Quantum search of partially ordered sets

Ashley Montanaro[1]

[1]Department of Computer Science
University of Bristol
Bristol, UK

28th February 2007

Motivation

Searching for a specific element in a set of elements that have some kind of structure is a fundamental task in computer science.

Let's consider the most basic such task: searching a list of n integers for a specific integer. How many queries to the list do we need?

Motivation

Searching for a specific element in a set of elements that have some kind of structure is a fundamental task in computer science.

Let's consider the most basic such task: searching a list of n integers for a specific integer. How many queries to the list do we need?

List is...	Classical	Quantum
Unsorted	$\Theta(n)$ queries	$\Theta(\sqrt{n})$ queries
Sorted	$\Theta(\log n)$ queries	$\Theta(\log n)$ queries

Motivation

Searching for a specific element in a set of elements that have some kind of structure is a fundamental task in computer science.

Let's consider the most basic such task: searching a list of n integers for a specific integer. How many queries to the list do we need?

List is...	Classical	Quantum
Unsorted	$\Theta(n)$ queries	$\Theta(\sqrt{n})$ queries
Sorted	$\Theta(\log n)$ queries	$\Theta(\log n)$ queries

How can we model some kind of partial structure between these two extremes?

Partially ordered sets

We can write a set unambiguously as a sorted list if it is totally ordered, i.e. every pair of elements is comparable.

We consider sets where some elements may be incomparable. This can be defined by a partial order on the set.

Posets

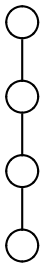
Definition: A partially ordered set (poset) is a set S equipped with an order relation $\leq$, such that, for $a, b, c \in S$:

1. $a \leq a$
2. $(a \leq b) \wedge (b \leq a) \Rightarrow a = b$
3. $(a \leq b) \wedge (b \leq c) \Rightarrow a \leq c$

(We define additional relations $\geq$, $<$, $>$ in the obvious way.)

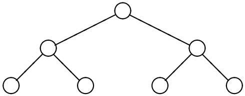
Partially ordered sets

We can visualise partially ordered sets using Hasse diagrams.



Totally ordered set

Unstructured set



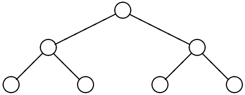
Tree-like poset

Partially ordered sets

We can visualise partially ordered sets using Hasse diagrams.

Totally ordered set

Unstructured set



Tree-like poset

Some more definitions:

- A *chain* is a subset whose elements are all comparable
- An *antichain* is a subset whose elements are all incomparable
- The *height* of a poset is the size of its largest chain
- The *width* of a poset is the size of its largest antichain

The poset we are searching will always be called S and will contain n elements.

Questions

I will attempt to answer the following questions:

- When can quantum computers achieve any reduction in the number of queries required to search posets?
- What are the limits of this reduction (e.g. can we beat Grover's algorithm for unstructured sets?)
- Can we come up with interesting quantum algorithms for searching posets?
- Are there any applications outside of this model?

Outline

I will discuss:

- Part 0: two models for quantum search
- Part 1: the abstract model
  1. General lower and upper bounds on query complexity
  2. Searching forest-like posets
- Part 2: the concrete model
  1. General lower and upper bounds on query complexity
  2. Searching a partially sorted array
  3. The intersection of two ordered lists

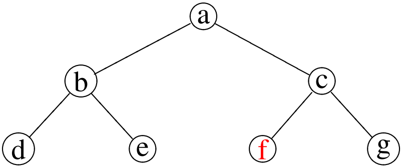
Part 0: two models for quantum search

It turns out that there are two natural models for poset search.

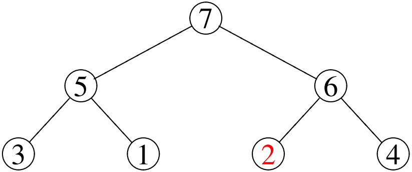
- In the abstract model, we search for an unknown "marked" element a of the set. Querying an element x returns one of $\{<, =, \not\leq\}$ according to whether $a < x$, $a = x$ or $a \not\leq x$.
- In the concrete model, we consider each element in the poset to store an integer. Querying an element returns that integer. The goal is to find where a known "target" integer is stored.

