Expected Optimal Selection on the PRAM

Laurence Boxer *

Abstract

The Selection Problem is the following: Given a totally ordered set S of n items and an integer k such that $0 < k \leq n$, find the k-th smallest member of S. A general algorithm for a PRAM using optimal $\Theta(n)$ cost is not currently known. In this paper, we show that if the input set S is uniformly (randomly) distributed on an interval $[a, b]$, then the Selection Problem may be solved by an algorithm with the following properties.

- The expected cost is an optimal $\Theta(n)$.
- The worst-case cost is $\Theta(n \log^{(2)} n)$, which matches that of the most efficient algorithm in the literature [3].

Key words and phrases: parallel algorithm, selection, parallel random access machine (PRAM), expected value, variance, Chebyshev's Inequality

1 Introduction

Selection is a fundamental problem in computer science. It is a generalization of such important problems as finding the *minimum value* ($k = 1$), *maximum value* ($k = n$), and *median value* ($k = \lceil n/2 \rceil$). Although a naive solution to the selection problem sorts the data and then returns the k-th entry, it has long been known that asymptotically more efficient sequential solutions exist. Sorting requires $\Theta(n \log n)$ sequential time, but an optimal $\Theta(n)$-time sequential solution is known for the Selection Problem (see [2, 4]). Thus, in the world of parallel algorithms, we seek an algorithm that requires $\Theta(n)$ cost, where we define cost to be the product of running time and the number of processors used, hence the number of computer cycles available during the running time of the algorithm.

In the following, the notation $\log n$ is short for $\log_2 n$. We use the notation $\log^{(2)} n$ to represent $\log \log n$, and $\log^2 n$ to abbreviate $(\log n)^2$.

*Niagara University and State University of New York - Buffalo. Department of Computer and Information Sciences, Niagara University, Niagara University, NY 14109, USA. E-mail: boxer@niagara.edu. Phone: 716-286-8447. Fax: 716-286-8445.

The best general algorithm in the literature for the PRAM Selection Problem is that found in [3], which runs in $O(\log n \log^{(2)} n)$ time on an EREW PRAM of $\Theta(n/\log n)$ processors, for a cost of $O(n \log^{(2)} n)$.

In [5], a solution to the Selection Problem is given that runs in $\Theta(\log n[\log^* n - \log^*(n/k)])$ time and $\Theta(n)$ cost on an EREW PRAM, provided the rank k of the desired element satisfies $0 < k \leq n/\log^{(c)} n$ for some constant c.

If we have some knowledge of the distribution of members of the input set S, we may be able to modify the algorithm of [3] to run faster. In this paper, we show that if the members of S are uniformly distributed on some interval, we can modify the algorithm of [3] to have expected cost of an optimal $O(n)$, and to have worst-case cost matching that of the algorithm of [3]. Note this situation parallels that of the sequential QuickSort algorithm, in the sense that QuickSort also has distinct expected and worst-case running times ($\Theta(n \log n)$ and $\Theta(n^2)$, respectively - see [2, 4]).

2 Preliminaries

2.1 EREW PRAM Model of Computation

A parallel random access machine (PRAM) is a computer with multiple processors that share memory. The exclusive read, exclusive write (EREW) PRAM, does not permit simultaneous reads from or writes to the same memory location.

2.2 Our problem

We assume our input is a set $S = \{s_i\}_{i=1}^{n}$ of real numbers uniformly (randomly) distributed over some interval $[a, b]$, and an integer k such that $0 < k \leq n$. Our problem is to return the k-th smallest member of S. The model of computation is the EREW PRAM with $\Theta(\frac{n}{\log n})$ processors.

2.3 Tools from Probability

For every integer m such that $0 < m \leq n$, let $X_{(m)}$ be a random variable representing the m-th order statistic of S, that is, the m-th smallest member of S. Assume $[a, b] = [0, 1]$. Then the expected value and variance, respectively, of $X_{(m)}$ are given by the following equations [6].

$$E(X_{(m)}) = \frac{m}{n+1}. \tag{1}$$

$$Var(X_{(m)}) = \frac{m(n+1-m)}{(n+1)^2(n+2)}. \tag{2}$$

Recall Chebyshev's Inequality:

Theorem 2.1 [6] *Let* $\mathbf{y}$ *be a random variable with expected value* $E(\mathbf{y}) = \mu$ *and variance* $V(\mathbf{y}) = v$. *Then for any* $t > 0$,

$$Pr[\,|\mathbf{y} - \mu| \; > \; t] \leq \frac{v}{t^2}.$$

3 Our algorithm

We give the following algorithm to compute

Function Select(S, k).

