Space Complexity of Two Adaptive Bitprobe Schemes Storing Three Elements

Deepanjan Kesh
Indian Institute of Technology Guwahati, Guwahati, Assam 781039, India
deepkesh@iitg.ac.in

Abstract

We consider the following set membership problem in the bitprobe model – that of storing subsets of size at most three from a universe of size m, and answering membership queries using two adaptive bitprobes. Baig and Kesh [2] proposed a scheme for the problem which takes $\mathcal{O}(m^{2/3})$ space. In this paper, we present a proof which shows that any scheme for the problem requires $\Omega(m^{2/3})$ amount of space. These two results together settles the space complexity issue for this particular problem.

2012 ACM Subject Classification Information systems → Data structures

Keywords and phrases Data structures, set membership problem, bitprobe model, lower bound

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Acknowledgements I want to thank Mirza Galib Anwarul Husain Baig for certain corrections and useful suggestions.

1 Introduction

Given a universe $\mathcal{U}$ containing m elements, consider the problem of storing an arbitrary subset $\mathcal{S}$ of size at most n. Once we have stored some such subset, we are required to answer membership queries of the form "Is x in $\mathcal{S}$?" The solutions to these problems are referred to as *schemes*. The resources that we consider to evaluate schemes for the problem are the space required by the data structure, denoted by s, and the number of bits of the data structure accessed to answer the membership queries, denoted by t. This particular class of static membership problem is referred to in the literature as the *bitprobe model*.

1.1 The Bitprobe Model

Schemes for the bitprobe model are further classified based on the how the decision is made to probe a particular bit of the data structure for some query. If for a given query, the location of a bitprobe is independent of the result obtained from the previous bitprobes, then such a scheme is called a *non-adaptive scheme*. On the other hand, if the location of the current bitprobe depends on the results obtained from the previous bitprobes, then such a scheme is called an *adaptive scheme*.

Given a universe $\mathcal{U}$ and the size of the subset to be stored, say n, the design of any scheme has two components – the storage scheme and the query scheme. Given an arbitrary subset $\mathcal{S}$ of size at most n, the storage scheme sets the bits of the data structure such that the membership queries can be answered correctly. The query scheme handles arbitrary queries of the form "Is x in $\mathcal{S}$?"

Radhakrishnan *et al.* [7] introduced the following notation to represent the various schemes in the model – a scheme that takes s amount of space and requires t bitprobes to answer membership queries correctly is denoted as $(n, m, s, t)_A$ or $(n, m, s, t)_N$ depending on whether the scheme is adaptive or non-adaptive, respectively. Sometimes, the notations $s_A(n, m, t)$ and $s_N(n, m, t)$ are used to denoted the space requirement for the respective schemes.

1.2 The Problem Statement

The space complexity for two adaptive bitprobe schemes storing subsets of size one ($n = 1, t = 2$) is well understood – the lower bound is $\Omega(m^{1/2})$, and there is an explicit scheme that matches this lower bound [1, 5]. For subsets of size two ($n = 2, t = 2$), there is a scheme due to Radhakrishnan *et al.* [6] that takes $\mathcal{O}(m^{2/3})$ amount of space. They further conjectured that this is also the minimum space required for the problem. Though progress has been made towards proving the lower bound [6, 7], the problem still remains open.

In this paper, we consider the problem of storing subsets of size at most three, and answering membership queries using two adaptive bitprobes, i.e. $n = 3$ and $t = 2$. Particularly, we look into the lower bound on space for the class schemes solving the problem.

Garg and Radhakrishnan [4] has proposed a general upper and lower bound for all adaptive schemes using two bitprobes. When applied to the particular case of storing three elements, the upper and lower bounds come out to be $\mathcal{O}(m^{12/13})$ and $\Omega(1)$, respectively. Garg [3] improved the general upper bound in his thesis, which improves the bound for the three element case to $\mathcal{O}(m^{11/12})$. Further improvement of the upper bound was made by Baig and Kesh [2] when they came up with a scheme that takes $\mathcal{O}(m^{2/3})$ space.

A much better lower bound was proposed by Radhakrishnan *et al.* [7] when they proved that for storing subsets of size at most two ($n = 2$), the space required is $\Omega(m^{4/7})$. As a corollary, their result puts a lower bound for the scenario when $n = 3$.

In this paper, we make the following claim – an adaptive scheme storing subsets of size at most three from a universe of size m and answering membership queries using two bitprobes requires $\Omega(m^{2/3})$ amount of space, i.e.