Illustrating the two models

Abstract model:

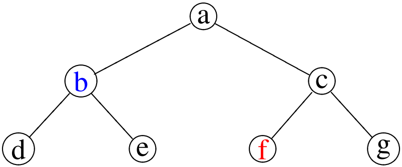


Concrete model:

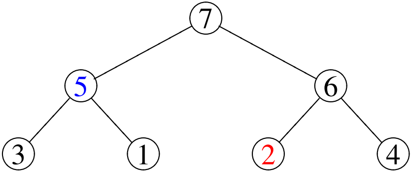


Illustrating the two models

Abstract model:

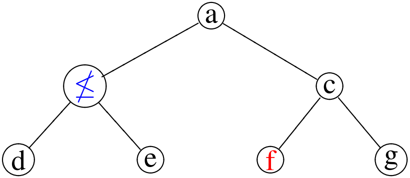


Concrete model:

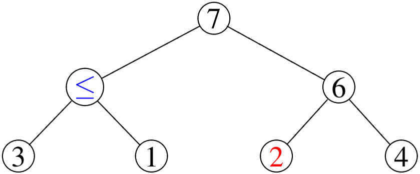


Illustrating the two models

Abstract model:

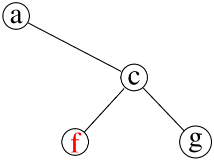


Concrete model:

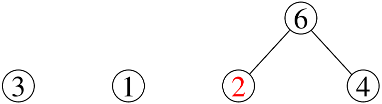


Illustrating the two models

Abstract model:

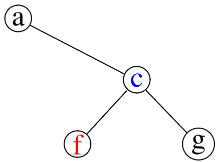


Concrete model:

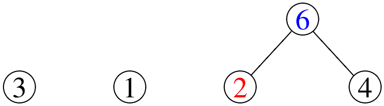


Illustrating the two models

Abstract model:

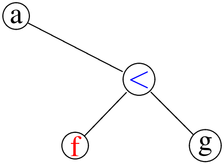


Concrete model:

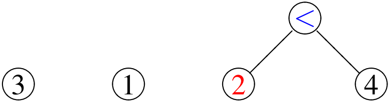


Illustrating the two models

Abstract model:



Concrete model:



Abstract model:

f g

Concrete model:

3 1 2 4

Measuring complexity in these models

We will measure the difficulty of poset search using the framework of query complexity.

- We count the number of queries to an appropriate oracle.
- When passed an element x of the set, an oracle query returns either the integer $S[x]$ stored there (concrete model), or the relationship $\{<, =, \not\leq\}$ between x and the unknown "marked" element a (abstract model).

Measuring complexity in these models

We will measure the difficulty of poset search using the framework of query complexity.

- We count the number of queries to an appropriate oracle.
- When passed an element x of the set, an oracle query returns either the integer $S[x]$ stored there (concrete model), or the relationship $\{<, =, \not\leq\}$ between x and the unknown "marked" element a (abstract model).

Definition

For a poset S, let the minimum number of queries required to find the target element be

$$\left.\begin{array}{c} D(S) \\ Q_E(S) \\ Q_2(S) \end{array}\right\} \text{for a} \left\{\begin{array}{c} \text{exact classical} \\ \text{exact quantum} \\ \text{bounded-error quantum} \end{array}\right\} \text{algorithm.}$$

We'll usually assume there is only one target element.

Previous results

Classical:

- The poset search problem in the concrete model was introduced by Linial and Saks[1], who gave asymptotically tight bounds on the query complexity.
- The abstract model was introduced by Ben-Asher, Farchi and Newman[2]. Subsequent authors have given algorithms for finding optimal query sequences for a given poset.
- Finding the optimal such sequence for a general poset is NP-complete but can be done efficiently for forest-like posets.

[1] N. Linial, M. Saks. Searching ordered structures. *Journal of Algorithms* 6

[2] Y. Ben-Asher, E. Farchi, I. Newman. Optimal search in trees. *SIAM J. Comp.* 28

Previous results

Classical:

- The poset search problem in the concrete model was introduced by Linial and Saks[1], who gave asymptotically tight bounds on the query complexity.
- The abstract model was introduced by Ben-Asher, Farchi and Newman[2]. Subsequent authors have given algorithms for finding optimal query sequences for a given poset.
- Finding the optimal such sequence for a general poset is NP-complete but can be done efficiently for forest-like posets.

Quantum:

- Problem only considered for totally ordered and unstructured sets.

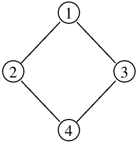
[1] N. Linial, M. Saks. Searching ordered structures. *Journal of Algorithms* 6

[2] Y. Ben-Asher, E. Farchi, I. Newman. Optimal search in trees. *SIAM J. Comp.* 28

Part 1: The abstract model

We obtain lower and upper bounds in this model by a reduction to the oracle identification problem of Ambainis et al[3].

- For each possible marked element a, we have an oracle which, given $x \in S$, returns $f_a(x) \in \{<, =, \not\leq\}$. Encode this output as 2 bits, giving a Boolean function.

