# Quantum walk algorithm for element distinctness

Andris Ambainis*

**Abstract**

We use quantum walks to construct a new quantum algorithm for element distinctness and its generalization. For element distinctness (the problem of finding two equal items among $N$ given items), we get an $O(N^{2/3})$ query quantum algorithm. This improves the previous $O(N^{3/4})$ quantum algorithm of Buhrman et al. [14] and matches the lower bound by [1]. We also give an $O(N^{k/(k+1)})$ query quantum algorithm for the generalization of element distinctness in which we have to find $k$ equal items among $N$ items.

## 1 Introduction

Element distinctness is the following problem.

**Element Distinctness.** Given numbers $x_1, \ldots, x_N \in [M]$, are they all distinct?

It has been extensively studied both in classical and quantum computing. Classically, the best way to solve element distinctness is by sorting which requires $\Omega(N)$ queries. In quantum setting, Buhrman et al. [14] have constructed a quantum algorithm that uses $O(N^{3/4})$ queries. Aaronson and Shi [1] have shown that any quantum algorithm requires at least $\Omega(N^{2/3})$ quantum queries.

In this paper, we give a new quantum algorithm that solves element distinctness with $O(N^{2/3})$ queries to $x_1, \ldots, x_N$. This matches the lower bound of [1, 5].

Our algorithm uses a combination of several ideas: quantum search on graphs [2] and quantum walks [30]. While each of those ideas has been used before, the present combination is new.

We first reduce element distinctness to searching a certain graph with vertices $S \subseteq \{1, \ldots, N\}$ as vertices. The goal of the search is to find a marked vertex. Both examining the current vertex and moving to a neighboring vertex cost one time step. (This contrasts with the usual quantum search [26], where only examining the current vertex costs one time step.)

We then search this graph by quantum random walk. We start in a uniform superposition over all vertices of a graph and perform a quantum random walk with one transition rule for unmarked vertices of the graph and another transition rule for marked vertices of the graph. The result is that the amplitude gathers in the marked vertices and, after $O(N^{2/3})$ steps, the probability of measuring the marked state is a constant.

*Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 2T2, Canada, e-mail: ambainis@math.uwaterloo.ca. Parts of this research done at University of Latvia, University of California, Berkeley and Institute for Advanced Study, Princeton. Supported by Latvia Science Council Grant 01.0354 (at University of Latvia), DARPA and Air Force Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-01-2-0524 (at UC Berkeley), NSF Grant DMS-0111298 (at IAS), NSERC, ARDA, IQC University Professorship and CIAR (at University of Waterloo).

We also give several extensions of our algorithm. If we have to find whether $x_1, \ldots, x_N$ contain $k$ numbers that are equal: $x_{i_1} = \ldots = x_{i_k}$, we get a quantum algorithm with $O(N^{k/(k+1)})$ queries for any constant[1] $k$.

If the quantum algorithm is restricted to storing $r$ numbers, $r \leq N^{2/3}$, then we have an algorithm which solves element distinctness with $O(N/\sqrt{r})$ queries which is quadratically better than the classical $O(N^2/r)$ query algorithm. Previously, such quantum algorithm was known only for $r \leq \sqrt{N}$ [14]. For the problem of finding $k$ equal numbers, we get an algorithm that uses $O(\frac{N^{k/2}}{r^{(k-1)/2}})$ queries and stores $r$ numbers, for $r \leq N^{(k-1)/k}$.

For the analysis of our algorithm, we develop a generalization of Grover's algorithm (Lemma 3) which might be of independent interest.

### 1.1 Related work

**Classical element distinctness.** Element distinctness has been extensively studied classically. It can be solved with $O(N)$ queries and $O(N \log N)$ time by querying all the elements and sorting them. Then, any two equal elements must be next one to another in the sorted order and can be found by going through the sorted list.

In the usual query model (where one query gives one value of $x_i$), it is easy to see that $\Omega(N)$ queries are also necessary. Classical lower bounds have also been shown for more general models (e.g. [25]).

The algorithm described above requires $\Omega(N)$ space to store all of $x_1, \ldots, x_N$. If we are restricted to space $S < N$, the running time increases. The straightforward algorithm needs $O(\frac{N^2}{S})$ queries. Yao [38] has shown that, for the model of comparison-based branching programs, this is essentially optimal. Namely, any space-$S$ algorithm needs time $T = \Omega(\frac{N^{2-o(1)}}{S})$. For more general models, lower bounds on algorithms with restricted space $S$ is an object of ongoing research [10].

**Related problems in quantum computing.** In *collision problem*, we are given a 2-1 function $f$ and have to find $x, y$ such that $f(x) = f(y)$. As shown by Brassard, Høyer and Tapp [17], collision problem can be solved in $O(N^{1/3})$ quantum steps instead of $\Theta(N^{1/2})$ steps classically. $\Omega(N^{1/3})$ is also a quantum lower bound [1, 31].

If element distinctness can be solved with $M$ queries, then collision problem can be solved with $O(\sqrt{M})$ queries. (This connection is credited to Andrew Yao in [1].) Thus, a quantum algorithm for element distinctness implies a quantum algorithm for collision but not the other way around.

**Quantum search on graphs.** The idea of quantum search on graphs was proposed by Aaronson and Ambainis [2] for finding a marked item on a $d$-dimensional grid (problem first considered by Benioff [12]) and other graphs with good expansion properties. Our work has a similar flavor but uses completely different methods to search the graph (quantum walk instead of "divide-and-conquer").

**Quantum walks.** There has been considerable amount of research on quantum walks (surveyed in [30]) and their applications (surveyed in [6]). Applications of walks [6] mostly fall into two classes. The first class is exponentially faster hitting times [21, 19, 29]. The second class is quantum walk search algorithms [36, 22, 8].

Our algorithm is most closely related to the second class. In this direction, Shenvi et al. [36] have constructed a counterpart of Grover's search [26] based on quantum walk on the hypercube. Childs and

[1]The big-O constant depends on $k$. For non-constant $k$, we can show that the number of queries is $O(k^2 N^{k/(k+1)})$. The proof of that is mostly technical and is omitted in this version.

Goldstone [22, 23] and Ambainis et al. [8] have used quantum walk to produce search algorithms on $d$-dimensional lattices ($d \geq 2$) which is faster than the naive application of Grover's search. This direction is quite closely related to our work. The algorithms by [36, 22, 8] and current paper solve different problems but all have similar structure.

**Recent developments.** After the work described in this paper, the results and ideas from this paper have been used to construct several other quantum algorithms. Magniez et al. [32] have used our element distinctness algorithm to give an $O(n^{1.3})$ query quantum algorithm for finding triangles in a graph. Ambainis et al. [8] have used ideas from the current paper to construct a faster algorithm for search on 2-dimensional grid. Childs and Eisenberg [20] have given a different analysis of our algorithm.

Szegedy [37] has generalized our results on quantum walk for element distinctness to an arbitrary graph with a large eigenvalue gap and cast them into the language of Markov chains. His main result is that, for a class of Markov chains, quantum walk algorithms are quadratically faster than the corresponding classical algorithm. An advantage of Szegedy's approach is that it can simultaneously handle any number of solutions (unlike in the present paper which has separate algorithms for single solution case (algorithm 2) and multiple-solution case (algorithm 3)).

Buhrman and Spalek [15] have used Szegedy's result to construct an $O(n^{5/3})$ quantum algorithm for verifying if a product of two $n \times n$ matrices $A$ and $B$ is equal to a third matrix $C$.

# 2 Preliminaries

## 2.1 Quantum query algorithms

Let $[N]$ denote $\{1, \ldots, N\}$. We consider

**Element Distinctness.** Given numbers $x_1, \ldots, x_N \in [M]$, are there $i, j \in [N]$, $i \neq j$ such that $x_i = x_j$?

Element distinctness is a particular case of

**Element $k$-distinctness.** Given numbers $x_1, \ldots, x_N \in [M]$, are there $k$ distinct indices $i_1, \ldots, i_k \in [N]$ such that $x_{i_1} = x_{i_2} = \ldots = x_{i_k}$?

We call such $k$ indices $i_1, \ldots, i_k$ a *$k$-collision*.

Our model is the quantum query model (for surveys on query model, see [7, 18]). In this model, our goal is to compute a function $f(x_1, \ldots, x_N)$. For example, $k$-distinctness is viewed as the function $f(x_1, \ldots, x_N)$ which is 1 if there exists a $k$-collision consisting of $i_1, \ldots, i_k \in [N]$ and 0 otherwise.

The input variables $x_i$ can be accessed by queries to an oracle $X$ and the complexity of $f$ is the number of queries needed to compute $f$. A quantum computation with $T$ queries is just a sequence of unitary transformations

$$U_0 \to O \to U_1 \to O \to \ldots \to U_{T-1} \to O \to U_T.$$

$U_j$'s can be arbitrary unitary transformations that do not depend on the input bits $x_1, \ldots, x_N$. $O$ are query (oracle) transformations. To define $O$, we represent basis states as $|i, a, z\rangle$ where $i$ consists of $\lceil \log N \rceil$ bits, $a$ consists of $\lceil \log M \rceil$ quantum bits and $z$ consists of all other bits. Then, $O$ maps $|i, a, z\rangle$ to $|i, (a + x_i) \bmod M, z\rangle$.

In our algorithm, we use queries in two situations. The first situation is when $a = |0\rangle$. Then, the state before the query is some superposition $\sum_{i,z} \alpha_{i,z} |i, 0, z\rangle$ and the state after the query is the same superposition with the information about $x_i$: $\sum_{i,z} \alpha_{i,z} |i, x_i, z\rangle$. The second situation is when the state before the

query is $\sum_{i,z} \alpha_{i,z}|i, -x_i \bmod M, z\rangle$ with the information about $x_i$ from a previous query. Then, applying the query transformation makes the state $\sum_{i,z} \alpha_{i,z}|i, 0, z\rangle$, erasing the information about $x_i$. This can be used to erase the information about $x_i$ from $\sum_{i,z} \alpha_{i,z}|i, x_i, z\rangle$. We first perform a unitary that maps $|x_i\rangle \rightarrow |-x_i \bmod M\rangle$, obtaining the state $\sum_{i,z} \alpha_{i,z}|i, -x_i \bmod M, z\rangle$ and then apply the query transformation.

The computation starts with a state $|0\rangle$. Then, we apply $U_0$, $O$, ..., $O$, $U_T$ and measure the final state. The result of the computation is the rightmost bit of the state obtained by the measurement.

We say that the quantum computation computes $f$ with bounded error if, for every $x = (x_1, \ldots, x_N)$, the probability that the rightmost bit of $U_T O_x U_{T-1} \ldots O_x U_0|0\rangle$ equals $f(x_1, \ldots, x_N)$ is at least $1 - \epsilon$ for some fixed $\epsilon < 1/2$.

To simplify the exposition, we occasionally describe a quantum computation as a classical algorithm with several quantum subroutines of the form $U_t O_x U_{t-1} \ldots O_x U_0|0\rangle$. Any such classical algorithm with quantum subroutines can be transformed into an equivalent sequence $U_T O_x U_{T-1} \ldots O_x U_0|0\rangle$ with the number of queries being equal to the number of queries in the classical algorithm plus the sum of numbers of queries in all quantum subroutines.

**Comparison oracle.** In a different version of query model, we are only allowed comparison queries. In a comparison query, we give two indices $i, j$ to the oracle. The oracle answers whether $x_i < x_j$ or $x_i \geq x_j$. In the quantum model, we can query the comparison oracle with a superposition $\sum_{i,j,z} a_{i,j,z}|i, j, z\rangle$, where $i, j$ are the indices being queried and $z$ is the rest of quantum state. The oracle then performs a unitary transformation $|i, j, z\rangle \rightarrow -|i, j, z\rangle$ for all $i, j, z$ such that $x_i < x_j$ and $|i, j, z\rangle \rightarrow |i, j, z\rangle$ for all $i, j, z$ such that $x_i \geq x_j$. In section 6, we show that our algorithms can be adapted to this model with a logarithmic increase in the number of queries.

## 2.2 $d$-wise independence

To make our algorithms efficient in terms of running time and, in the case of multiple-solution algorithm in section 5, also space, we use $d$-wise independent functions. A reader who is only interested in the query complexity of the algorithms may skip this subsection.

**Definition 1** *Let $\mathcal{F}$ be a family of functions $f : [N] \rightarrow \{0, 1\}$. $\mathcal{F}$ is $d$-wise independent if, for all $d$-tuples of pairwise distinct $i_1, \ldots, i_d \in [N]$ and all $c_1, \ldots, c_d \in \{0, 1\}$,*

$$Pr[f(i_1) = c_1, f(i_2) = c_2, \ldots, f(i_d) = c_d] = \frac{1}{2^d}.$$

**Theorem 1** *[4] There exists a $d$-wise independent family $\mathcal{F} = \{f_j | j \in [R]\}$ of functions $f_j : [N] \rightarrow \{0, 1\}$ such that:*

1. *$R = O(N^{\lceil d/2 \rceil})$;*
2. *$f_j(i)$ is computable in $O(d \log^2 N)$ time, given $j$ and $i$.*

We will also use families of permutations with a similar properties. It is not known how to construct small $d$-wise independent families of permutations. There are, however, constructions of approximately $d$-wise independent families of permutations.

**Definition 2** *Let $\mathcal{F}$ be a family of permutations on $f : [n] \to [n]$. $\mathcal{F}$ is $\epsilon$-approximately $d$-wise independent if, for all $d$-tuples of pairwise distinct $i_1, \ldots, i_d \in [n]$ and pairwise distinct $j_1, \ldots, j_d \in [n]$,*

$$Pr[f(i_1) = j_1, f(i_2) = j_2, \ldots, f(i_d) = j_d] \in \left[\frac{1-\epsilon}{n(n-1)\ldots(n-d+1)}, \frac{1+\epsilon}{n(n-1)\ldots(n-d+1)}\right].$$

**Theorem 2** *[28] Let $n$ be an even power of a prime number. For any $d \leq n$, $\epsilon > 0$, there exists an $\epsilon$-approximate $d$-wise independent family $\mathcal{F} = \{\pi_j | j \in [R]\}$ of permutations $\pi_j : [n] \to [n]$ such that:*

1. *$R = O((n^{d^2}/\epsilon^d)^{3+o(1)})$;*
2. *$\pi_j(i)$ is computable in $O(d \log^2 n)$ time, given $j$ and $i$.*

# 3 Results and algorithms

Our main results are

**Theorem 3** *Element $k$-distinctness can be solved by a quantum algorithm with $O(N^{k/(k+1)})$ queries. In particular, element distinctness can be solved by a quantum algorithm with $O(N^{2/3})$ queries.*

**Theorem 4** *Let $r \geq k$, $r = o(N)$. There is a quantum algorithm that solves element distinctness with $O(\max(\frac{N}{\sqrt{r}}, r))$ queries and and $k$-distinctness with $O(\max(\frac{N^{k/2}}{r^{(k-1)/2}}, r))$ queries, using $O(r(\log M + \log N))$ qubits of memory.*

Theorem 3 follows from Theorem 4 by setting $r = \lfloor N^{2/3} \rfloor$ for element distinctness and $r = \lfloor N^{k/(k+1)} \rfloor$ for $k$-distinctness. (These values minimize the expressions for the number of queries in Theorem 4.)

Next, we present Algorithms 2 which solves element distinctness if we have a promise that $x_1, \ldots, x_N$ are either all distinct or there is exactly one pair $i, j$, $i \neq j$, $x_i = x_j$ (and $k$-distinctness if we have a promise that there is at most one set of $k$ indices $i_1, \ldots, i_k$ such that $x_{i_1} = x_{i_2} = \ldots = x_{i_k}$). The proof of correctness of algorithm 2 is given in section 4. After that, in section 5, we present Algorithm 3 which solves the general case, using Algorithm 2 as a subroutine.

## 3.1 Main ideas

We start with an informal description of main ideas. For simplicity, we restrict to element distinctness and postpone the more general $k$-distinctness till the end of this subsection.

