Selection from Read-Only Memory with Limited Workspace

Srinivasa Rao Satti
Seoul National University

Joint work with Amr Elmasry, Daniel Dahl Juhl, and Jyrki Katajainen

Outline

- Models and Problem definition
- Results
- Wavelet stack
- Main ideas
- Conclusions

Space-constrained algorithms for read-only memory

- Model:
  - The input is stored on read-only memory.
  - A small amount of additional workspace is given, to store intermediate results.
  - (Output is written onto a write-only memory.)
- Interested in studying the trade-offs between the amount of additional workspace and the running time of the algorithm.

Motivation and Applications

- **Massive-parallel computing**

  Input data are shared with many processors.

  Easy to design algorithms with read-only arrays,

  (to avoid concurrent writes).

- **Flash Memory**

  Reading is fast but writing is slow; no in-place updates; writing also reduces the lifetime of the memory.

- **Embedded software**

  e.g., digital camera, scanner, wireless sensor array

  Input data are stored outside with random access.

- **Theoretical curiosity.**

Selection Problem

Input: An array A of n elements (integers)

Queries: `select`(i) returns the i^{th} smallest element in A

Models: A is stored in read-only memory;

S bits of additional workspace is allowed.

- Multipass streaming model: input can only be accessed sequentially (several passes over the input are allowed).
- Space-restricted random-access model: random access to the input is allowed.

Memory model

- Word RAM model with word size $\Theta(\log n)$ supporting
  - read/write
  - addition, subtraction, multiplication, division
  - left/right shifts
  - AND, OR, XOR, NOT

operations on words in constant time.

(n is the “problem size”)

Results

	Workspace in bits	Running time
Munro and Raman	$\Theta(\lg N)$	$O(N^{1+\varepsilon})$
Raman and Ramnath	$\Theta(\lg^2 N)$	$O(N \lg^2 N)$
Frederickson	$\Theta(\lg^3 N)$	$O(N \lg N / \lg\lg N)$
Frederickson	$\Theta(N)$	$O(N \log^* N)$
Blum et al.	$\Theta(N \lg N)$	$\Theta(N)$
This paper	$O(N)$	$\Theta(N)$

Trade-off results: with O(S) bits, where $S = \Omega(\lg^3 N)$

Frederickson: $O(N \lg^* ((N \lg N)/S) + N (\lg N)/(\lg S))$

This paper: $O(N \lg^* (N/S) + N (\lg N)/(\lg S))$

Lower bounds

- [Chan, 2010]
  - In the multipass streaming model, any selection algorithm that uses a working space of $O(N)$ bits requires $\Omega(N \lg^* N)$ time.

- Our result separates the multipass streaming model from the space-restricted random-access model, as it “surpasses” this lower bound.

General approach

When space is $\Omega(\log^2 N)$ bits

- Maintain a pair of *filters*, m and M such that the i^{th} element is between these values. Elements within this range are called *active* elements.
- In each iteration:
  - Scan through the array to find an *approximate median* of the active elements.
  - Split the active elements into two halves using the approximate median and recurse on the appropriate half (setting one of the filters to be the approximate median).

Using O(N) bits

- We can keep track of the active elements after each level using bit vectors.
- We start with a bit vector of length N, and shrink it by a constant factor after each level. So, the total space is O(N) bits.
- Need to efficiently scan through the ones or zeros in a bit vector. We build a *wavelet stack* to do this. It uses a data structure for a bit vector that supports *rank/select*.