$s_A(3, m, 2) = \Omega(m^{2/3})$ (Theorem 18).

This claim, along with the scheme due to Baig and Kesh [2] resolves the space complexity question for $n = 3$ and $t = 2$.

2 Two Bitprobe Schemes

In this section, we discuss the components of an adaptive two-bitprobe scheme, restate a few notations from the literature, and introduce some new ones used in the proof of our claim.

2.1 The Decision Tree

The data structure for two-bitprobe adaptive schemes consists of three tables, namely $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$. Every element in our universe has a location reserved in each of the three tables, a location which stores a single bit. For an element x, we use the notations $\mathcal{A}(x), \mathcal{B}(x)$, and $\mathcal{C}(x)$ to denote its location in the three tables. We abuse this notation a bit, and use these notations to also denote the bits stored in those locations.

Given a subset $\mathcal{S}$ of the universe $\mathcal{U}$, the storage scheme sets the bits of these tables in such a way that the query scheme answers membership questions correctly. The arrangement and the purpose of these three tables will become more apparent from the query scheme, discussed below.

The design of the query scheme is as follows. Upon the query "Is x in $\mathcal{S}$?", the first bitprobe in made in table $\mathcal{A}$ at the location $\mathcal{A}(x)$. Depending on whether the bit stored in

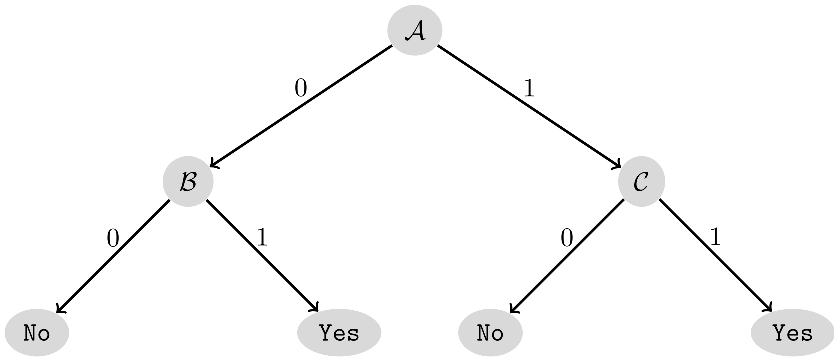

Figure 1 The decision tree of an element.

the location is 0 or 1, the second bitprobe in made in table $\mathcal{B}$ or $\mathcal{C}$, respectively. Finally, if the second bitprobe returned 1, we declare that the element x is a member of $\mathcal{S}$, else if 0 is returned, we declare that x is not a member of $\mathcal{S}$.

The description of the query scheme can be represented in the form of tree, shown in Figure 1, and is known as the *decision tree*.

2.2 Blocks

We borrow the terminology introduced in Radhakrishnan *et al.* [6] and define the notion of *blocks*.

▶ **Definition 1.** The set of all elements of the universe $\mathcal{U}$ that query the same location in table $\mathcal{A}$ is said to form a *block*.

It follows that if elements u and v belong to the same block, then $\mathcal{A}(u) = \mathcal{A}(v)$. Consequently, we have as many blocks as there are bits in table $\mathcal{A}$. Blocks are significant for the following reason – all the elements of a block will either query table $\mathcal{B}$ or $\mathcal{C}$, depending on whether the bit corresponding to the block stores a 0 or a 1, respectively.

Given a block, each of its elements will be numbered uniquely starting from 1. We will call this number corresponding to an element within a block as the *index* of the element.

The notion of blocks together with the notion of indices gives us a unique way of identifying the elements of the universe $\mathcal{U}$ – the block number in table $\mathcal{A}$, and the index within that block. Henceforth, we are going to use the following notation to label any element. If an element belongs to block a, and its index within the block is i, then we are going to address that element as a_i.

As a block is essentially a set, we will use the notation $|a|$ to denote the number of elements block a contains. Table $\mathcal{A}$ being a collection of blocks, we use the notation $|\mathcal{A}|$ to denote its size.

2.3 Sets

In tables $\mathcal{B}$ and $\mathcal{C}$, for the sake of convenience, which will become apparent as the proof progresses, we use the term **sets** instead of blocks for elements querying the same location.

▶ **Definition 2.** Elements that query the same location in table $\mathcal{B}$ are said to belong the same *set*. The same terminology is used for elements that query the same location in table $\mathcal{C}$.

