

Lower bounds on the size of selection and rank indexes

Peter Bro Miltersen*

1 Introduction

The *rank index problem* is the following: Preprocess and store a bit string $x \in \{0, 1\}^n$ on a random access machine with word size w so that *rank* queries “What is $\sum_{i=1}^j x_i$?” for arbitrary values of j can afterwards be easily answered. The *selection index problem* is the following: Preprocess and store a bit string $x \in \{0, 1\}^n$ so that *selection* queries “What is the index of the j ’th 1-bit in x ?” for arbitrary values of j can afterwards be easily answered. The data structure representing x should be an *index structure*, i.e., the n -bit string x is kept *verbatim* in $\lceil n/w \rceil$ words and the preprocessing phase adds an r -bit *index* $\phi(x)$ with additional information contained in $\lceil r/w \rceil$ words. We are interested in tradeoffs between r , the size of the index measured in bits (the *redundancy* of the scheme), and t , the worst case time for answering a query.

Upper bounds for the rank and selection index problems were discussed by Jacobson [4], Clark [1], Munro [5], Munro, Raman and Rao [6] and Raman, Raman and Rao [7]. The most relevant case is $w = O(\log n)$ and $t = O(1)$. For these parameters, the best known solutions for both problems have $r = O(n \log \log n / \log n)$ (the upper bound for selection indexes being the most recent one, explicit in the journal version of [7] only).

Here we show lower bounds for the rank and selection index problems in the cell probe model with word size w . That is, when measuring the query time, we are charged one unit of time each time we access a w -bit register, while computation is for free. Such lower bounds clearly also apply to random access machines with the same word size. We show:

THEOREM 1.1. *For any index for selection queries using word size w with redundancy r and query time t it holds that $3(r + 2)(tw + 1) \geq n$. Also, for any index for rank queries using word size w with redundancy r and query time t it holds that $2(2r + \log_2(w + 1))tw \geq n \log_2(w + 1)$.*

Thus, for the case of $t = O(1)$ and $w = O(\log n)$, we obtain the lower bound $r \geq \Omega(n / \log n)$ for selection

indexes and the lower bound $r \geq \Omega(n \log n \log n / \log n)$ for rank indexes. The latter matches the best known upper bound while the former does not. We leave as an open problem to get tight bounds for the size of a selection index. The two proofs are based on two quite different counting arguments, both being rather simple. The proof for selection indexes is related to (but simpler than) the lower bound proof for substring search of Demaine and López-Ortiz [2] and the lower bound for rank indexes is based on the “exposure game” technique of Gál and Miltersen [3].

2 Lower bound for selection

Fix a scheme with parameters n, w, r, t . The form of the lower bound to be proved makes it without loss of generality to assume that $w = 1$. We can also assume $t \geq 1$ as this is true in any valid scheme and $n \geq 12$, as there is otherwise nothing to prove. Given a bit string $x \in \{0, 1\}^n$, we construct another bit string $\tau(x)$ and argue that the index structure $\phi(x)$ together with $\tau(x)$ is sufficient to reconstruct the original data x . Let $m = \lfloor (n/3 - 1)/t \rfloor$ and restrict attention to inputs x of Hamming weight m . Given x , we first construct $\phi(x)$ and then define a string $\tau'(x) \in \{0, 1\}^{mt}$ as follows: We run the selection query operation with parameter j for each $j = 1, \dots, m$ and concatenate the contents of the registers read by these operations. As each query operation inspects t registers, each containing one bit, the total length of $\tau'(x)$ is mt bits. We now obtain $\tau(x)$ by initially setting $\tau(x) = \tau'(x)$ and then erasing certain bits of $\tau(x)$ by setting them to zero by applying the following two rules:

- If $\tau'(x)_i$ corresponds to a bit of the index structure $\phi(x)$ we set $\tau(x)_i$ to 0.
- If $\tau'(x)_i$ and $\tau'(x)_k$ with $i < k$ are both copies of the *same* bit in the input x , we let $\tau(x)_k = 0$.

We now observe that if we have access to $\phi(x)$ and $\tau(x)$ and know the query algorithm, we can reconstruct $\tau'(x)$ without having access to x . This is done by simulating the selection queries with parameter j for each $j = 1, \dots, m$ in increasing order while scanning the string $\tau(x)$ from left to right, thus associating to each bit of $\tau(x)$ an address in the structure $x \cdot \phi(x)$. Each

*Department of Computer Science, University of Aarhus. Supported by **BRICS**, Basic Research in Computer Science, a centre of the Danish National Science Foundation.

time a bit of value 0 is read from $\tau(x)$ we check if this bit should be 1 instead of 0 in $\tau'(x)$ by checking if the bit is associated with a bit in the index structure $\phi(x)$ and if that bit in the index structure is 1, or if a bit in $\tau(x)$ of value 1 we previously read is a copy of the same bit of x . Having in this way reconstructed $\tau'(x)$ we can reconstruct x , as we can now tell the answers to all selection queries of parameters $j = 1, \dots, m$.