Rank/Select on a bit vector

Given a bit vector B

$rank_1(i)$ = # 1's up to position i in B

$select_1(i)$ = position of the i-th 1 in B

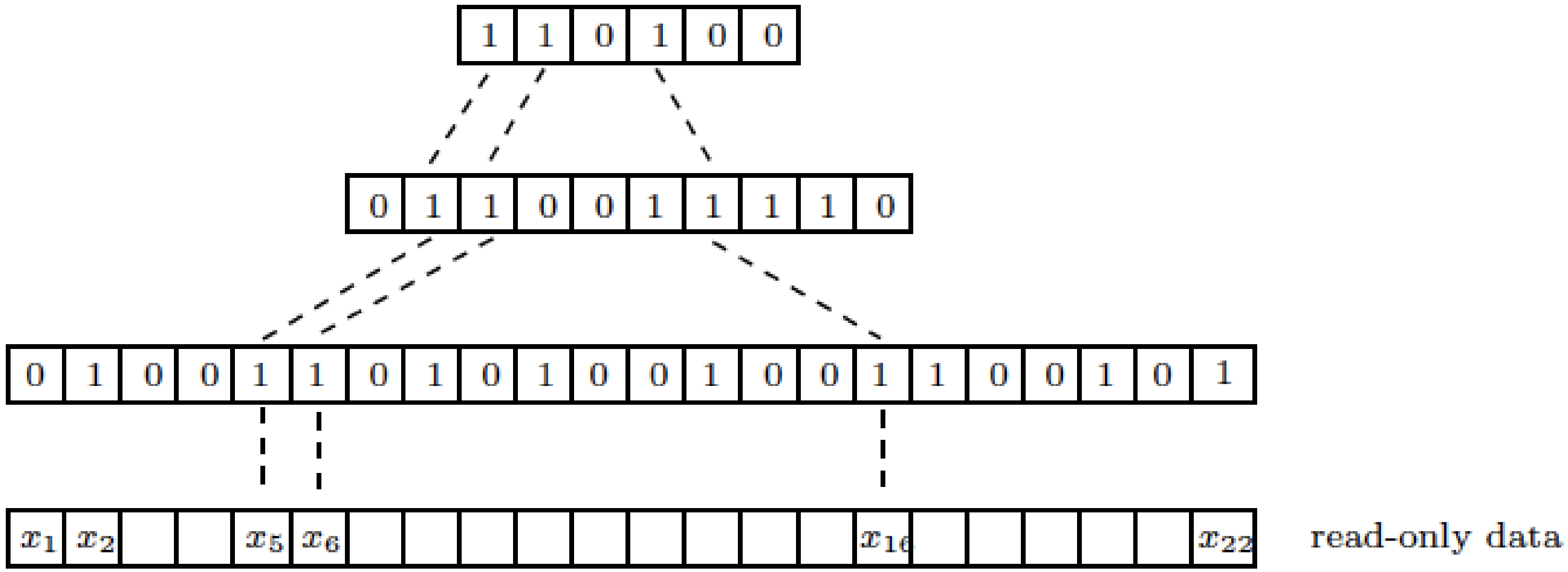
(similarly $rank_0$ and $select_0$)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B:	0	1	1	0	1	0	0	0	1	1	0	1	1	1	1

$rank_1(5) = 3$
$select_1(4) = 9$
$rank_0(5) = 2$
$select_0(4) = 7$

Given a bit vector of length n, by storing an additional o(n)-bit structure, we can support all four operations in O(1) time.

Wavelet stack



With each bit vector, we also store an auxiliary structure to support *rank/select* in constant time.

Checking whether an element is active at level h, or finding the i^{th} active element at level h takes O(h) time.

Selection using O(N) bits

- Scanning all the active elements at level h takes $O(h\ N/c^h)$ time, for some constant c.

$$\sum_{h=0}^{\infty} h\ N/c^h = O(N)$$

- In i^{th} iteration, the median algorithm and the partitioning takes $O(i\ N/c^i)$ time.
- Thus the total running time is O(N).

"Pruning" algorithms

- Munro-Paterson find an $(S/\lg^c N)$-sample in $O(N \lg S)$ time (and in one pass).
  - Gives an $O(N \lg S + N \lg_S N)$ algorithm for selection.

- Frederickson describes a way to prune a set (of active elements) of size $N/\lg^{(a)} N$ elements down to $N/\lg^{(a-1)} N$ elements in $O(N)$ time.
  - Gives an $O(N \lg^* S + N \lg_S N)$ time algorithm for selection.

Selection with S bits

$S = \Omega(\lg^3 N)$

- Apply Frederikson's "pruning" algorithm until there are only S active elements left.
  - Takes $O(N \lg^* (N/S))$ time.

- Divide the input array into blocks of size N/S, and use a wavelet stack of size O(S) bits to maintain active blocks, and their cardinalities.

- Overall time: $O(N \lg^* (N/S) + N (\lg N)/(\lg S))$

Conclusions

- We obtained a selection algorithm for the random-access model that supports queries in O(N) time using O(N) bits of working space.
- Also obtained a better trade-offs when we are given a working space of S bits, for $\lg^3 N \leq S \leq N$.

- Determining the exact complexity of the selection problem is still open.

References

- Blum, Floyd, Pratt, Rivest, Tarjan. JCSS-1973
- Timothy Chan. ACM TAlg-2010
- Frederickson. JCSS-1987
- Munro, Paterson. TCS-1980
- Munro, Raman. TCS-1996
- Raman, Ramnath. NJC-1999.

Thank you