Quanutum Bitprobe

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Shubham Vivek Pawar

IISER PUNE

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

April, 2019

Supervisor: Jaikumar Radhakrisnan

Certificate

This is to certify that this dissertation entitled Quanutum Bitprobe towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Shubham Vivek Pawar at Indian Institute of Science Education and Research under the supervision of Jaikumar Radhakrisnan, Senior Professor, Department of Mathematics , during the academic year 2018-2019.

Jaikumar

Jaikumar Radhakrisnan

Committee:

Jaikumar Radhakrisnan

Soumen Maity

This thesis is dedicated to my parents.

Declaration

I hereby declare that the matter embodied in the report entitled Quanutum Bitprobe are the results of the work carried out by me at the Department of Mathematics, Indian Institute of Science Education and Research, Pune, under the supervision of Jaikumar Radhakrisnan and the same has not been submitted elsewhere for any other degree.

[illegible]

Shubham Vivek Pawar

Acknowledgments

First of all I would like to thank Prof. Jaikumar Radhakrishnan for patiently guiding me throughout this project, which is a really tough task. I would like to thank him for letting me explore my ideas and getting involved whenever needed. It was good to work with Shyam Dhamapurkar for a part of the project.
I am grateful to the people at TIFR who made my stay here very comfortable. I met many interesting people here and learned a lot from them.
A special thanks to Abhishek Ojha, Kr. Amit Singh Bhati and Varun Narayanan, who helped me with the diagrms, also to Aanjaneya Kumar and Aayush Vijayvargia who helped me with the last moment finishing touches
I owe a lot my parents for their constant support throughout.
Last but not the least, I would like to thank my family in Pune, my friends at IISER Pune. More specifically "Badaks of common room", these are my friends who made my stay at IISER amazing.

Abstract

We use the quantum bit probe model to study the static membership problem (data structure problem). The data structure is stored classically by the probing scheme is quantum. Like most problem the goal is to reduce the resources used. The resources used here are space and number of probes (to the data structure). We study the space complexity by keep the number of probes to two. Our results -

1. We give a exact non-adaptive scheme that has sample complexity $\mathcal{O}(m^{1/2})$ which show the lower bound got by the polynomial method in ([1]) is tight.

2. We show a way to construct quantum scheme that handle subsets of size $2k+1$, is the same as drawing bipartite graphs of girth $> 2k$.

3. We show that sample complexity for the quantum scheme is less than the classical scheme which was shown to be m in ([2]). That is we give an upper bound for quantum schemes.

4. We borrow ideas we used to construct the quantum schemes to construct a adaptive classical scheme to handle subsets of size at most 4, whose sample complexity is $\mathcal{O}(m^{3/4})$. This results is better than the currently best know result of $\mathcal{O}(m^{5/6})$, a result in ([3]).This shows that "quantum way" of "looking" at a problem at time gives better insights to classical problems.

Contents

Introduction

Like all thesis, this thesis will also be graded. The committee doing so are at IISER Pune. So, we will start with the following story - IISER Pune is a very prestigious institute in India. A large number of students apply to get in every year. To handle such large scale process, everything is done online. A student can check the status of his application online. To do so one needs access to IISER's database, but it would be best if the access is minimal. However there is a trade off in doing so. Lesser the access to the data base a student has more is the size of the data base stored. If each student is given access to one bit to find out whether he or she has gotten in then the size of the data base has to be equal to the number of applicants. Results in ([2]) show that even if two bit (non-adaptive) access is given we cannot do better. So with the recent hype about quantum computing one would feel that why not make the data base quantum. But due to limited funds one could either store the data set in a 'quantum way' or give access in a 'quantum way'. By a toss of coin it was deiced that the access would be quantum, i.e. one could query the data base in suppositions of bits. This problem is studied in ([1]). They used the polynomial method to give lower bounds on the size of the data base. This gives us hope that quantum schemes could do better. So far no explicit construction is given hence we plan to give a few explicit construction of quantum schemes that output the query perfectly (exactly).

Chapter 1

Preliminaries

1.1 Classical Bit-probe model

I will begin by introducing the Static membership problem. We are given a universe $\mathcal{U}$ of size m ($\mathcal{U} = [m]$). Our goal is to store it's subset S in such a way that the query "Is x in S" could be answered with the least resources possible. The resource that we would consider here are the space required to store the data structure and the number of probes to the data structure.

We will be using the bitprobe model to study this problem. In this model we store each subset S as a as a table of cells each capable of storing one bit. A single probe at a given location would give us the value of the bit stored there. A scheme is a method to store the subset. We will define it below as given in ([4])

Definition 1.1.1. *An (n, m, s)-scheme is a map ϕ from the subsets of size at most n of $\{1, ..., m\}$ to $\{0, 1\}^s$. A deterministic (m, s, t)-query scheme is a family of m Boolean decision trees $\{T_i\}$ of depth at most t. Each node is marked with a label from {1,...,m} indicating the address of the bit in $\{0, 1\}^s$. The leaf is labelled either 0 or 1. Hence each tree T_i is a map from $\{0, 1\}^s$ to $\{0, 1\}$.*

An (m, n, s)-storing scheme ϕ and an (m, s, t)-query scheme T_i together form an (m, n, s, t)-scheme if it correctly solves the static set membership problem i.e. $\forall x \in S, T_x(\phi(S) = 1$ iff $x \in S$. Let $s(m, n, t)$ be the smallest s such that there exists an (m, n, s, t)-scheme.

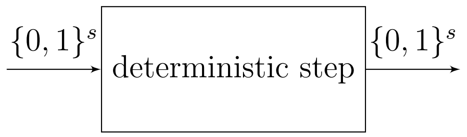
A non-adaptive query scheme is a deterministic scheme where all the nodes on the same level are marked with the same index.

For example, if your had only one query the best you could do is a characteristic vector of the universe, i.e. 1 in the ith if the ith belongs to the subset.

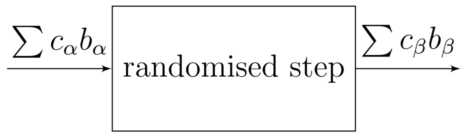
1.2 Quantum computation

Here I would like to give a basic ideas of quantum computing. Below we look at a reversible computing steps.

In every step of classical deterministic computation, the input is a binary string and the output is also a binary string.



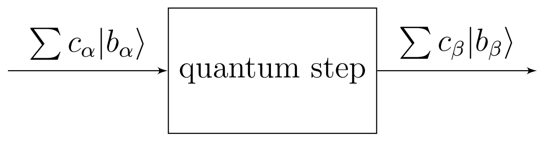
In randomised computation the following occurs, the input is a linear combinations of a string of bit and so is the output. The coefficients are positive real number and to maintain the law of total probability the sum up to one. This is because the probability of b_β occurring is c_β.