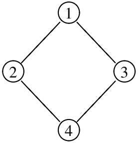


[3] A. Ambainis et al. Quantum identification of Boolean oracles. *Proc. STACS'04*

Part 1: The abstract model

We obtain lower and upper bounds in this model by a reduction to the oracle identification problem of Ambainis et al[3].

- For each possible marked element a, we have an oracle which, given $x \in S$, returns $f_a(x) \in \{<, =, \nleq\}$. Encode this output as 2 bits, giving a Boolean function.



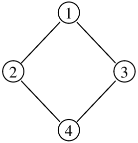
$$\Rightarrow \quad \begin{array}{cc} & \longleftarrow x \longrightarrow \\ \begin{array}{c} \uparrow \\ a \\ \downarrow \end{array} & \begin{pmatrix} = & \nleq & \nleq & \nleq \\ < & = & \nleq & \nleq \\ < & \nleq & = & \nleq \\ < & < & < & = \end{pmatrix} \end{array}$$

[3]A. Ambainis et al. Quantum identification of Boolean oracles. *Proc. STACS'04*

Part 1: The abstract model

We obtain lower and upper bounds in this model by a reduction to the oracle identification problem of Ambainis et al[3].

- For each possible marked element a, we have an oracle which, given $x \in S$, returns $f_a(x) \in \{<, =, \not\leq\}$. Encode this output as 2 bits, giving a Boolean function.



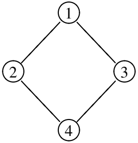
$$\Rightarrow \begin{array}{cc} & \longleftarrow x \longrightarrow \\ \begin{array}{c}\uparrow\\ a\\ \downarrow\end{array} & \begin{pmatrix} = & \not\leq & \not\leq & \not\leq \\ < & = & \not\leq & \not\leq \\ < & \not\leq & = & \not\leq \\ < & < & < & = \end{pmatrix} \end{array} \Rightarrow \begin{array}{cc} & \longleftarrow x \longrightarrow \\ \begin{array}{c}\uparrow\\ a\\ \downarrow\end{array} & \begin{pmatrix} 01 & 10 & 10 & 10 \\ 00 & 01 & 10 & 10 \\ 00 & 10 & 01 & 10 \\ 00 & 00 & 00 & 01 \end{pmatrix} \end{array}$$

[3] A. Ambainis et al. Quantum identification of Boolean oracles. *Proc. STACS'04*

Part 1: The abstract model

We obtain lower and upper bounds in this model by a reduction to the oracle identification problem of Ambainis et al[3].

- For each possible marked element a, we have an oracle which, given $x \in S$, returns $f_a(x) \in \{<, =, \not\leq\}$. Encode this output as 2 bits, giving a Boolean function.



$$\Rightarrow \begin{array}{cc} & \longleftarrow x \longrightarrow \\ \begin{array}{c}\uparrow\\ a\\ \downarrow\end{array} & \begin{pmatrix} = & \not\leq & \not\leq & \not\leq \\ < & = & \not\leq & \not\leq \\ < & \not\leq & = & \not\leq \\ < & < & < & = \end{pmatrix}\end{array} \Rightarrow \begin{array}{cc} & \longleftarrow x \longrightarrow \\ \begin{array}{c}\uparrow\\ a\\ \downarrow\end{array} & \begin{pmatrix} 01 & 10 & 10 & 10 \\ 00 & 01 & 10 & 10 \\ 00 & 10 & 01 & 10 \\ 00 & 00 & 00 & 01 \end{pmatrix}\end{array}$$

- So the problem becomes: distinguish the n different Boolean functions corresponding to different values of a.
- Strong upper and lower bounds are already known for this problem!

[3] A. Ambainis et al. Quantum identification of Boolean oracles. *Proc. STACS'04*

The bounds of Atici, Gortler and Servedio

- Atici, Gortler and Servedio studied this problem from the perspective of computational learning theory[4].
- Servedio and Gortler find upper and lower bounds in terms of a parameter γ^S:

$$\gamma^S = \min_{S' \subseteq S, |S'| \geq 2} \max_{a \in \{0,1\}^m} \min_{b \in \{0,1\}} \frac{|S'_{a,b}|}{|S'|}$$

where:

- m is the number of bits functions are defined on
- $S'_{a,b}$ is the subset of S' taking value b on input a

Informally: the maximum fraction of functions which a classical algorithm can be sure of removing from consideration with a single query.

[4] R. Servedio, S. Gortler. Quantum vs. classical learnability. *Proc. CCC'01*
A. Atici, R. Servedio. Improved bounds on quantum learning algorithms. *Quantum Information Processing* 4

Bounds on poset search in the abstract model

Translating their results into our setting, we have:

Theorem

For a poset S with n elements,

$$Q_2(S) = \Omega\left(\frac{1}{\sqrt{\gamma^S}}\right) \text{ and } D(S) = O\left(\frac{\log n}{\gamma^S}\right)$$

So $D(S) = O(Q_2(S)^2 \log n)$.