Note that $\tau(x) \in \{0, 1\}^{tm}$ is a bit vector of Hamming weight at most m . Since the information in $\tau(x)$ and $\phi(x) \in \{0, 1\}^r$ is sufficient to reconstruct x , we have $r + \log_2 \sum_{i=0}^m \binom{tm}{i} \geq \log_2 \binom{n}{m}$. In particular, since $m < n/3$, we have $r + \log_2 \binom{tm}{m} + 1 \geq \log_2 \binom{n}{m}$, or $r + 1 \geq \log_2 \binom{n}{m} - \log_2 \binom{tm}{m} = \log_2 \frac{n(n-1)\dots(n-m+1)}{tm(tm-1)\dots(tm-m+1)} \geq \log_2(((n-m+1)/((t-1)m+1))^m) \geq \log_2(2^m) = m$. So $r+1 \geq \lfloor (n/3-1)/t \rfloor$ which implies that $(r+2)(t+1) \geq n/3$, as desired.

3 Lower bound for rank

Consider the following *vector addition problem*. Let G be a finite commutative group. Given k vectors $y_1, y_2, \dots, y_k \in G^m$, our task is to find their sum $\sum y_i \in G^m$, the km entries in the vectors not initially being revealed to us. Instead we are allowed to *expose* any desired entry $y_{ij} \in G$. Which entry to expose may be decided adaptively, i.e., we can base our choice of which entry y_{ij} to expose next on the values of the entries already exposed. We are only allowed to expose $km/2$ entries and must then make a guess for the value of $\sum y_i$. Our objective is to maximize the probability that our answer is correct when the vectors are chosen uniformly at random and independently from G^m . We claim that any strategy for solving the vector addition problem will give the correct output with probability at most $|G|^{-m/2}$. To see this, consider an execution of any strategy on randomly chosen vectors y_1, \dots, y_k . By the *trace* T of the execution we mean a full description of the indices and values of the entries exposed and the final guess given. Our claim follows by showing that the conditional probability of the guess being correct is at most $|G|^{-m/2}$, conditioned by any particular trace T . Fix a trace T . Let $S = \{j \mid T \text{ leaves } y_{ij} \text{ unexposed for some value of } i\}$. As only $km/2$ entries have been exposed in T , we have that for at least $m/2$ values of j , it holds for some i , y_{ij} is unexposed, i.e., $|S| \geq m/2$. Since G is a group, for fixed T , the variables $(\sum_{i=1}^n y_{ij})$, $j \in S$ are mutually independent and uniformly distributed in G . Thus, the probability that the answers written in T for the set of entries in S are all correct is at most $|G|^{-|S|} \leq |G|^{-m/2}$, as desired.

Now let us consider a solution to the rank index

problem with parameters n, w, r, t . Let $k = 2t$ and let $m = \lfloor \frac{n}{kw} \rfloor$. We shall consider the vector addition problem for $y_1, y_2, \dots, y_k \in G^m$ with $G = \mathbf{Z}/(w+1)\mathbf{Z}$ and show that the claimed rank index structure leads to a strategy for solving this problem with a certain good success probability. For $v \in G$, let $b(v) \in \{0, 1\}^w$ be the *unary* representation of v , i.e., $b(i) = 1^i 0^{w-i}$. Also, given vectors $y_1, y_2, \dots, y_k \in G^m$ let $b(y_1, \dots, y_k)$ be the bit string $b(y_{11}) \cdot b(y_{21}) \cdots b(y_{k1}) \cdot b(y_{12}) \cdot b(y_{22}) \cdots b(y_{k2}) \cdots \cdots b(y_{1m}) \cdots b(y_{km})$, padded with zeroes in the end to make its length exactly n . Our strategy for the vector addition problem is as follows. We *guess* the index structure $\phi(x)$ for the string $x = b(y_1, \dots, y_k)$ by picking a random bit vector in $\{0, 1\}^r$. Now we may simulate the query algorithm of the data structure as follows: Each time we want to read part of the index, we already know it. Each time we want to read a word of the original data x (i.e., w consecutive bits), we may do so by exposing an entry of the vectors. Thus, we may simulate the query algorithm for all queries “What is $\sum_{i=1}^{jwk} x_i$?”, $j = 1, \dots, m$, thereby exposing at most $mt = km/2$ entries of the vectors y_1, y_2, \dots, y_k . If the answers to the queries are all correct, we can output the correct value of $\sum y_i$ and this will happen if we guessed the index structure correctly, i.e., with probability 2^{-r} . Thus, by our bound on the vector addition problem, we have $2^{-r} \leq |G|^{-m/2}$ which gives us $2(2r + \log_2(w+1))tw \geq n \log_2(w+1)$, as desired.

Acknowledgements I would like to thank S. Srinivasa Rao, Roberto Grossi and an anonymous SODA reviewer for very helpful comments.

References

- [1] David R. Clark. “Compact Pat Trees”. PhD-thesis. Department of Computer Science, University of Waterloo, Canada, 1996.
- [2] Erik D. Demaine, Alejandro López-Ortiz. A linear lower bound on the index size for text retrieval. *J. Algorithms* 48(1), pages 2-15 (2003).
- [3] Anna Gál, Peter Bro Miltersen. The cell probe complexity of succinct data structures. *ICALP 2003*, pages 332-344.
- [4] Guy Jacobson. Space-efficient Static Trees and Graphs. *FOCS 1989*, pages 549-554.
- [5] J. Ian Munro: Tables. *FSTTCS 1996*, pages 37-40.
- [6] J. Ian Munro, Venkatesh Raman, S. Srinivasa Rao. Space Efficient Suffix Trees. *J. Algorithms* 39(2), pages 205-222 (2001)
- [7] R. Raman, V. Raman and S. S. Rao. Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. *SODA 2002*, pages 233-242.