Where $b_\alpha, b_\beta \in \{0,1\}^s$ and $c_\alpha, c_\beta \in \mathbb{R}_{\geq 0}$ such that $\sum c_\alpha = 1$ and $\sum c_\beta = 1$

Quantum computation is a generalization of randomised computing. Here the coefficients belong to complex numbers number rather than positive real numbers and to ensure the total law of probability holds the summation modulus square of these coefficients is one. This is because the probability of $|b_\beta\rangle$ occurring is $|c_\beta|^2$

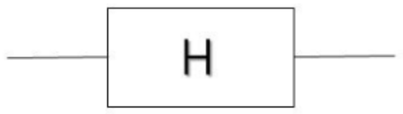
Where $b_\alpha, b_\beta \in \{0,1\}^s$ and $c_\alpha, c_\beta \in$ such that $\sum |c_\alpha|^2 = 1$ and $\sum |c_\beta|^2 = 1$


$\sum c_\alpha |b_\alpha\rangle$
quantum step
$\sum c_\beta |b_\beta\rangle$

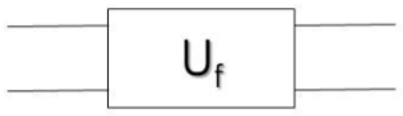
These steps are called gates. In the quantum case notice that, $|b_\beta\rangle$ belong to a vector space whose orthonormal basis are $|0..00\rangle, |0..01\rangle, ..., |1..11\rangle$ and the quantum gates are unitary transforms (preserve probability). These gates act linearly, hence to describe a gate all we need to know is how it acts on the basis vectors

Here I'll will describe two gates which will be required for the deutch algorithm.

1.Hadamard gate H - It maps $|0\rangle$ to $\frac{1}{2}(|0\rangle + |1\rangle)$ and $|1\rangle$ to $\frac{1}{2}(|0\rangle - |1\rangle)$.



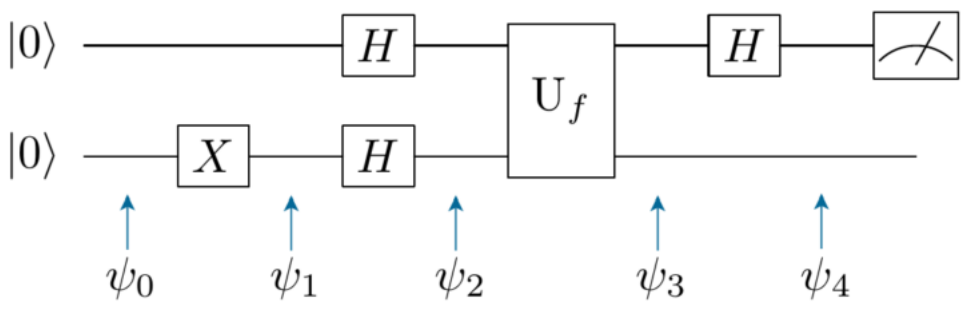
2.Oracle U_f - It maps $|x, b\rangle$ to $|x\rangle, b \oplus f(x)\rangle$



1.3 Deutch Algorithm

We will now present the famous Deutch Algorithm ([5])

INPUT: $f : \{0, 1\} \to \{0, 1\}$
OUTPUT: Indicate whether the function is constant or not (i.e. Is $f(0) = f(1)$?)
The resource we would like to minimise is the number of time the function is evaluated.
Classical algorithms: To solve this problem one would have to evaluate the function twice to answer the question with probability greater than 1/2.
Quantum algorithm: The quantum algorithm is described using the quantum circuit given below ([6])



$|\psi_0\rangle = |0\rangle|0\rangle$

$|\psi_1\rangle = |0\rangle|1\rangle$

$|\psi_2\rangle = H|\psi_1\rangle = \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$

$|\psi_3\rangle = \mathcal{U}_f|\phi_2\rangle$

$= \frac{1}{2}(|0\rangle(|f(0) \oplus 0\rangle - |f(0) \oplus 1\rangle) + |1\rangle(|f(1) \oplus 0\rangle - |f(1) \oplus 1\rangle))$

$= \frac{1}{2}((-1)^{f(0)}|0\rangle(|0\rangle - |1\rangle) + (-1)^{f(1)}|1\rangle(|0\rangle - |1\rangle))$

$= (-1)^{f(0)}\frac{1}{2}\left(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\right)(|0\rangle - |1\rangle)$

$= \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle)$

$|\psi_4\rangle = H \otimes I|\psi_3\rangle$

$= \frac{1}{2}(|0\rangle + |1\rangle + (-1)^{f(0)\oplus f(1)}|0\rangle - (-1)^{f(0)\oplus f(1)}|1\rangle)$

$= \frac{1}{2}((1 + (-1)^{f(0)\oplus f(1)})|0\rangle + (1 - (-1)^{f(0)\oplus f(1)})|1\rangle)$

If you look closely at $|\psi_3\rangle$ you can see that $f(0) \oplus f(1) = 0$ if and only if we measure $|0\rangle$ and $f(0) \oplus f(1) = 1$ if and only if we measure $|1\rangle$. This shows that we can indicate whether a function is a constant or not with only one query to the oracle (i.e evaluating the function once), and this can be done without error.

What we can take from the above is that if we are allowed exact (without errors) quantum queries, we could do so with just one query. This helps the quantum bitprobe perform better than it's classical counterpart.

1.4 Quantum Bitprobe model

In this model the data structure is classical but the probe is quantum. In a sense, we are allowed to query the data structure as a superposition of the memory locations. The data structure acts like a oracle, let it be denoted by $\mathcal{O}_D$. Written below is the action of $\mathcal{O}_D$ on the basis state $|i, \beta, w\rangle$ where $i \in [m]$ is the location of the address of the bit, $\beta \in \{0, 1\}$ and w is the work bit.

$$\mathcal{O}_S : |i, \beta, w\rangle \rightarrow (-1)^{\beta \bigoplus b_i} |i, \beta, w\rangle$$

where b_i is the value of the bit at the address i.

The ability to query in superposition gives us many advantages. One of the advantages it gives for the exact quantum schemes is that we could in one probe get the equality of two bits located at two addresses.

A exact quantum (m, s, t)-query scheme **consists** of a family of m Boolean decision trees $\{T_i\}$ of depth at most t. Each node is marked with a label from either {1,...,m} or $\{i \bigoplus j\}$, where $i, j \in [m]$, the second indicating the XOR (equality) of the bits corresponding to addresses i and j in $\{0, 1\}^s$. The leaf is labelled either 0 or 1.

