of (1) and (2) are also well known. They can be derived elementary or by means of generating functions. Both of the recurrences fall into the class of so called Quickselect type recurrences.

$$X_{n,j} = a_{n,j} + \frac{1}{n} \sum_{k=1}^{j-1} X_{n-k,j-k} + \frac{1}{n} \sum_{k=j+1}^{n} X_{k-1,j}. \tag{3}$$

We prove the following theorem.

Theorem 1. *The value $X_{n,j}$, defined by (3) with arbitrary fixed values $a_{n,j}$, $1 \le j \le n$ is given by*

$$X_{n,j} = \sum_{k=n+2-j}^{n} A(k, k-n+j) + \sum_{k=2}^{n+1-j} \frac{k a_{k,1} - (k-1) a_{k-1,1}}{k} + a_{1,1}\,,$$

where $A(n,j)$ is given by

$$\begin{aligned} A(n,j) := \sum_{k=j+1}^{n} & \frac{k a_{k,j} - (k-1) a_{k-1,j-1} - (k-1) a_{k-1,j} + (k-2) a_{k-2,j-1}}{k} \\ & + \frac{j a_{j,j} - (j-1) a_{j-1,j-1}}{j}\,. \end{aligned} \tag{4}$$

Proof. The proof relies heavily on the difference argument of Knuth as shown in [5] (p. 12-15). There he solved (3) for $a_{n,j} = n-1$ and $a_{n,j} = 1$. We use this approach for an arbitrary toll function $a_{n,j}$. If $j = 1$ or $j = n$ the formula is seen to be true by solving the resulting full history recursion. Now we proceed to the case $1 < j < n$. At first we set up

$$F_{n,j} = \sum_{k=1}^{j-1} X_{n-k,j-k,}, \quad G_{n,j} = \sum_{k=j+1}^{n} X_{k-1,j}, \tag{5}$$

to get rid of the sums in (3). Hence (3) can be expressed as

$$X_{n,j} = a_{n,j} + \frac{1}{n} \cdot F_{n,j} + \frac{1}{n} \cdot G_{n,j}. \tag{6}$$

Note that

$$F_{n+1,j+1} = F_{n,j} + X_{n,j}, \quad G_{n+1,j} = G_{n,j} + X_{n,j}. \tag{7}$$

The key part of the proof is the following equation,

$$\begin{aligned} & n X_{n,j} - (n-1) X_{n-1,j} - (n-1) X_{n-1,j-1} + (n-2) X_{n-2,j-1} \\ & = n a_{n,j} - (n-1) a_{n-1,j} - (n-1) a_{n-1,j-1} + (n-2) a_{n-2,j-1} + X_{n-1,j-1} + X_{n-1,j} \\ & \quad - 2 X_{n-2,j-1}, \end{aligned} \tag{8}$$

which follows by using (6) and (7). We get the recursion

$$X_{n,j} - X_{n-1,j-1} = \frac{na_{n,j} - (n-1)a_{n-1,j-1} - (n-1)a_{n-1,j} + (n-2)a_{n-2,j-1}}{n} + X_{n-1,j} - X_{n-2,j-1}. \tag{9}$$

This relation can be iterated to get

$$X_{n,j} - X_{n-1,j-1} = \sum_{k=j+1}^{n} \frac{ka_{k,j} - (k-1)a_{k-1,j-1} - (k-1)a_{k-1,j} + (k-2)a_{k-2,j-1}}{k} + \frac{ja_{j,j} - (j-1)a_{j-1,j-1}}{j} = A(n,j), \tag{10}$$

which simplifies to

$$X_{n,j} = X_{n-1,j-1} + A(n,j), \tag{11}$$

as claimed. In (10) we have used

$$X_{j,j} - X_{j-1,j-1} = \frac{ja_{j,j} - (j-1)a_{j-1,j-1}}{j}, \tag{12}$$

which follows easily from (6). Iteration of (11) leads to

$$X_{n,j} = \sum_{k=n+2-j}^{n} A(k, k-n+j) + X_{n-j+1,1}. \tag{13}$$

This simplifies to

$$X_{n,j} = \sum_{k=n+2-j}^{n} A(k, k-n+j) + \sum_{k=2}^{n+1-j} \frac{ka_{k,1} - (k-1)a_{k-1,1}}{k} + a_{1,1}. \tag{14}$$