Our algorithm is based on the one presented in [3], but is modified to take advantage of the assumed uniform distribution of the set S over an interval $[a, b]$.

1. Perform the initializations

$$n_0 = n; \; c = 0; \; k_0 = k.$$

   This takes $\Theta(1)$ time.

2. In $\Theta(\log n)$ time, use a sweep (semigroup) operation to construct

$$S_0 = \{(s_i, i) \mid s_i \in S\}.$$

3. While $n_c > n/\log n$, do

   (a) Add 1 to c. This takes $\Theta(1)$ time.

   (b) If the interval $[a, b]$, over which the members of S_{c-1} are uniformly distributed, is unknown, compute

$$a = \min\{s \mid (s, j) \in S_{c-1}\}, \; b = \max\{s \mid (s, j) \in S_{c-1}\}$$

   via two semigroup operations. This can be done in $\Theta(\log n)$ time [4].

(c) In $\Theta(1)$ time, if $a = b$ then exit the algorithm, returning as the k-th smallest value in S the value $s_j \in S$ for any j such that $(s_j, j) \in S_{c-1}$. Otherwise, $a < b$, and we proceed as follows.

(d) In $\Theta(1)$ time, compute the unique linear function $f_c : [a, b] \to [0, 1]$ such that $f_c(a) = 0$, $f_c(b) = 1$.

(e) In $\Theta(\log n)$ time, use a sweep (semigroup) operation to compute

$$S'_{c-1} \;=\; \{(f_c(s), j) \mid (s, j) \in S_{c-1}\}.$$

Thus, the resulting set S'_{c-1} is uniformly distributed with respect to members' first coordinates on $[0, 1]$.

(f) (**Approximate Interval Step**) In $\Theta(1)$ time, compute a subinterval $[a_0, a_1] \subset [0, 1]$ that has a high probability of containing (first components of) members of S_{c-1} that index about $\frac{n}{4\log n}$ to $\frac{n}{2\log n}$ members of S, including the k-th smallest member of S. This may be done by choosing a_0 and a_1 as follows.

- If $(k_{c-1} - \frac{n}{4\log n}) \leq 0$, take $a_0 = 0$; otherwise,
- take a_0 to be the expected value of the $\lceil k_{c-1} - \frac{n}{4\log n}\rceil$-th smallest (with respect to the first component) member of S_{c-1}, using equation (1).

Similarly,

- If $(k_{c-1} + \frac{n}{4\log n}) > n_{c-1}$, take $a_1 = 1$; otherwise,
- take a_1 to be the expected value of the $\lceil k_{c-1} + \frac{n}{4\log n}\rceil$-th smallest member of S_{c-1}, using equation (1).

(g) Perform a semigroup operation to determine sets

$$T_0 \;=\; \{(s, j) \in S'_{c-1} \mid s \leq a_0\},\; T_1 \;=\; \{(s, j) \in S'_{c-1} \mid a_0 < s < a_1\},$$

$$T_2 \;=\; \{(s, j) \in S'_{c-1} \mid a_1 \leq s\}$$

and their respective cardinalities. This takes $\Theta(\log n)$ time.

(h) Consider the following possibilities.

- If $k_{c-1} \leq |T_0|$, the desired selection value is indexed by the second component of a member of T_0. Let $U = T_0$, $n_c = |T_0|$, $k_c = k_{c-1}$.
- Otherwise, if $k_{c-1} \leq |T_0| + |T_1|$ the desired selection value is indexed by a member of T_1. Let $U = T_1$, $n_c = |T_1|$, $k_c = k_{c-1} - |T_0|$.
- Otherwise, the desired selection value is indexed by a member of T_2. Let $U = T_2$, $n_c = |T_2|$, $k_c = k_{c-1} - |T_0| - |T_1|$.

This step may be performed in $\Theta(1)$ time.