The only thing one has to know about quantum to get through the thesis, is that we can in one probe for an equality of bits at two locations

Chapter 2

Results so far

2.1 Characteristic vector is optimal

In this section we will state a result found in ([2]). The result shows that the space complexity of non-adaptive classical bitprobe model given access to two (classical) queries is m.

Theorem 2.1.1. *For a non-adaptive classical* $(m, n, s, t) -$ *scheme we have*

$$s(m, n = 2, t = 2) = m$$

This characteristic vector is the best classical scheme for two non-adaptive probes.

2.2 Quantum scheme lower bounds

The polynomial method was used to get lower bounds for the space complexity in quantum schemes by ([1]) gives a lower bound for space complexity of quantum exact model i.e. a quantum scheme that answers the query correctly always.

Theorem 2.2.1. *If there exists an (n,m,s,t)-exact quantum scheme for the static membership* *problem then we have the following inequality*

$$\sum_{i=0}^{n} \binom{m}{i} \leq \sum_{i=0}^{nt} \binom{s}{i}$$

using the above we get lower bounds for s in terms of n and m.

Chapter 3

Our Contribution

3.1 Results for exact non-adaptive quantum scheme

3.1.1 $s_Q(m, n = 2, t = 2) = O(m^{1/2})$

We have a non-adaptive exact quantum scheme for $n = 2$ and $t = 2$ that uses $s = 4m^{1/2}$ bits for storage. This is an interesting result as it matches the lower bound for quantum schemes.

Arrangement of elements

Each x is assigned a unique pair (a_x, b_x) and (c_x, d_x) such that $a_x = c_x$ and $b_x = d_x$. Notice it is possible to assign such a unique pair since the size of the sub-strings are $\lceil m^{1/2} \rceil$.

Data structure

Store one bit at each location $(a, 0), (0, b), (c, 0), (0, d)$, where $a, b, c, d \in \{1, 2, ..., \lceil m^{1/2} \rceil\}$. Hnce we have the following lemma.

Lemma 3.1.1. $s_Q(m, n = 2, t = 2) = O(m^{1/2})$

Notation - The bit stored at adress i is denoted by $m(i)$

Probing scheme

When queried Is x in S? we answer with an affirmative if an only if $m(a_x) \neq m(b_x)$ and $m(c_x) \neq m(d_x)$.

Storage scheme

There are many cases to look at, given below are all the possible cases similar up to permutation.

subset of size 0

Assign bit value 0 to all locations i.e. all zero vector

subset of size 1

Let the element α belong to the subset. Assign bit value 1 at locations a_α and d_α and 0 everywhere else. What happens when you query is x in S?

If $x = \alpha$ we have $m(a_x) \neq m(b_x)$ and $m(c_x) \neq m(d_x)$

If $x \neq \alpha$ then none of the memory locations of x and α are equal then we are done because all these locations assigned with a bit value of zero.

In this case the memory location of α and x would be different let's say they differ at location a, i.e. $a_\alpha \neq a_x$, hence $m(a_x) = m(b_\alpha) = 0$. This implies the element x does not belong to the subset. Similar analysis can be done for the other cases.

subset of size 2

Let the elements α and β belong to the subset.

Case 1 - none of the memory locations of α and β are the same

Set $m(a_\alpha), m(b_\beta), m(c_\alpha), m(d_\beta)$ to 1 and every other location to 0.

If $x \in \{\alpha, \beta\}$ then we have $m(a_x) \neq m(b_x)$ and $m(c_x) \neq m(d_x)$.

If $x \notin \{\alpha, \beta\}$ then x differs from α and β in at-least one position each. Lets say $a_x \neq a_\alpha$, hence $c_x \neq c_\alpha$. In this case we have $c_x = d_x = 0$. This implies that the element x doesn't belong to the subset. Similarly the case when $b_x \neq b_\alpha$ can be analysed.

Case 2 - one of the memory locations of α and β is the same

Let us assume that $a_\alpha = a_\beta$. In this case set $m(a_\alpha) = m(a_\beta)$, $m(d_\alpha)$ and $m(d_\beta)$ are set to 1 and the rest are set to 0.

If $x \in \{\alpha, \beta\}$ then we have $m(a_x) \neq m(b_x)$ and $m(c_x) \neq m(d_x)$.

If $x \notin \{\alpha, \beta\}$ then either $a_x = a - \alpha = a_\beta$ or $a_x \neq a - \alpha = a_\beta$. First let us look at the case when $a_x = a_\alpha = a_\beta$ in that case we have $c_x = c_\alpha = c_\beta = 0$ and since $m(d_x) = 0$ we have $m(c_x) = m(d_x)$, this implies $x \notin \{\alpha, \beta\}$. Now the case when $a_x \neq a - \alpha = a_\beta$, here we have $m(a_x) = m(b_x)$, hence $x \notin \{\alpha, \beta\}$.

3.1.2 $s_Q(m, n = 3, t = 2) = O(m^{2/3})$

We have a non-adaptive exact quantum scheme for n = 3 and t = 2.

Arrangement of elements

Each element x from the universe is assigned a unique point on the three dimensional cube of dimensions $m^{1/3} \times m^{1/3} \times m^{1/3}$ placed in the first quadrant of an $XYZ - 3D$ space with one vertex at the origin, more precisely each element x is associated a point (a_x, b_x, c_x) where $a_x, b_x, c_x \in \{1, 2, ...\lceil m^{1/3} \rceil\}$

Data structure

A way set up the data structure is assign on bit from the string to each point in the XY, XZ, YZ - plane, more precisely each point $(a, b, 0)$, $(a, 0, c)$ and $(0, b, c)$, where $a, b, c \in \{1, 2, ..., \lceil m^{1/3} \rceil\}$, is associated a unique bit from the string $\{0, 1\}^s$. Hence we have the following easy to verify lemma.

Lemma 3.1.2. $s_Q(m, n = 3, t = 2) = O(m^{2/3})$

Probing Scheme

When queried Is x in S? one has to follow the probing scheme given below to exactly answer the query. First (classically) probe for the value of the bit stored at location $(a, b, 0)$, then we probe for the equality of the bits stored at $(a, 0, c)$ and $(0, b, c)$, let the output '0' imply that they are equal and '1' if not. Let b_1 and b_2 be the outputs of the two queries. The answer to the query would be the AND of the two output bits, i.e. $b_1 b_2$, and if the result is '1' then we say that the element "x is in S." else if '0' we say "x is not in S.".