□

3. Applications of theorem 1

3.1. **Higher moments for Quickselect.** By introducing the probability generating functions $\mathcal{P}_{n,j}(v) = \sum_{k\geq 1} \mathbb{P}(\xi_{n,j} = k)v^k$ and $\mathcal{C}_{n,j}(v) = \sum_{k\geq 1} \mathbb{P}(\vartheta_{n,j} = k)v^k$ of $\xi_{n,j}$ and $\vartheta_{n,j}$ we get the following recurrences (see [10]) for $\mathcal{P}_{n,j}(v)$ and $\mathcal{C}_{n,j}(v)$.

$$\mathcal{P}_{n,j}(v) = \frac{v}{n}\left(\sum_{k=1}^{j-1} \mathcal{P}_{n-k,j-k}(v) + 1 + \sum_{k=j+1}^{n} \mathcal{P}_{k-1,j}(v)\right), \tag{15}$$

$$\mathcal{C}_{n,j}(v) = \frac{v^{n-1}}{n}\left(\sum_{k=1}^{j-1} \mathcal{C}_{n-k,j-k}(v) + 1 + \sum_{k=j+1}^{n} \mathcal{C}_{k-1,j}(v)\right). \tag{16}$$

To get the lth factorial moments we differentiate l times with respect to v and evaluate at $v = 1$. We get the following recurrence.

$$P_{n,j}^{[l]} = \frac{l}{n}\left(\sum_{k=1}^{j-1} P_{n-k,j-k}^{[l-1]} + \sum_{k=j+1}^{n} P_{k-1,j}^{[l-1]} + \delta_{l,1}\right) + \frac{1}{n}\cdot\sum_{k=1}^{j-1} P_{n-k,j-k}^{[l]} + \frac{1}{n}\cdot\sum_{k=j+1}^{n} P_{k-1,j}^{[l]}, \tag{17}$$

$$C_{n,j}^{[l]} = \sum_{m=1}^{l} \frac{(n-1)^{\underline{l}}\binom{l}{m}}{n}\left(\sum_{k=1}^{j-1} C_{n-k,j-k}^{[l-m]} + \sum_{k=j+1}^{n} C_{k-1,j}^{[l-m]} + \delta_{m,l}\right) + \frac{1}{n}\sum_{k=1}^{j-1} C_{n-k,j-k}^{[l]} + \frac{1}{n}\sum_{k=j+1}^{n} C_{k-1,j}^{[l]}. \tag{18}$$

Since we know $P^{[1]}_{n,j}$ and $C^{[1]}_{n,j}$ we can recursively find a (although involved) closed form for the higher moments of $\xi_{n,j}$ and $\vartheta_{n,j}$. The variance can explicitly be calculated this way avoiding some of the complications of the generating functions approach [4].

3.2. **Partial Quicksort.** Partial Quicksort was first proposed and analyzed by Martínez [9]. The algorithm finds the p smallest elements of a given array $A[1\dots n]$ containing n elements and sorts them. It works as follows. First we choose a pivot element. After partitioning we compare the rank k of the pivot element with p. If $k \le p$ we sort the array $A[1\dots k-1]$ and recursively call the algorithm for $A[k-1\dots n]$ searching for the remaining $p-k$ smallest elements. If $k > p$ we recursively call the algorithm for $A[1\dots k-1]$. Let $C_{n,p}$ denote the average number of key comparisons of partial quicksort to sort the p smallest out of n elements. We consider again $\pi_{n,k} = 1/n$ for the analysis. We get the recursion

$$\begin{aligned} &C_{n,p} = n-1+\sum_{k=1}^{p}\pi_{n,k}(C_{k-1,k-1}+C_{n-k,p-k}) + \sum_{k=p+1}^{n}\pi_{n,k}C_{k-1,p} \ \text{ if } n>0\,, p>0\,, \\ &C_{n,0} = 0 \ \forall n\in\mathbb{N}, \quad C_{0,p} = 0 \ \forall p \in \mathbb{N}\,. \end{aligned} \tag{19}$$