Let $r = N^{2/3}$. We define a graph $G$ with $\binom{N}{r} + \binom{N}{r+1}$ vertices. The vertices $v_S$ correspond to sets $S \subseteq [N]$ of size $r$ and $r+1$. Two vertices $v_S$ and $v_T$ are connected by an edge if $T = S \cup \{i\}$ for some $i \in [N]$. A vertex is marked if $S$ contains $i, j$, $x_i = x_j$.

Element distinctness reduces to finding a marked vertex in this graph. If we find a marked vertex $v_S$, then we know that $x_i = x_j$ for some $i, j \in S$, i.e. $x_1, \ldots, x_N$ are not all distinct.

The naive way to find a marked vertex would be to use Grover's quantum search algorithm [26, 16]. If $\epsilon$ fraction of vertices are marked, then Grover's search finds a marked vertex after $O(\frac{1}{\sqrt{\epsilon}})$ vertices. Assume

that there exists a single pair $i, j \in [N]$ such that $i \neq j$, $x_i = x_j$. For a random $S$, $|S| = N^{2/3}$, the probability of $v_S$ being marked is

$$Pr[i \in S; j \in S] = Pr[i \in S]Pr[j \in S | i \in S] = \frac{N^{2/3}}{N}\frac{N^{2/3}-1}{N-1} = (1-o(1))\frac{1}{N^{2/3}}.$$

Thus, a quantum algorithm can find a marked vertex by examining $O(\frac{1}{\sqrt{\epsilon}}) = O(N^{1/3})$ vertices. However, to find out if a vertex is marked, the algorithm needs to query $N^{2/3}$ items $x_i$, $i \in S$. This makes the total query complexity $O(N^{1/3}N^{2/3}) = O(N)$, giving no speedup compared to the classical algorithm which queries all items.

We improve on this naive algorithm by re-using the information from previous queries. Assume that we just checked if $v_S$ is marked by querying all $x_i$, $i \in S$. If the next vertex $v_T$ is such that $T$ contains only $m$ elements $i \notin S$, then we only need to query $m$ elements $x_i$, $i \in T \setminus S$ instead of $r = N^{2/3}$ elements $x_i$, $i \in T$.

To formalize this, we use the following model. At each moment, we are at one vertex of $G$ (superposition of vertices in quantum case). In one time step, we can examine if the current vertex $v_S$ is marked and move to an adjacent vertex $v_T$. Assume that there is an algorithm $A$ that finds a marked vertex with $M$ moves between vertices. Then, there is an algorithm that solves element distinctness in $M + r$ steps, in a following way:

1. We use $r$ queries to query all $x_i$, $i \in S$ for the starting vertex $v_S$.
2. We then repeat the following two operations $M$ times:
   (a) Check if the current vertex $v_S$ is marked. This can be done without any queries because we already know all $x_i$, $i \in S$.
   (b) We simulate the algorithm $A$ until the next move, find the vertex $v_T$ to which it moves from $v_S$. We then move to $v_T$, by querying $x_i$, $i \in T \setminus S$. After that, we know all $x_i, i \in T$. We then set $S = T$.

The total number of queries is at most $M + r$, consisting of $r$ queries for the first step and 1 query to simulate each move of $A$.

In the next sections, we will show how to search this graph by quantum walk in $O(N^{2/3})$ steps for element distinctness and $O(N^{k/(k+1)})$ steps for $k$-distinctness.

### 3.2 The algorithm

Let $x_1, \ldots, x_N \in [M]$. We consider two Hilbert spaces $\mathcal{H}$ and $\mathcal{H}'$. $\mathcal{H}$ has dimension $\binom{N}{r}M^r(N-r)$ and the basis states of $\mathcal{H}$ are $|S, x, y\rangle$ with $S \subseteq [N]$, $|S| = r$, $x \in [M]^r$, $y \in [N] \setminus S$. $\mathcal{H}'$ has dimension $\binom{N}{r+1}M^{r+1}(r+1)$. The basis states of $\mathcal{H}'$ are $|S, x, y\rangle$ with $S \subseteq [N]$, $|S| = r+1$, $x \in [M]^{r+1}$, $y \in S$. Our algorithm thus uses

$$O\left(\binom{N}{r}M^r(N-r) + \binom{N}{r+1}M^{r+1}(r+1)\right) = O(r(\log N + \log M))$$

qubits of memory.

1. Apply the transformation mapping $|S\rangle|y\rangle$ to

$$|S\rangle \left( \left( -1 + \frac{2}{N-r} \right) |y\rangle + \frac{2}{N-r} \sum_{y' \notin S, y' \neq y} |y'\rangle \right).$$

   on the $S$ and $y$ registers of the state in $\mathcal{H}$. (This transformation is a variant of "diffusion transformation" in [26].)

2. Map the state from $\mathcal{H}$ to $\mathcal{H}'$ by adding $y$ to $S$ and changing $x$ to a vector of length $k+1$ by introducing 0 in the location corresponding to $y$:

3. Query for $x_y$ and insert it into location of $x$ corresponding to $y$.

4. Apply the transformation mapping $|S\rangle|y\rangle$ to

$$|S\rangle \left( \left( -1 + \frac{2}{r+1} \right) |y\rangle + \frac{2}{r+1} \sum_{y' \in S, y' \neq y} |y'\rangle \right).$$

   on the $y$ register.

5. Erase the element of $x$ corresponding to new $y$ by using it as the input to query for $x_y$.

6. Map the state back to $\mathcal{H}$ by removing the 0 component corresponding to $y$ from $x$ and removing $y$ from $S$.

Algorithm 1: One step of quantum walk

In the states used by our algorithm, $x$ will always be equal to $(x_{i_1}, \ldots, x_{i_r})$ where $i_1, \ldots, i_r$ are elements of $S$ in increasing order.

We start by defining a quantum walk on $\mathcal{H}$ and $\mathcal{H}'$ (algorithm 1). Each step of the quantum walk starts in a superposition of states in $\mathcal{H}$. The first three steps map the state from $\mathcal{H}$ to $\mathcal{H}'$ and the last three steps map it back to $\mathcal{H}$.

If there is at most one $k$-collision, we apply Algorithm 2 ($t_1$ and $t_2$ are $c_1\sqrt{r}$ and $c_2(\frac{N}{r})^{k/2}$ for constants $c_1$ and $c_2$ which can be calculated from the analysis in section 4). This algorithm alternates quantum walk with a transformation that changes the phase if the current state contains a $k$-collision. We give a proof of correctness for Algorithm 2 in section 4.

If there can be more one $k$-collision, element $k$-distinctness is solved by algorithm 3. Algorithm 3 is a classical algorithm that randomly selects several subsets of $x_i$ and runs algorithm 2 on each subset. We give Algorithm 3 and its analysis in section 5.

1. Generate the uniform superposition $\frac{1}{\sqrt{\binom{N}{r}(N-r)}} \sum_{|S|=r, y \notin S} |S\rangle|y\rangle$.
2. Query all $x_i$ for $i \in S$. This transforms the state to
$$\frac{1}{\sqrt{\binom{N}{r}(N-r)}} \sum_{|S|=r, y \notin S} |S\rangle|y\rangle \bigotimes_{i \in S} |x_i\rangle.$$
3. $t_1 = O((N/r)^{k/2})$ times repeat:
   (a) Apply the conditional phase flip (the transformation $|S\rangle|y\rangle|x\rangle \rightarrow -|S\rangle|y\rangle|x\rangle$) for $S$ such that $x_{i_1} = x_{i_2} = \ldots = x_{i_k}$ for $k$ distinct $i_1, \ldots, i_k \in S$.
   (b) Perform $t_2 = O(\sqrt{r})$ steps of the quantum walk (algorithm 1).
4. Measure the final state. Check if $S$ contains a $k$-collision and answer "there is a $k$-collision" or "there is no $k$-collision", according to the result.

Algorithm 2: Single-solution algorithm

# 4 Analysis of single $k$-collision algorithm

## 4.1 Overview

The number of queries for algorithm 2 is $r$ for creating the initial state and $O((N/r)^{k/2}\sqrt{r}) = O(\frac{N^{k/2}}{r^{(k-1)/2}})$ for the rest of the algorithm. Thus, the overall number of queries is $O(\max(r, \frac{N^{k/2}}{r^{(k-1)/2}}))$. The correctness of algorithm 2 follows from

**Theorem 5** *Let the input $x_1$, ..., $x_N$ be such that $x_{i_1} = \ldots = x_{i_k}$ for exactly one set of $k$ distinct values $i_1, \ldots, i_k$. With a constant probability, measuring the final state of algorithm 2 gives $S$ such that $i_1, \ldots, i_k \in S$.*

**Proof:** The main ideas are as follows. We first show (Lemma 1) that algorithm's state always stays in a $2k+1$-dimensional subspace of $\mathcal{H}$. After that (Lemma 2), we find the eigenvalues for the unitary transformation induced by one step of the quantum walk (algorithm 1), restricted to this subspace. We then look at algorithm 2 as a sequence of the form $(U_2U_1)^{t_1}$ with $U_1$ being a conditional phase flip and $U_2$ being a unitary transformation whose eigenvalues have certain properties (in this case, $U_2$ is $t_2$ steps of quantum walk). We then prove a general result (Lemma 3) about such sequences, which implies that the algorithm finds the $k$-collision with a constant probability.

Let $|S, y\rangle$ be a shortcut for the basis state $|S\rangle \otimes_{i \in S} |x_i\rangle|y\rangle$. In our algorithm, the $|x\rangle$ register of a state $|S, x, y\rangle$ always contains the state $\otimes_{i \in S}|x_i\rangle$. Therefore, the state of the algorithm is always a linear combination of the basis states $|S, y\rangle$.

We classify the basis states $|S, y\rangle$ ($|S| = r$, $y \notin S$) into $2k+1$ types. A state $|S, y\rangle$ is of type $(j, 0)$ if $|S \cap \{i_1, \ldots, i_k\}| = j$ and $y \notin \{i_1, \ldots, i_k\}$ and of type $(j, 1)$ if $|S \cap \{i_1, \ldots, i_k\}| = j$ and $y \in \{i_1, \ldots, i_k\}$.

For $j \in \{0, \ldots, k-1\}$, there are both type $(j,0)$ and type $(j,1)$ states. For $j = k$, there are only $(k,0)$ type states. ($(k,1)$ type is impossible because, if, $|S \cap \{i_1, \ldots, i_k\}| = k$, then $y \notin S$ implies $y \notin \{i_1, \ldots, i_k\}$.)

Let $|\psi_{j,l}\rangle$ be the uniform superposition of basis states $|S, y\rangle$ of type $(j,l)$. Let $\tilde{H}$ be the $(2k+1)$-dimensional space spanned by states $|\psi_{j,l}\rangle$.

For the space $\mathcal{H}'$, its basis states $|S, y\rangle$ ($|S| = r+1$, $y \in S$) can be similarly classified into $2k+1$ types. We denote those types $(j,l)$ with $j = |S \cap \{i_1, \ldots, i_k\}|$, $l = 1$ if $y \in \{i_1, \ldots, i_k\}$ and $l = 0$ otherwise. (Notice that, since $y \in S$ for the space $\mathcal{H}'$, we have type $(k,1)$ but no type $(0,1)$.) Let $|\varphi_{j,l}\rangle$ be the uniform superposition of basis states $|S, y\rangle$ of type $(j,l)$ for space $\mathcal{H}'$. Let $\tilde{H}'$ be the $(2k+1)$-dimensional space spanned by $|\varphi_{j,l}\rangle$. Notice that the transformation $|S, y\rangle \to |S \cup \{y\}, y\rangle$ maps

$$|\psi_{i,0}\rangle \to |\varphi_{i,0}\rangle,\ |\psi_{i,1}\rangle \to |\varphi_{i+1,1}\rangle.$$

We claim

**Lemma 1** *In algorithm 1, steps 1-3 map $\tilde{\mathcal{H}}$ to $\tilde{\mathcal{H}}'$ and steps 4-6 map $\tilde{\mathcal{H}}'$ to $\tilde{\mathcal{H}}$.*

**Proof:** In section 4.2. ∎

Thus, algorithm 1 maps $\tilde{\mathcal{H}}$ to itself. Also, in algorithm 2, step 3a maps $|\psi_{k,0}\rangle \to -|\psi_{k,0}\rangle$ and leaves $|\psi_{j,l}\rangle$ for $j < k$ unchanged (because $|\psi_{j,l}\rangle$, $j < k$ are superpositions of states $|S, y\rangle$ which are unchanged by step 3b and $|\psi_{k,0}\rangle$ is a superposition of states $|S, y\rangle$ which are mapped to $-|S, y\rangle$ by step 3b). Thus, every step of algorithm 2 maps $\tilde{\mathcal{H}}$ to itself. Also, the starting state of algorithm 2 can be expressed as a combination of $|\psi_{j,l}\rangle$. Therefore, it suffices to analyze algorithms 1 and 2 on subspace $\tilde{\mathcal{H}}$.

In this subspace, we will be interested in two particular states. Let $|\psi_{start}\rangle$ be the uniform superposition of all $|S, y\rangle$, $|S| = r$, $y \notin S$. Let $|\psi_{good}\rangle = |\psi_{k,0}\rangle$ be the uniform superposition of all $|S, y\rangle$ with $i_1, \ldots, i_k \in S$. $|\psi_{start}\rangle$ is the algorithm's starting state. $|\psi_{good}\rangle$ is the state we would like to obtain (because measuring $|\psi_{good}\rangle$ gives a random set $S$ such that $\{i_1, \ldots, i_k\} \subseteq S$).

We start by analyzing a single step of quantum walk.

**Lemma 2** *Let $U$ be the unitary transformation induced on $\tilde{\mathcal{H}}$ by one step of the quantum walk (algorithm 1). $U$ has $2k+1$ different eigenvalues in $\tilde{\mathcal{H}}$. One of them is 1, with $|\psi_{start}\rangle$ being the eigenvector. The other eigenvalues are $e^{\pm\theta_1 i}$, …, $e^{\pm\theta_k i}$ with $\theta_j = (2\sqrt{j} + o(1))\frac{1}{\sqrt{r}}$.*

**Proof:** In section 4.2. ∎

We set $t_2 = \lceil \frac{\pi}{3\sqrt{k}}\sqrt{r} \rceil$. Since one step of quantum walk fixes $\tilde{\mathcal{H}}$, $t_2$ steps fix $\tilde{\mathcal{H}}$ as well. Moreover, $|\psi_{start}\rangle$ will still be an eigenvector with eigenvalue 1. The other $2k$ eigenvalues become $e^{\pm i(\frac{2\pi\sqrt{j}}{3\sqrt{k}} + o(1))}$. Thus, every of those eigenvalues is $e^{i\theta}$ with $\theta \in [c, 2\pi - c]$, for a constant $c$ independent of $N$ and $r$.