A small detour

We will define a few things before we a few things before giving a storage scheme. This terminology will help generalise the above scheme to subsets of size greater then 3.

In all the schemes below we arrange the elements in a cuboid of dimensions $m^\gamma \times m^\beta \times m^\alpha$. To minimise the space we set $\gamma = \beta$ (AM-GM inequality). When this cuboid is view from the positive Z-axis, i.e. project all the element on to the XY-plane, then we get a gird, such that each square box in the grid consists of one and only one integral point. The elements in a square of the grid form a block.

Definition 3.1.1. *An (a, b) block is a set whose elements are the coordinates (a, b, i) where $i \in \{1, 2, ..., m^\alpha\}$.*

All blocks need not contain elements, in fact, for our scheme to work we need a few blocks to be empty. To make simplify things, each block either is packed with elements or does not contain any elements at all (i.e. number of elements in a block is either m^α or 0).

Definition 3.1.2. *An (a, b) element-block is the set of element from the universe, whose first query address is $(a, b, 0)$. An (a, b) element-block vector is the characteristic vector of the (a, b) element-block, whose i-th element is 1 if the element located at (a, b, i) is in the subset S else it is 0. Let the (a, b) element-block vector be denoted by $\vec{e}_{(a,b)}$.*

We shall define bit-block vector in the same way as we defined the element-block vector.

Definition 3.1.3. *An $(a, 0)$ bit-block vector is a vector whose i-th element is the bit value of the string located at $(a, 0, i)$. Let the (a, b) bit-block vector be denoted by $\vec{b}_{(a,0)}$.*

Similarly we define $(0, b)$ bit-block vector. To avoid ambiguity we will define them below.

$$(\vec{b}_{(a,0)})_i = m((a, 0, i))$$
$$(\vec{b}_{(0,b)})_i = m((0, b, i))$$

Remember we defined $m(i) =$ bit value stored at location i.

Notice that for every non-zero vector $\vec{e}_{(a,b)}$, we have the following relation $\vec{e}_{(a,b)} = \vec{b}_{(a,0)} \oplus \vec{b}_{(0,b)}$. We can says each element-block vector $\vec{e}_{(a,b)}$ is associated to a set of bit-block vectors $\{\vec{b}_{(a,0)}, \vec{b}_{(0,b)}\}$.

Associated graph

The above exact quantum scheme in section 3.1.2 can be represented as a bipartite graph. We will call this graph the associated graph of the scheme. The construction of the graph is given below.

Let us put in graph theoretic terminology. We can represent the quantum scheme as a bipartite graph (U, V, E), where vertex sets U and V are the set of $(a, 0)$ bit-block vector and $(0, b)$ bit-block vector respectively and edge set E is the set containing (a, b) element-block vectors, more precisely $U = \{\vec{b}_{(a,0)}\}$, $V = \{\vec{b}_{(0,b)}\}$ and $E = U \times V = \vec{e}_{(a,b)} = \{\vec{b}_{(a,0)}, \vec{b}_{(0,b)}\}$.

Now we would like to incorporate the first query into the graph, to do so we will colour the edges. An element of the edge, say $\vec{e}_{(a,b)}$, is coloured black if the first probe output is 1, i.e. $m((a, b)) = 1$ (this happens if an element of the subset lie in this block), else colour the edge grey.

○	∗	○		⋯	⋯
∗	∗			⋯	⋯
○			○	⋯	⋯
○	○	∗	○	⋯	⋯
⋮	⋮	⋮	⋮	⋱	
⋮	⋮	⋮	⋮		⋱

Figure 3.1: ○ indicates an empty block

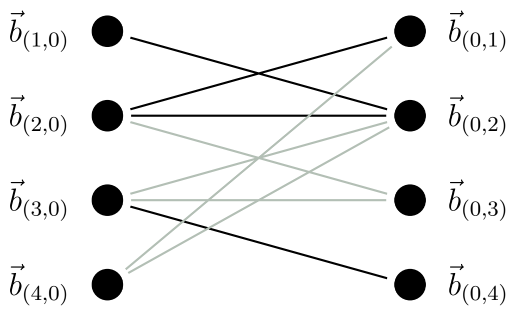


Figure 3.2: The associate graph of the above

Now we will give the storage scheme for the non-adaptive exact quantum scheme for $n = 3$ and $t = 2$. This will help in getting some familiarity with the terminology, help devoloping an intiution toward getting a generalised scheme, and also to complete the section.

storage scheme

There are many cases to look at, given below are all the possible cases similar up to permutation.

subsets of size 0

Assign bit value 0 to all locations i.e. all zero vector.

subsets of size 1

have to describe the figure better

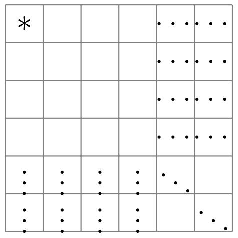


Figure 3.3: This is a view of the arrangement of elements from above such that XY plane, with the y axis increasing in the downward direction and the x axis increasing in the right direction

In the above figure we need to solve the following equation

$$\vec{e}_{(1,1)} = \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)}$$

One of the solution, and hence a storage scheme is -

$$\vec{b}_{(1,0)} = \vec{e}_{(1,1)}$$

and set all other bit-block vectors to zero.

subsets of size 2 - for this we have three cases

Case 1

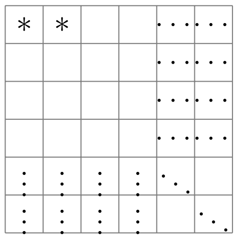


Figure 3.4: grid

here we have the following equations to solve (or constrains to satisfy) -

$$\vec{e}_{(1,1)} = \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)}$$
$$\vec{e}_{(2,1)} = \vec{b}_{(2,0)} \oplus \vec{b}_{(0,1)}$$

Hence a storage scheme -

$$\vec{b}_{(1,0)} = \vec{e}_{(1,1)}$$
$$\vec{b}_{(2,0)} = \vec{e}_{(2,1)}$$

and set all other bit-block vectors to zero.

Case 2

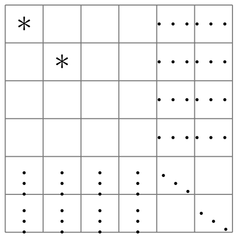


Figure 3.5: grid

here we have the following equations to solve (or constrains to satisfy) -

$$\vec{e}_{(1,1)} = \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)}$$
$$\vec{e}_{(2,2)} = \vec{b}_{(2,0)} \oplus \vec{b}_{(0,1)}$$

Hence a storage scheme -

$$\vec{b}_{(1,0)} = \vec{e}_{(1,1)}$$
$$\vec{b}_{(2,0)} = \vec{e}_{(2,2)}$$

and set all other bit-block vectors to zero.

