Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Finding modes with equality comparisons ☆

Varukumar Jayapaul [a,*], J. Ian Munro [b], Venkatesh Raman [c], Srinivas Rao Satti [d]

[a] Chennai Mathematical Institute, H1, SIPCOT IT Park, Siruseri, Chennai 603 103, India
[b] Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
[c] The Institute of Mathematical sciences, C.I.T. Campus, Chennai 600 113, India
[d] Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, 151-744, Republic of Korea

ARTICLE INFO

Article history:
Received 4 January 2017
Received in revised form 30 May 2017
Accepted 8 September 2017
Available online xxxx
Communicated by G.F. Italiano

Keywords:
Mode finding
Equality comparisons
Turán's theorem
Dirac's theorem
Lower bound
Adversary argument

ABSTRACT

We consider the comparison complexity of finding modes (the most frequently occurring elements) in a list of elements that are not necessarily from a totally ordered set. Here, the relation between elements is determined by *equality* comparisons whose outcome is $=$ when the two elements being compared are equal, and $\neq$ otherwise. The problem generalizes the classical majority problem studied in this model (using equalities). We show that $n^2/2m - n/2$ comparisons are necessary and $n^2/m + n$ comparisons are sufficient to find an element that appears at least m times. This is in sharp contrast to the bound of $\Theta(n \log(n/m))$ bound in the model where comparisons are $<, =, >$ or $\leq, >$.

We give three algorithms for finding mode, including one that is a generalization of a classical majority finding algorithm due to Fischer and Salzberg (1982) [9]. We also discuss upper and lower bounds for sorting (i.e., finding the frequency of every element) and for finding the least frequent element. Sorting problem (under the equality comparisons) also known as *equivalence class sorting*, has applications in several scenarios where the total order of elements is either not possible or can not be revealed for security reasons.

1. Introduction

Sorting and selection are fundamental well studied problems in computer science. We consider these problems when the input sequence is not necessarily from a totally ordered set and so there is no notion of an inherent linear or total order.

The only way the relation between a pair of elements is determined in this scenario is by making equality comparisons. While this is a natural variant that occurs when dealing with heterogenous sets of elements or elements that do not have a total order (say, for example, the elements are subsets of a universe), to the best of our knowledge the only problem studied extensively in this model is the problem of determining the majority element (an element that appears at least $\lceil n/2 \rceil$ times) if it exists, and there is a classical linear time algorithm for this [5]. Exact comparison complexity including upper and lower bounds, and average case complexity of the majority problem have been studied [1–3,9,16].

In this paper, we explore the natural problem of determining a mode (a most frequently occurring element), of sorting (determining the frequency of every element) and of finding a least frequent element in this model. We show that $\Theta(n^2/m)$

☆ Preliminary versions of this paper appeared in WADS 2015 [10] and WALCOM 2016 [11].

* Corresponding author.
E-mail addresses: varukumarj@cmi.ac.in (V. Jayapaul), imunro@uwaterloo.ca (J. Ian Munro), vraman@imsc.res.in (V. Raman), ssrao@cse.snu.ac.kr (S.R. Satti).

http://dx.doi.org/10.1016/j.tcs.2017.09.007
0304-3975/

(equality) comparisons are necessary and sufficient to find an element that appears at least m times. This is in sharp contrast to the bound of $\Theta(n \log(n/m))$ [8] bound in the traditional comparison model. The lack of transitivity of the inequality comparisons throws interesting challenges.

We begin the development of our mode finding algorithms in Section 2 with a simple algorithm that takes at most $2n^2/m$ comparisons where m is the frequency of the mode. This appears in Section 2.1. Then, by generalizing a classical majority algorithm due to Fischer and Salzberg [9], we improve the number of comparisons to at most $(3/2)(n^2/m) + O(n^2/m^2)$ in Section 2.2. Our final algorithm in Section 2.3 takes at most $n^2/m + n$ comparisons and can be implemented in $\tilde{O}(n^2)$ time.[1] Section 4 proves asymptotically matching lower bound for finding a mode.