Let step $U_1$ be step 3a of algorithm 2 and $U_2 = U^{t_2}$ be step 3b. Then, the entire algorithm consists of applying $(U_2 U_1)^{t_1}$ to $|\psi_{start}\rangle$. We will apply

**Lemma 3** *Let $\mathcal{H}$ be a finite dimensional Hilbert space and $|\psi_1\rangle$, …, $|\psi_m\rangle$ be an orthonormal basis for $\mathcal{H}$. Let $|\psi_{good}\rangle$, $|\psi_{start}\rangle$ be two states in $\mathcal{H}$ which are superpositions of $|\psi_1\rangle$, …, $|\psi_m\rangle$ with real amplitudes and $\langle\psi_{good}|\psi_{start}\rangle = \alpha$. Let $U_1$, $U_2$ be unitary transformations on $\mathcal{H}$ with the following properties:*

1. *$U_1$ is the transformation that flips the phase on $|\psi_{good}\rangle$ ($U_1|\psi_{good}\rangle = -|\psi_{good}\rangle$) and leaves any state orthogonal to $|\psi_{good}\rangle$ unchanged.*

2. *$U_2$ is a transformation which is described by a real-valued $m \times m$ matrix in the basis $|\psi_1\rangle, \ldots, |\psi_m\rangle$. Moreover, $U_2|\psi_{start}\rangle = |\psi_{start}\rangle$ and, if $|\psi\rangle$ is an eigenvector of $U_2$ perpendicular to $|\psi_{start}\rangle$, then $U_2|\psi\rangle = e^{i\theta}|\psi\rangle$ for $\theta \in [\epsilon, 2\pi - \epsilon]$, $\theta \neq \pi$ (where $\epsilon$ is a constant, $\epsilon > 0$)*[2]

*Then, there exists $t = O(\frac{1}{\alpha})$ such that $|\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle| = \Omega(1)$. (The constant under $\Omega(1)$ is independent of $\alpha$ but can depend on $\epsilon$.)*

**Proof:** In section 4.3. ∎

By Lemma 3, we can set $t_1 = O(\frac{1}{\alpha})$ so that the inner product of $(U_2U_1)^{t_1}|\psi_{start}\rangle$ and $|\psi_{good}\rangle$ is a constant. Since $|\psi_{good}\rangle$ is a superposition of $|S, y\rangle$ over $S$ satisfying $\{i_1, \ldots, i_k\} \subseteq S$, measuring $(U_2U_1)^{t_1}|\psi_{start}\rangle$ gives a set $S$ satisfying $\{i_1, \ldots, i_k\} \subseteq S$ with a constant probability.

It remains to calculate $\alpha$. Let $\alpha'$ be the fraction of $S$ satisfying $\{i_1, \ldots, i_k\} \subseteq S$. Since $|\psi_{start}\rangle$ is the uniform superposition of all $|S, y\rangle$ and $|\psi_{good}\rangle$ is the uniform superposition of $|S, y\rangle$ with $\{i_1, \ldots, i_k\} \subseteq S$ we have $\alpha = \sqrt{\alpha'}$.

$$\alpha' = Pr[\{i_1, \ldots, i_k\} \subseteq S] = \frac{\binom{N-k}{r-k}}{\binom{N}{r}} = \frac{r}{N}\prod_{j=1}^{k-1}\frac{r-j}{N-j} = (1 - o(1))\frac{r^k}{N^k}.$$

Therefore, $\alpha = \Omega(\frac{r^{k/2}}{N^{k/2}})$ and $t_1 = O((N/r)^{k/2})$. ∎

Lemma 3 might also be interesting by itself. It generalizes one of analyses of Grover's algorithm [3]. Informally, the lemma says that, in Grover-like sequence of transformations $(U_2U_1)^t$, we can significantly relax the constraints on $U_2$ and the algorithm will still give similar result. It is quite likely that such situations might appear in analysis of other algorithms.

For the quantum walk for element $k$-distinctness, Childs and Eisenberg [20] have improved the analysis of lemma 3, by showing that $\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle$ (and, hence, algorithm's success probability) is $1-o(1)$. Their result, however, does not apply to arbitrary transformations $U_1$ and $U_2$ satisfying conditions of lemma 3.

### 4.2 Proofs of Lemmas 1 and 2

**Proof:** [of Lemma 1] To show that $\tilde{\mathcal{H}}$ is mapped to $\tilde{\mathcal{H}}'$, it suffices to show that each of basis vectors $|\psi_{j,l}\rangle$ is mapped to a vector in $\tilde{\mathcal{H}}'$. Consider vectors $|\psi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$ for $j \in \{0, 1, \ldots, k-1\}$. Fix $S$, $|S \cap \{i_1, \ldots, i_k\}| = j$. We divide $[N] \setminus S$ into two sets $S_0$ and $S_1$. Let

$$S_0 = \{y : y \in [N] \setminus S, y \notin \{i_1, \ldots, i_k\}\},$$
$$S_1 = \{y : y \in [N] \setminus S, y \in \{i_1, \ldots, i_k\}\}.$$

Since $|S \cap \{i_1, \ldots, i_k\}| = j$, $S_1$ contains $s_1 = k - j$ elements. Since $S_0 \cup S_1 = [N] \setminus S$ contains $N - r$ elements, $S_0$ contains $s_0 = N - r - k + j$ elements. Define $|\psi_{S,0}\rangle = \frac{1}{\sqrt{N-r-k+j}}\sum_{y\in S_0}|S, y\rangle$ and $|\psi_{S,1}\rangle = \frac{1}{\sqrt{k-j}}\sum_{y\in S_1}|S, y\rangle$. Then, we have

$$|\psi_{j,0}\rangle = \frac{1}{\sqrt{\binom{k}{j}\binom{N-k}{r-j}}} \sum_{\substack{S:|S|=r \\ |S\cap\{i_1,\ldots,i_k\}|=j}} |\psi_{S,0}\rangle \qquad (1)$$

[2]The requirement $\theta \neq \pi$ is made to simplify the proof of the lemma. The lemma remains true if $\theta = \pi$ is allowed. At the end of section 4.3, we sketch how to modify the proof for this case.

and, similarly for $|\psi_{j,1}\rangle$ and $|\psi_{S,1}\rangle$.

Consider the step 1 of algorithm 1, applied to the state $|\psi_{S,0}\rangle$. Let $|\psi'_{S,0}\rangle$ be the resulting state. Since the $|S\rangle$ register is unchanged, $|\psi'_{S,0}\rangle$ is some superposition of states $|S,y\rangle$. Moreover, both the state $|\psi_{S,0}\rangle$ and the transformation applied to this state in step 1 are invariant under permutation of states $|S,y\rangle$, $y \in S_0$ or states $|S,y\rangle$, $y \in S_1$. Therefore, the resulting state must be invariant under such permutations as well. This means that every $|S,y\rangle$, $y \in S_0$ and every $|S,y\rangle$, $y \in S_1$ has the same amplitude in $|\psi'_{S,0}\rangle$. This is equivalent to $|\psi'_{S,0}\rangle = a|\psi_{S,0}\rangle + b|\psi_{S,1}\rangle$ for some $a$, $b$. Because of equation (1), this means that step 1 maps $|\psi_{j,0}\rangle$ to $a|\psi_{j,0}\rangle + b|\psi_{j,1}\rangle$. Steps 2 and 3 then map $|\psi_{j,0}\rangle$ to $|\varphi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$ to $|\varphi_{j+1,1}\rangle$. Thus, $|\psi_{j,0}\rangle$ is mapped to a superposition of two basis states of $\tilde{\mathcal{H}}'$: $|\varphi_{j,0}\rangle$ and $|\varphi_{j+1,1}\rangle$. Similarly, $|\psi_{j,1}\rangle$ is mapped to a (different) superposition of those two states.

For $j = k$, we only have one state $|\psi_{k,0}\rangle$. A similar argument shows that this state is unchanged by step 1 and then mapped to $|\varphi_{k,0}\rangle$ which belongs to $\tilde{\mathcal{H}}'$.

Thus, steps 1-3 map $\tilde{\mathcal{H}}$ to $\tilde{\mathcal{H}}'$. The proof that steps 4-6 map $\tilde{\mathcal{H}}'$ to $\tilde{\mathcal{H}}$ is similar. ∎

**Proof:** [of Lemma 2] We fix a basis for $\tilde{\mathcal{H}}$ consisting of $|\psi_{j,0}\rangle$, $|\psi_{j,1}\rangle$, $j \in \{0,\ldots,k-1\}$ and $|\psi_{k,0}\rangle$ and a basis for $\tilde{\mathcal{H}}'$ consisting of $|\varphi_{0,0}\rangle$ and $|\varphi_{j,1}\rangle$, $|\varphi_{j,0}\rangle$, $j \in \{1,\ldots,k\}$. Let $D_\epsilon$ be the matrix

$$D_\epsilon = \begin{pmatrix} -1+2\epsilon & 2\sqrt{\epsilon-\epsilon^2} \\ 2\sqrt{\epsilon-\epsilon^2} & 1-2\epsilon \end{pmatrix}.$$

**Claim 1** *Let $U_1$ be the unitary transformation mapping $\tilde{\mathcal{H}}$ to $\tilde{\mathcal{H}}'$ induced by steps 1-3 of quantum walk. Then, $U_1$ is described by a block diagonal matrix*

$$U_1 = \begin{pmatrix} D_{\frac{k}{N-r}} & 0 & \ldots & 0 & 0 \\ 0 & D_{\frac{k-1}{N-r}} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & D_{\frac{1}{N-r}} & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{pmatrix},$$

*where the columns are in the basis $|\psi_{0,0}\rangle$, $|\psi_{0,1}\rangle$, $|\psi_{1,0}\rangle$, $|\psi_{1,1}\rangle$, …, $|\psi_{k,0}\rangle$ and the rows are in the basis $|\varphi_{0,0}\rangle$, $|\varphi_{1,1}\rangle$, $|\varphi_{1,0}\rangle$, $|\varphi_{2,1}\rangle$, …, $|\varphi_{k,1}\rangle$, $|\varphi_{k,0}\rangle$.*

**Proof:** Let $\mathcal{H}_j$ be the 2-dimensional subspace of $\tilde{\mathcal{H}}$ spanned by $|\psi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$. Let $\mathcal{H}'_j$ be the 2-dimensional subspace of $\tilde{\mathcal{H}}'$ spanned by $|\varphi_{j,0}\rangle$ and $|\varphi_{j+1,1}\rangle$.

From the proof of Lemma 1, we know that the subspace $\mathcal{H}_j$ is mapped to the subspace $\mathcal{H}'_j$. Thus, we have a block diagonal matrices with $2 \times 2$ blocks mapping $\mathcal{H}_j$ to $\mathcal{H}'_j$ and $1 \times 1$ identity matrix mapping $|\psi_{k,0}\rangle$ to $|\varphi_{k,0}\rangle$. It remains to show that the transformation from $\mathcal{H}_j$ to $\mathcal{H}'_j$ is $D_{\frac{k-j}{N-r}}$. Let $S$ be such that $|S \cap \{i_1,\ldots,i_k\}| = j$. Let $S_0, S_1$, $|\psi_{S,0}\rangle$, $|\psi_{S,1}\rangle$ be as in the proof of lemma 1. Then, step 1 of algorithm 1 maps $|\psi_{S,0}\rangle$ to

$$\frac{1}{\sqrt{s_0}} \sum_{y \in S_0} \left( \left( -1 + \frac{2}{N-r} \right) |S,y\rangle + \sum_{y' \neq y, y' \notin S} \frac{2}{N-r} |S,y'\rangle \right)$$

$$= \frac{1}{\sqrt{s_0}} \left( -1 + \frac{2}{N-r} + (s_0 - 1)\frac{2}{N-r} \right) \sum_{y \in S_0} |S,y\rangle + s_0 \frac{1}{\sqrt{s_0}} \frac{2}{N-r} \sum_{y \in S_1} |S,y\rangle$$

$$= \left(-1 + \frac{2s_0}{N-r}\right)|\psi_{S,0}\rangle + \frac{2\sqrt{s_0 s_1}}{N-r}|\psi_{S,1}\rangle.$$

By a similar calculation, $|\psi_{S,1}\rangle$ is mapped to

$$\left(-1 + \frac{2s_1}{N-r}\right)|\psi_{S,1}\rangle + \frac{2\sqrt{s_0 s_1}}{N-r}|\psi_{S,0}\rangle = \left(1 - \frac{2s_0}{N-r}\right)|\psi_{S,1}\rangle + \frac{2\sqrt{s_0 s_1}}{N-r}|\psi_{S,0}\rangle.$$

Thus, step 1 produces the transformation $D_{\frac{k-j}{N-r}}$ on $|\psi_{S,0}\rangle$ and $|\psi_{S,1}\rangle$. Since $|\psi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$ are uniform superpositions of $|\psi_{S,0}\rangle$ and $|\psi_{S,1}\rangle$ over all $S$, step 1 also produces the same transformation $D_{\frac{k-j}{N-r}}$ on $|\psi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$. Steps 2 and 3 just map $|\psi_{j,0}\rangle$ to $|\varphi_{j,0}\rangle$ and $|\psi_{j,1}\rangle$ to $|\varphi_{j+1,1}\rangle$. ∎

Similarly, steps 4-6 give the transformation $U_2$ described by block-diagonal matrix

$$U_2 = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & D'_{\frac{1}{r+1}} & 0 & \dots & 0 \\ 0 & 0 & D'_{\frac{2}{r+1}} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & D'_{\frac{k}{r+1}} \end{pmatrix}.$$

from $\tilde{\mathcal{H}}'$ to $\tilde{\mathcal{H}}$. Here, $D'_\epsilon$ denotes the matrix

$$D'_\epsilon = \begin{pmatrix} -1+2\epsilon & 2\sqrt{\epsilon-\epsilon^2} \\ 2\sqrt{\epsilon-\epsilon^2} & 1-2\epsilon \end{pmatrix}.$$

A step of quantum walk is $U = U_2 U_1$. Let $V$ be the diagonal matrix with odd entries on the diagonal being -1 and even entries being 1. Since $V^2 = I$, we have $U = U_2 V^2 U_1 = U'_2 U'_1$ for $U'_2 = U_2 V$ and $U'_1 = V U_1$. Let

$$E_\epsilon = \begin{pmatrix} 1-2\epsilon & 2\sqrt{\epsilon-\epsilon^2} \\ -2\sqrt{\epsilon-\epsilon^2} & 1-2\epsilon \end{pmatrix}.$$

Then, $U'_1$ and $U'_2$ are equal to $U_1$ and $U_2$, with every $D_\epsilon$ or $D'_\epsilon$ replaced by corresponding $E_\epsilon$. 7We will first diagonalize $U'_1$ and $U'_2$ separately and then argue that eigenvalues of $U'_2 U'_1$ are almost the same as eigenvalues of $U'_2$.

Since $U'_2$ is block diagonal, it suffices to diagonalize each block. $1 \times 1$ identity block has eigenvalue 1. For a matrix $E_\epsilon$, its characteristic polynomial is $\lambda^2 - (2-4\epsilon)\lambda + 1 = 0$ and its roots are $1 - 2\epsilon \pm 2\sqrt{\epsilon - \epsilon^2} i$. For $\epsilon = o(1)$, this is equal to $e^{\pm(2+o(1))i\sqrt{\epsilon}}$. Thus, the eigenvalues of $U'_2$ are 1, and $e^{\pm(2+o(1))\frac{\sqrt{j}}{\sqrt{r+1}}i}$ for $j \in \{1, 2, \dots, k\}$. Similarly, the eigenvalues of $U'_1$ are 1, and $e^{\pm(2+o(1))\frac{\sqrt{j}}{\sqrt{N-r}}i}$ for $j \in \{1, 2, \dots, k\}$.

To complete the proof, we use the following bound on the eigenvalues of the product of two matrices which follows from Hoffman-Wielandt theorem in matrix analysis [27].

**Theorem 6** *Let $A$ and $B$ be unitary matrices. Assume that $A$ has eigenvalues $1+\delta_1, \dots, 1+\delta_m$, $B$ has eigenvalues $\mu_1, \dots, \mu_m$ and $AB$ has eigenvalues $\mu'_1, \dots, \mu'_m$. Then,*

$$|\mu_j - \mu'_j| \leq \sum_{i=1}^{m} |\delta_i|$$

*for all $j \in [m]$.*

**Proof:** In section 4.4. ∎

Let $A = U_1'$ and $B = U_2'$. Since $|e^{\epsilon i} - 1| \leq |\epsilon|$, each of $|\delta_i|$ is of order $O(\frac{1}{\sqrt{N-r}})$. Therefore, their sum is of order $O(\frac{1}{\sqrt{N-r}})$ as well. Thus, for each eigenvalue of $U_2'$, there is a corresponding eigenvalue of $U_2'U_1'$ that differs by at most by $O(\frac{1}{\sqrt{N-r}})$. The lemma now follows from $\frac{1}{\sqrt{N-r}} = o(\frac{1}{\sqrt{r+1}})$. ∎

### 4.3 Proof of Lemma 3

We assume that $|\alpha| < c\epsilon^2$ for some sufficiently small positive constant $c$. Otherwise, we can just take $t = 0$ and get $|\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle| = |\langle\psi_{good}|\psi_{start}\rangle| = |\alpha| \geq c\epsilon^2$.