Case 3

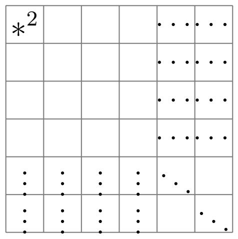


Figure 3.6: $*^2$ this implies there are 2 elements of the subset in this block

In the above figure we need to solve the following equation

$$\vec{e}_{(1,1)} = \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)}$$

One of the solution, and hence a storage scheme is -

$$\vec{b}_{(1,0)} = \vec{e}_{(1,1)}$$

and set all other bit-block vectors to zero.

subsets of size 3 - Here we will have to look at 6 cases

Case 1

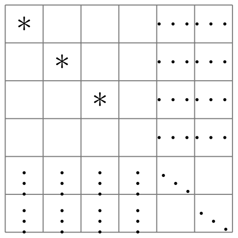


Figure 3.7: grid

here we have the following equations to solve (or constrains to satisfy) -

$$\begin{aligned}
\vec{e}_{(1,1)} &= \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)} \\
\vec{e}_{(2,2)} &= \vec{b}_{(2,0)} \oplus \vec{b}_{(0,2)} \\
\vec{e}_{(3,3)} &= \vec{b}_{(3,0)} \oplus \vec{b}_{(0,3)}
\end{aligned}$$

Hence a storage scheme -

$$\begin{aligned}
\vec{b}_{(1,0)} &= \vec{e}_{(1,1)} \\
\vec{b}_{(2,0)} &= \vec{e}_{(2,2)} \\
\vec{b}_{(3,0)} &= \vec{e}_{(3,3)}
\end{aligned}$$

and set all other bit-block vectors to zero.

Case 2

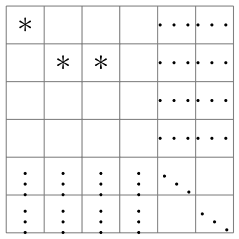


Figure 3.8: grid

here we have the following equations to solve (or constrains to satisfy) -

$$\begin{aligned}\vec{e}_{(1,1)} &= \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)}\\ \vec{e}_{(2,2)} &= \vec{b}_{(2,0)} \oplus \vec{b}_{(0,2)}\\ \vec{e}_{(3,2)} &= \vec{b}_{(3,0)} \oplus \vec{b}_{(0,2)}\end{aligned}$$

Hence a storage scheme -

$$\begin{aligned}\vec{b}_{(1,0)} &= \vec{e}_{(1,1)}\\ \vec{b}_{(2,0)} &= \vec{e}_{(2,2)}\\ \vec{b}_{(3,0)} &= \vec{e}_{(3,3)}\end{aligned}$$

and set all other bit-block vectors to zero.

Case 3

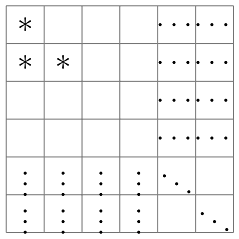


Figure 3.9: grid

here we have the following equations to solve (or constrains to satisfy) -

$$\begin{aligned}\vec{e}_{(1,1)} &= \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)}\\ \vec{e}_{(1,2)} &= \vec{b}_{(1,0)} \oplus \vec{b}_{(0,2)}\\ \vec{e}_{(2,2)} &= \vec{b}_{(2,0)} \oplus \vec{b}_{(0,2)}\end{aligned}$$

There are simpler storage schemes, but the one given below will help build intuition to understand the proof of theorem 3.2.1. Given below is a storage scheme -

$$\begin{aligned}\vec{b}_{(1,0)} &= \vec{e}_{(1,1)} \\ \vec{b}_{(0,1)} &= \vec{e}_{(1,2)} \oplus \vec{b}_{(1,0)} \\ \vec{b}_{(2,0)} &= \vec{e}_{(2,2)} \oplus \vec{b}_{(0,1)}\end{aligned}$$

and set all other bit-block vectors to zero.

Case 4

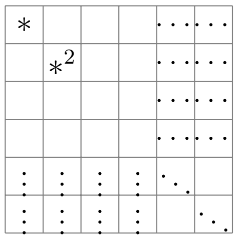


Figure 3.10: grid

here we have the following equations to solve (or constrains to satisfy) -

$$\begin{aligned}\vec{e}_{(1,1)} &= \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)} \\ \vec{e}_{(2,2)} &= \vec{b}_{(2,0)} \oplus \vec{b}_{(0,1)}\end{aligned}$$

Hence a storage scheme -

$$\begin{aligned}\vec{b}_{(1,0)} &= \vec{e}_{(1,1)} \\ \vec{b}_{(2,0)} &= \vec{e}_{(2,2)}\end{aligned}$$

and set all other bit-block vectors to zero.

Case 5

here we have the following equations to solve (or constrains to satisfy) -

$$\begin{aligned}\vec{e}_{(1,1)} &= \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)} \\ \vec{e}_{(2,1)} &= \vec{b}_{(2,0)} \oplus \vec{b}_{(0,1)}\end{aligned}$$

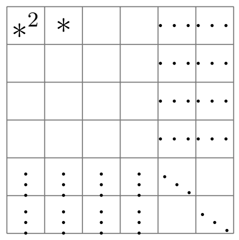


Figure 3.11: grid

Hence a storage scheme -

$$\vec{b}_{(1,0)} = \vec{e}_{(1,1)}$$
$$\vec{b}_{(2,0)} = \vec{e}_{(2,1)}$$

and set all other bit-block vectors to zero.

Case 6

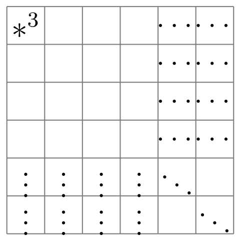


Figure 3.12: $*^2$ this implies there are 2 elements of the subset in this block

In the above figure we need to solve the following equation

$$\vec{e}_{(1,1)} = \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)}$$

One of the solution, and hence a storage scheme is -

$$\vec{b}_{(1,0)} = \vec{e}_{(1,1)}$$

and set all other bit-block vectors to zero.

If you look closely you can see the above is exhaustive, up to permutation

A question we could ask now is can we extend the same idea to subsets of size 4. The following diagram shows why this scheme cannot be extended to subsets of size at most 4.