Atici and Servedio give a quantum algorithm that almost achieves this query complexity, giving $Q_2(S) = O\left(\log n \log\log n / \sqrt{\gamma^S}\right)$. Can we do better?

Sketch of general poset search algorithm

Idea behind Atici and Servedio's algorithm:

1. Maintain a set of items that could be the marked item
2. Use Grover search on a set of size about $\frac{1}{\gamma^S}$ items to reduce the size of this set by about half
3. Repeat until only one possible item remains

Sketch of general poset search algorithm

Idea behind Atici and Servedio's algorithm:

1. Maintain a set of items that could be the marked item
2. Use Grover search on a set of size about $\frac{1}{\gamma^S}$ items to reduce the size of this set by about half
3. Repeat until only one possible item remains

Ends up almost giving an $O\left(\log n / \sqrt{\gamma^S}\right)$ bounded-error algorithm: the $O(\log \log n)$ factor comes from needing to repeat the Grover search step to improve its success probability.

(in fact, that bit can be improved to $O(\sqrt{\log \log n})$ using a version of amplitude amplification for high success probabilities)

Searching forest-like posets

We give a new algorithm based on similar ideas that searches forest-like posets. Advantages:

- The Grover search step can be made exact, giving an overall time complexity of $O\left(\log n/\sqrt{\gamma^S}\right)$
- The new algorithm is exact...
- ...and it can easily be extended to searching posets with multiple marked elements.
    - (though we regain an $O(\sqrt{\log \log n})$ penalty in query complexity and the algorithm becomes bounded-error again)

Idea: find a set G of about $\frac{1}{\gamma^S}$ items such that the marked element is a "descendant" of at most one element of G.

The algorithm

To find the marked element $a \in S$:

1. Maintain a set T of items that could be the marked item.
2. Find the most central element $v \in T$.
   - If v is maximal, set $G = \{\text{maximal elements of } T\}$
   - Otherwise, set $G = \{\text{siblings of } v\}$
3. Perform exact Grover search on G to find $g \in G$ such that $a \leq g$.
4. Remove everything from T that isn't a descendant of g.
5. Repeat until T has one or zero elements.

The algorithm

To find the marked element $a \in S$:

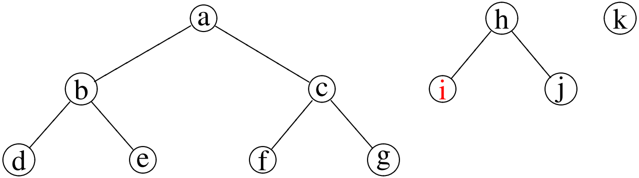
1. Maintain a set T of items that could be the marked item.
2. Find the most central element $v \in T$.
   - If v is maximal, set $G = \{\text{maximal elements of } T\}$
   - Otherwise, set $G = \{\text{siblings of } v\}$
3. Perform exact Grover search on G to find $g \in G$ such that $a \leq g$.
4. Remove everything from T that isn't a descendant of g.
5. Repeat until T has one or zero elements.

Explanation:

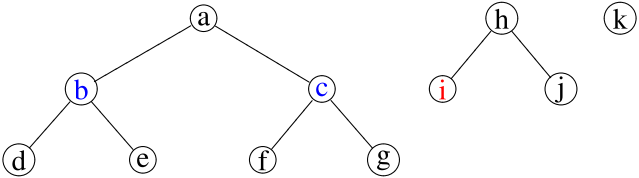
- The weight of $v \in S$ is $wt(v) = |\{x : (x \in S) \wedge (x \leq v)\}|$
- The most central element $v \in T$ has maximal weight, given that $wt(v) \leq \lceil |S|/2 \rceil$.

One can prove that $|G| \approx 1/\gamma^S$ and each step removes $\approx 1/2$ of the elements in T.

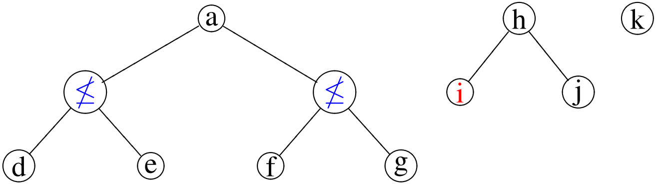
Example



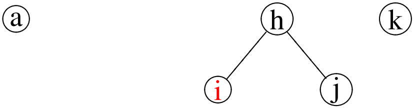
Example



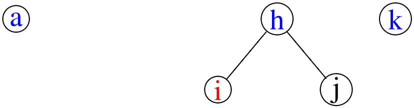
Example



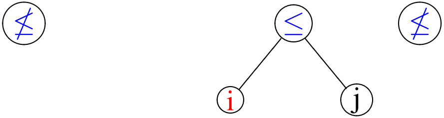
Example



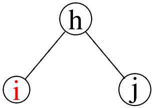
Example



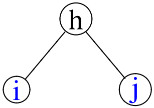
Example



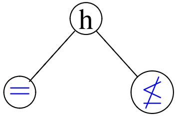
Example



Example



Example



Part 2: the concrete model

Recap: In the concrete model, we consider each element x in the poset to store an integer $S[x]$. Querying an element returns that integer. The goal is to find where a known "target" integer is stored.