Consider the eigenvalues of $U_2$. Since $U_2$ is described by a real $m \times m$ matrix (in the basis $|\psi_1\rangle$, ..., $|\psi_m\rangle$), its characteristic polynomial has real coefficients. Therefore, the eigenvalues are 1, -1, $e^{\pm i\theta_1}$, ..., $e^{\pm i\theta_l}$. From conditions of the lemma, we know that the eigenvalue of $e^{i\pi} = -1$ never occurs.

Let $|w_{j,+}\rangle$, $|w_{j,-}\rangle$ be the eigenvectors of $U_2$ with eigenvalues $e^{i\theta_j}$, $e^{-i\theta_j}$. Let $|w_{j,+}\rangle = \sum_{j'=1}^{l} c_{j,j'}|\psi_{j'}\rangle$. Then, we can assume that $|w_{j,-}\rangle = \sum_{j'=1}^{l} c_{j,j'}^*|\psi_{j'}\rangle$. (Since $U_2$ is a real matrix, taking $U_2|w_{j,+}\rangle = e^{i\theta_j}|w_{j,+}\rangle$ and replacing every number with its complex conjugate gives $U_2|w\rangle = e^{-i\theta_j}|w\rangle$ for $|w\rangle = \sum_{j=1}^{l} c_{j,j'}^*|\psi_{j'}\rangle$.)

We write $|\psi_{good}\rangle$ in a basis consisting of eigenvectors of $U_2$:

$$|\psi_{good}\rangle = \alpha|\psi_{start}\rangle + \sum_{j=1}^{l}(a_{j,+}|w_{j,+}\rangle + a_{j,-}|w_{j,-}\rangle). \tag{2}$$

W. l. o. g., assume that $\alpha$ is a positive real. (Otherwise, multiply $|\psi_{start}\rangle$ by an appropriate factor to make $\alpha$ a positive real.)

We can also assume that $a_{j,+} = a_{j,-} = a_j$, with $a_j$ being a positive real number. (To see that, let $|\psi_{good}\rangle = \sum_{j'=1}^{l} b_{j'}|\psi_{j'}\rangle$. Then, $b_{j'}$ are real (by the assumptions of Lemma 3). We have $\langle w_{j,+}|\psi_{good}\rangle = a_{j,+} = \sum_{j'=1}^{l} b_{j'}c_{j,j'}^*$ and $\langle w_{j,-}|\psi_{good}\rangle = a_{j,-} = \sum_{j'=1}^{l} b_{j'}(c_{j,j'}^*)^* = (\sum_{j'=1}^{l} b_{j'}c_{j,j'}^*)^* = a_{j,+}^*$. Multiplying $|w_{j,+}\rangle$ by $\frac{a_{j,+}^*}{|a_{j,+}|}$ and $|w_{j,-}\rangle$ by $\frac{a_{j,+}}{|a_{j,+}|}$ makes both $a_{j,+}$ and $a_{j,-}$ equal to $\frac{a_{j,+}a_{j,+}^*}{|a_{j,+}|} = |a_{j,+}|$ which is a positive real.)

Consider the vector

$$|v_\beta\rangle = \alpha\left(1 + i\cot\frac{\beta}{2}\right)|\psi_{start}\rangle + \sum_{j=1}^{l} a_j\left(1 + i\cot\frac{-\theta_j+\beta}{2}\right)|w_{j,+}\rangle + \sum_{j=1}^{l} a_j\left(1 + i\cot\frac{\theta_j+\beta}{2}\right)|w_{j,-}\rangle. \tag{3}$$

We will prove that, for some $\beta = \Omega(\alpha)$, $|v_\beta\rangle$ and $|v_{-\beta}\rangle$ are eigenvectors of $U_2U_1$, with eigenvalues $e^{\pm i\beta}$. After that, we show that the starting state $|\psi_{start}\rangle$ is close to the state $\frac{1}{\sqrt{2}}|v_\beta\rangle + \frac{1}{\sqrt{2}}|v_{-\beta}\rangle$. Therefore, repeating $U_2U_1$ $\frac{\pi}{2\beta}$ times transforms $|\psi_{start}\rangle$ to a state close to $\frac{i}{\sqrt{2}}|v_\beta\rangle + \frac{-i}{\sqrt{2}}|v_{-\beta}\rangle$ which is equivalent to $\frac{1}{\sqrt{2}}|v_\beta\rangle - \frac{1}{\sqrt{2}}|v_{-\beta}\rangle$. We then complete the proof by showing that this state has a constant inner product with $|\psi_{good}\rangle$.

We first state some bounds on trigonometric functions that will be used throughout the proof.

**Claim 2** *1.* $\frac{2x}{\pi} \leq \sin x \leq x$ *for all* $x \in [0, \frac{\pi}{2}]$*;*

*2.* $\frac{\pi}{4x} \leq \cot x \leq \frac{1}{x}$ *for all* $x \in [0, \frac{\pi}{4}]$.

We now start the proof by establishing a sufficient condition for $|v_\beta\rangle$ and $|v_{-\beta}\rangle$ to be eigenvectors. We have $|v_\beta\rangle = |\psi_{good}\rangle + i|v'_\beta\rangle$ where

$$|v'_\beta\rangle = \alpha \cot\frac{\beta}{2}|\psi_{start}\rangle + \sum_{j=1}^{l} a_j \cot\frac{-\theta_j+\beta}{2}|w_{j,+}\rangle + \sum_{j=1}^{l} a_j \cot\frac{\theta_j+\beta}{2}|w_{j,-}\rangle. \tag{4}$$

**Claim 3** *If $|v'_\beta\rangle$ is orthogonal to $|\psi_{good}\rangle$, then $|v_\beta\rangle$ is an eigenvector of $U_2U_1$ with an eigenvalue of $e^{i\beta}$ and $|v_{-\beta}\rangle$ is an eigenvector of $U_2U_1$ with an eigenvalue of $e^{-i\beta}$.*

**Proof:** Since $|v'_\beta\rangle$ is orthogonal to $|\psi_{good}\rangle$, we have $U_1|v'_\beta\rangle = |v'_\beta\rangle$ and $U_1|v_\beta\rangle = -|\psi_{good}\rangle + i|v'_\beta\rangle$. Therefore,

$$U_2U_1|v_\beta\rangle = \alpha\left(-1+i\cot\frac{\beta}{2}\right)|\psi_{start}\rangle + \sum_{j=1}^{l} a_j e^{i\theta_j}\left(-1+i\cot\frac{-\theta_j+\beta}{2}\right)|w_{j,+}\rangle +$$

$$\sum_{j=1}^{l} a_j e^{-i\theta_j}\left(-1+i\cot\frac{\theta_j+\beta}{2}\right)|w_{j,-}\rangle.$$

Furthermore,

$$1+i\cot x = \frac{\sin x + i\cos x}{\sin x} = \frac{e^{i(\frac{\pi}{2}-x)}}{\sin x},$$
$$-1+i\cot x = \frac{-\sin x + i\cos x}{\sin x} = \frac{e^{i(\frac{\pi}{2}+x)}}{\sin x},$$

Therefore,

$$\left(-1+i\cot\frac{\beta}{2}\right) = e^{i\beta}\left(1+i\cot\frac{\beta}{2}\right),$$

$$e^{i\theta_j}\left(-1+i\cot\frac{-\theta_j+\beta}{2}\right) = \frac{e^{i(\frac{\pi}{2}+\frac{\theta_j}{2}+\frac{\beta}{2})}}{\sin\frac{-\theta_j+\beta}{2}} = e^{i\beta}\left(1+i\cot\frac{-\theta_j+\beta}{2}\right)$$

and similarly for the coefficient of $|w_{j,-}\rangle$. This means that $U_2U_1|v_\beta\rangle = e^{i\beta}|v_\beta\rangle$.

For $|v_{-\beta}\rangle$, we write out the inner products $\langle\psi_{good}|v'_\beta\rangle$ and $\langle\psi_{good}|v'_{-\beta}\rangle$. Then, we see that $\langle\psi_{good}|v'_{-\beta}\rangle = -\langle\psi_{good}|v'_\beta\rangle$. Therefore, if $|\psi_{good}\rangle$ and $|v'_\beta\rangle$ are orthogonal, so are $|\psi_{good}\rangle$ and $|v'_{-\beta}\rangle$. By the argument above, this implies that $|v_{-\beta}\rangle$ is an eigenvector of $U_2U_1$ with an eigenvalue $e^{-i\beta}$. ∎

Next, we use this necessary condition to bound $\beta$ for which $|v_\beta\rangle$ and $|v_{-\beta}\rangle$ are eigenvectors.

**Claim 4** *There exists $\beta$ such that $|v'_\beta\rangle$ is orthogonal to $|\psi_{good}\rangle$ and $\frac{\epsilon\alpha}{\sqrt{\pi}} \le \beta \le 2.6\alpha$.*

**Proof:** Let $f(\beta) = \langle\psi_{good}|v'_\beta\rangle$. We have

$$f(\beta) = \alpha^2\cot\frac{\beta}{2} + \sum_{j=1}^{l}|a_j|^2\left(\cot\frac{-\theta_j+\beta}{2} + \cot\frac{\theta_j+\beta}{2}\right).$$

We bound $f(\beta)$ from below and above, for $\beta \in [0, \frac{\epsilon}{2}]$. For the first term, we have $\frac{\pi}{2\beta} \leq \cot\frac{\beta}{2} \leq \frac{2}{\beta}$ (by claim 2). For the second term, we have

$$\cot\frac{-\theta_j+\beta}{2} + \cot\frac{\theta_j+\beta}{2} = -\frac{\sin\beta}{\sin\frac{\theta_j+\beta}{2}\sin\frac{\theta_j-\beta}{2}}. \tag{5}$$

For the numerator, we have $\frac{2\beta}{\pi} \leq \sin\beta \leq \beta$, because of Claim 2. The denominator can be bounded from below as follows:

$$\sin\frac{\theta_j+\beta}{2}\sin\frac{\theta_j-\beta}{2} \geq \sin\frac{\epsilon}{2}\sin\frac{\epsilon}{4} \geq \frac{\epsilon^2}{2\pi^2},$$

with the first inequality following from $\theta_j \geq \epsilon$ and $\beta \leq \frac{\epsilon}{2}$ and the last inequality following from claim 2. This means

$$\alpha^2\frac{\pi}{2\beta} - \frac{(1-\alpha^2)\pi^2}{\epsilon^2}\beta \leq f(\beta) \leq \alpha^2\frac{2}{\beta} - \frac{1-\alpha^2}{\pi}\beta, \tag{6}$$

where we have used $\|\psi_{good}\|^2 = |\alpha|^2 + 2\sum_{j=1}^{l}|a_j|^2$ (by equation (2)) and $\|\psi_{good}\| = 1$ to replace $\sum_{j=1}^{l}|a_j|^2$ by $\frac{1-\alpha^2}{2}$.

The lower bound of equation (6) implies that $f(\beta) \geq 0$ for $\beta = \frac{\epsilon}{\sqrt{2\pi(1-\alpha^2)}}\alpha$. The upper bound implies that $f(\beta) \leq 0$ for $\beta = \frac{\sqrt{2\pi}}{\sqrt{1-\alpha^2}}\alpha$. Since $f$ is continuous, it must be the case that $f(\beta) = 0$ for some $\beta \in [\frac{\epsilon}{\sqrt{2\pi(1-\alpha^2)}}\alpha, \frac{\sqrt{2\pi}}{\sqrt{1-\alpha^2}}\alpha]$. The claim now follows from $0 \leq \alpha \leq 0.1$. ∎

Let $|u_1\rangle = \frac{|v_\beta\rangle}{\|v_\beta\|}$ and $|u_2\rangle = \frac{|v_{-\beta}\rangle}{\|v_{-\beta}\|}$. We show that $|\psi_{start}\rangle$ is almost a linear combination of $|u_1\rangle$ and $|u_2\rangle$. Define $|\psi_{end}\rangle = \frac{|v_{end}\rangle}{\|v_{end}\|}$ where

$$|v_{end}\rangle = \sum_{j=1}^{l} a_j\left(1 + i\cot\frac{-\theta_j}{2}\right)|w_{j,+}\rangle + \sum_{j=1}^{l} a_j\left(1 + i\cot\frac{\theta_j}{2}\right)|w_{j,-}\rangle. \tag{7}$$

**Claim 5**

$$|u_1\rangle = c_{start}i|\psi_{start}\rangle + c_{end}|\psi_{end}\rangle + |u_1'\rangle,$$

$$|u_2\rangle = -c_{start}i|\psi_{start}\rangle + c_{end}|\psi_{end}\rangle + |u_2'\rangle$$

*where $c_{start}$, $c_{end}$ are positive real numbers and $u_1'$, $u_2'$ satisfy $\|u_1'\| \leq \frac{3\beta}{\epsilon}$ and $\|u_2'\| \leq \frac{3\beta}{\epsilon}$, for $\beta$ from Claim 4.*

**Proof:** By regrouping terms in equation (3), we have

$$|v_\beta\rangle = \alpha i\cot\frac{\beta}{2}|\psi_{start}\rangle + |v_{end}\rangle + |v_\beta''\rangle \tag{8}$$

where

$$|v_\beta''\rangle = \alpha|\psi_{start}\rangle + \sum_{j=1}^{l} a_j i\left(\cot\frac{-\theta_j+\beta}{2} - \cot\frac{-\theta_j}{2}\right)|w_{j,+}\rangle$$

$$+\sum_{j=1}^{l} a_j i \left(\cot\frac{\theta_j+\beta}{2} - \cot\frac{\theta_j}{2}\right)|w_{j,-}\rangle.$$

We claim that $\|v''_\beta\| \leq \frac{3\beta}{\epsilon}\|v_\beta\|$. We prove this by showing that the absolute value of each of coefficients in $|v''_\beta\rangle$ is at most $\frac{3\beta}{\epsilon}$ times the absolute value of corresponding coefficient in $|v_\beta\rangle$. The coefficient of $|\psi_{start}\rangle$ is $\alpha$ in $|v''_\beta\rangle$ and $\alpha(1+i\cot\frac{\beta}{2})$ in $|v_\beta\rangle$. We have

$$|\alpha(1+i\cot\frac{\beta}{2})| \geq \alpha\cot\frac{\beta}{2} \geq \alpha\frac{8}{\pi\beta},$$

which means that the absolute value of the coefficient of $|\psi_{start}\rangle$ in $|v''_\beta\rangle$ is at most $\frac{\pi\beta}{8}$ times the absolute value of the coefficient in $|v_\beta\rangle$. For the coefficient of the $|w_{j,+}\rangle$, we have

$$\cot\frac{-\theta_j+\beta}{2} - \cot\frac{-\theta_j}{2} = \frac{\sin\frac{\beta}{2}}{\sin\frac{-\theta_j+\beta}{2}\sin\frac{-\theta_j}{2}}$$

If $\theta_j - \beta \geq \frac{\pi}{2}$, then

$$\left|\frac{\sin\frac{\beta}{2}}{\sin\frac{-\theta_j+\beta}{2}\sin\frac{-\theta_j}{2}}\right| \leq \frac{\frac{\beta}{2}}{\sin\frac{\pi}{4}\sin\frac{\pi}{4}} = \frac{\frac{\beta}{2}}{\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}} = \beta \leq \beta\left|1+i\cot\frac{-\theta_j+\beta}{2}\right|.$$

If $\theta_j - \beta \leq \frac{\pi}{2}$, then

$$\left|\frac{\sin\frac{\beta}{2}}{\sin\frac{-\theta_j+\beta}{2}\sin\frac{-\theta_j}{2}}\right| = \left|\frac{\sin\frac{\beta}{2}}{\cos\frac{-\theta_j+\beta}{2}\sin\frac{-\theta_j}{2}}\cot\frac{-\theta_j+\beta}{2}\right| \leq \frac{\frac{\beta}{2}}{\frac{1}{\sqrt{2}}\frac{\theta_j}{\pi}}\cot\left|\frac{-\theta_j+\beta}{2}\right| \leq 3\frac{\beta}{\epsilon}\left|\cot\frac{-\theta_j+\beta}{2}\right|,$$

with the first inequality following from $|\cos\frac{-\theta_j+\beta}{2}| \geq |\cos\frac{\pi}{4}| = \frac{1}{\sqrt{2}}$ and $|\sin x| = \sin|x| \geq \frac{2|x|}{\pi}$ (using Claim 2). Therefore, the absolute value of coefficient of $|w_{j,+}\rangle$ in $|v''_\beta\rangle$ is at most $\frac{3\beta}{\epsilon}$ times the absolute value of the coefficient of $|w_{j,+}\rangle$ in $|v_\beta\rangle$ (which is $|a_j(1+i\cot\frac{-\theta_j+\beta}{2})|$). Similarly, we can bound the absolute value of coefficient of $|w_{j,-}\rangle$.