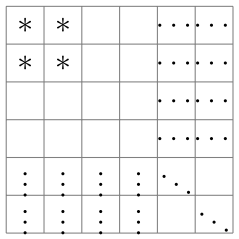


Figure 3.13: grid

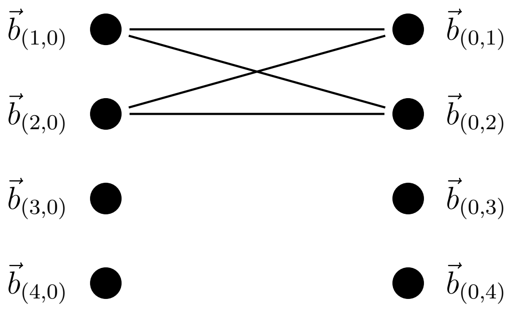


Figure 3.14: We have a 4-cycle, only the black edges are shown in the figure

Why does the above setup cause problems in the storage scheme. To answer this let us look at the set of equations -

$$\begin{aligned}
\vec{e}_{(1,1)} &= \vec{b}_{(1,0)} \oplus \vec{b}_{(0,1)} \\
\vec{e}_{(2,1)} &= \vec{b}_{(2,0)} \oplus \vec{b}_{(0,2)} \\
\vec{e}_{(1,2)} &= \vec{b}_{(1,0)} \oplus \vec{b}_{(0,2)} \\
\vec{e}_{(2,2)} &= \vec{b}_{(2,0)} \oplus \vec{b}_{(0,2)}
\end{aligned}$$

By summing all the four equations we get

$$\vec{e}_{(1,1)} \oplus \vec{e}_{(2,1)} \oplus \vec{e}_{(1,2)} \oplus \vec{e}_{(2,2)} = \vec{0}$$

Hence there exists subsets S,of size 4, for which we do not have a storage scheme. For example $S = \{e(1,1,1), e(1,2,1), e(2,1,1), e(2,2,2)\}$. If you check, you will see that the LHS of the above equation will not give you a zero vector, hence a contradiction.

To understand what is happening we look this grid as graphs, more specifically bipartite graphs as we did at the end of page 14. When figure 3.18 is converted to it's graph form, we see that it contains a cycle of length 4 whose edges are all coloured black. What if I ensure there are no black 4-cycles in the graph? This can be done by spreading the grid. Not every block must contain an element block, i.e. there may exist a block with no elements. All blocks either contain the same number of element or no elements at all, this helps reduce wastage. Now if we ensure that every 2×2 sub-block contains one block with no elements, then we can say that the corresponding graph cannot contain a black 4-cycle. Hence such an arrangement scheme help incorporate subsets of size 4 (which will be shown later). In fact it also works fine with subsets of size 5. So, what happens in the case the where we have subsets of size 6. The following diagram below show an arrangement in which shows why we have problems handeling subsets of size 6.

	$*$	$*$		$\cdots$	$\cdots$
$*$	$*$			$\cdots$	$\cdots$
$*$		$*$		$\cdots$	$\cdots$
				$\cdots$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\ddots$

Figure 3.15: If we analyse it the same way a did for figure 3.18, one could see why such a configuration could cause problems.

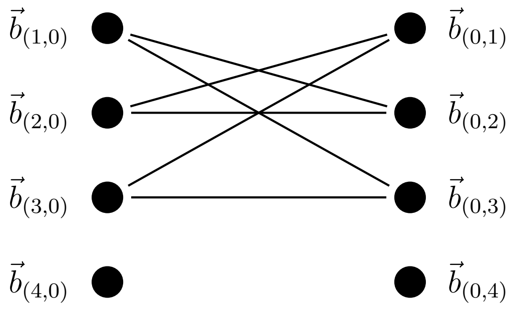


Figure 3.16: We have a 6-cycle, only the black edges are shown in the figure

If you look at the graph associated to the above figure contains a black 6-cycle. Intuitively it seems that if we remove all cycles of size less than $2k$, where $k \in \mathbb{Z}_+$, then we can handle subsets of size $2k$ and $2k+1$ with the above query scheme. This is precisely the next theorem in the thesis.

Notice removing cycle come at a cost, the space complexity increases as we remove cycles of bigger size.

Theorem 3.1.3. *There exists an exact quantum scheme, as above, that can handle subsets of size 2k and 2k+1 if and only if the associated graph to this scheme has girth greater than 2k, where $k \in \mathbb{Z}_+$.*

Proof. ($\Longleftarrow$) Proof by contrapositon - Let us say that the associated graph has a cycle of length 2k. Let the edges in the cycle be $(\vec{e}_{i_1}, \vec{e}_{i_2}, ..., \vec{e}_{i_{2k}})$, where $i_1, i_2, ..., i_{2k} \in \{1, 2, ..., m^\beta\} \times \{1, 2, ..., m^\beta\}$, and let the corresponding vertices be $(\vec{u}_{j_{(2k,1)}}, \vec{u}_{j_{(2,3)}}, ..., \vec{u}_{j_{(2k-2,2k-1)}}, \vec{v}_{h_{(1,2)}}, \vec{v}_{h_{(3,4)}}, ..., \vec{v}_{h_{(2k-1,2k)}})$, where $j_{(2k,1)}, j_{(2,3)}, ..., j_{(2k-2,2k-1)} \in \{1, 2, ..., |E|\} \times \{0\}$ and $h_{(1,2)}, h_{(2,3)}, ..., h_{(2k-1,2k)} \in \{0\} \times \{1, 2, ..., |E|\}$. Here we have the following set of equations -

$$\begin{aligned}
\vec{e}_{i_1} &= \vec{u}_{j_{(2k,1)}} \oplus \vec{v}_{h_{(1,2)}} \\
\vec{e}_{i_2} &= \vec{u}_{j_{(2,3)}} \oplus \vec{v}_{h_{(1,2)}} \\
&\vdots \\
\vec{e}_{i_{2k}} &= \vec{u}_{j_{(2k,1)}} \oplus \vec{v}_{h_{(2k,1)}}
\end{aligned}$$

Summing them all together, we get the following

$$\vec{e}_{i_1} \oplus \vec{e}_{i_2} \oplus ... \oplus \vec{e}_{i_{2k}} = \vec{0}$$

So, if our subset are the elements located at $(i_1, 1), (i_2, 1), ..., (i_{2k-1}, 1), (i_{2k}, 2)$ then in the LHS of the above the 1st entry is a 1, which is a contradiction. This says that if there is a 2k-cycle we cannot represent susets of size 2k in our scheme. now to prove the converse.