Part 2: the concrete model

Recap: In the concrete model, we consider each element x in the poset to store an integer $S[x]$. Querying an element returns that integer. The goal is to find where a known "target" integer is stored.

- This model appears harder to analyse because the query complexity depends not only on the poset structure, but on the integers stored in the poset.
- We will show a general lower bound on query complexity based on the width $w(S)$ of a poset S – i.e. the size of the "largest unsorted subset".
- It turns out that this almost completely determines the query complexity.

The lower bound

$T \subseteq S$ is a section of S if $(x \in T) \wedge (z \in T) \wedge (x < y < z) \Rightarrow y \in T$.

Theorem

Let T be a section of S. Then $D(S) \geq D(T)$, $Q_E(S) \geq Q_E(T)$ and $Q_2(S) \geq Q_2(T)$.

(NB: not trivial! e.g. doesn't hold if T is a general subset of S)

The lower bound

$T \subseteq S$ is a section of S if $(x \in T) \wedge (z \in T) \wedge (x < y < z) \Rightarrow y \in T$.

Theorem

Let T be a section of S. Then $D(S) \geq D(T)$, $Q_E(S) \geq Q_E(T)$ and $Q_2(S) \geq Q_2(T)$.

(NB: not trivial! e.g. doesn't hold if T is a general subset of S)

So, as:

- an antichain $T \subseteq S$ is a section of S.
- we can use a standard lower bound on inverting a permutation to lower bound $Q_2(T)$

we have:

Theorem

$D(S) = \Omega(w(S))$ and $Q_2(S) = \Omega(\sqrt{w(S)})$.

Upper bounds

Upper bounds in the concrete model follow from Dilworth's Theorem:

Dilworth's Theorem

Let S be an n-element poset with $w(S) = k$. Then S is the union of k disjoint chains.

Upper bounds

Upper bounds in the concrete model follow from Dilworth's Theorem:

Dilworth's Theorem

Let S be an n-element poset with $w(S) = k$. Then S is the union of k disjoint chains.

We can search a chain of length l in time $O(\log l)$ using binary search. As there are $w(S)$ chains, we have

Upper bounds

$D(S) = O(w(S) \log h(S))$
$Q_E(S) = O(\sqrt{w(S)} \log h(S))$

because we can search the chains in quantum parallel!

Recap

So we've shown the following theorem.

Theorem

Let S be an n-element poset, and let $D(S)$ and $Q_2(S)$ be the number of queries required for an exact classical or bounded-error quantum (respectively) algorithm to find the target element in S, in either of the two models discussed above. Then

$$\begin{aligned} D(S) &= O(Q_2(S)^2 \log n) \\ Q_2(S) &= \begin{cases} O(\sqrt{D(S)} \log n \sqrt{\log \log n}) & \text{(abstract model)} \\ O(\sqrt{D(S)} \log n) & \text{(concrete model)} \end{cases} \end{aligned}$$

...we can't do much better than Grover's algorithm for any poset, but we can almost achieve the optimal quantum improvement.

An interesting problem in the concrete model: search a multidimensional integer array A sorted in each dimension, i.e. $(i_1 \leq j_1) \wedge (i_2 \leq j_2) \wedge \cdots \wedge (i_d \leq j_d) \Rightarrow A(i_1, \dots, i_d) \leq A(j_1, \dots, j_d)$.

1	3	6
2	4	8
5	7	9

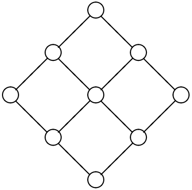


Figure: A 3×3 2-dimensional array sorted by rows and columns, and its corresponding Hasse diagram.

This immediately gives rise to a poset; but we'll mostly think in terms of the original array.

Searching partially sorted arrays

We're particularly interested in d-dimensional $m \times m \times \cdots \times m$ arrays.

- One can show that for this poset $w(S) = \Theta(d^{m-1})$. So the general algorithm gives a query complexity of $O(d^{(m-1)/2} d \log m)$. Can we improve on this?
- Yes! We can achieve the optimal query complexity of $O(d^{(m-1)/2})$.