The recurrence (19) reflects the cost for splitting the array $C_{k-1,p}, C_{n-k,p-k}$ and the cost for sorting $C_{k-1,k-1}$. Partial Quicksort works the same way as Quicksort when $p=n$. We get $C_{n,n} = 2(n+1)H_n - 4n$ where $H_n = \sum_{k=1}^{n} 1/k$ denotes the nth harmonic number. Let $a_{n,p}$ denote the toll function given by

$$a_{n,p} = n-1+\sum_{k=1}^{p}\pi_{n,k}C_{k-1,k-1}\,. \tag{20}$$

For the calculation of $a_{n,p}$ we need the well known formulae

$$\sum_{k=1}^{n} kH_k \;=\; \binom{n+1}{2}\Big[H_{n+1}-\frac{1}{2}\Big], \quad \sum_{k=1}^{n} H_k \;=\; (n+1)(H_{n+1}-1)\,, \tag{21}$$

which can be found in [1]. We get

$$a_{n,p} = n-1+\frac{p(p+1)H_p + p(-\frac{5}{2}p+\frac{1}{2})}{n}\,. \tag{22}$$

Hence we get for the value $C_{n,p}$ a recursion of the form

$$X_{n,p} = a_{n,p} + \frac{1}{n}\sum_{k=1}^{p-1}X_{n-k,p-k} + \frac{1}{n}\sum_{k=p+1}^{n}X_{k-1,p}\,. \tag{23}$$

The standard approach for solving (23) would be setting up bivariate generating functions and solving the arising differential equation. This approach was successfully carried out in [9]. We get the following corollary much more easy by using Theorem 1 and (22).

Corollary 1. *The average number of key comparisons of partial quicksort to sort the p smallest elements out of n elements is given by*

$$C_{n,p} = 2(n+1)H_n + 2n - 6p + 6 - 2(n+3-p)H_{n+1-p}. \tag{24}$$

Let $P_{n,p}$ denote the average number of passes of Partial Quicksort to sort the p smallest elements out of n elements. $P_{n,p}$ satisfies (23) with the toll function $a_{n,p}$ given by

$$a_{n,p} = 1+\sum_{k=1}^{p}\pi_{n,k}P_{k-1,k-1} = 1+\frac{p(p-1)}{2n}, \tag{25}$$

where $P_{k,k} = k$ denotes the average number of passes by quicksort to sort an array containing k elements. We get the following result.

Corollary 2. *The average number of passes of Partial Quicksort to sort the p smallest elements out of n elements is given by*

$$P_{n,p} = H_{n+1-p} + p - 1. \tag{26}$$

Proof. We present the derivation of Corollary 1, Corollary 2 follows by a similar argument. We split the toll function into two parts $a_{n,p} = \hat{a}_{n,p} + \tilde{a}_{n,p}$, with $\hat{a}_{n,p} = n - 1$ and $\tilde{a}_{n,p} = \frac{p(p+1)H_p + p(-\frac{5}{2}p + \frac{1}{2})}{n}$. We already know the solution for $\hat{a}_{n,p}$ from Knuth [5], p. 14, which is just $C^{[1]}_{n,p}$, the average number of key comparisons of Quickselect for selecting p smallest element.

$$C^{[1]}_{n,p} = 2(n+1)H_n - 2(n+3-p)H_{n+1-p} - 2(p+2)H_p + 2n + 6, \tag{27}$$

Now we can restrict ourselves to $\tilde{a}_{n,p}$. We apply Theorem 1 with $\tilde{a}_{n,p}$. It easy to see that

$$A(n,p) = \frac{p\tilde{a}_{p,p} - (j-1)\tilde{a}_{p-1,p-1}}{p} = 2H_p - 4 + \frac{2}{p}, \tag{28}$$

