Journal of Discrete Algorithms 36 (2016) 3–17

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Near-optimal online multiselection in internal and external memory ☆

Jérémy Barbay [a,1], Ankur Gupta [b,2], Srinivasa Rao Satti [c,*,3], Jon Sorenson [b]

[a] Departamento de Ciencias de la Computación (DCC), Universidad de Chile, Chile
[b] Department of Computer Science and Software Engineering, Butler University, United States
[c] School of Computer Science and Engineering, Seoul National University, Republic of Korea

ARTICLE INFO

Article history:
Available online 10 November 2015

Keywords:
Selection
Sorting
Multiselection
Online algorithms
Deferred data structures
Dynamic
External memory

ABSTRACT

We introduce an online version of the *multiselection* problem, in which q selection queries are requested on an unsorted array of n elements. We provide the first online algorithm that is 1-competitive with the offline algorithm proposed by Kaligosi et al. [14] in terms of comparison complexity. Our algorithm also supports online *search* queries efficiently. We then extend our algorithm to the dynamic setting, while retaining online functionality, by supporting arbitrary *insertions* and *deletions* on the array. Assuming that the insertion of an element is immediately preceded by a search for that element, our dynamic online algorithm performs an optimal number of comparisons, up to lower order terms and an additive $O(n)$ term.

For the external memory model, we describe the first online multiselection algorithm that is $O(1)$-competitive. This result improves upon the work of Sibeyn [20] when $q = \omega(m^{1-\epsilon})$ for any fixed positive constant ϵ, where m is the number of blocks that can be stored in main memory. We also extend it to support searches, insertions, and deletions of elements efficiently.

1. Introduction

The *multiselection* problem asks for elements of rank r_i from the sequence $R = \{r_1, r_2, \ldots, r_q\}$ on an unsorted array $\mathtt{A}$ of size n drawn from an ordered universe of elements. We define $\mathcal{B}(S_q)$ as the information-theoretic lower bound on the number of comparisons required (in the comparison model) to answer q selection queries, where $S_q = \{s_1, s_2, \ldots, s_q\}$ denotes the queries ordered by rank. This lower bound can be obtained by taking the number of comparisons needed to sort the entire array, and then subtracting the comparisons needed to sort the query gaps. (Please see Section 2.2 for more details on this bound.) The *online multiselection* problem asks for elements of rank r_i, where the sequence R is given one element at a time. The lower bound $\mathcal{B}(S_q)$ also applies to *search* queries in the offline model, as well as to both types of queries in the online model.

☆ A preliminary version of these results have appeared in the proceedings of ESA-2013 [2] and WALCOM-2015 [3].

* Corresponding author.
E-mail addresses: jeremy@barbay.cl (J. Barbay), agupta@butler.edu (A. Gupta), ssrao@cse.snu.ac.kr (S.R. Satti), jsorenso@butler.edu (J. Sorenson).

[1] Supported in part by the Millennium Nucleus "Information and Coordination in Networks" ICM/FIC RC130003.
[2] Supported in part by the Areté Initiative at the University of Chicago and the Butler Holcomb Awards grant.
[3] Supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (Grant number 2012-0008241).

http://dx.doi.org/10.1016/j.jda.2015.11.001
1570-8667/

The *dynamic online multiselection* problem supports *select*, *search*, *insert* and *delete* operations, described below:

- *select*(s), returns the position of the sth smallest element in $\mathtt{A}$;
- *search*(p), returns the position of the largest element $y \leq p$ from $\mathtt{A}$;
- *insert*(p), inserts p into $\mathtt{A}$; and
- *delete*(s), deletes the sth smallest element from $\mathtt{A}$.

1.1. Previous work

Offline Multiselection. Several papers have analyzed the offline multiselection problem, but all of these approaches required the queries to be known in advance. Dobkin and Munro [8] gave a deterministic algorithm for this problem using $3\mathcal{B}(S_q) + O(n)$ comparisons. Prodinger [18] proved the expected comparisons with random pivot selection is at most $2\mathcal{B}(S_q)\ln 2 + O(n)$. More recently, Kaligosi et al. [14] described a randomized algorithm that uses $\mathcal{B}(S_q) + O(n)$ expected comparisons, along with a deterministic algorithm that performs $\mathcal{B}(S_q) + o(\mathcal{B}(S_q) + O(n)$ comparisons. Jiménez and Martínez [13] later improved the bound on the number of comparisons in the expected case to $\mathcal{B}(S_q) + n + o(n)$, when $q = o(\sqrt{n})$. Cardinal et al. [7] generalized the problem to *partial order production* (of which multiselection is a special case), and they used multiselection as a subroutine after an initial preprocessing phase.