[5] H. Buhrman, C. Durr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, R. de Wolf. Quantum algorithms for element distinctness. *SIAM J. Comput.* 34

Searching partially sorted arrays

We're particularly interested in d-dimensional $m \times m \times \cdots \times m$ arrays.

- One can show that for this poset $w(S) = \Theta(d^{m-1})$. So the general algorithm gives a query complexity of $O(d^{(m-1)/2} d \log m)$. Can we improve on this?
- Yes! We can achieve the optimal query complexity of $O(d^{(m-1)/2})$.

Some results that were already known:

- An optimal classical algorithm achieving $D(S) = O(d^{m-1})$
- A result of Buhrman et al[5] can be adapted to this setting to achieve $O(d^{(m-1)/2} c^{\log^* m})$ for some constant c
  - $\log^*(x)$ is the iterated log function (number of logs needed to reduce x to ≤ 2)

---

[5] H. Buhrman, C. Durr, M. Heiligman, P. Høyer, F. Magniez, M. Santha, R. de Wolf. Quantum algorithms for element distinctness. *SIAM J. Comput.* 34

Searching partially sorted arrays

- What if we could search an $m \times m$ array with a bounded-error quantum algorithm using $O(\sqrt{m})$ queries?
    - Would imply the optimal d-dimensional algorithm: we can split the array into m^{d-2} disjoint 2-dimensional arrays and use each 2-dimensional search as an oracle within an overall application of quantum search.

Searching partially sorted arrays

- What if we could search an $m \times m$ array with a bounded-error quantum algorithm using $O(\sqrt{m})$ queries?
  - Would imply the optimal d-dimensional algorithm: we can split the array into m^{d-2} disjoint 2-dimensional arrays and use each 2-dimensional search as an oracle within an overall application of quantum search.
- The general algorithm given earlier nests binary search on the rows within Grover search on the columns to achieve $O(\sqrt{m}\log m)$ queries.
- Goal: beat this and find the optimal 2-dimensional quantum search algorithm.
- Idea: find a classical 2-dimensional algorithm that's amenable to quantum speed-up.

An asymptotically optimal classical algorithm

Given an $m \times m$ array A:

1. Perform binary search on the middle row/column of A.
2. After binary search, can eliminate two subarrays of A containing about half the elements in A.
3. We're left with two subarrays which might contain the target element: recurse on these subarrays.

Can show query complexity $D(m) \leq O(\log m) + 2D(m/2) = O(m)$.

(a different optimal classical algorithm was already known, but seems harder to "make quantum")

Example: search of a 5×5 array

1	3	5	10	13
2	4	7	**11**	14
6	8	9	15	21
12	16	17	20	24
18	19	22	23	25

Searching for the element 11:

- Yellow squares are those that are searched in each round
- Light grey squares have been excluded from consideration
- White squares are still to be searched
- Here, 11 is found with only 2 levels of recursion.

Example: search of a 5 × 5 array

1	3	5	10	13
2	4	7	**11**	14
6	8	9	15	21
12	16	17	20	24
18	19	22	23	25

1	3	5	10	13
2	4	7	**11**	14
6	8	9	15	21
12	16	17	20	24
18	19	22	23	25

Searching for the element 11:

- Yellow squares are those that are searched in each round
- Light grey squares have been excluded from consideration
- White squares are still to be searched
- Here, 11 is found with only 2 levels of recursion.

Example: search of a 5 × 5 array

1	3	5	10	13
2	4	7	**11**	14
6	8	9	15	21
12	16	17	20	24
18	19	22	23	25

1	3	5	10	13
2	4	7	**11**	14
6	8	9	15	21
12	16	17	20	24
18	19	22	23	25

1	3	5	10	13
2	4	7	**11**	14
6	8	9	15	21
12	16	17	20	24
18	19	22	23	25

Searching for the element 11:

- Yellow squares are those that are searched in each round
- Light grey squares have been excluded from consideration
- White squares are still to be searched
- Here, 11 is found with only 2 levels of recursion.

Optimal quantum search of a 2-dimensional array

- Idea: perform the recursive search of the two subarrays in quantum parallel.
- Want to end up with a recurrence like
  $Q_2(m) \leq O(\log m) + \sqrt{2}\, Q_2(m/2) = O(\sqrt{m})$.
- Immediate from Grover's algorithm?

Optimal quantum search of a 2-dimensional array

- Idea: perform the recursive search of the two subarrays in quantum parallel.
- Want to end up with a recurrence like $Q_2(m) \leq O(\log m) + \sqrt{2}\, Q_2(m/2) = O(\sqrt{m})$.
- Immediate from Grover's algorithm?