In Section 3 we discuss the sorting problem where one needs to determine all the distinct elements and their frequencies in the given input. This has applications, for example in Graph Mining, where we are given a collection of graphs, and we need to group them into those that are isomorphic to each other, and the only oracle we know is an algorithm to detect whether two graphs are isomorphic or not. See [17] for examples of other application scenarios of this sorting problem (known as *equivalence class sorting*) in security and distributed computing. We give upper bounds for sorting and finding a least frequent element, and also give lower bounds when all elements have the same frequency.[2] Finally, Section 5 concludes with remarks and open problems.

1.1. Related work

As mentioned earlier, we know of only the majority problem [5] studied extensively in the equality comparison model. Demaine, Lopez-Ortiz and Munro [6] studied the (lower and upper bounds on the) number of passes required to find (a superset of) elements that appear(s) at least $n/(k+1)$ times in the equality comparison model. Our first algorithm for mode in Section 2.1 is essentially the same algorithm though we analyze the number of comparisons (as opposed to the number of passes). In one of the earliest papers studying optimal algorithms on sets, Reingold [15] proved lower bounds for determining the intersection/union of two sets if only $=, \neq$ comparisons are allowed. Regarding mode, Munro and Spira [13] considered optimal algorithms and lower bounds to find a mode and the spectrum (the frequencies of all elements), albeit in the three way comparison model. Misra and Gries [14] gave algorithms to determine an element that appears at least n/k times for various values of k, in the three way comparison model.

2. Finding mode (or elements with specific frequency)

Given a set of elements, a *mode* of the input is defined as any element which occurs the maximum number of times in the input. The input can have several modes. A natural randomized algorithm to find a mode (given its frequency m) in this model is to pick a random element and find its frequency by comparing it with all other elements. If m is the frequency of a mode, then the probability that this algorithm picks a mode in any given round is m/n, and hence in about n/m rounds, the algorithm finds a mode with high probability. As it makes $n-1$ comparisons in each round, the expected number of comparisons is around n^2/m. A high confidence bound can then be shown for this randomized approach. We show that this bound of $O(n^2/m)$ is achievable by a deterministic algorithm even without the knowledge of m. In addition, we give an adversary argument to show that $\Omega(n^2/m)$ comparisons are necessary.

In this section, we provide three algorithms to find a mode, each subsequent algorithm improving upon the earlier one in terms of the number of comparisons performed. The first two algorithms not only use $O(n^2/m)$ comparisons, but also spend only $O(n^2/m)$ time for the rest of the operations. The first one takes at most $2n^2/m$ comparisons, while the number of comparisons made by the second one is $3n^2/2m + O(n^2/m^2)$. The first one is relatively simple to argue correctness, and the second algorithm generalizes a classical majority finding algorithm. Both these algorithms have a 'selection phase' where a candidate set of elements for a mode are selected, and a 'confirmation phase' where the candidates are confirmed and those who pass the confirmation tests are output. The third algorithm uses at most $n^2/m + n$ comparisons although our implementation uses $\tilde{O}(n^2)$ time for the other operations.

For now, we assume that m is known, and later, we explain how this assumption can be removed.

2.1. A simple mode finding algorithm

The first phase of the SimpleMode algorithm in this subsection is essentially the one (called FREQUENT) that appears in [6].

Let k be the smallest integer such that $\lfloor n/k \rfloor \leq m-1$, i.e., $\lfloor n/k \rfloor \leq m-1 < \lfloor n/(k-1) \rfloor$. Let $a_1, a_2, \ldots a_n$ be the given list of n elements. We give an algorithm that finds all elements with frequency more than $\lfloor n/k \rfloor$ from an input of size n. The pseudocode description is given in Algorithm 1. Here B is a set of distinct elements with some frequencies associated with each element.

[1] $\tilde{O}$ ignore logarithmic factors.

[2] Our lower bound for sorting elements when all elements have the same frequency has since been improved to match with our upper bound. See [17] for details on the improved lower bound, parallel algorithms for sorting, and several applications of the sorting problem, which is termed as 'sorting equivalence classes' in this model.

Please cite this article in press as: V. Jayapaul et al., Finding modes with equality comparisons, Theoret. Comput. Sci. (2017), http://dx.doi.org/10.1016/j.tcs.2017.09.007