So, there are as many sets in tables $\mathcal{B}$ and $\mathcal{C}$ as there are bits. Similar to table $\mathcal{A}$, we will use the notation $|\mathcal{B}|$ and $|\mathcal{C}|$ to denote the sizes of the respective tables.

We now define two of the key notions employed in the proof of the lower bound, that of the *mass* of a set and the *universe* of a set.

▶ **Definition 3.** The *mass* of a set is the total number of elements in all of those blocks of table $\mathcal{A}$ which has one or more elements in the set. For a set W, its mass is denoted by m_W.

▶ **Definition 4.** Given the set W, we construct a new set corresponding to W using the following steps.

Step 1. Collect all the elements in all of those blocks of table $\mathcal{A}$ which has a member in set W.

Step 2. From the resulting set, remove the elements of set W.

This set will be denoted as U_W, the *universe* of W. The size of this set is

$$|U_W| = m_W - |W|. \tag{1}$$

To take an example, suppose the set $W = \{\ a_1, e_1, f_2, h_3\ \}$. Let the members of the relevant blocks a, e, f, and h be

$$\begin{aligned} a &= \{a_1, a_2, a_3\}; \\ e &= \{e_1, e_2\}; \\ f &= \{f_1, f_2, f_3, f_4\}; and \\ h &= \{h_1, h_2, h_3, h_4, h_5\}, \end{aligned}$$

then,

$$\begin{aligned} m_W &= |a| + |e| + |f| + |h| \\ U_W &= \{a_2, a_3, e_2, f_1, f_3, f_4, h_1, h_2, h_4, h_5\}. \end{aligned}$$

3 Clean and Dirty Sets

In this section, we define and discuss two categories of sets, namely *clean* sets and *dirty* sets.

▶ **Definition 5.** A set is said to be *dirty* if the set contains more than one element from some block of table $\mathcal{A}$. On the other hand, if all of the elements of a set are from distinct blocks of table $\mathcal{A}$, then that set is said to be *clean*.

For example, the set $\{\ a_1, a_2, b_3, c_4\ \}$ is a dirty set as it contains two elements from block a. On the other hand, the set $\{\ e_1, f_1, g_3\ \}$ is a clean set. Note that same indices, as in the later set, are allowed, but same block numbers are not.

We now make an important observation about the relationship between blocks and dirty sets.

▶ **Lemma 6.** *If a set in any of the tables is dirty due to the elements of a block of table $\mathcal{A}$, then all of the elements of that block must belong to distinct sets in the other table.*

In other words, if some block of table $\mathcal{A}$ make some set dirty in table $\mathcal{B}$, then it cannot make any set dirty in table $\mathcal{C}$, and vice versa.

Proof. Without loss of generality, let the elements a_1 and a_2 (the first and the second elements of block a) belong to the same set W in table $\mathcal{B}$. So, the set W is dirty due to block a. We will prove that the elements of this block will necessarily belong to distinct sets in table $\mathcal{C}$.

Let us construct the subset $\mathcal{S}$ so as to contain the element a_1 but not the element a_2. In this case, $\mathcal{A}(a)$ cannot be 0. If $\mathcal{A}(a)$ is indeed 0, then upon query for the element a_1, we would get a 0 from table $\mathcal{A}$, and the second query for a_1 must be in table $\mathcal{B}$. As a_1 belongs to the set W of table $\mathcal{B}$, and as a_1 belongs to the subset $\mathcal{S}$, the bit corresponding to the set W must be set to 1.

Under this assignment, we look into the queries for the element a_2. As $\mathcal{A}(a) = 0$, the second query for a_2 will be in table $\mathcal{B}$. As we have assumed that a_1 and a_2 belong to the set W in $\mathcal{B}$, the second query for a_2 will be to the bit corresponding to the set W. As the bit stored is 1, we will deduce that a_2 belongs to $\mathcal{S}$, which would be incorrect.

So, $\mathcal{A}(a)$ cannot store 0, and hence it must store 1. So, the second query for all the elements of block a will be made in table $\mathcal{C}$.

In table $\mathcal{C}$, if two elements of block a are again together in some set, we can put one of the elements in $\mathcal{S}$ but not the other, and we will reach a contradiction similar to the one above. Note that the subset $\mathcal{S}$ is allowed to contain at most three elements, and to arrive at the contradiction we need to put at most two elements in $\mathcal{S}$.