(i) (**Reduction Decision**) If $n_c \leq \max\{\frac{n}{\log n}, 0.75n_{c-1}\}$, then let $S_c = U$ (note this takes $\Theta(1)$ time); *Else* U is a set whose cardinality is not sufficiently reduced with respect to n_{c-1}, so we proceed to compute S_c in subsequent steps of the loop as in the algorithm of [3], as follows.

  i. The processors partition U so that each processor P_i is responsible for a block B_i of $O(\log n)$ members of U. In parallel, each processor P_i finds the median value b_i of (the first coordinates of members of) B_i using the linear-time sequential algorithm for the Selection Problem [2, 4]. This takes $O(\log n)$ time.

  ii. Compute the median A of the set

$$M = \{b_i\}_{i=1}^{n/\log n}$$

  by sorting M and reporting its median entry. Since M has one item per processor, this may be done using the algorithm of [1] in $\Theta(\log n)$ time.

  iii. Perform a semigroup operation to determine sets

$$U_0 = \{(s,j) \in U \mid s < A\},\ U_1 = \{(s,j) \in U \mid s = A\},\ U_2 = \{(s,j) \in U \mid s > A\}$$

  and their respective cardinalities. This takes $\Theta(\log n)$ time.

  iv. Consider the following possibilities.

  - If $k_c \leq |U_0|$, the desired selection value is indexed by a member of U_0. Let $S_c = U_0$ and let $n_c = |U_0|$.

- Otherwise, if $k_c \leq |U_0| + |U_1|$, the desired selection value is indexed by a member of U_1. Since all members of U_1 have value A in the first component, exit the algorithm, returning the value $s_{k_c} = A \in S$ as the k-th smallest member of S.
- Otherwise, the desired selection value is indexed by a member of U_2. Let $S_c = U_2$, $n_c = |U_2|$, and $k_c \leftarrow k_c - |U_0| - |U_1|$.

This step (consideration of three cases) takes $\Theta(1)$ time.

End Else.

This *If - Else* step (the block consequent to the Reduction Decision) takes $\Theta(\log n)$ time.

End while

One performance of the loop body requires $\Theta(\log n)$ time. We discuss below the number of performances of the loop body.

4. Sort the members of S_c by the first component, in ascending order. Since $|S_c| \leq \frac{n}{\log n}$, this may be done in $O(\log n)$ time.

5. Let (s_j, j) be the k_c-th member of the sorted list S_c. In $\Theta(1)$ time, report $s_j \in S$ as the k-th smallest member of S.

4 Analysis

Clearly, the loop dominates the running time of the algorithm. Thus, we must determine the number of performances of the loop body. It is useful for this purpose to note that the function

$$F_n(x) = x(n+1-x)$$

satisfies, for all real x,

$$F_n(x) \leq F_n(\frac{n+1}{2}) = \frac{(n+1)^2}{4}. \tag{3}$$

Note it follows from equation (2) and inequality (3) that for all m,

$$Var(X_{(m)}) = \frac{F_n(m)}{(n+1)^2(n+2)} \leq \frac{1}{4(n+2)}. \tag{4}$$

4.1 Worst Case

In the worst case, each performance of the loop body reaches the Reduction Decision having to perform its *Else* construction. This is the case, for example, for $S = \{2^{-i}\}_{i=1}^{n}$, and, say, $k = \lceil \frac{n}{2} \rceil$, since, then, at each pass through the loop, the selection value is indexed by a member of T_0, which contains all but a constant number of members of S'_{c-1}. It follows from our analysis in section 4.2, below, that examples that cause worst-case running time are highly improbable under the assumption of uniformly distributed data values.

In the worst case, the loop body is performed $\Theta(\log^{(2)} n)$ times. This is seen as follows. First, we show that every performance of the loop body reduces the size of the index set remaining under consideration by a factor of 0.75 or smaller:

- Suppose the selection value is indexed by a member of U_0. Note $(s, j) \in U_0$ implies $s < A$. Since A is the median of the members of M, s is smaller than at least $\lceil \frac{n}{2 \log n} \rceil$ members $b_i \in M$, namely those that satisfy $b_i \geq A$, and each such b_i is less than or equal to at least

$$\lceil \frac{n_{c-1}}{2\frac{n}{\log n}} \rceil = \lceil \frac{n_{c-1} \log n}{2n} \rceil$$

members of $B_i \subset S_{c-1} \setminus U_0$. Therefore,

$$|S_{c-1} \setminus U_0| \geq \lceil \frac{n}{2 \log n} \rceil \lceil \frac{n_{c-1} \log n}{2n} \rceil \geq 0.25 n_{c-1}.$$

Hence $|U_0| \leq 0.75 n_{c-1}$.