Not so fast!

- We can't use $\sqrt{2}$ queries for the search (it's not even an integer!)

Optimal quantum search of a 2-dimensional array

- Idea: perform the recursive search of the two subarrays in quantum parallel.
- Want to end up with a recurrence like $Q_2(m) \leq O(\log m) + \sqrt{2}\, Q_2(m/2) = O(\sqrt{m})$.
- Immediate from Grover's algorithm?

Not so fast!

- We can't use $\sqrt{2}$ queries for the search (it's not even an integer!)
- What if we performed k levels of recursion then used Grover search on the resulting 2^k subarrays?
- Seems to give $Q_2(m) \leq 2^k O(\log m) + O(2^{k/2})\, Q_2(m/2^k) = O(m^{1/2+c})$
  - for some constant c that turns up because of the hidden constant in the big-O notation

Optimal quantum search of a 2-dimensional array

- Idea: perform the recursive search of the two subarrays in quantum parallel.
- Want to end up with a recurrence like
  $Q_2(m) \le O(\log m) + \sqrt{2}\, Q_2(m/2) = O(\sqrt{m})$.
- Immediate from Grover's algorithm?

Not so fast!

- We can't use $\sqrt{2}$ queries for the search (it's not even an integer!)
- What if we performed k levels of recursion then used Grover search on the resulting 2^k subarrays?
- Seems to give
  $Q_2(m) \le 2^k O(\log m) + O(2^{k/2})\, Q_2(m/2^k) = O(m^{1/2+c})$
  - for some constant c that turns up because of the hidden constant in the big-O notation

Moral: We have to be very careful about constants in this recursive algorithm!

A general recursive quantum search algorithm

- Goal: a "cookbook" way of "quantising" recursive classical search algorithms
- We extend a powerful result of Aaronson and Ambainis[6] on quantum search of spatial regions
- Idea: it's more efficient to do fewer iterations of amplitude amplification

[6] S. Aaronson, A. Ambainis. Quantum search of spatial regions. *Theory of Computing* 1

A general recursive quantum search algorithm

- Goal: a "cookbook" way of "quantising" recursive classical search algorithms
- We extend a powerful result of Aaronson and Ambainis[6] on quantum search of spatial regions
- Idea: it's more efficient to do fewer iterations of amplitude amplification
- So our recursive algorithm performs "a small amount of" amplitude amplification on an algorithm that consists of:
  - Divide the input into some number of subinputs
  - Pick one of these subinputs at random
  - Call yourself on that subinput
- Then it does "lots" of amplitude amplification at the end.
- Importantly, can find exact bounds on the number of queries required to achieve a certain success probability!

[6] S. Aaronson, A. Ambainis. Quantum search of spatial regions. *Theory of Computing* 1

A general "recursive quantum search theorem"

- Let problem P_n be: search an abstract database, with an abstract "size" n, for a known element which may or may not be in the database.
- Let $T(n)$ be time required for a bounded-error quantum algorithm to solve P_n.

A general "recursive quantum search theorem"

- Let problem P_n be: search an abstract database, with an abstract "size" n, for a known element which may or may not be in the database.
- Let $T(n)$ be time required for a bounded-error quantum algorithm to solve P_n.

Let P_n satisfy the following conditions:

- If $n \leq n_0$ for some constant n_0: can find the element in time $T(n) \leq t_0$, for some constant t_0.

A general "recursive quantum search theorem"

- Let problem P_n be: search an abstract database, with an abstract "size" n, for a known element which may or may not be in the database.
- Let $T(n)$ be time required for a bounded-error quantum algorithm to solve P_n.

Let P_n satisfy the following conditions:

- If $n \leq n_0$ for some constant n_0: can find the element in time $T(n) \leq t_0$, for some constant t_0.
- If $n > n_0$: the database can be divided into k sub-databases of size at most $\lceil n/k \rceil$, for some constant $k > 1$.

A general "recursive quantum search theorem"

- Let problem P_n be: search an abstract database, with an abstract "size" n, for a known element which may or may not be in the database.
- Let $T(n)$ be time required for a bounded-error quantum algorithm to solve P_n.

Let P_n satisfy the following conditions:

- If $n \leq n_0$ for some constant n_0: can find the element in time $T(n) \leq t_0$, for some constant t_0.
- If $n > n_0$: the database can be divided into k sub-databases of size at most $\lceil n/k \rceil$, for some constant $k > 1$.
- If the element is in the original database, then it is in exactly one of these sub-databases.

A general "recursive quantum search theorem"

- Let problem P_n be: search an abstract database, with an abstract "size" n, for a known element which may or may not be in the database.
- Let $T(n)$ be time required for a bounded-error quantum algorithm to solve P_n.