We can thus conclude that the elements of block a must belong to distinct sets in table $\mathcal{C}$. ◀

The next lemma shows the relationship between multiple blocks that create dirty sets in table $\mathcal{B}$

▶ **Lemma 7.** *Consider all of those blocks of table $\mathcal{A}$ that make one or more sets dirty in table $\mathcal{B}$. All of the elements from all of those blocks must necessarily be in distinct sets of table $\mathcal{C}$.*

Proof. Without loss of generality, let the elements a_1 and a_2 of block a make some set dirty in table $\mathcal{B}$. Putting one of them in subset $\mathcal{S}$ but not the other, and reasoning along the lines of the proof of Lemma 6, we will have $\mathcal{A}(a) = 1$. Similarly, if we have the elements b_1 and b_2 of block b making some set dirty in table $\mathcal{B}$, we can put one of them in $\mathcal{S}$ and not the other, and ensure that $\mathcal{A}(b) = 1$.

Now, suppose that the elements a_i and b_j belong to a set X in table $\mathcal{C}$. In this scenario, we will add a_i to subset $\mathcal{S}$ as its third member. As the second query for a_i will be in table $\mathcal{C}$, we will have to set the bit corresponding to the set X to 1.

With this assignment, we look in the query "Is b_j in $\mathcal{S}$?" As $\mathcal{A}(b)$ is 1, the second query of b_j will be in table $\mathcal{C}$. As it belongs to set X, we will get a 1 for the second query and incorrectly deduce that b_j is a member of $\mathcal{S}$.

This tells us that all the elements of blocks a and b must belong to distinct sets of table $\mathcal{C}$. It is to be noted that it has been implicitly assumed that the elements a_1, a_2, b_1, and b_2 are distinct from a_i and b_j, which need not necessarily be true. In such a case too it can be argued, as above, that the elements of the two blocks cannot share a set in table $\mathcal{C}$. ◀

The aforementioned restrictions on the blocks creating dirty sets help us to estimate the total number elements in all of those blocks of table $\mathcal{A}$ which are responsible for creating dirty sets in table $\mathcal{B}$.

Consider those blocks of table $\mathcal{A}$ that create dirty sets in table $\mathcal{B}$. Let the total number of elements in all of those blocks combined be $N_{\mathcal{B}}$. Lemma 7 tells us that of those elements must belong to distinct sets in table $\mathcal{C}$. This observation immediately puts the following bound on $N_{\mathcal{B}}$ –

$$N_{\mathcal{B}} \leq |\mathcal{C}|.$$

We can do the same exercise for table $\mathcal{C}$, and count the total number of elements in all of the blocks responsible for creating dirty sets in table $\mathcal{C}$. If that number is $N_{\mathcal{C}}$, then we will arrive at the relation

$$N_{\mathcal{C}} \leq |\mathcal{B}|.$$

In our data structure, let us remove all of those $N_{\mathcal{B}}$ elements from their respective sets and put in singleton sets in table $\mathcal{B}$, and we do the same for the $N_{\mathcal{C}}$ elements in table $\mathcal{C}$. This will make all of the sets in the tables $\mathcal{B}$ and $\mathcal{C}$ clean. Of course, this comes with an additional cost to the size our data structure, and the its new size will be

$$|\mathcal{A}| + 2|\mathcal{B}| + 2|\mathcal{C}|.$$

If the sizes of all of the tables in our initial data structure be s each, resulting in the total size to be $3s$ to begin with, after the adjustment mentioned above we will have a data structure whose size is at most $5s$, an increase by a constant factor, and no asymptotic penalty.

We can further introduce s empty blocks in table $\mathcal{A}$ and make the sizes of the three tables uniform. With these observations, we can make the following claim.

▶ **Theorem 8.** *Given a $(3, m, s, 2)_A$-scheme, we can have an equivalent $(3, m, 2 \times s, 2)_A$ adaptive scheme where the data structure has only clean sets.*

Henceforth, we will talk exclusively about schemes with clean sets only, and whose table sizes are all equal, and prove the lower bound for this class of schemes. Theorem 8 guarantess that the lower bound claim will also hold for the general class of schemes, with or without dirty sets.

4 Mass of a set

The following relationship holds between the total mass of all the sets of tables $\mathcal{B}$ and $\mathcal{C}$, and the sizes of the blocks of table $\mathcal{A}$.