- If the selection value is indexed by a member of U_1, we immediately conclude the algorithm.

- If the selection value is indexed by a member of U_2, then an argument similar to that given for the case of U_0 above shows $|U_2| \leq 0.75 n_{c-1}$.

Thus, the number of performances of the loop body is, in the worst case, at most the smallest integer c satisfying

$$(0.75)^c n \leq \frac{n}{\log n},$$

which implies $c = \Theta(\log^{(2)} n)$. Therefore, in the worst case, the running time of the algorithm is $\Theta(\log n \log^{(2)} n)$, and the cost of the algorithm is $\Theta(n \log^{(2)} n)$, which matches that of [3].

4.2 Best Case

In the best case, we exit the loop immediately following the Reduction Decision during the first performance of the loop body. In this case, the running time of the algorithm is $\Theta(\log n)$, and the cost of the algorithm is an optimal $\Theta(n)$. The probability of the best case occurring may be analyzed as follows. We use the notation $P[U]$ to denote the probability of the event U, and $\overline{U}$ denotes the event "not U."

Let $a_0 = E(X_{(\lceil k-\frac{n}{4\log n}\rceil)})$, $a_1 = E(X_{(\lceil k+\frac{n}{4\log n}\rceil)})$. Then we have, by equation (1),

$$a_0 = \max\{0, \frac{\lceil k-\frac{n}{4\log n}\rceil}{n+1}\},\ a_1 = \min\{1, \frac{\lceil k+\frac{n}{4\log n}\rceil}{n+1}\}.$$

Note that $[a_0, a_1]$ is the interval used at Approximate Interval Step of the algorithm.

Let B be the event that $X_{(k)} \in [a_0, a_1]$. Let C be the event that $[a_0, a_1]$ contains at most $\frac{n}{\log n}$ members of S. Then the event $B \cap C$ is a subset of the event that the While loop performs its body exactly once. Thus, we are interested in showing that $P[B \cap C]$ is nearly 1 for large n.

Since $E(X_{(k)}) = \frac{k}{n+1}$, we have

$$P[\overline{B}] \leq P[\,|X_{(k)} - E(X_{(k)})| > \frac{\frac{n}{4\log n} - 1}{n+1}].$$

It follows that there is a positive N_0 such that $n > N_0$ implies

$$P[\overline{B}] \leq P[\,|X_{(k)} - E(X_{(k)})| > \frac{n}{5(n+1)\log n}] \leq \text{ (by Chebyshev's inequality and inequality (4))}$$

$$\frac{1}{4(n+2)} \frac{25(n+1)^2\log^2 n}{n^2}.$$

Thus,

$$n > N_0 \text{ implies } P[\overline{B}] \leq \frac{25(n+1)^2\log^2 n}{4n^2(n+2)}. \tag{5}$$

Also, notice that

$$\overline{C} \subset (X_{(\lceil k-\frac{n}{2\log n}\rceil)} > a_0) \cup (X_{(\lceil k+\frac{n}{2\log n}\rceil)} < a_1),$$

so

$$P[\overline{C}] \leq P[(X_{(\lceil k-\frac{n}{2\log n}\rceil)} > a_0) \cup (X_{(\lceil k+\frac{n}{2\log n}\rceil)} < a_1)] \leq$$

$$P[X_{(\lceil k-\frac{n}{2\log n}\rceil)} > a_0] \ + \ P[X_{(\lceil k+\frac{n}{2\log n}\rceil)} < a_1] \leq$$

$$P[\,|X_{(\lceil k-\frac{n}{2\log n}\rceil)} \;-\; E(X_{(\lceil k-\frac{n}{2\log n}\rceil)})| \;>\; \frac{\frac{n}{4\log n}-1}{n+1}\,] +$$

$$P[\,|X_{(\lceil k+\frac{n}{2\log n}\rceil)} \;-\; E(X_{(\lceil k+\frac{n}{2\log n}\rceil)})| \;>\; \frac{\frac{n}{4\log n}-1}{n+1}\,].$$