($\Longrightarrow$) Here we have to show that given that the associated graph (biparite) has girth greater than $2k$, there exists a storage scheme that can store subsets of size $2k$, $2k+1$, i.e. give a method to produce the bit-block vectors. Notice that the only element blocks we are concerned with are those that contain at least one element from the subset. The edges associated to these edges are coloured black (section?). The graph contains a black forest, i.e. the subgraph formed by the black edges is a forest, this is because girth is $2k+2$ and number of black edges (subsets) is atmost $2k+1$. If we can give a valid storage scheme for a black tree then we have a valid scheme for the black forest too, this is because the assignment of value to the bit-block vector(vertex) in one black tree does not affect the value of the elements of the bit-block vector in another black tree. By doing so we get a storage scheme. Pick an arbitrary vertex in the tree assign it a random value from $0, 1^{m^\alpha}$. The adjacent vertices can be assigned values as the sum of bit-block vectors (vetices) gives us the corresponding element-block vector, i.e.

$$\vec{e}_{(a,b)} = \vec{b}_{(a,0)} \oplus \vec{b}_{(0,b)}$$
$$Edge = Vertex \oplus Vertex$$

This process of selecting values of the vertex is well defined and terminates because it is a tree (no cycles and connected). □

One application of the above result is to show the existence of the following exact non-adaptive quantum scheme, $s_Q(m, n = 4, 5, t = 2) = O(m^{3/4})$.

3.1.3 $s_Q(m, n = 4, 5, t = 2) = O(m^{3/4})$

Arrangement of elements

Here not every block contains elements. All blocks either contain m^α element or no elements at all, α will be calculated later. Lets us first see how to arrange the elements. Since we have to ensure girth of the associate graph is greater than 4, if we construct a bipartite graph with no $K_{2,2}$ subgraph we are done. So answering the following question would help, what is the maximum number of edges in a bipartite?

The above question is answer by a result in extremal graph theory, which we state below -

Theorem 3.1.4. *Let $M(U, V, k)$ be the maximum number of edges in a bipartite graph $G = (U, V, E)$, if $|U| = |V| = n$ We have the following*

$$M(U, V, k) = \mathcal{O}(n^{3/2})$$

The above result say that if I have e blocks with elements(i.e. if there are e edges in the graph with no complete $K_{2,2} subgraph$), then the dimensions of the grid are $e^{2/3} \times e^{2/3}$ (i.e in the associated graph $|V| = m^2/3$ and $|V| = m^2/3$). Also note the following, since in the whole grid we need to fill m elements and each block has m^α elements, this implies that the number block with elements must be $m^{1-\alpha}$. Hence we see that since $e = m^{1-\alpha}$ the dimensions of the grid are $e^{2/3} \times e^{2/3} = m^{2(1-\alpha)/3} \times m^{2(1-\alpha)/3}$. The next step would be to choose α such that the space required (string length) is minimised.

What is the optimal α

The size of the string on the XY - plane is $\mathcal{O}(m^{1-\alpha})$ (i.e. the number of non empty blocks), and the size of the string in the XZ and YZ - planes are $\mathcal{O}(m^{2(1-\alpha)/3}) \times \mathcal{O}(m^{\alpha})$ each. Hence we have the following -

$$s_Q = \mathcal{O}(2m^{2(1-\alpha)/3}) + \mathcal{O}(m^{\alpha} + m^{1-\alpha}) = \mathcal{O}(2m^{2/3+\alpha/3}) + \mathcal{O}(m^{1-\alpha}) \tag{3.1}$$

AM-GM inequality tells us to equate the powers of the sum above, since we are looking at the asymptotic behaviour we can drop the constants.

$$\begin{aligned} 2/3 + \alpha/3 &= 1 - \alpha \\ \alpha &= 1/4 \end{aligned}$$

Plugging back the value of α in equation 3.1, we get

$$s_Q = \mathcal{O}(m^{3/4})$$

Hence we have the following lemma.

Lemma 3.1.5. $s_Q(m, n = 4, 5, t = 2) = \mathcal{O}(m^{3/4})$

Quantum upper bound

We are going to write the space complexity in terms of the number of edges in a bipartite graph of girth greater than $2k$, where $k \in \mathbb{Z}^+$.

Let $E(k)$ be the number of edges in a bipartite graph with girth greater than $2k$. This gives us the number of non empty blocks. Hence, we have have the number of nonempty blocks. We know that each non-empty block is assigned a bit. This takes care of the first probe, now we count the number of bits bits do we require for the second equality probe.

The equality probe are on the bits on the XZ and XY-planes. The dimensions of these planes are $m^\alpha \times m^\beta$, where m^β is the length of the (square) grid and m^α as above. Hence the space complexity to represent subsets of size atmost $2k+1$, which we will denote by $s_Q(2k+1)$, is given below -

$$s_Q(2k+1) = 2(m^\alpha \times m^\beta) + |E(k)|$$

Since all the element lie in some block and each block either contains m^α elements or none, we have, the following -

$$\begin{aligned} m &= m^\alpha |E(k)| \\ \implies m^\alpha &= m/|E(k)| \end{aligned}$$

Using the above, we get

$$\begin{aligned} s_Q(2k+1) &= 2(m^\beta \times m/|E(k)|) + |E(k)| \\ &= \mathcal{O}(m^{1+\beta}/|E(k)| + |E(k)|) \end{aligned} \tag{3.2}$$

Now that we have the space complexity written down in terms of the number of edges, we will use know results graph theory to give use upper bounds for the sample complexity of exact non-adaptive quantum schemes.

A result from ([7]) is as follows - Let $k \geq 1$ be an odd integer, $t = \lfloor (k+2)/4 \rfloor$. They construct a bipartite, q-regular graph with girth $g \geq k+5$, with number of edges $e = \Omega(v^{1+\frac{1}{k-t+1}})$. Since in our bipartite graph we have m^β vertices, this gives us $m^{\beta(1+\frac{1}{k-t+1})}$ edges from the result just mentioned above. We use this in equation 3.2 to get the following

$$s_Q(2k+1) = \mathcal{O}(m^{1+\beta}/(m^{\beta(1+\frac{1}{k-t+1})}) + m^{\beta(1+\frac{1}{k-t+1})}) \tag{3.3}$$

To minimise the space complexity, we will equate the powers.

$$\begin{aligned} 1 + \beta - \beta(1 + \frac{1}{k-t+1}) &= \beta(1 + \frac{1}{k-t+1}) \\ \beta &= \frac{k-t+1}{k-t+3} \end{aligned}$$

Substituting this back into equation 3.3, we get -

$$s_Q(2k+1) = \mathcal{O}(m^{\frac{k-t+2}{k-t+3}})$$

Hence we get an upper bound for quantum schemes. The theorem is stated below

Theorem 3.1.6. *Let $k \geq 1$ be an odd integer, $t = \lfloor (k+2)/4 \rfloor$ and $g \geq k+5$, where $g \in \mathbb{Z}_+$. We have a quantum scheme that can handle subsets of size at most $g-1$, whose space complexity is $\mathcal{O}(m^{\frac{k-t+2}{k-t+3}})$*

Setting $k = 3$ we get the following

Lemma 3.1.7. $s_Q(m, n = 7, t = 2) = \mathcal{O}(m^{4/5})$

Next we will show how the above trick of all removing 4-cycles (complete $K_{2,2}$ subgraphs),the help managing subsets of size 4 in the quantum non-adaptive scheme also help us in getting a classical adaptive scheme that can manage subsets of size 4.