▶ **Lemma 9.** *For the sets of tables $\mathcal{B}$ and $\mathcal{C}$, and the blocks of table $\mathcal{A}$, the following equality is true.*

$$\sum_{W \in \mathcal{B}} m_W = \sum_{X \in \mathcal{C}} m_X = \sum_{a \in \mathcal{A}} |a|^2 \tag{2}$$

Proof. Consider a block a of table $\mathcal{A}$. As we are dealing with schemes containing clean sets only, the elements of the block a will be distributed in exactly $|a|$ sets of table $\mathcal{B}$. This implies that block a will contribute to the masses of $|a|$ sets of table $\mathcal{B}$. In other words, the term $|a|$ will occur as summand in the masses of $|a|$ sets of table $\mathcal{B}$.

So, in the total mass of all the sets of table $\mathcal{B}$, $|a|$ will occur as a summand exactly $|a|$ times. In other words, the contribution of block a to the total mass of table $\mathcal{B}$ is $|a|^2$, and the equality follows.

We can similarly argue about table $\mathcal{C}$. ◀

▶ **Lemma 10.** *The following inequality holds between the masses of the sets of tables $\mathcal{B}$ and $\mathcal{C}$, and the size of table $\mathcal{A}$ –*

$$\sum_{W \in \mathcal{B}} m_W = \sum_{X \in \mathcal{C}} m_X \geq \frac{m^2}{|\mathcal{A}|}. \tag{3}$$

Proof. Consider the sum from Equation 2

$$\sum_{a \in \mathcal{A}} |a|^2.$$

Using the arithmetic mean geometric mean inequality, we can show that the sum is minimized when all the summands are equal, i.e.

$$\sum_{a \in \mathcal{A}} |a|^2 \geq |\mathcal{A}| \times \left(\frac{\sum_{a \in \mathcal{A}} |a|}{|\mathcal{A}|} \right)^2.$$

By using the fact that $\sum_{a \in \mathcal{A}} |a| = m$, we get the desired R.H.S. ◀

It is interesting to note that the total mass of either of the tables $\mathcal{B}$ and $\mathcal{C}$ is minimized when all of the blocks of table $\mathcal{A}$ are of equal size.

5 Bad Elements

In this section, we give a characterisation of certain elements of our universe $\mathcal{U}$ as being *bad* for some particular sets of table $\mathcal{C}$.

▶ **Definition 11.** Suppose an element a_i from block a of table $\mathcal{A}$ belongs to a set W of table $\mathcal{B}$, and to a set X in table $\mathcal{C}$. Such an element is said to be a *bad element* for the set X if the following holds:

1. a_i shares the set W with two other elements b_j and c_k, from blocks b and c, respectively.
2. There exists elements b_l and c_n, different from the elements b_j and c_k, such that they share a set in table $\mathcal{C}$.

Figure 2 describes pictorially the notion of a bad element.

As in the above instance, let us suppose that the element a_i is a bad element for the set X of table $\mathcal{C}$. We discuss below why, given such an arrangement of elements, is a_i referred to as bad for the set X.

5.1 Property of a bad element

Consider the following subset $\mathcal{S} = \{ a_i, b_l \}$. We show that if we want to store this subset, then $\mathcal{A}(a)$ must be set to 1, and we show this by contradiction.

If it is indeed the case that $\mathcal{A}(a) = 0$, then upon query for the element a_i, we will go to table $\mathcal{B}$ for the second query. As a_i belongs to the set W in table $\mathcal{B}$, and as it is also a member of subset $\mathcal{S}$, the bit corresponding to W must be set to 1.

This would imply that $\mathcal{A}(b) = 1$. If it is not, and $\mathcal{A}(b)$ is set to 0, then the second query for the element b_j would in table $\mathcal{B}$. As b_j is a member of the set W in table $\mathcal{B}$, we would get a 1 against the second query and incorrectly assume b_j is a member of $\mathcal{S}$. So, we see that $\mathcal{A}(b)$ must be 1. We can similarly argue that $\mathcal{A}(c)$ must also be 1.

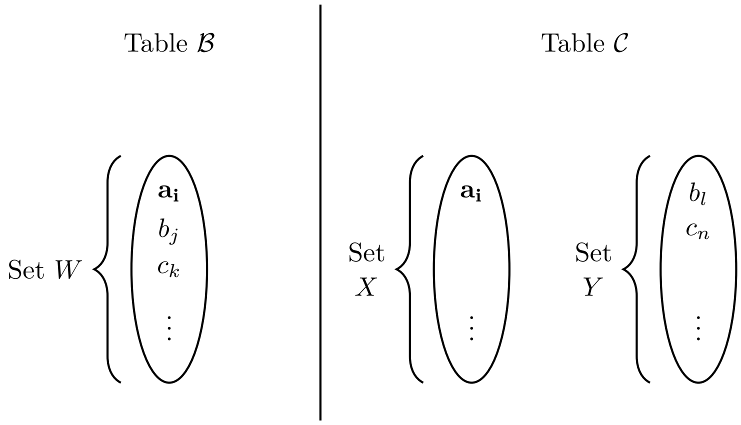

Figure 2 In this arrangement, a_i is a bad element for the set X. Note that $j \neq l$ and $k \neq n$.