since $k\tilde{a}_{k,p} - (k-1)\tilde{a}_{k-1,p-1} - (k-1)\tilde{a}_{k-1,p} + (k-2)\tilde{a}_{k-2,p-1} = 0$ for $k = p+1, \dots, n$. Hence

$$\sum_{k=n+2-j}^{n} A(k, k-n+j) + \sum_{k=2}^{n+1-j} \frac{ka_{k,1} - (k-1)a_{k-1,1}}{k} + a_{1,1} = \sum_{k=n+2-j}^{n} A(k, k-n+j)$$
$$= 2(p+1)H_p - 2p - 4p + 2H_p = 2(p+2)H_p - 6p, \tag{29}$$

where we have used (21). Combining the solutions corresponding to $\hat{a}_{n,p} = n - 1$ and $\tilde{a}_{n,p} = \frac{p(p+1)H_p + p(-\frac{5}{2}p + \frac{1}{2})}{n}$ finishes the proof. □

4. General Partial Quicksort

Now we modify the Partial Quicksort algorithm to solve the problem of sorting the elements of the ranks $r, r+1, \dots, r+p-1 = s$ with $1 \le r \le r+p-1 \le n$ of a given array $A[1 \dots n]$ containing n elements. General Partial Quicksort merges Quickselect, Quicksort and Partial Quicksort into a single algorithm providing also the possibility to search arrays of type $A[r \dots s]$. General Partial Quicksort works as follows. First we choose a pivot element. After partitioning we compare the rank k of the pivot element with the searched ranks $r, r+1, \dots, r+p-1$. If $r - 1 \le k \le r + p$ we call the algorithm for the left subarray $A[1 \dots k-1]$ sorting the ranks $r, r+1, \dots, k-1$ and upon the right subarray $A[k+1 \dots n]$ sorting the ranks $k+1, k+2, \dots, r+p-1$. If $1 \le k < r-1$ we recursively call the algorithm for $A[k+1 \dots n]$ sorting the ranks $r-k, r-k+1, \dots, r-k+p-1$. If $r + p < k \le n$ we recursively call the algorithm for $A[1 \dots k-1]$ sorting the ranks $r, r+1, \dots, r+p-1$.

4.1. Analysis of general partial quicksort.

Let $C_{n,r,p}$ denote the average number of key comparisons of General Partial Quicksort to sort the elements of the ranks $r, r+1, \dots, r+p-1$ out of n elements. We get the recursion

$$\begin{aligned} C_{n,r,p} = n - 1 + \frac{1}{n}\sum_{k=1}^{r-2} C_{n-k,r-k,p} + \frac{1}{n}\sum_{k=r+p+1}^{n} C_{k-1,r,p} \\ + \frac{1}{n}\sum_{k=r-1}^{r+p} \left(C_{k-1,r,k-r} + C_{n-k,1,r+p-k-1}\right). \end{aligned} \tag{30}$$

General Partial Quicksort works the same way as partial quicksort when $r+p-1=n$ or $r=1$. In the first case we get $C_{n,r,p}=2(n+1)H_n-4n+2-6r-2(r+2)H_r$ and in the second case we get $C_{n,1,p}=2(n+1)H_n+2n-6p+6-2(n+3-p)H_{n+1-p}$. Let $a_{n,r,p}$ denote the toll function given by

$$a_{n,r,p}=n-1+\frac{1}{n}\sum_{k=r-1}^{r+p}(C_{k-1,r,k-r}+C_{n-k,1,r+p-k-1}). \tag{31}$$

The explicit formula for $a_{n,r,p}$ is quite complicated, it can be easily obtained by using (21). We get the recursion

$$C_{n,r,p}=a_{n,r,p}+\frac{1}{n}\sum_{k=1}^{r-2}(C_{n-k,r-k,p})+\frac{1}{n}\sum_{k=r+p+1}^{n}C_{k-1,r,p}. \tag{32}$$

Again, we do not use generating functions to solve the recurrence above. Instead, we stick to Knuth's difference method. We get the following theorem.