3.2 Adaptive Classical scheme

In the adaptive scheme the second probe address depends on the output of the first probe. Below we will show that results of the quantum scheme gives us insight in the adaptive classical scheme. Arrangement of element is the same but has a different probing scheme (obviously).

Probing scheme

The first probe is the same, i.e. probe the XY-plane. If we get a 1 we look at the XZ-plane else look at the YZ-plane. Writing it down more formally, if we have a querry "Is x in S?" and the location on x is at (x_1, y_1, z_1), we output the following

$$\text{Outputs to the query} = \begin{cases} m(x_1, 0, z_1), & \text{if } m(x_1, y_1, 0) = 1 \\ m(0, y_1, z_1), & \text{if } m(x_1, y_1, 0) = 0 \end{cases} \tag{3.4}$$

where 1 indicates YES and 0 indicating NO. Note the every element-block vector is equal to one of the bit-block vector adjacent to it, which one depends on the first probe.

Using the same ideas we used in the quantum schemes, we will use the same ideas for a two probe adaptive classical scheme.

3.2.1 $s(m, n = 4, t = 2) = O(m^{3/4})$

The arrangements of elements and the data structure is the same as in the quantum case. Since the probing scheme is different, we give give a storage scheme that works in this case.

Storage scheme

We will look at the cases where the size of the subsets are equal to four.

When all four element have either different x or y coordinates

If y coordinates are different, project the element-block vector onto the corresponding bit-block vector.

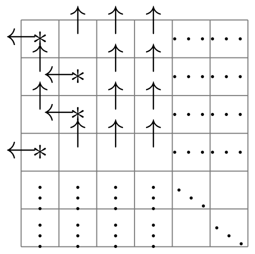


Figure 3.17:

Similarly the other such cases can be handled.

When there is one element that shares an x coordinate with one element and the y coordinate with another element

Hence we have the following set up -

Notice that the fourth element can be placed in one of the five positions and the others follow similar storage schemes. We will show a storage scheme for the following six cases -

Case 1

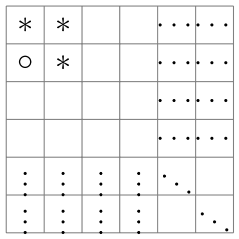


Figure 3.18: An empty block is represented by ∘

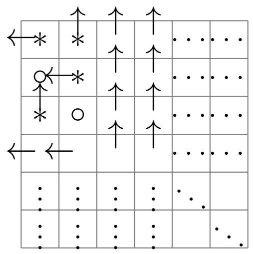


Figure 3.19:

Case 2

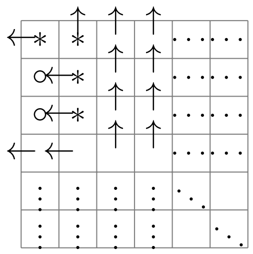


Figure 3.20:

Case 3

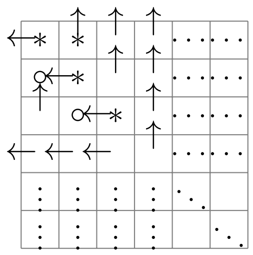


Figure 3.21:

Case 4

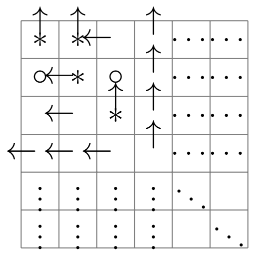


Figure 3.22:

Case 5

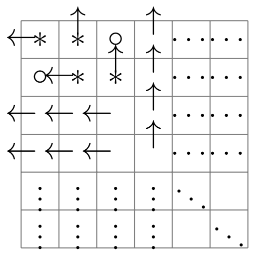
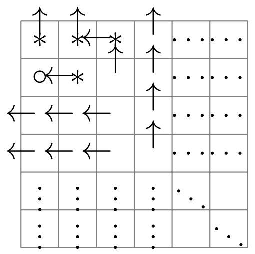


Figure 3.23:

Case 6



One cn check that the cases are exhaustively checked up to permutation. Hence we end with the following lemma.

Lemma 3.2.1. $s_A(m, n = 4, 5, t = 2) = \mathcal{O}(m^{3/4})$

Where A stands for adaptive.

Chapter 4

Future Directions

The quantum scheme helped give insight to construct a better adaptive classical scheme. We would like to do better. Hopefully get an upper bound in the adaptive classical scheme. A way to do this would be to construct graphs the same way we did for the quantum case and show some equivalence. Another thing that we could have done was to use experiment with other advantage quantum probes could provide, not only the equality probes.

Bibliography

[1] J. Radhakrishnan, P. Sen, and S. Venkatesh, "The quantum complexity of set membership," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, ser. FOCS '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 554–. [Online]. Available: http://dl.acm.org/citation.cfm?id=795666.796593

[2] H. Buhrman, P. B. Miltersen, J. Radhakrishnan, and S. Venkatesh, "Are bitvectors optimal?" *SIAM J. Comput.*, vol. 31, no. 6, pp. 1723–1744, Jun. 2002. [Online]. Available: https://doi.org/10.1137/S0097539702405292

[3] M. Galib Anwarul Husain Baig, D. Kesh, and C. Sodani, "An improved scheme in the two query adaptive bitprobe model," 12 2018.

[4] M. Garg and J. Radhakrishnan, "Set membership with non-adaptive bit probes," *CoRR*, vol. abs/1612.09388, 2016. [Online]. Available: http://arxiv.org/abs/1612.09388

[5] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," Bristol, UK, UK, Tech. Rep., 1992.

[6] D. Kopczyk, "Quantum algorithms: Deutschs algorithm." [Online]. Available: http://dkopczyk.quantee.co.uk/deutschs-algorithm/#easy-footnote-1-441

[7] F. Lazebnik and V. A. Ustimenko, "Explicit construction of graphs with an arbitrary large girth and of large size," *Discrete Applied Mathematics*, vol. 60, no. 1-3, pp. 275–284, 1995. [Online]. Available: https://doi.org/10.1016/0166-218X(94)00058-L