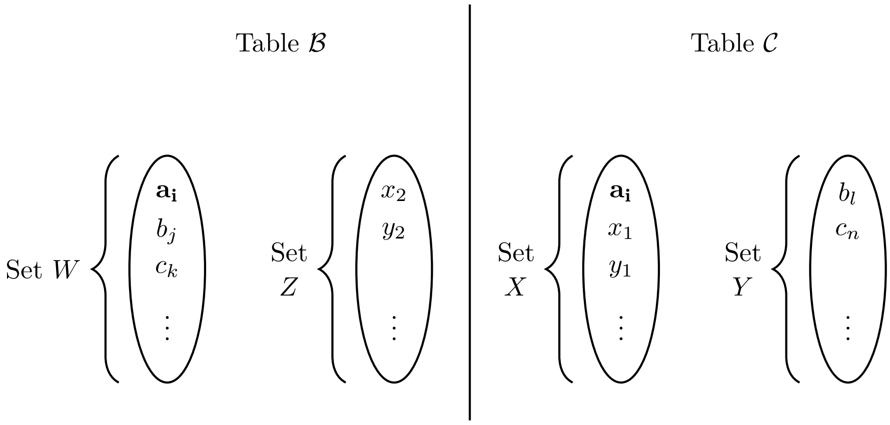

Figure 3 In this arrangement, a_i is a bad element. Note that $j \neq l$ and $k \neq n$.

As shown in Figure 2, the elements b_l and c_n belong to the set Y in table $\mathcal{C}$. From the arrangement of elements above, we can further deduce that as b_l is a member of $\mathcal{S}$, and it is also a member of the set Y in table $\mathcal{C}$, the bit corresponding to the set Y must be set to 1. If we now consider the query "Is c_n in $\mathcal{S}$?", we would find out that we would incorrectly get that c_n is a member of $\mathcal{S}$.

This shows that if the subset $\mathcal{S}$ contains the elements a_i and b_l, then $\mathcal{A}(a)$ cannot be 0, and hence it must be set to 1. We summarise our findings in the following lemma.

▶ **Lemma 12.** *If the subset $\mathcal{S}$ contains the elements a_i and b_l, then $\mathcal{A}(a)$ must be set to 1.*

5.2 Universe of X

We would show that no two elements of $U_X \setminus a$, the universe of X minus the elements of block a, can share a set in table $\mathcal{B}$, and we arrive at this by contradiction.

Let us suppose, without loss of generality, that the elements x_1 and y_1 belong to set X, implying that the elements of the blocks x and y will be part of U_X. Let us further assume that the elements x_2 and y_2, which are members of U_X, share the set Z in table $\mathcal{B}$. The arrangement of the elements can be seen in Figure 3.

In this scenario we will show next that while trying to store the subset $\mathcal{S} = \{ a_i, b_l, x_2 \}$, the query for the element y_2 will give an incorrect answer.

As the subset $\mathcal{S}$ contains the elements a_i and b_l, Lemma 12 tells us that $\mathcal{A}(a)$ must be equal to 1. This means that the second query for a_i will be in table $\mathcal{C}$. a_i belongs to the set X in this table, and hence the bit corresponding to the X must be set to 1.

The element x_1 is not a member of $\mathcal{S}$, so the second query for this element cannot be in table $\mathcal{C}$. If it is, then it will query the bit corresponding to the set X and get a 1, implying x_1 *is* a member of $\mathcal{S}$. To ensure that the second query is in table $\mathcal{B}$, we ought to have $\mathcal{A}(x) = 0$. We can argue similarly about the block y, and conclude that $\mathcal{A}(y) = 0$.

With $\mathcal{A}(x) = 0$, the second query for the element x_2 must be in table $\mathcal{B}$. Considering the fact that x_2 is a member of the subset $\mathcal{S}$, the bit corresponding to the set Z in table $\mathcal{B}$ must be set to 1(Figure 3).