Theorem 2. *The value $C_{n,r,p}$, defined by* (30)*, with with arbitrary fixed values $a_{n,r,p}$, is given by is given by*

$$C_{n,r,p}=\sum_{k=n+2-r}^{n}M_{k,k+r-n,p}+\sum_{k=p+2}^{n+1-r}\frac{ka_{k,1,p}-(k-1)a_{k-1,1,p}}{k}+a_{p+1,r,p}\,,$$

where $M_{n,r,p}$ is given by

$$\begin{aligned}M_{n,r,p}:=&\sum_{k=r+p+1}^{n}\frac{ka_{k,r,p}-(k-1)a_{k-1,r,p}-(k-1)a_{k-1,r-1,p}+(k-2)a_{k-2,r-1,p}}{k}\\&+\frac{(r+p)a_{r+p,r,p}-(r+p-1)a_{r+p-1,r-1,p}}{r+p}.\end{aligned} \tag{33}$$

Proof. The proof is a straightforward extension of the previous proof of Theorem 1. □

This leads to the following result.

Corollary 3. *The average number of key comparisons of general partial quicksort to sort the elements of the ranks $r,r+1,\dots,r+p-1$ out of n elements is given by*

$$C_{n,r,p}\;=\;2(n+1)H_n+2n-6p+12-2(r+2)H_r-2(n+4-r-p)H_{n+2-r-p}\,. \tag{34}$$

5. Partial sorting and multiple quickselect

The results appearing in (24) and (34) are not surprising when considering the Multiple Quickselect algorithm. Multiple Quickselect is an extension of Hoare's FIND algorithm (Quickselect) for finding p ranks $j_1,j_2,\dots,j_p$, with $1\le j_1<\dots<j_p\le n$, simultaneously in a given array $A[1\dots n]$ containing n elements. Hence Multiple Quickselect also solves the partial sorting problem. The algorithm works as follows. First we choose a pivot element. After partitioning we compare the rank k of the pivot element with the searched ranks $j_1,\dots,j_p$. If k equals a searched rank j_i we recursively search for the ranks $j_1,\dots,j_{i-1}$ in the left subarray $A[1\dots k-1]$ and for the ranks $j_{i+1}-k,\dots,j_p-k$ in the right subarray $A[k+1\dots n]$. Otherwise we recursively search for the ranks on one subarray or both of them, according to in which interval $[j_i,j_{i+1}]$ the rank k lies. A nice description of the algorithm can be found in [11]. The algorithm was analyzed by elementary means in [12] and by using generating

functions in [11] under the condition that only key comparisons were counted (the extra index comparisons are neglected).

The average number of key comparisons $C(n; j_1, \dots, j_p)$ for finding the elements of the ranks $1 \le j_1 < j_2 < \dots < j_p \le n$ out of n elements was given by Prodinger in [12]. Now (24) and (34) can easily be obtained from

$$\begin{aligned} C(n; j_1, \dots, j_p) &= 2n + j_p - j_1 + 2(n+1)H_n - 2(j_1+2)H_{j_1} + 8p - 2 \\ &\quad - 2(n+3-j_p)H_{n+1-j_p} - 2\sum_{l=2}^{p}(j_l + 4 - j_{l-1})H_{j_l+1-j_{l-1}} \end{aligned} \tag{35}$$

by simply setting $j_k = k$ for $1 \le k \le p$ for partial quicksort. To get the result for General Partial Quicksort we set $j_k = r + k - 1$ for $1 \le k \le p$. We can also obtain the average number of passes from the formula

$$P(n; j_1, \dots, j_p) = H_{j_1} + H_{n+1-j_p} + 2\sum_{l=2}^{p} H_{j_l+1-j_{l-1}} - 2p + 1, \tag{36}$$

also given by Prodinger in [12]. The formulas (35) and (36) where found by Prodinger "through extensive computer experiments with the computer algebra system Maple". We present a way to actually find solutions like (35) and (36) by hand.