It follows that there is a positive N_1 such that $n > N_1$ implies $P[\overline{C}] \leq$

$$P[\,|X_{(\lceil k-\frac{n}{2\log n}\rceil)} \;-\; E(X_{(\lceil k-\frac{n}{2\log n}\rceil)})| \;>\; \frac{n}{5(n+1)\log n}\,] +$$

$$P[\,|X_{(\lceil k+\frac{n}{2\log n}\rceil)} \;-\; E(X_{(\lceil k-\frac{n}{2\log n}\rceil)})| \;>\; \frac{n}{5(n+1)\log n}\,]$$

$$\leq \text{ (by Chebyshev's inequality and inequality (4))}$$

$$2 \cdot \frac{1}{4(n+2)} \, \frac{25(n+1)^2 \log^2 n}{n^2}.$$

Thus,

$$n > N_1 \text{ implies } P[\overline{C}] \leq \frac{50(n+1)^2 \log^2 n}{4n^2(n+2)}. \tag{6}$$

Therefore, it follows from inequalities (5) and (6) that

$$n > \max\{N_0, N_1\} \text{ implies } P[\,\overline{B \cap C}\,] \;=\; P[\,\overline{B} \cup \overline{C}\,] \leq P[\,\overline{B}\,] \;+\; P[\,\overline{C}\,] \leq \frac{75(n+1)^2 \log^2 n}{4n^2(n+2)}. \tag{7}$$

It follows that

$$lim_{n\to\infty}\, P[B \cap C] \;=\; lim_{n\to\infty}\, [\,1 - \frac{75(n+1)^2 \log^2 n}{4n^2(n+2)}\,] \;=\; 1.$$

Thus, for large n, there is high probability that the algorithm finishes its work in $\Theta(\log n)$ time, hence at optimal $\Theta(n)$ cost.

4.3 Expected Case

We have observed that the loop dominates the running time of the algorithm, that each performance of the loop body requires $\Theta(\log n)$ time, and that the number c of performances of the loop body satisfies $c = O(\log^{(2)} n)$. Then there is a positive constant D such that $c \leq D \log^{(2)} n$. Without loss of generality, we may assume $D \log^{(2)} n$ is an integer. It follows that the expected running time of our algorithm is $O(\Sigma_{i=1}^{D\log^{(2)} n} P[A_i]\, i \, \log n)$, where A_i is the event that the loop body is performed exactly i times. But notice that for $i > 1$, $A_i \subset \overline{B \cap C}$. Therefore, the expected running time of our algorithm is

$$O(P[A_1]\log n \ + \ (\log n)\Sigma_{i=2}^{D\log^{(2)} n} i\, P[A_i]) \ = \ \text{(using inequality (7))}$$

$$O((\log n)\{P[A_1] \ + \ \frac{(D\log^{(2)} n)(1 \ + \ D\log^{(2)} n)}{2} \, \frac{75(n+1)^2 \log^2 n}{4n^2(n+2)}\}) \ = \ O(\log n).$$

Since one performance of the loop body requires $\Theta(\log n)$ time, it follows that the expected running time is $\Theta(\log n)$, and the expected cost is an optimal $\Theta(n)$.

5 Further Remarks

In this paper, we have given a parallel algorithm to solve the Selection Problem on an EREW PRAM of $\Theta(\frac{n}{\log n})$ processors. Our algorithm has the same worst-case running time, $\Theta(\log n \log^{(2)} n)$, as that of the best previous algorithm in the literature [3]. However, our algorithm has an expected running time of $\Theta(\log n)$, which is expected-case optimal for this architecture, since the expected cost is an optimal $\Theta(n)$.

6 Acknowledgment

We thank Richard Cramer-Benjamin for his suggestions.

References

[1] R. Cole, Parallel merge sort, *SIAM J. Comput.* 17(4) (1988), 770-785.

[2] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, McGraw-Hill, New York, 1989.

[3] J. JáJá, *An Introduction to Parallel Algorithms*, Addison-Wesley, Reading, MA, 1992.

[4] Miller, R. and L. Boxer, *Algorithms Sequential and Parallel*, Prentice Hall, 2000.

[5] Ramnath, S., and V. Raman, Selecting small ranks in EREW PRAM, *Information Processing Letters* 71 (1999), 183-186.

[6] Rice, J.A., *Mathematical Statistics and Data Analysis*, 2nd ed., International Thomson Publishing, Belmont, CA, 1995.