Let us now consider the query “Is y_2 in $\mathcal{S}$?” As $\mathcal{A}(y) = 0$, the second query for y_2 will be in table $\mathcal{B}$. In this table, y_2 belongs to the set Z, and hence, the second query for this element will return a 1, implying incorrectly that y_2 is a member of $\mathcal{S}$.

We summarise our findings in the following lemma.

▶ **Lemma 13.** *Suppose that an element a_i is bad for a set X in table $\mathcal{C}$. Then, the elements of $U_X \setminus a$ must belong to distinct sets in table $\mathcal{B}$.*

5.3 Bounded Sets of Table $\mathcal{C}$

Lemma 13 tells us that the elements of $U_X \setminus a$ must belong to distinct sets in table $\mathcal{B}$. Hence, the size of U_X is bounded by the size of $\mathcal{B}$.

$$
\begin{aligned}
&|U_X| - |a| \leq |\mathcal{B}| \\
&\implies m_X \leq |\mathcal{B}| + |a| + |X| \text{ (from Equation 1),}
\end{aligned}
$$

giving us the following corollary to the lemma above.

▶ **Corollary 14.** *If a set X of table $\mathcal{C}$ contains a bad element from a block a of table $\mathcal{A}$, then the mass of X must satisfy the following inequality.*

$$m_X \leq |\mathcal{B}| + |a| + |X|. \tag{4}$$

We now come to reason why elements with the property, as stated in Definition 11, are said to be bad for sets of table $\mathcal{C}$. A bad element in a set of table $\mathcal{C}$ puts an upper bound on the mass of that set. For small data structures, the sizes of the sets of the tables $\mathcal{B}$ and $\mathcal{C}$ must be large, so that the number of distinct sets is small. A bad element in a set, on the other hand, restricts the size of the set.

For easy reference, we characterise these sets as *bounded sets* because their mass has an upper bound.

▶ **Definition 15.** A set in table $\mathcal{C}$ which contains one or more bad elements is called a *bounded set.*

5.4 Large Sets of Table $\mathcal{B}$.

▶ **Lemma 16.** *Consider such a set W of table $\mathcal{B}$ whose mass satisfies the following inequality.*

$$m_W \geq 2 \times |\mathcal{C}| + |W| + 1. \tag{5}$$

Then, all of the elements of set W are bad elements.

Proof. If some set W of table $\mathcal{B}$ satisfies the above inequality, then the size of the universe of W, which is $m_W - |W|$, has more than twice the number of elements than there are sets in table $\mathcal{C}$. Hence, there is at least one set in table $\mathcal{C}$ which contains three elements or more of U_W.

Without loss of generality, let us assume that all of the elements of W have index 1, i.e. $W = \{ a_1, b_1, c_1, d_1, \dots \}$. Let us assume further that the set X of table $\mathcal{C}$ is the set containing at least three elements of U_W; let those elements be a_2, b_2, and c_2.

If we consider the element c_1, it satisfies the definition of a bad element - a_1 and b_1 along with c_1 belong to the set W in table $\mathcal{B}$, and the elements a_2 and b_2 belong to the set X in table $\mathcal{C}$. We can say the same for every element of W except for a_1 and b_1. So, we get $|W| - 2$ bad elements in the set W.

a_1 is also a bad element due to b_1, c_1, b_2, c_2, and similarly b_1. So, all of the elements of W are bad for one or the other set of table $\mathcal{C}$. ◀

We next highlight an important relation between the masses of large sets of table $\mathcal{B}$ and the bounded sets of table $\mathcal{C}$.

▶ **Lemma 17.** *The total mass of the large sets of table $\mathcal{B}$ is less than or equal to the total mass of the bounded sets of table $\mathcal{C}$.*

Proof. Let a_i be an element that belongs to a large set W in table $\mathcal{B}$. Then, two things hold true – a_i is a bad element, and a_i contributes the amount $|a_i|$ to the mass of the large set W.

In table $\mathcal{C}$, if a_i belongs to set X, then two things hold true here as well – X is a bounded set, and a_i contributes the amount $|a_i|$ to the mass of X.

So, every contribution to the total mass of a large set will also have an equal amount of contribution to the total mass of bounded sets.

Additionally, there could be bad elements due to sets that are not large, or there would be elements in bounded sets that are not bad. Both of these will contribute to the mass of bounded sets, but not to that of large sets. Consequently, the inequality follows. ◀

6 The Lower Bound

Baig and Kesh [2] have shown that there exists an adaptive scheme that stores subsets of size at most three elements from a universe of size m, and answers memberships queries using two bitprobes. The space required by the scheme is $3 \times m^{2/3}$. In other words, we have a $(3, m, 3 \times m^{2/3}, 2)_A$-scheme.