By dividing equation (8) by $\|v_\beta\|$, we get

$$|u_1\rangle = c_{start}i|\psi_{start}\rangle + c_{end}|\psi_{end}\rangle + |u'_1\rangle$$

for $c_{start} = \frac{\alpha\cot\frac{\beta}{2}}{\|v_\beta\|}$, $c_{end} = \frac{\|v_{end}\|}{\|v_\beta\|}$ and $|u'_1\rangle = \frac{1}{\|v_\beta\|}|v''_\beta\rangle$. Since $\|v''_\beta\| \leq \frac{3\beta}{\epsilon}\|v_\beta\|$, we have $\|u'_1\| \leq \frac{3\beta}{\epsilon}$. The proof for $u_2$ is similar. ∎

Since $|u_1\rangle$ and $|u_2\rangle$ are eigenvectors of $U_2U_1$ with different eigenvalues, they must be orthogonal. Therefore,

$$\langle u_1|u_2\rangle = -c^2_{start} + c^2_{end} + O(\frac{\beta}{\epsilon}) = 0,$$

where $O(\frac{\beta}{\epsilon})$ denotes a term that is at most $const\frac{\beta}{\epsilon}$ in absolute value for some constant $const$ that does not depend on $\beta$ and $\epsilon$. Also,

$$\|u_1\|^2 = c^2_{start} + c^2_{end} + O(\frac{\beta}{\epsilon}) = 1.$$

These two equalities together with $c_{start}$ and $c_{end}$ being positive reals imply that $c_{start} = \frac{1}{\sqrt{2}} + O(\beta/\epsilon)$ and $c_{end} = \frac{1}{\sqrt{2}} + O(\beta/\epsilon)$. Therefore,

$$|u_1\rangle = \frac{1}{\sqrt{2}} i|\psi_{start}\rangle + \frac{1}{\sqrt{2}}|\psi_{end}\rangle + |u_1''\rangle,$$

$$|u_2\rangle = -\frac{1}{\sqrt{2}} i|\psi_{start}\rangle + \frac{1}{\sqrt{2}}|\psi_{end}\rangle + |u_2''\rangle,$$

with $\|u_1''\| = O(\beta/\epsilon)$ and $\|u_2''\| = O(\beta/\epsilon)$. This means that

$$|\psi_{start}\rangle = -\frac{i}{\sqrt{2}}|u_1\rangle + \frac{i}{\sqrt{2}}|u_2\rangle + |w'\rangle,$$

$$|\psi_{end}\rangle = \frac{1}{\sqrt{2}}|u_1\rangle + \frac{1}{\sqrt{2}}|u_2\rangle + |w''\rangle,$$

where $w'$ and $w''$ are states with $\|w'\| = O(\beta/\epsilon)$ and $\|w''\| = O(\beta/\epsilon)$. Let $t = \lfloor \frac{\pi}{2\beta} \rfloor$. Then, $(U_2U_1)^t|u_1\rangle$ is almost $i|u_1\rangle$ (plus a term of order $O(\beta)$) and $(U_2U_1)^t|u_2\rangle$ is almost $-i|u_2\rangle$. Therefore,

$$(U_2U_1)^t|\psi_{start}\rangle = |\psi_{end}\rangle + |v'\rangle$$

where $\|v'\| = O(\beta/\epsilon)$. This means that

$$|\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle| \geq |\langle\psi_{good}|\psi_{end}\rangle| - O(\frac{\beta}{\epsilon}). \tag{9}$$

Since $\beta \leq 2.6\alpha$ and $\alpha = c\epsilon^2$, we have $O(\beta/\epsilon) = O(\epsilon)$. By choosing $c$ to be sufficiently small, we can make the $O(\beta/\epsilon)$ term to be less than $0.1\epsilon$. Then, Lemma 3 follows from

**Claim 6**

$$|\langle\psi_{good}|\psi_{end}\rangle| \geq \min\left(\frac{1-\alpha^2}{2}, \frac{1-\alpha^2}{4}\epsilon\right).$$

**Proof:** Since $|\psi_{end}\rangle = \frac{|v_{end}\rangle}{\|v_{end}\|}$, we have $\langle\psi_{good}|\psi_{end}\rangle = \frac{\langle\psi_{good}|v_{end}\rangle}{\|v_{end}\|}$. By definition of $|v_{end}\rangle$ (equation (7)), $\langle\psi_{good}|v_{end}\rangle = 2\sum_{j=1}^{l} a_j^2$. By equation (2), $\|\psi_{good}\|^2 = \alpha^2 + 2\sum_{j=1}^{l} a_j^2$. Since $\|\psi_{good}\|^2 = 1$, we have $\langle\psi_{good}|v_{end}\rangle = 1 - \alpha^2$. Therefore, $\langle\psi_{good}|\psi_{end}\rangle \geq \frac{1-\alpha^2}{\|v_{end}\|}$.

We have $\|v_{end}\|^2 = 2\sum_{j=1}^{l} a_j^2(1+\cot^2\frac{\theta_j}{2})$. Since $\theta_k \in [\epsilon, 2\pi-\epsilon]$, $\|v_{end}\|^2 \leq 2\sum_{j=1}^{l} a_j^2(1+\cot^2\frac{\epsilon}{2}) \leq (1+\cot^2\frac{\epsilon}{2})$ and

$$\langle\psi_{good}|\psi_{end}\rangle \geq \frac{1-\alpha^2}{\sqrt{1+\cot^2(\epsilon/2)}} \geq \frac{1-\alpha^2}{2\max(1,\cot\frac{\epsilon}{2})} \geq \min\left(\frac{1-\alpha^2}{2}, \frac{1-\alpha^2}{4}\epsilon\right).$$

∎

If $\alpha$ is set to be sufficiently small, $|\langle\psi_{good}|\psi_{end}\rangle|$ is close to $0.5\epsilon$ and, together with equation (9), this means that $|\langle\psi_{good}|(U_2U_1)^t|\psi_{start}\rangle|$ is of order $\Omega(\epsilon)$. ∎

**Remark.** If $U_2$ has eigenvectors with eigenvalue -1, the equation (2) becomes

$$|\psi_{good}\rangle = \alpha|\psi_{start}\rangle + \sum_{j=1}^{l}(a_{j,+}|w_{j,+}\rangle + a_{j,-}|w_{j,-}\rangle) + a_{l+1}|w_{l+1}\rangle,$$

with $|w_{l+1}\rangle$ being an eigenvector with eigenvalue -1. We also add $a_{l+1}(1-i\tan\frac{\beta}{2})|w_{l+1}\rangle$, $-a_{l+1}i\tan\frac{\beta}{2}|w_{l+1}\rangle$ and $a_{l+1}|w_{l+1}\rangle$ terms to the right hand sides of equations (3), (4) and (8), respectively. Claims 3, 4, 5 and 6 remain true, but proofs of claims require some modifications to handle the $|w_{l+1}\rangle$ term.

### 4.4 Derivation of Theorem 6

In this section, we derive Theorem 6 (which was used in the proof of Lemma 2) from Hoffman-Wielandt inequality.

**Definition 3** *For a matrix $C = (c_{ij})$, we define its $l_2$-norm as $\|C\| = \sqrt{\sum_{i,j}|c_{ij}^2|}$.*

**Theorem 7** *[27, pp. 292] If $U$ is unitary, then $\|UC\| = \|C\|$ for any $C$.*

**Theorem 8** *[27, Theorem 6.3.5] Let $C$ and $D$ be $m \times m$ matrices. Let $\mu_1$, ..., $\mu_m$ and $\mu'_1, \ldots, \mu'_m$ be eigenvalues of $C$ and $D$, respectively. Then,*

$$\sum_{i=1}^{m}(\mu_i - \mu'_i)^2 \leq \|C - D\|^2.$$

To derive theorem 6 from theorem 8, let $C = B$ and $D = AB$. Then, $C - D = (I - A)B$. Since $B$ is unitary, $\|C - D\| = \|I - A\|$ (Theorem 7). Let $U$ be a unitary matrix that diagonalizes $A$. Then, $U(I-A)U^{-1} = I-UAU^{-1}$ and $\|I-A\| = \|I-UAU^{-1}\|$. Since $UAU^{-1}$ is a diagonal matrix with $1+\delta_i$ on the diagonal, $I-UAU^{-1}$ is a diagonal matrix with $\delta_i$ on the diagonal and $\|I-UAU^{-1}\|^2 = \sum_{i=1}^{m}|\delta_i|^2$ By applying Theorem 8 to $I$ and $UAU^{-1}$, we get

$$\sum_{i=1}^{m}(\mu_i - \mu'_i)^2 \leq \sum_{i=1}^{m}|\delta_i|^2.$$

In particular, for every $i$, we have $(\mu_i - \mu'_i)^2 \leq (\sum_{i=1}^{m}|\delta_i|^2)$ and

$$|\mu_i - \mu'_i| \leq \sqrt{\sum_{i=1}^{m}|\delta_i|^2} \leq \sum_{i=1}^{m}|\delta_i|.$$

## 5 Analysis of multiple $k$-collision algorithm

To solve the general case of $k$-distinctness, we run Algorithm 2 several times, on subsets of the input $x_i, i \in [N]$.

The simplest approach is as follows. We first run Algorithm 2 on the entire input $x_i, i \in [N]$. We then chose a sequence of subsets $T_1 \subseteq [N], T_2 \subseteq [N], \ldots$ with $T_i$ being a random subset of size $|T_i| = (\frac{2k}{2k+1})^i N$,

1. Let $T_1 = [N]$. Let $j = 1$.
2. While $|T_j| > \max(r, \sqrt{N})$ repeat:
   (a) Run Algorithm 2 on $x_i, i \in T_j$, using memory size $r_j = \frac{r|T_j|}{N}$. Measure the final state, obtaining a set $S$. If there are $k$ equal elements $x_i$, $i \in S$, stop, answer "there is a $k$-collision".
   (b) Let $q_j$ be an even power of a prime with $|T_j| \leq q_j \leq (1 + \frac{1}{2k^2})|T_j|$. Select a random permutation $\pi_j$ on $[q_j]$ from an $\frac{1}{N}$-approximately $2k \log N$-wise independent family of permutations (Theorem 2).
   (c) Let
   $$T_{j+1} = \left\{ \pi_1^{-1} \pi_2^{-1} \dots \pi_j^{-1}(i), i \in \left[ \left\lceil \frac{2k}{2k+1} q_j \right\rceil \right] \right\}.$$
   (d) Let $j = j + 1$;
3. If $|T_j| \leq r$, query all $x_i$, $i \in T_j$ classically. If $k$ equal elements are found, answer "there is a $k$-collision", otherwise, answer "there is no $k$-collision".
4. If $|T_j| \leq \sqrt{N}$, run Grover search on the set of at most $N^{k/2}$ $k$-tuples $(i_1, \dots, i_k)$ of pairwise distinct $i_1, \dots, i_k \in T_j$, searching for a tuple $(i_1, \dots, i_k)$ such that $x_{i_1} = \dots = x_{i_k}$. If such a tuple is found, answer "there is a $k$-collision", otherwise, answer "there is no $k$-collision".

Algorithm 3: Multiple-solution algorithm

and run Algorithm 2 on $x_i, i \in T_1$, then on $x_i, i \in T_2$ and so on. It can be shown that, if the input $x_i, i \in [N]$ contains a $k$-collision, then with probability at least 1/2, there exists $j$ such that $x_i, i \in T_j$ contains exactly one $k$-collision. This means that running algorithm 2 on $x_i, i \in T_j$ finds the $k$-collision with a constant probability.

The difficulty with this solution is choosing subsets $T_j$. If we chose a subset of size $\frac{2k}{2k+1}N$ uniformly at random, we need $\Omega(N)$ space to store the subset and $\Omega(N)$ time to generate it. Thus, the straightforward implementation of this solution is efficient in terms of query complexity but not in terms of time or space. Algorithm 3 is a more complicated implementation of the same approach that also achieves time-efficiency and space-efficiency.

We claim

**Theorem 9** *(a) Algorithm 3 uses $O(r + \frac{N^{k/2}}{r^{(k-1)/2}})$ queries.*

*(b) Let $p$ be the success probability of algorithm 2, if there is exactly one $k$-collision. For any $x_1, \dots, x_N$ containing at least one $k$-collision, algorithm 3 finds a $k$-collision with probability at least $(1 - o(1))p/2$.*

**Proof:**

**Part (a).** The second to last step of algorithm 3 use at most $r$ queries. The last step uses $O(N^{k/4})$ queries and is performed only if $\sqrt{N} \geq r$. In this case, $\frac{N^{k/2}}{r^{(k-1)/2}} \geq \frac{N^{k/2}}{N^{(k-1)/4}} \geq N^{k/4}$. Thus, the last two

steps use $O(r + \frac{N^{k/2}}{r^{(k-1)/2}})$ queries and it suffices to show that algorithm 3 uses $O(r + \frac{N^{k/2}}{r^{(k-1)/2}})$ queries in its second step (the while loop).

Let $T_j$ and $r_j$ be as in algorithm 3. Then $|T_1| = N$ and $|T_{j+1}| \leq \frac{2k}{2k+1}(1 + \frac{1}{2k^2})|T_j|$. The number of queries in the $j^{\text{th}}$ iteration of the while loop is of the order

$$\frac{|T_j|^{k/2}}{r_j^{(k-1)/2}} + r_j = \frac{|T_j|^{k/2}}{(|T_j|r/N)^{(k-1)/2}} + \frac{|T_j|r}{N} = \frac{N^{(k-1)/2}}{r^{(k-1)/2}}\sqrt{|T_j|} + \frac{|T_j|r}{N}.$$

The total number of queries in the while loop is of the order

$$\sum_j \left( \frac{N^{(k-1)/2}}{r^{(k-1)/2}}\sqrt{|T_j|} + \frac{|T_j|r}{N} \right) \leq \sum_{j=0}^{\infty} \left( \left( \frac{2k}{2k+1}\frac{2k^2+1}{2k^2} \right)^{j/2} \frac{N^{k/2}}{r^{(k-1)/2}} + \left( \frac{2k}{2k+1}\frac{2k^2+1}{2k^2} \right)^{j} r \right)$$

$$= O\left( \frac{N^{k/2}}{r^{(k-1)/2}} + r \right). \tag{10}$$

**Part (b).** If $x_1, \ldots, x_N$ contain exactly one $k$-collision, then running algorithm 2 on all of $x_1, \ldots, x_N$ finds the $k$-collision with probability at least $p$. If $x_1, \ldots, x_N$ contain more than one $k$-collision, we can have three cases:

1. For some $j$, $T_j$ contains more than one $k$-collision but $T_{j+1}$ contains exactly one $k$-collision.
2. For some $j$, $T_j$ contains more than one $k$-collision but $T_{j+1}$ contains no $k$-collisions.
3. All $T_j$ contain more than one $k$-collision (till $|T_j|$ becomes smaller than $\max(r, \sqrt{N})$ and the loop is stopped).