The general Multiple Quickselect recurrence is stated below.

$$\begin{aligned} X(n; j_1, \dots, j_p) &= a_n + \frac{1}{n} \cdot \sum_{i=1}^{p}\Big(\sum_{j_{i-1} \le k < j_i} X(n-k; j_i - k, \dots, j_p - k)\Big) \\ &\quad + \frac{1}{n} \cdot \sum_{i=1}^{p}\Big(\sum_{j_i < k \le j_{i+1}} X(k-1; j_1, \dots, j_i)\Big), \end{aligned}$$

with $j_0 := 1$ and $j_{p+1} := n$. If we apply Knuth's difference method we can express $X(n; j_1, \dots, j_p)$ by a full history recursion only involving values of the type $X(n; s_1, \dots, s_m)$, with $1 \le m \le p-1$, therefore reducing the number of searched elements by one.

Hence for a given toll function a_n one can find formulas like (35) and (36) by proceeding as follows.

(1) Use Theorem 1 to solve the case $p = 1$.
(2) Apply Knuth's difference method for recursive calculation of case $p = 2, p = 3, \dots$ until an explicit formula can be guessed.
(3) Prove the formula using induction.

It is possible to formalize step (2), we omit the lengthy formula.

For example for $a_n = \frac{H_n}{n}$ one can verify the following formula.

$$\begin{aligned} X(n; j_1, \dots, j_p) &= (n+1)H_n^{(2)} - j_1 H_{j_1}^{(2)} - (n - j_p + 1)H_{n-j_p+1}^{(2)} \\ &\quad - \sum_{t=2}^{p}\Big[(j_t - j_{t-1})H_{j_t - j_{t-1}+1}^{(2)} - \frac{H_{j_t - j_{t-1}+1}}{j_t - j_{t-1} + 1}\Big] + p. \end{aligned} \tag{37}$$

6. Conclusion

Using this theorem it is possible to build up a repertoire of solutions of the Multiple Quickselect recursion for various toll functions a_n. The author is working on a more detailed analysis of the Multiple Quickselect recursion.

References

[1] D. H. Greene and D. E. Knuth, *Mathematics for the Analysis of Algorithms.* Birkhäuser, Boston, 2nd edition, 1982.

[2] C. A. R. Hoare, *Find (Algorithm 65).* Comm. ACM, 4:321322, 1961.

[3] C. A. R. Hoare, *Quicksort.* Computer Journal, 5:1015, 1962.

[4] P Kirschenhofer and H Prodinger, *Comparisons in Hoare's FIND Algorithm.* Combinatorics, Probability and Computing, 7, 111-120, 1998.

[5] D. E. Knuth, *Selected Papers on Analysis of Algorithms.* CLSI Publications, 2000.

[6] D. E. Knuth, *The Art of Computer Programming: Fundamental Algorithms, volume 1.* Addison-Wesley, Reading, Mass., 3rd edition, 1997.

[7] D. E. Knuth, *The Art of Computer Programming: Sorting and Searching, volume 3.* Addison-Wesley, Reading, Mass., 2nd edition, 1998.

[8] M. Kuba, *Multiple Quickselect und die Steinerdistanz in binären Suchbäumen.* Diploma thesis, Technische Universität Wien, 2004.

[9] C. Martínez, *Partial Quicksort.* Partial quicksort.*Proc. of the 6th ACM-SIAM Workshop on Algorithm Engineering and Experiments and the 1st ACM-SIAM Workshop on Analytic Algorithmics and Combinatorics*, 224-228, 2004.

[10] A. Panholzer and H. Prodinger, *Binary search tree recursions with harmonic toll funktions.* Journal of Computational and Applied Mathematics, 142, 211-225, 2002.

[11] A. Panholzer and H. Prodinger, *A Generating Function Approach for the Analysis of Grand Averages for Multiple QUICKSELECT.* Random Struct. Alg., 13, 189-209, 1998.

[12] H. Prodinger, *Multiple Quickselect, Hoare's Find algorithm for several elements.* Information Processing Letters 56, 123-129, 1995.

Markus Kuba, Institut für Diskrete Mathematik und Geometrie, Technische Universität Wien, Wiedner Hauptstr. 8-10/104, 1040 Wien, Austria

E-mail address: `markus.kuba@tuwien.ac.at`