Let P_n satisfy the following conditions:

- If $n \leq n_0$ for some constant n_0: can find the element in time $T(n) \leq t_0$, for some constant t_0.
- If $n > n_0$: the database can be divided into k sub-databases of size at most $\lceil n/k \rceil$, for some constant $k > 1$.
- If the element is in the original database, then it is in exactly one of these sub-databases.
- Each division into sub-databases uses time $f(n)$, where $f(n) = O(n^{1/2-\epsilon})$ for some $\epsilon > 0$.

A general "recursive quantum search theorem"

- Let problem P_n be: search an abstract database, with an abstract "size" n, for a known element which may or may not be in the database.
- Let $T(n)$ be time required for a bounded-error quantum algorithm to solve P_n.

Let P_n satisfy the following conditions:

- If $n \leq n_0$ for some constant n_0: can find the element in time $T(n) \leq t_0$, for some constant t_0.
- If $n > n_0$: the database can be divided into k sub-databases of size at most $\lceil n/k \rceil$, for some constant $k > 1$.
- If the element is in the original database, then it is in exactly one of these sub-databases.
- Each division into sub-databases uses time $f(n)$, where $f(n) = O(n^{1/2-\epsilon})$ for some $\epsilon > 0$.

Then $T(n) = O(\sqrt{n})$.

The intersection of two ordered lists

Problem: Given two lists of n sorted integers, output an element that occurs in both lists, or "not found".

1	2	4	4	8	9	10

3	4	5	6	6	7	8

The intersection of two ordered lists

Problem: Given two lists of n sorted integers, output an element that occurs in both lists, or "not found".

1	2	4	4	8	9	10

3	4	5	6	6	7	8

- Obvious classical lower bound is $2n$ queries (have to read all the input in)
- "Obvious" quantum algorithm uses $O(\sqrt{n} \log n)$ queries (wrap binary search in one list within Grover search on the other)
- Buhrman et al gave an ingenious $O(\sqrt{n} c^{\log^* n})$ algorithm
- Lower bound is $\Omega(\sqrt{n})$ queries

We give an algorithm matching this lower bound.

Reducing the problem to poset search

We can (almost) use the algorithm for searching the 2-dimensional array.

- Consider a notional $n \times n$ array T where $T(x, y) = L_x - M_{m+1-y}$ for lists L and M
- Then finding a zero in T finds a match in the two lists.

	8	7	6	6	5	4	3
1	-7	-6	-5	-5	-4	-3	-2
2	-6	-5	-4	-4	-3	-2	-1
4	-4	-3	-2	-2	-1	0	1
4	-4	-3	-2	-2	-1	0	1
8	0	1	2	2	3	4	5
9	1	2	3	3	4	5	6
10	2	3	4	4	5	6	7

Finishing off the algorithm

Problem: The poset search algorithm can only cope with at most one marked element.

Solution:

- Note that the zeroes only occur in rectangular blocks, with at most one block per row and column
- If there's only one such "zero block", can modify the search algorithm to pretend that it only contains one element
- If not, to reduce to the single-block case, repeatedly throw away random rows and columns over several rounds
- Can show that with constant probability, one round will have only one zero block remaining
- Can also show that the asymptotic query complexity isn't hurt by doing this

Summary and further work

- We have obtained general and almost tight upper and lower bounds on quantum search of posets.
- Quantum computers can achieve at most a polynomial reduction in queries.
- We've given an optimal algorithm for searching a $r \times c$ array of integers sorted by rows and columns...
- ...leading to an optimal $O(\sqrt{n})$ algorithm to find the intersection of two n element sorted lists.

Summary and further work

- We have obtained general and almost tight upper and lower bounds on quantum search of posets.
- Quantum computers can achieve at most a polynomial reduction in queries.
- We've given an optimal algorithm for searching a $r \times c$ array of integers sorted by rows and columns...
- ...leading to an optimal $O(\sqrt{n})$ algorithm to find the intersection of two n element sorted lists.

Remaining annoyances:

- Searching for multiple elements in general posets in the abstract model?
- A general $\Omega(\log n)$ lower bound in the abstract model?
- Tightening the upper bounds in both models.

The End

- Further reading: “Quantum search of partially ordered sets”, quant-ph/0702196

- Thanks for your time!

Search for multiple target elements

How can we extend these models to cope with multiple target elements?

- Concrete model: just allow the set to store integers that are not distinct
- Abstract model: one approach is to use an oracle that returns $\leq$ if *any* of the "marked" elements are less than x
- Lower bounds go through in both cases
- General upper bound holds in concrete model, but not in abstract model (no obvious reduction to OIP)
- Not obvious how to extend recursive quantum search theorem to this case...