In this section, we will show that a $(3, 2 \times m, 3 \times m^{2/3}, 2)_A$ scheme cannot exist. Section 3 tells us that we will only have to look into schemes with data structures having the following properties – all of the sets are clean, and the sizes of the three tables are equal. So we have the following setting.

$$\begin{aligned} |U| &= 2m \\ n &= 3 \\ t &= 2 \\ |\mathcal{A}| = |\mathcal{B}| = |\mathcal{C}| &= m^{2/3} \end{aligned}$$

If the total number of elements is $2m$, then Lemma 10 tells us that the total mass of all the sets of table $\mathcal{B}$ or table $\mathcal{C}$ is at least

$$\frac{(2m)^2}{|\mathcal{A}|} = \frac{(2m)^2}{m^{2/3}} = 4m^{4/3}.$$

The mass of a set which is not large is at most

$2 \times |C| + |W| = 2m^{2/3} + |W|$ (from Equation 5).

In the worst case, the total number of such sets could at most $m^{2/3}$ – the size of table $\mathcal{B}$ – and the total number of elements belonging to such sets could be m. So, the total mass of all such non-large sets of table $\mathcal{B}$ is

$$m^{2/3} \times 2 \times m^{2/3} + m = 2 \times m^{4/3} + m.$$

This means that the total mass of the large sets of table $\mathcal{B}$ is at least

$$4m^{4/3} - 2 \times m^{4/3} - m = 2m^{4/3} - m \tag{6}$$

If table $\mathcal{C}$, the mass of a bounded set X is at most

$|\mathcal{B}| + |a| + |X| = m^{2/3} + |a| + |X|$ (Corollary 14).

Here, a is the block to which the bad element belongs. In the case that all of the sets of table $\mathcal{C}$ are bounded, then the total mass of all bounded sets is at most

$$m^{2/3} \times m^{2/3} + m + m = m^{4/3} + 2m. \tag{7}$$

Equations 6 and 7 tell us that as long as $m \geq 27$, the total mass of large sets of table $\mathcal{B}$ is strictly greater than the total mass of bounded sets of table $\mathcal{C}$, which is absurd as it contradicts Lemma 17.

This tells us that a $(3, 2\times, 3 \times m^{2/3}, 2)_A$-scheme cannot exist. We now arrive at the final result.

▶ **Theorem 18.** $s_A(3, m, 2) = \Omega(m^{2/3})$.

7 Conclusion

In this paper, we have provided a lower bound for two-bitprobe adaptive schemes storing subsets of size at most three (Theorem 18), which matches with the upper bound for the problem proposed by Baig and Kesh [2]. This, as alluded to earlier, settles the space complexity problem for this particular n and t.

The lower bound for the problem where $n = 2$ and $t = 2$, conjectured by Radhakrishnan *et al.* [6] to be $\Omega(m^{2/3})$, still remains open. We hope that the notions of the mass of a set (Definition 3) and the universe of a set (Definition 4) would help us better understand the data structure for this problem, and consequently resolve the conjecture.

References

1 Noga Alon and Uriel Feige. On the power of two, three and four probes. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 346–354, 2009.

2 Mirza Galib Anwarul Husain Baig and Deepanjan Kesh. Two new schemes in the bitprobe model. In *WALCOM: Algorithms and Computation - 12th International Conference, WALCOM 2018, Dhaka, Bangladesh, March 3-5, 2018, Proceedings*, pages 68–79, 2018.

3 Mohit Garg. *The Bit-probe Complexity of Set Membership*. PhD thesis, School of Technology and Computer Science, Tata Institute of Fundamental Research, Homi Bhabha Road, Navy Nagar, Colaba, Mumbai 400005, India, 2016.

4 Mohit Garg and Jaikumar Radhakrishnan. Set membership with a few bit probes. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 776–784, 2015.

5 Moshe Lewenstein, J. Ian Munro, Patrick K. Nicholson, and Venkatesh Raman. Improved explicit data structures in the bitprobe model. In *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 630–641, 2014.

6 Jaikumar Radhakrishnan, Venkatesh Raman, and S. Srinivasa Rao. Explicit deterministic constructions for membership in the bitprobe model. In *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, pages 290–299, 2001.

7 Jaikumar Radhakrishnan, Smit Shah, and Saswata Shannigrahi. Data structures for storing small sets in the bitprobe model. In *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part II*, pages 159–170, 2010.