In the first case, performing algorithm 2 on $x_j$, $j \in T_{i+1}$ finds the $k$-collision with probability at least $p$. In the second case, we have no guarantees about the probability at all. In the third case, the last step of algorithm 3 finds one of $k$-collisions with probability 1.

We will show that the probability of the second case is always less than the probability of the first case plus an asymptotically small quantity. This implies that, with probability at least $1/2 - o(1)$, either first or third case occurs. Therefore, the probability of algorithm 3 finding a $k$-collision is at least $(1/2 - o(1))p$. To complete the proof, we show

**Lemma 4** *Let $T$ be a set containing a $k$-collision. Let $None_j$ be the event that $x_i, i \in T_j$ contains no $k$-collision and $Unique_j$ be the event that $x_i, i \in T_j$ contains a unique $k$-collision. Then,*

$$Pr[Unique_{j+1}|T_j = T] > Pr[None_{j+1}|T_j = T] - o\left(\frac{1}{N^{1/4}}\right) \tag{11}$$

*where $Pr[Unique_{j+1}|T_j = T]$ and $Pr[None_{j+1}|T_j = T]$ denote the conditional probabilities of $Unique_{j+1}$ and $None_{j+1}$, if $T_j = T$.*

The probability of the first case is just the sum of probabilities

$$Pr[Unique_{j+1} \wedge T_j = T] = Pr[T_j = T]Pr[Unique_{j+1}|T_j = T]$$

over all $j$ and $T$ such that $|T| > \max(r, \sqrt{N})$ and $T$ contains more than one $k$-collision. The probability of the second case is a similar sum of probabilities

$$Pr[None_{j+1} \wedge T_j = T] = Pr[T_j = T]Pr[None_{j+1}|T_j = T].$$

Therefore, $Pr[Unique_{j+1}|T_j = T] > Pr[None_{j+1}|T_j = T] + o(\frac{1}{N^{1/4}})$ implies that the probability of the second case is less than the probability of the first case plus a term of order $\frac{1}{N^{1/4}}$ times the number of repetitions for the while loop. The number of repetitions is $O(k \log N)$, because $|T_{j+1}| \leq \frac{2k}{2k+1}(1 + \frac{1}{2k^2})|T_j| \leq (1 - \frac{1}{5k})|T_j|$. Therefore, the probability of the second case is less than the probability of the first case plus a term of order $o(\frac{k \log N}{N^{1/4}}) = o(1)$.

It remains to prove the lemma.

**Proof:** [of Lemma 4] We fix the permutations $\pi_1, \ldots, \pi_{j-1}$ and let $\pi_j$ be chosen uniformly at random from the family of permutations given by Theorem 2.

We consider two cases. The first case is when $T_j$ contains many $k$-collisions. We show that, in this case, the lemma is true because the probability of $None_{j+1}$ is small (of order $o(\frac{1}{N^{1/4}})$). The second case is if $T_j$ contains few $k$-collisions. In this case, we pick one $x$ such that there are at least $k$ elements $i$, $x_i = x$. We compare the probabilities that

- $T_{j+1}$ contains no $k$-collisions;
- $T_{j+1}$ contains exactly one $k$-collision, consisting of $i$ with $x_i = x$.

The first event is the same as $None_{j+1}$, the second event implies $Unique_{j+1}$. We prove the lemma by showing that the probability of the second event is at least the probability of the first event minus a small amount. This is proven by first conditioning on $T_{j+1}$ containing no $k$-collisions consisting of $i$ with $x_i \neq x$ and then comparing the probability that less than $k$ of $i : x_i = x$ belong to $T_{j+1}$ with the probability that exactly $k$ of $i : x_i = x$ belong to $T_{j+1}$.

**Case 1.** $T_j$ contains at least $\log N$ pairwise disjoint sets $S_l = \{i_{l,1}, \ldots, i_{l,k}\}$ with $x_{i_{l,1}} = \ldots = x_{i_{l,k}}$.

Let $S = S_1 \cup S_2 \ldots \cup S_{\log N}$. If event $None_{j+1}$ occurs, at least $\log N$ of $\pi_j \pi_{j-1} \ldots \pi_1(i)$, $i \in S$ (at least one from each of sets $S_1$, ..., $S_{\log N}$) must belong to $\{\lceil \frac{2k}{2k+1} q_j \rceil + 1, \ldots, q_j\}$. By the next claim, this probability is almost the same as the probability that at least $\log N$ of $k \log N$ random elements of $[q_j]$ belong to $\{\lceil \frac{2k}{2k+1} q_j \rceil + 1, \ldots, q_j\}$.

**Claim 7** *Let $S \subseteq T_j$, $|S| \leq 2k \log N$. Let $V \subseteq [q_j]^{|S|}$. Let $p$ be the probability that $(\pi_j \pi_{j-1} \ldots \pi_1(i))_{i \in S}$ belongs to $V$ and let $p'$ be the probability that a tuple consisting of $|S|$ uniformly random elements of $[q_j]$ belongs to $V$. Then,*

$$|p - p'| \leq \frac{|S|^2 + 1}{q_j}.$$

**Proof:** Let $S' = \{\pi_{j-1} \ldots \pi_1(i) | i \in S\}$. Then, $p$ is the probability that $(\pi_j(i))_{i \in S'}$ belongs to $V$. Let $p''$ be the probability that $(v_1, \ldots, v_{|S|})$ belongs to $V$, for $(v_1, \ldots, v_{|S|})$ picked uniformly at random among all tuples of $|S|$ distinct elements of $[q_j]$. By Definition 2, $|p - p''| \leq \frac{1}{N}$.

It remains to bound $|p'' - p'|$. If $(v_1, \ldots, v_{|S|})$ is picked uniformly at random among tuples of distinct elements, every tuple of $|S|$ distinct elements has a probability $\frac{1}{q_j(q_j-1)\ldots(q_j-|S|+1)}$ and the tuples of non-distinct elements have probability 0. If $(v_1, \ldots, v_{|S|})$ is uniformly at random among all tuples, every tuple

has probability $\frac{1}{q_j^{|S|}}$. Therefore,

$$\frac{q_j(q_j-1)\dots(q_j-|S|+1)}{q_j^{|S|}}p'' \le p' \le \frac{q_j\dots(q_j-|S|+1)}{q_j^{|S|}}p'' + \left(1-\frac{q_j\dots(q_j-|S|+1)}{q_j^{|S|}}\right),$$

which implies

$$|p'-p''| \le 1-\frac{q_j(q_j-1)\dots(q_j-|S|+1)}{q_j^{|S|}}.$$

We have

$$1-\frac{q_j(q_j-1)\dots(q_j-|S|+1)}{q_j^{|S|}} \le 1-\left(\frac{q_j-|S|}{q_j}\right)^{|S|} \le 1-\left(1-\frac{|S|^2}{q_j}\right) = \frac{|S|^2}{q_j}.$$

∎

The probability that, out of $k\log N$ uniformly random $i_1,\dots,i_{k\log N} \in \{1,\dots,q_j\}$, at least $\log N$ belong to $\{\lceil\frac{2k}{2k+1}q_j\rceil+1,\dots,q_j\}$ can be bounded using Chernoff bounds [33]. Let $X_l$ be a random variable that is 1 if $i_l \in \{\lceil\frac{2k}{2k+1}q_j\rceil+1,\dots,q_j\}$. Let $X = X_1+\dots+X_{k\log N}$. We need to bound $Pr[X \ge \log N]$. We have $E[X] = k\log N\cdot E[X_1] = \frac{k}{2k+1}\log N - o(1)$ and

$$Pr[X \ge \log N] < \left(\frac{e^{(k+1)/(2k+1)}}{\frac{2k+1}{k}}\right)^{\log N} = e^{-0.316..\log N} = o\left(\frac{1}{N^{1/4}}\right),$$

with the first inequality following from Theorem 4.4 of [33] ($Pr[X \ge (1+\delta)E[X]] < (\frac{e^\delta}{(1+\delta)^{1+\delta}})^{E[X]}$ for $X$ that is a sum of independent identically distributed 0-1 valued random variables). By combining this bound with Claim 7, the probability of $None_{j+1}$ is

$$o\left(\frac{1}{N^{1/4}}\right) + \frac{(k\log N)^2+1}{q_j} = o\left(\frac{1}{N^{1/4}}\right),$$

where we used $q_j \ge |T_j| \ge \sqrt{N}$ (otherwise, the algorithm finishes the while loop).

**Case 2.** $T_j$ contains less than $\log N$ pairwise disjoint sets $S_l = \{i_{l,1},\dots,i_{l,k}\}$ with $x_{i_{l,1}} = \dots = x_{i_{l,k}}$.

Let $S$ be the set of all $i$ such that $x_i$ is a part of a $k$-collision among $x_i$, $i \in T_j$.

**Claim 8** $|S| < 2k\log N$.

**Proof:** We first select a maximal collection of pairwise disjoint $S_l$. This collection contains less than $k\log N$ elements. It remains to prove that $|S - \cup_l S_l| < k\log N$.

Since the collection $\{S_l\}$ is maximal, any $k$-collision between $x_i$, $i \in T_j$ must involve at least one element from $\cup_l S_l$. Therefore, for any $x$, $S \setminus \cup_l S_l$ contains at most $k-1$ values $i$ with $x_i = x$. Also, there are less than $\log N$ possible $x$ because any $k$-collision must involve an element from one of sets $S_l$ and there are less than $\log N$ sets $S_l$. This means that $|S - \cup_l S_l| < (k-1)\log N$. ∎

Let $y_1, y_2, \dots$ be an enumeration of all distinct $y$ such that $T_j$ contains a $k$-collision $i_1,\dots,i_k$ with $x_{i_1} = \dots = x_{i_k} = y$. Let $UniqueColl_l$ be the event that $T_{j+1}$ contains exactly one $k$-collision $i_1,\dots,i_k$

with $x_{i_1} = \ldots = x_{i_k} = y_l$ and $NoColl_l$ be the event that $T_{j+1}$ contains no such collision. The event $None_{j+1}$ is the same as $\bigwedge_l NoColl_l$. The event $Unique_{j+1}$ is implied by $UniqueColl_1 \wedge \bigwedge_{l>1} NoColl_l$. Therefore, it suffices to show

$$Pr\left[\bigwedge_l NoColl_l\right] < Pr\left[UniqueColl_1 \wedge \bigwedge_{l>1} NoColl_l\right] + \frac{2((2k\log N)^2+1)}{q_j}. \tag{12}$$

The events $UniqueColl_l$ and $NoColl_l$ are equivalent to the cardinality of

$$\left\{i : x_i = y_l, i \in T_j \text{ and } \pi_j \ldots \pi_1(i) \in \left\{1, \ldots, \left\lceil \frac{2k}{2k+1} q_j \right\rceil\right\}\right\}$$

being exactly $k$ and less than $k$, respectively.

By Claim 7, the probabilities of both $\bigwedge_l NoColl_l$ and $UniqueColl_1 \wedge \bigwedge_{l>1} NoColl_l$ change by at most $\frac{(2k\log N)^2+1}{N}$ if we replace $(\pi_j \ldots \pi_1(i))_{i\in S}$ by a tuple of $|S|$ random elements of $[q_j]$. Then, the events $NoColl_l$ and $UniqueColl_l$ are independent of events $NoColl_{l'}$ and $UniqueColl_{l'}$ for $l' \neq l$. Therefore,

$$Pr\left[\bigwedge_l NoColl_l\right] = Pr[NoColl_1] \prod_{l>1} Pr[NoColl_l],$$

$$Pr\left[UniqueColl_1 \wedge \bigwedge_{l>1} NoColl_l\right] = Pr[UniqueColl_1] \prod_{l>1} Pr[NoColl_l].$$

This means that, to show (12) for the actual probability distribution $(\pi_j \ldots \pi_1(i))_{i\in S}$, it suffices to prove $Pr[UniqueColl_1] \geq Pr[NoColl_1]$ for tuples consisting of $|S|$ random elements.

Let $I$ be the set of all $i \in T_j$ such that $x_i = y_1$. Let $m = |I|$. Notice that $m \geq k$ (by definition of $x$ and $I$). Let $P_l$ be the event that exactly $l$ of $\pi_j \ldots \pi_1(i)$, $i \in I$ belong to $T_{j+1}$. Then, $Pr[UniqueColl_1] = Pr[P_k]$ and $Pr[NoColl_1] = \sum_{l=0}^{k-1} Pr[P_l]$. When $\pi_j \ldots \pi_1(i)$, $i \in I$ are replaced by random elements of $[q_j]$, we have

$$Pr[P_l] = \binom{m}{l}\left(1 - \frac{1}{2k+1}\right)^l \left(\frac{1}{2k+1}\right)^{m-l},$$

$$\frac{Pr[P_l]}{Pr[P_{l+1}]} = \frac{\binom{m}{l}}{\binom{m}{l+1}} \cdot \frac{1}{2k+1} \cdot \frac{1}{1-\frac{1}{2k+1}} = \frac{l+1}{m-l} \cdot \frac{1}{2k}.$$

For $l \leq k-1$, we have $\frac{l+1}{m-l}\frac{1}{2k} \leq k\frac{1}{2k} = \frac{1}{2}$. This implies $Pr[P_l] \leq \frac{1}{2^{k-l}} Pr[P_k]$ and

$$\sum_{l=0}^{k-1} Pr[P_l] \leq \left(\sum_{l=0}^{k-1} \frac{1}{2^{k-l}}\right) Pr[P_k] \leq Pr[P_k]$$

which is equivalent to $Pr[NoColl_1] \leq Pr[UniqueColl_1]$. ∎

# 6 Running time and other issues

## 6.1 Comparison model

Our algorithm can be adapted to the model of comparison queries similarly to the algorithm of [14]. Instead of having the register $\otimes_{j\in S}|x_j\rangle$, we have a register $|j_1, j_2, \ldots, j_r\rangle$ where $|j_l\rangle$ is the index of the $l^{\text{th}}$ smallest element in the set $S$. Given such register and $y \in [N]$, we can add $y$ to $|j_1, \ldots, j_r\rangle$ by binary search which takes $O(\log N^{k/(k+1)}) = O(\log N)$ queries. We can also remove a given $x \in [N]$ in $O(\log N)$ queries by reversing this process. This gives an algorithm with $O(N^{k/(k+1)} \log N)$ queries.

## 6.2 Running time

So far, we have shown that our algorithm solves element $k$-distinctness with $O(N^{k/(k+1)})$ queries. In this section, we consider the actual running time of our algorithm (when non-query transformations are taken into account).

**Overview.** All that we do between queries is Grover's diffusion operator which can be implemented in $O(\log N)$ quantum time and some data structure operations on set $S$ (for example, insertions and deletions).

We now show how to store $S$ in a classical data structure which supports the necessary operations in $O(\log^4(N+M))$ time. In a sufficiently powerful quantum model, it is possible to transform these $O(\log^4(N+M))$ time classical operations into $O(\log^c(N+M))$ step quantum computation. Then, our quantum algorithm runs in $O(N^{k/(k+1)} \log^c(N+M))$ steps. We will first show this for the standard query model and then describe how the implementation should be modified for it to work in the comparison model.

**Required operations.** To implement algorithm 2, we need the following operations:

1. Adding $y$ to $S$ and storing $x_y$ (step 2 of algorithm 1);
2. Removing $y$ from $S$ and erasing $x_y$ (step 5 of algorithm 1);
3. Checking if $S$ contains $i_1, \ldots, i_k$, $x_{i_1} = \ldots = x_{i_k}$ (to perform the conditional phase flip in step 3a of algorithm 2);
4. Diffusion transforms on $|x\rangle$ register in steps 1 and 4 of algorithm 1.