In the external memory model [1] with internal memory M and block size B, we use N to denote the number of elements in $\mathtt{A}$. We also define $n = N/B$ and $m = M/B$ in external memory. Sibeyn [20] solved external multiselection using $n + nq/m^{1-\epsilon}$ I/Os, where ϵ is any fixed positive constant. The first term comes from creating a static index structure using n I/Os, and the remaining $nq/m^{1-\epsilon}$ comes from answering q searches using that index. In addition, this result requires the condition that $B = \Omega(\log_m n)$.[4] When $q = m$, Sibeyn's multiselection algorithm takes $O(nm^{\epsilon})$ I/Os, whereas the optimum is $\Theta(n)$ I/Os. In fact, his bounds are $\omega(\mathcal{B}_m(S_q))$, for any $q = \omega(m^{1-\epsilon})$, where $\mathcal{B}_m(S_q)$ is the lower bound on the number of I/Os required. (See Section 6.1 for the definition of $\mathcal{B}_m(S_q)$.)

Online Multiselection. Motwani and Raghavan [17] introduced the static online multiselection problem, where selection and search queries arrive one at a time, as a "Deferred Data Structure" for sorting. (In other words, the input array is sorted over time, as queries are answered.) Barbay et al. [2] described a simpler solution with an improved analysis that matched the offline results of Kaligosi et al. [14]. Ching et al. [21] extended Motwani and Raghavan's solution [17] to support insertion and deletion, with optimal amortized complexity in the worst case over instances with a fixed number q of queries. Our solution is simpler, and our analysis finer, in the worst case over instances where the query positions are fixed. To the best of our knowledge, there are no existing dynamic results for the multiselection problem in the external memory model.

1.2. Our results

For the *dynamic online multiselection* problem in internal memory, we describe the *first* algorithm that supports a sequence R of q selection, search, insert, and delete operations, of which q' are search, insert, and delete, using $\mathcal{B}(S_q) + o(\mathcal{B}(S_q) + O(n + q'\log n)$ comparisons.[5] For the *online multiselection* problem (when $q' = 0$), our algorithm is 1-competitive with the offline algorithm of Kaligosi et al. [14] in the number of comparisons performed. In addition, we obtain a randomized result that matches (i.e., is 1-competitive with) the performance of Kaligosi et al. [14], while only using $O((\log n)^{O(1)})$ sampled elements instead of $O(n^{3/4})$ elements.

For the external memory model [1], we describe an *external online multiselection* algorithm on an unsorted array $\mathtt{A}$ of size N, using $O(\mathcal{B}_m(S_q))$ I/Os, where $\mathcal{B}_m(S_q)$ is a lower bound on the number of I/Os required to support the given queries. This result improves upon the work of Sibeyn [20] when $q = \omega(m^{1-\epsilon})$ for any fixed positive constant ϵ. We also extend it to support search, insert, and delete operations using $O(\mathcal{B}_m(S_q) + q\log_B N)$ I/Os.

1.3. Preliminaries

Given an unsorted array $\mathtt{A}$ of length n, the *median* of $\mathtt{A}$ is the element x such that $\lceil n/2 \rceil$ elements in $\mathtt{A}$ are no greater than x. It is well-known that the median can be computed in $O(n)$ comparisons, and many [11,5,19] have analyzed the exact constants involved. Dor and Zwick [9] provided the best known constant, yielding $2.942n + o(n)$ comparisons.

With a linear time median-finding algorithm, one can obtain a linear time algorithm to select the element of a specified rank r in $\mathtt{A}$. Dobkin and Munro [8] considered the extension of this selection problem to the multiselection problem, and gave an algorithm that requires an asymptotically optimal number of comparisons. As mentioned earlier, Kaligosi et al. [14]

[4] We use the notation $\log_b a$ to refer to the base b logarithm of a. By default, we let $b = 2$. We also define $\ln a$ as the base e logarithm of a.

[5] For the dynamic portion of the result, we make the (mild) assumption that the insertion of an element is immediately preceded by a search for that element. In that case, our dynamic online algorithm performs an optimal number of comparisons, up to lower order terms and an additive $O(n)$ term.