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

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Note

Determining membership with 2 simultaneous queries

David Howard

Department of Mathematics, Colgate University, Hamilton, NY 13346, United States

ARTICLE INFO

Article history:
Received 29 April 2013
Received in revised form 21 March 2014
Accepted 27 May 2014
Available online xxxx
Communicated by G.F. Italiano

Keywords:
Searching
Sorting
Information retrieval
Cell-probe
Membership

ABSTRACT

Alice and *Bob* are playing a cooperative game in which *Alice* must devise a scheme to store n elements in an array from a universe U of size m. Her goal is to store in such a way that for every $x \in U$ *Bob* can observe the values of two positions (dependent on x) in the array and determine whether x is in the array or not. *Alice* may share her storage scheme with *Bob* and they win if such an arrangement is made. The question is how large can the universe U be in terms of n so that *Alice* and *Bob* can win? In this paper we give upper and lower bounds on this question for general n and the special case when $n = 3$. We also pose conjectures and further questions for research.

1. Introduction

Sorting and searching are fundamental operations that we use everyday in many different ways. Our assumptions change from problem to problem and we are frequently concerned with having these operations done as quickly as possible. We are sometimes concerned with how long an operation will take us to complete in the worst case, other times knowing the average amount of time to complete such an operation is important. There is a large amount of research in this area and for an encyclopedic but by no means comprehensive discussion in the area we refer the reader to [4].

Let us look at a particular but normal instance of sorting and searching. Suppose *Alice* has a list of n objects taken from a Universe U and we must choose a good storing scheme so that if *Bob* wants to determine whether a particular element x is among our n objects he doesn't need to search the entire string to decide whether x is in the list or not. Well immediately questions come to mind:

- Is *Bob* aware of the storage scheme?
- What do we know about the universe?
  - Do we know what all the possible items are in the universe?
  - Are the items comparable with each other in some way (i.e. can they be ordered)?
  - Is the universe finite and if so how does it compare to n?

Well the answers to both knowing the storage scheme and comparability must be yes otherwise *Bob* may have to traverse the whole list on some searches. Given these two assumptions perhaps the most natural thing to do is to look at the storage scheme that completely sorts the n element input. Now when *Bob* wants to determine if a particular element is in the list he can perform a binary search of the list which runs in a logarithmic number of steps. Knowing nothing else

E-mail address: dmhoward@colgate.edu.

http://dx.doi.org/10.1016/j.tcs.2014.05.020
0304-3975/

Please cite this article in press as: D. Howard, Determining membership with 2 simultaneous queries, Theor. Comput. Sci. (2014), http://dx.doi.org/10.1016/j.tcs.2014.05.020

about U this strategy has the best possible average run time. That being said it is often the case that we have additional assumptions on U. For example, perhaps we have knowledge of the elements in U and that $|U|$ is finite. Now choosing to sort the elements in order and having *Bob* perform binary search may not be the best method of quickly finding membership among the list.

Let us suppose we have knowledge of the elements of the universe and that the universe is finite. Knowing the elements of the set means that we can assign numerical values to each element and impose an order. Thus, we can assume our universe is the set of integers from 1 to some integer m. Without any knowledge of the value of m compared to n, sorting in order and using a binary search to determine membership in our list may seem like a reasonable strategy. What if, however, $m = n + 1$? In this case in-order sorting and performing binary search is a rather poor strategy. If *Alice*, as a storage scheme, places the integer above (modular n) the missing element in the first position of the list, then *Bob* by looking at the value stored in the first position can determine exactly who is in the list. *Bob* in this case needs to query a position in the list just once to complete his goal. This naturally begs the question: How big can m be so that only one query is needed to determine membership of any particular value from the list?

This question has been solved [16] and the answer is $m = 2n - 2$ for $n > 2$ (Note: for $n = 2$ the answer is 3). To show that $2n - 2$ is possible we list the storage scheme from [16].

Let the universe be the set $[2n - 2]$ and our storage list be denoted as rooms with labels $1, 2, \ldots, n$. We say that each element $p \in [2n - 2]$ prefers to stay in the room $(p \bmod n)$ (In the case of 0 choose room n instead). We also denote the first n elements as lower elements and the last $n - 2$ elements as upper elements. We say room r is *full* if all members who prefer room r from the universe are among the n elements to be stored and *empty* if no members prefer room r. Note there are always at least two full rooms and rooms $n - 1$ and n are either full or empty whereas the other rooms may be neither. Storage is as follows:

- Any element that prefers a room that is not full is stored in that room.
- If an upper element prefers a room that is full it is stored in a room that is empty.
- If a lower element prefers a room that is full it is stored in a different room that is full.

To determine membership for an element $p \in [2n - 2]$ *Bob* queries room $p' = (p \bmod n)$ (In the case of 0 choose room n instead). If p' contains p or another lower element that does not prefer p' as a room then p is in one of the rooms, otherwise it's not.

We leave it to the reader to verify that this scheme will work. That being said perhaps the next logical question is: How big can m be if you allow two queries of the list to determine membership? Well, this question isn't well-defined because it isn't clear whether *Bob* can use the information obtained by one query to help decide where in the list *Bob* makes the second query. In an upcoming paper Paul Horn, Adam Jobson, Andre Kezdy, and Jake Wildstrom [5] give an adaptive membership algorithm (i.e. *Bob* can use the information from the first query), which is similar to Yao's algorithm, where $m = 3n - 4$. At the time of writing there has been *no* upper bound found for m regarding an adaptive strategy. The main results of this paper give upper and lower bounds for the non-adaptive question (*Bob* apriori must decide which two positions in the list he will query and then based on these answers determine whether a particular element from the universe is in the list).

We make one other point of note from Yao's article that given special additional arrangements two queries is enough to determine membership under the adaptive scheme. In fact, in this special case when Bob determines that someone (Person x) *is* among the stored rooms then the member seen by the second query will actually be Person x. Specifically, given an arbitrarily large universe and an additional room which is occupied with whomever Alice chooses, Bob can determine membership with only two queries. Bob determines who Alice has put in the additional room and this person acts as an index (Alice has many different storage schemes and the additional person tells Bob which scheme to choose from). Bob can then query a specific room which will either contain Person x or Bob will claim that Person x is not among the stored list.

2. Related work and previous results

Despite the simplicity of the question "Is x in one of the rooms?" from a computer science standpoint the question can quickly become quite difficult to even ask:

- How is the information stored (e.g. A linked list of integers? A zero-one array which has the size of my universe? etc.)
- How many rooms do I have?
- What type of answer do I get from my queries (numerical/boolean)?
- How many such queries am I entitled to? What is my universe size?
- Are my queries adaptive?

In this article, we assume that queries give a numerical answer (which for a computer means the size of the answer is logarithmic in the size of the finite universe). This model is known as the cell-probe model. In general there are a variety

Please cite this article in press as: D. Howard, Determining membership with 2 simultaneous queries, Theor. Comput. Sci. (2014), http://dx.doi.org/10.1016/j.tcs.2014.05.020