**Additional requirements.** Making a data structure part of quantum algorithm creates two subtle issues. First, there is the uniqueness problem. In many classical data structures, the same set $S$ can be stored in many equivalent ways, depending on the order in which elements were added and removed. In the quantum case, this would mean that the basis state $|S\rangle$ is replaced by many states $|S^1\rangle$, $|S^2\rangle$, ... which in addition to $S$ store some information about the previous sets. This can have a very bad result. In the original quantum algorithm, we might have $\alpha|S\rangle$ interfering with $-\alpha|S\rangle$, resulting in 0 amplitude for $|S\rangle$. If $\alpha|S\rangle - \alpha|S\rangle$ becomes $\alpha|S^1\rangle - \alpha|S^2\rangle$, there is no interference between $|S^1\rangle$ and $|S^2\rangle$ and the result of the algorithm will be different.

To avoid this problem, we need a data structure where the same set $S \subseteq [N]$ is always stored in the same way, independent of how $S$ was created.

Second, if we use a classical subroutine, it must terminate in a fixed time $t$. Only then, we can replace it by an $O(poly(t))$ time quantum algorithm. The subroutines that take time $t$ on average (but might take longer time sometimes) are not acceptable.

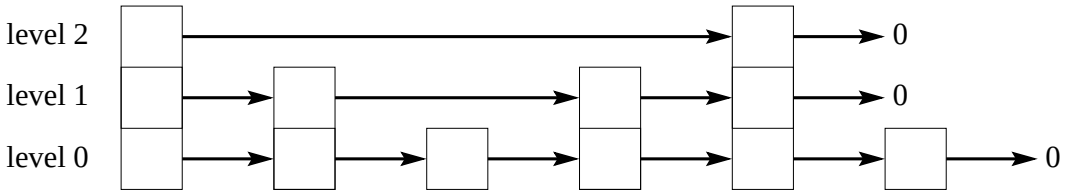


Figure 1: A skip list with 3 levels

**Model.** To implement our algorithm, we use standard quantum circuit model, augmented with gates for random access to a quantum memory. A random access gate takes three inputs: $|i\rangle$, $|b\rangle$ and $|z\rangle$, with $b$ being a single qubit, $z$ being an $m$-qubit register and $i \in [m]$. It then implements the mapping

$$|i, b, z\rangle \rightarrow |i, z_i, z_1 \dots z_{i-1} b z_{i+1} \dots z_m\rangle.$$

Random access gates are not commonly used in quantum algorithms but are necessary in our case because, otherwise, simple data structure operations (for example, removing $y$ from $S$) which require $O(\log N)$ time classically would require $\Omega(r)$ time quantumly.

In addition to random access gates, we allow the standard one and two qubit gates [9].

**Data structure:overview.** Our data structure is a combination of a hash table and a skip list. We use the hash table to store pairs $(i, x_i)$ in the memory and to access them when we need to find $x_i$ for a given $i$. We use the skip list to keep the items sorted in the order of increasing $x_i$ so that, when a new element $i$ is added to $S$, we can quickly check if $x_i$ is equal to any of $x_j$, $j \in S$.

We also maintain a variable $v$ counting the number of different $x \in [M]$ such that the set $S$ contains $i_1, \dots, i_k$ with $x_{i_1} = \dots = x_{i_k} = x$.

**Data structure:hash table.** Our hash table consists of $r$ buckets, each of which contains memory for $\lceil \log N \rceil$ entries. Each entry uses $O(\log^2 N + \log M)$ qubits. The total memory is, thus, $O(r \log^3(N + M))$, slightly more than in the case when we were only concerned about the number of queries.

We hash $\{1, \dots, N\}$ to the $r$ buckets using a fixed hash function $h(i) = \lfloor i \cdot r/N \rfloor + 1$. The $j^{\text{th}}$ bucket stores pairs $(i, x_i)$ for $i \in S$ such that $h(i) = j$, in the order of increasing $i$.

In the case if there are more than $\lceil \log N \rceil$ entries with $h(i) = j$, the bucket only stores $\lceil \log N \rceil$ of them. This means that our data structure misfunctions. We will show that the probability of that happening is small.

Besides the $\lceil \log N \rceil$ entries, each bucket also contains memory for storing $\lfloor \log r \rfloor$ counters $d_1, \dots, d_{\lfloor \log r \rfloor}$. The counter $d_1$ in the $j^{\text{th}}$ bucket counts the number of $i \in S$ such that $h(i) = j$. The counter $d_l$, $l > 1$ is only used if $j$ is divisible by $2^l$. Then, it counts the number of $i \in S$ such that $j - 2^l + 1 \leq h(i) \leq j$.

The entry for $(i, x_i)$ contains $(i, x_i)$, together with a memory for $\lceil \log N \rceil + 1$ pointers to other entries that are used to set up a skip list (described below).

**Data structure:skip list.** In a skip list [35], each $i \in S$ has a randomly assigned level $l_i$ between 0 and $l_{max} = \lceil \log N \rceil$. The skip list consists of $l_{max} + 1$ lists, from the level-0 list to the level-$l_{max}$ list. The level-$l$ list contains all $i \in S$ with $l_i \geq l$. Each element of the level-$l$ level list has a level-$l$ pointer pointing to the next element of the level-$l$ list (or 0 if there is no next element). The skip list also uses one additional

"start" entry. This entry does not store any $(i, x_i)$ but has $l_{max} + 1$ pointers, with the level-$l$ pointer pointing to the first element of the level-$l$ list. An example is shown in figure 1.

In our case, each list is in the order of increasing $x_i$. (If several $i$ have the same $x_i$, they are ordered by $i$.) Instead of storing an adress for a memory location, pointers store the value of the next element $i \in S$. Given $i$, we can find the entry for $(i, x_i)$ by computing $h(i)$ and searching the $h(i)^{\text{th}}$ bucket.

Given $x$, we can search the skip list as follows:

1. Traverse the level-$l_{max}$ list until we find the last element $i_{l_{max}}$ with $x_{i_{l_{max}}} < x$.

2. For each $l = l_{max} - 1, l_{max} - 2, \ldots, 0$, traverse the level-$l$ list, starting at $i_{l+1}$, until the last element $i_l$ with $x_{i_l} < x$.

The result of the last stage is $i_0$, the last element of the level-0 list (which contains all $i \in S$) with $x_{i_0} < x$. If we are given $i$ and $x_i$, a similar search can find the last element $i_0$ which satisfies either $x_{i_0} < x_i$ or $x_{i_0} = x_i$ and $i_0 < i$. This is the element which would precede $i$, if $i$ was inserted into the skip list.

It remains to specify the levels $l_i$. The level $l_i$ is assigned to each $i \in [N]$ before the beginning of the computation and does not change during the computation. $l_i$ is equal to $j$ with probability $1/2^{j+1}$ for $j < l_{max}$ and probability $1/2^{l_{max}}$ for $j = l_{max}$.

The straightforward implementation (in which we chose the level independently for each $i$) has the drawback that we have to store the level for each of $N$ possible $i \in [N]$ which requires $\Omega(N)$ time to choose the levels and $\Omega(N)$ space to store them. To avoid this problem, we define the levels using $l_{max}$ functions $h_1, h_2, \ldots, h_{l_{max}} : [N] \to \{0, 1\}$. $i \in [N]$ belongs to level $l$ (for $l < l_{max}$) if $h_1(i) = \ldots = h_l(i) = 1$ but $h_{l+1}(i) = 0$. $i \in [N]$ belongs to level $l_{max}$ if $h_1(i) = \ldots = h_{l_{max}}(i) = 1$. Each hash function is picked uniformly at random from a $d$-wise independent family of hash functions (Theorem 1), for $d = \lceil 4 \log_2 N + 1 \rceil$.

In the quantum case, we augment the quantum state by an extra register holding $|h_1, \ldots, h_{l_{max}}\rangle$. The register is initialized to a superposition in which every basis state $|h_1, \ldots, h_{l_{max}}\rangle$ has an equal amplitude. The register is then used to perform transformations dependent on $h_1, \ldots, h_{l_{max}}$ on other registers.

**Operations: insertion and deletion.** To add $i$ to $S$, we first query the value $x_i$. Then, we compute $h(i)$ and add $(i, x_i)$ to the $h(i)^{\text{th}}$ bucket. If the bucket already contains some entries, we may move some of them so that, after inserting $(i, x_i)$, the entries are still in the order of increasing $i$. We then add 1 to the counter $d_1$ for the $h(i)^{\text{th}}$ bucket and the counter $d_l$ for the $(\lceil \frac{h(i)}{2^l} \rceil 2^l)^{\text{th}}$ bucket, for each $l \in \{2, \ldots, \lfloor \log r \rfloor\}$. We then update the skip list:

1. Run the search for the last element before $i$ (as described earlier). The search finds the last element $i_l$ before $i$ on each level $l \in \{0, \ldots, l_{max}\}$.

2. For each level $l \in \{0, \ldots, l_i\}$, let $j_l$ be the level-$l$ pointer of $i_l$. Set the level-$l$ pointer of $i$ to be equal to $j_l$ and the level-$l$ pointer of $i_l$ to be equal to $i$.

After the update is complete, we use the skip list to find the smallest $j$ such that $x_j = x_i$ and then use level-0 pointers to count if the number of $j : x_j = x_i$ is less than $k$, exactly $k$ or more than $k$. If there are exactly $k$ such $j$, we increase $v$ by 1. (In this case, before adding $i$ to $S$, there were $k - 1$ such $j$ and, after adding $i$, there are $k$ such $j$. Thus, the number of $x$ such that $S$ contains $i_1, \ldots, i_k$ with $x_{i_1} = \ldots = x_{i_k} = x$ has increased by 1.)

An element $i$ can be deleted from $S$ by running this procedure in reverse.

**Operations: checking for $k$-collisions.** To check for $k$-collisions in set $S$, we just check if $v > 0$.

**Operations: diffusion transform.** As shown by Grover[26], the following transformation on $|1\rangle$, ..., $|n\rangle$ can be implemented with $O(\log n)$ elementary gates:

$$|i\rangle \rightarrow \left(-1 + \frac{2}{n}\right)|i\rangle + \sum_{i' \in [n], i' \neq i} \frac{2}{n}|i'\rangle. \tag{13}$$

To implement our transformation in the step 4 of Algorithm 1, we need to implement a 1-1 mapping $f$ between between $S$ and $\{1, \ldots, |S|\}$. Once we have such mapping, we can carry out the transformation $|y\rangle \rightarrow |f(y)\rangle$ by $|y\rangle|0\rangle \rightarrow |y\rangle|f(y)\rangle \rightarrow |0\rangle|f(y)\rangle$ where the first step is a calculation of $f(y)$ from $y$ and the second step is the reverse of a calculation of $y$ from $f(y)$. Then, we perform the transformation (13) on $|1\rangle$, ..., $||S|\rangle$ and then apply the transformation $|f(y)\rangle \rightarrow |y\rangle$, mapping $\{1, \ldots, |S|\}$ back to $S$.

The mapping $f$ can be defined as follows. $f(y) = f_1(y) + f_2(y)$ where $f_1(y)$ is the number of items $i \in S$ that are mapped to buckets $j$, $j < h(y)$ and $f_2(y)$ is the number of items $y' \leq y$ that are mapped to bucket $h(y)$. It is easy to see that $f$ is 1-1 mapping from $S$ to $\{1, \ldots, |S|\}$. $f_2(y)$ can be computed by counting the number of items in bucket $h(y)$ in time $O(\log N)$. $f_1(y)$ can be computed as follows:

1. Let $i = 0$, $l = \lfloor \log r \rfloor$, $s = 0$.
2. While $l \geq 0$ repeat:
   (a) If $i + 2^l < y$, add $d_l$ from the $(i + 2^l)^{\text{th}}$ bucket to $s$; let $i = i + 2^l$;
   (b) Let $l = l - 1$;
3. Return $s$ as $f_1(y)$;

The transformation in step 1 of algorithm 1 is implemented, using a similar 1-1 mapping $f$ between between $[N] \setminus S$ and $\{1, \ldots, N - |S|\}$.

**Uniqueness.** It is easy to see that a set $S$ is always stored in the same way. The values $i \in S$ are always hashed to buckets by $h$ in the same way and, in each bucket, the entries are located in the order of increasing $i$. The counters counting the number of entries in the buckets are uniquely determined by $S$. The structure of the skip list is also uniquely determined, once the functions $h_1, \ldots, h_{l_{max}}$ are fixed.

**Guaranteed running time.** We show that, for any $S$, the probability that lookup, insertion or deletion of some element takes more than $O(\log^4(N + M))$ steps is very small. We then modify the algorithms for lookup, insertion or deletion so that they abort after $c \log^4(N + M)$ steps and show that this has no significant effect on the entire quantum search algorithm. More precisely, let

$$|\psi_t\rangle = \sum_{S, y, h_1, \ldots, h_{l_{max}}} \alpha^t_{S,y} |\psi_{S,h_1,\ldots,h_{l_{max}}}\rangle |y\rangle |h_1, \ldots, h_{l_{max}}\rangle$$

be the state of the quantum algorithm after $t$ steps (each step being the quantum translation of one data structure operation), using quantum translations of the perfect data structure operations (which do not fail but may take more than $c \log^4 N$ steps). Here, $|\psi_{S,h_1,\ldots,h_{l_{max}}}\rangle$ stands for the basis state corresponding to our

data structure storing $S$ and $x_i$, $i \in S$, using the hash functions $h_1, \ldots, h_{l_{max}}$. (Notice that the amplitude $\alpha^i_{S,y}$ is independent of $h_1, \ldots, h_{l_{max}}$, since $h_1, \ldots, h_{l_{max}}$ all are equally likely.)

We decompose $|\psi_t\rangle = |\psi_t^{good}\rangle + |\psi_t^{bad}\rangle$, with $|\psi_t^{good}\rangle$ consisting of $(S, h_1, \ldots, h_{l_{max}})$ for which the next operation successfully completes in $c\log^4(N+M)$ steps and $|\psi_t^{bad}\rangle$ consisting of $(S, h_1, \ldots, h_{l_{max}})$ for which the next operation fails to complete in $c\log^4(N+M)$ steps. Let $|\psi'_t\rangle$ be the state of the quantum algorithm after $t$ steps using the imperfect data structure algorithms which may abort. The next lemma is an adaptation of "hybrid argument" by Bennett et al. [11] to our context.

**Lemma 5**

$$\|\psi_t - \psi'_t\| \leq \sum_{t'=1}^{t} 2\|\psi_{t'}^{bad}\|.$$

**Proof:** By induction. It suffices to show that

$$\|\psi_t - \psi'_t\| \leq \|\psi_{t-1} - \psi'_{t-1}\| + 2\|\psi_t^{bad}\|.$$

To show that, we introduce an intermediate state $|\psi''_t\rangle$ which is obtained by applying the perfect transformations in the first $t-1$ steps and the transformation which may fail in the last step. Then,

$$\|\psi_t - \psi'_t\| \leq \|\psi_t - \psi''_t\| + \|\psi''_t - \psi'_t\|.$$

The second term, $\|\psi''_t - \psi'_t\|$ is the same as $\|\psi_{t-1} - \psi'_{t-1}\|$ because the states $|\psi''_t\rangle$ and $|\psi'_t\rangle$ are obtained by applying the same unitary transformation (quantum translation of a data structure transformation which may fail) to states $|\psi_{t-1}\rangle$ and $|\psi'_{t-1}\rangle$, respectively. To bound the first term, $\|\psi_t - \psi''_t\|$, let $U_p$ and $U_i$ be the unitary transformations corresponding to perfect and imperfect version of the $t^{\text{th}}$ data structure operation. Then, $|\psi_t\rangle = U_p|\psi_{t-1}\rangle$ and $|\psi'_t\rangle = U_i|\psi_{t-1}\rangle$. Since $U_p$ and $U_i$ only differ for $(S, h_1, \ldots, h_{l_{max}})$ for which the data structure operation does not finish in $c\log^4 N$ steps, we have

$$\|\psi_t - \psi'_t\| = \|U_p|\psi_{t-1}\rangle - U_i|\psi_{t-1}\rangle\| = \|U_p|\psi_{t-1}^{bad}\rangle - U_i|\psi_{t-1}^{bad}\rangle\| \leq 2\|\psi_{t-1}^{bad}\|.$$

∎

**Lemma 6** *For every $t$, $\|\psi_t^{bad}\| = O(\frac{1}{N^{1.5}})$.*

**Proof:** We assume that there is exactly one $k$-collision $x_{i_1} = \ldots = x_{i_k}$. (If there is no $k$-collisions, the checking step at the end of algorithm 2 ensures that the answer is correct. The case with more than one $k$-collision reduces to the case with exactly one $k$-collision because of the analysis in section 5.)

By Lemma 1, every basis state $|S, x\rangle$ of the same type has equal amplitude. Also, all $h_1, \ldots, h_{l_{max}}$ have equal probabilities. Therefore, it suffices to show that, for any fixed $s = |S \cap \{i_1, \ldots, i_k\}|$ and $t = |\{x\} \cap \{i_1, \ldots, i_k\}|$, the fraction of $|S, x, h_1, \ldots, h_{l_{max}}\rangle$ for which the operation fails is at most $\frac{1}{N^3}$.

There are two parts of the update operation which can fail:

1. Hash table can overflow if more than $\lceil \log N \rceil$ elements $i \in S$ have the same $h(i) = h$;
2. Update or lookup in the skip list can take more than $c\log^4 N$ steps.

For the first part, let $s = |S \cap \{i_1, \ldots, i_k\}|$. If more than $\lceil \log N \rceil$ elements $i \in S$ have $h(i) = j$, then at least $\lceil \log N \rceil - s$ of them must belong to $[N] \setminus \{i_1, \ldots, i_k\}$. We now show that, for a random set $S \subseteq [N] \setminus \{i_1, \ldots, i_k\}$, $|S| = r - s$ the probability that more than $\lceil \log N \rceil - s$ of $i \in S$ satisfy $h(i) = j$ is small.

We introduce random variables $X_1, \ldots, X_{r-s}$ with $X_l = 1$ if $h$ maps the $l^{\text{th}}$ element of $S$ to $j$. We need to bound $X = X_1 + \ldots + X_{r-s}$. We have $\frac{N/r-s}{N-k} \leq E[X_l] \leq \frac{N/r}{N-k}$, which means that $E[X_l] = \frac{1}{r} + O(\frac{1}{N})$. (Here, we are assuming that $k$ is a constant. $s$ is also a constant because $s \leq k$.) Therefore, $E[X] = (r-s)E[X_l] = 1 + o(1)$.

The random variables $X_l$ are negatively correlated: if one or more of $X_l$ is equal to 1, then the probability that other variables $X_{l'}$ are equal to 1 decreases. Therefore [34], we can apply Chernoff bounds to bound $Pr[X > \log N - s]$. By using the bound $Pr[X \geq (1+\delta)E[X]] < (\frac{e^\delta}{(1+\delta)^{1+\delta}})^{E[X]}$ [33, 34], we get

$$Pr[X > \log N - s] < \frac{e^{\log N - s - 1}}{(\log N - s)^{\log N - s}} = o\left(\frac{1}{N^4}\right).$$

For the second part, we consider the time required for insertion of a new element. (Removing an element requires the same time, because it is done by running the insertion algorithm in reverse.) Adding $(i, x_i)$ to the $(h(i))^{\text{th}}$ bucket requires comparing $i$ to entries already in the bucket and, possibly, moving some of the entries so that they remain sorted in the order of increasing $i$. Since a bucket contains $O(\log N)$ entries and each entry uses $\log^2(N+M)$ bits, this can be done in $O(\log^3(N+M))$ time. Updating counters $d_l$ requires $O(\log N)$ time, for each of $O(\log r) = O(\log N)$ counters.

To update the skip list, we first need to compute $h_1(i)$, ..., $h_{l_{max}}(i)$. This is the most time-consuming step, requiring $O(d \log^2 N) = O(\log^3 N)$ steps for each of $l_{max} = \lceil \log N \rceil$ functions $h_l$. The total time for this step is $O(\log^4 N)$. We then need to update the pointers in the skip list. We show that, for any fixed $S, y$ (and random $h_1, \ldots, h_{l_{max}}$), the probability that updating the pointers in the skip list takes more than $c \log^4 N$ steps, is small.

Each time when we access a pointer in the skip list, it may take $O(\log^2 N)$ steps, because a pointer stores the number $i$ of the next entry and, to find the entry $(i, x_i)$ itself, we have to compute $h(i)$ and search the $h(i)^{\text{th}}$ bucket which may contain $\log N$ entries, each of which uses $\log N$ bits to store $i$. Therefore, it suffices to show that the probability of a skip list operation accessing more than $c \log^2 N$ pointers is small.

We do that by proving that at most $d = 4 \log N + 1$ pointer accesses are needed on each of $\log N + 1$ levels $l$. We first consider level 0. Let $j_1, j_2, \ldots$ be the elements of $S$ ordered so that $x_{j_1} \leq x_{j_2} \leq x_{j_3} \ldots$ (and, if $x_{j_l} = x_{j_{l+1}}$ for some $j$, then $j_l < j_{l+1}$). If the algorithm requires more than $d$ pointer accesses on level 0, it must be the case that, for some $i'$, $j_{i'}$, ..., $j_{i'+d-1}$ are all at level 0. That is equivalent to $h_1(j_{i'}) = h_1(j_{i'+1}) = \ldots = h_1(j_{i'+d-1}) = 0$. Since $h_1$ is $d$-wise independent, the probability that $h_1(j_{i'}) = \ldots = h_1(j_{i'+d-1}) = 0$ is $2^{-d} < N^{-4}$.

For level $l$ ($0 < l < l_{max}$), we first fix the hash functions $h_1, \ldots, h_l$. Let $j_1, j_2, \ldots$ be the elements of $S$ for which $h_1$, ..., $h_l$ are all 1, ordered so that $x_{j_1} \leq x_{j_2} \leq x_{j_3} \ldots$. By the same argument, the probability that the algorithm needs $d$ or more pointer accesses on level $l$ is the same as the probability that $h_{l+1}(j_{i'}) = \ldots = h_{l+1}(j_{i'+d-1}) = 0$ for some $i'$ and this probability is at most $2^{-d} < N^{-4}$. For level $l_{max}$, we fix hash functions $h_1, \ldots, h_{l_{max}-1}$ and notice that $i$ is on level $l_{max}$ whenever $h_{l_{max}}(i) = 1$. The rest of the argument is as before, with $h_{l_{max}}(j_{i'}) = h_{l_{max}}(j_{i'+1}) = \ldots = h_{l_{max}}(j_{i'+d-1}) = 1$ instead of $h_1(j_{i'}) = h_1(j_{i'+1}) = \ldots = h_1(j_{i'+d-1}) = 0$.

Since there are $\log N + 1$ levels and $r$ elements of $S$, the probability that the algorithm spends more than $k-1$ steps on one level for some element of $S$ is at most $O(\frac{|S| \log N}{N^4}) = O(\frac{1}{N^3})$.

Therefore, $\|\psi_t^{bad}\|^2 = O(\frac{1}{N^3})$ and $\|\psi_t^{bad}\| = O(\frac{1}{N^{1.5}})$, proving the lemma. ∎

By Lemmas 5 and 6, the distance between the final states of the ideal algorithm (where the data structures never fail) and the actual algorithm is of order $O(\frac{r}{N^{3/2}}) = O(\frac{1}{N^{1/2}})$. This also means that the probability distributions obtained by measuring the two states differ by at most $O(\frac{1}{N^{1/2}})$, in variational distance [13]. Therefore, the imperfectness of the data structure operations does not have a significant effect.

**Implementation in comparison model.** The implementation in comparison model is similar, except that the hash table only stores $i$ instead of $(i, x_i)$.

# 7 Open problems

1. **Time-space tradeoffs.** Our optimal $O(N^{2/3})$-query algorithm requires space to store $O(N^{2/3})$ items.

   How many queries do we need if algorithm's memory is restricted to $r$ items? Our algorithm needs $O(\frac{N}{\sqrt{r}})$ queries and this is the best known. Curiously, the lower bound for deterministic algorithms in comparison query model is $\Omega(\frac{N^2}{r})$ queries [38] which is quadratically more. This suggests that our algorithm might be optimal in this setting as well. However, the only lower bound is the $\Omega(N^{2/3})$ lower bound for algorithms with unrestricted memory [1].

2. **Optimality of $k$-distinctness algorithm.** While element distinctness is known to require $\Omega(N^{2/3})$ queries, it is open whether our $O(N^{k/(k+1)})$ query algorithm for $k$-distinctness is optimal.

   The best lower bound for $k$-distinctness is $\Omega(N^{2/3})$, by a following argument. We take an instance of element distinctness $x_1, \ldots, x_N$ and transform it into $k$-distinctness by repeating every element $k-1$ times. If $x_1, \ldots, x_N$ are all distinct, there is no $k$ equal elements. If there are $i, j$ such that $x_i = x_j$ among original $N$ elements, then repeating each of them $k-1$ times creates $2k-2$ equal elements. Therefore, solving $k$-distinctness on $(k-1)N$ elements requires at least the same number of queries as solving distinctness on $N$ elements (which requires $\Omega(N^{2/3})$ queries).

3. **Quantum walks on other graphs.** A quantum walk search algorithm based on similar ideas can be used for Grover search on grids [8, 22]. What other graphs can quantum-walks based algorithms search? Is there a graph-theoretic property that determines if quantum walk algorithms work well on this graph?

   [8] and [37] have shown that, for a class of graphs, the performance of quantum walk depends on certain expressions consisting of graph's eigenvalues. In particular, if a graph has a large eigenvalue gap, quantum walk search performs well [37]. A large eigenvalue gap is, however, not necessary, as shown by quantum search algorithms for grids [8, 37].

**Acknowledgments.** Thanks to Scott Aaronson for showing that $k$-distinctness is at least as hard as distinctness (remark 2 in section 7), to Robert Beals, Greg Kuperberg and Samuel Kutin for pointing out the "uniqueness" problem in section 6 and to Boaz Barak, Andrew Childs, Daniel Gottesman, Julia Kempe, Samuel Kutin, Frederic Magniez, Oded Regev, Mario Szegedy, Tathagat Tulsi and anonymous referees for comments and discussions.

## References


[1] S. Aaronson, Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51:595-605, 2004.

[2] S. Aaronson, A. Ambainis. Quantum search of spatial structures. *Theory of Computing*, 1:47-79, 2005. Earlier version at FOCS'03.

[3] D. Aharonov. Quantum computation - a review. *Annual Review of Computational Physics(ed. Dietrich Stauffer)*, vol. VI, World Scientific, 1998.

[4] N. Alon, L. Babai, A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4): 567-583, 1986.

[5] A. Ambainis. Quantum lower bounds for collision and element distinctness with small range, *Theory of Computing*, 1:37-46.

[6] A. Ambainis. Quantum walks and their algorithmic applications. *International Journal of Quantum Information*, 1:507-518 (2003).

[7] A. Ambainis. Quantum query algorithms and lower bounds. *Proceedings of FOTFS'III*, Trends in Logic, 23:15-32, Kluwer, 2004. Journal version under preparation.

[8] A. Ambainis, J. Kempe, A. Rivosh. Coins make quantum walks faster, *Proceedings of SODA'05*, pp. 1099-1108.

[9] A. Barenco, C. Bennett, R. Cleve, D. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, H. Weinfurter, Elementary gates for quantum computation, *Physical Review A* 52:34573467, 1995.

[10] P. Beame, M. Saks, X. Sun, E. Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *Journal of the ACM*, 50: 154-195, 2003. Also FOCS'00.

[11] C. Bennett, E. Bernstein, G. Brassard, U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing* 26(5): 1510-1523, 1997.

[12] P. Benioff. Space searches with a quantum robot.*Quantum Computation and Quantum Information: A Millennium Volume*, AMS Contemporary Mathematics Series, 2002. Also quant-ph/0003006.

[13] E. Bernstein, U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5): 1411-1473, 1997.

[14] H. Buhrman, C. Durr, M. Heiligman, P. Hoyer, F. Magniez, M. Santha, and R. de Wolf. Quantum algorithms for element distinctness. *SIAM Journal on Computing*, 34:1324-1330, 2005. Earlier version in Complexity'01.

[15] H. Buhrman, R. Spalek. Quantum verification of matrix products, quant-ph/0409035.

[16] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation, *Quantum Computation and Quantum Information: A Millennium Volume*, AMS Contemporary Mathematics Series, 2002.

[17] G. Brassard, P. Høyer, and A. Tapp. Quantum cryptanalysis of hash and claw-free functions. Proceedings of LATIN'98, *Lecture Notes in Computer Science*, 1380:163-169, 1998.

[18] H. Buhrman, R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288:21-43, 2002.

[19] A. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, D. Spielman Exponential algorithmic speedup by quantum walk. *Proceedings of STOC'03*, pp. 59-68.

[20] A. Childs, J. Eisenberg. Quantum algorithms for subset finding. quant-ph/0311038.

[21] A. Childs, E. Farhi, S. Gutmann. An example of the difference between quantum and classical random walks. *Journal of Quantum Information Processing*, 1:35, 2002.

[22] A. Childs, J. Goldstone. Spatial search by quantum walk, *Physical Review A*, 70:022314, 2004.

[23] A. Childs, J. Goldstone. Spatial search and the Dirac equation, *Physical Review A*, 70:042312, 2004.

[24] E. Farhi, S. Gutmann. Quantum computation and decision trees. *Physical Review A*, 58:915-928, 1998.

[25] D. Grigoriev, M. Karpinski, F. Meyer auf der Heide, and R. Smolensky. A lower bound for randomized algebraic decision trees. *Proceedings of STOC'96*, pp. 612–619.

[26] L. Grover. A fast quantum mechanical algorithm for database search. *Proceedings of STOC'96*, pp. 212-219.

[27] R. Horn, C. Johnson *Matrix Analysis*, Cambridge University Press, 1985.

[28] T. Itoh, Y. Takei, J. Tarui. Explicit construction of $k$-wise nearly random permutations by Feistel transform. *Workshop on Randomness and Computation*, Sendai, Japan, July 2005.

[29] J. Kempe. Quantum random walks hit exponentially faster Proceedings of RANDOM'03, *Lecture Notes in Computer Science*, 2764:354-369, 2003.

[30] J. Kempe. Quantum random walks - an introductory overview, *Contemporary Physics*, 44:307-327, 2003.

[31] S. Kutin. A quantum lower bound for the collision problem. *Theory of Computing*, 1:29-36, 2005.

[32] F. Magniez, M. Santha, M. Szegedy. An $O(\tilde{n^{1.3}})$ quantum algorithm for the triangle problem. *Proceedings of SODA'05*, pp. 1109-1117.

[33] M. Mitzenmacher, E. Upfal. *Probability and Computing. Randomized Algorithms and Probabilistic Analysis*, Cambridge University Press, 2005.

[34] A. Panconesi, A. Srinivasan. Randomized distributed edge coloring via an extension of the Chernoff-Hoeffding bounds. *SIAM Journal on Computing*, 26:350-368, 1997.

[35] W. Pugh. Skip lists: A probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, 1990.

[36] N. Shenvi, J. Kempe, K. Whaley. Quantum random-walk search algorithm, *Physical Review A*, 67:052307, 2003.

[37] M. Szegedy. Quantum speed-up of Markov Chain based algorithms. *Proceedings of FOCS'04*, pp. 32-41. Earlier version at quant-ph/0401053.

[38] A. Yao. Near-optimal time-space tradeoff for element distinctness. *Proceedings of FOCS'88*, pp. 91-97.