Simple Quantum Algorithms for k-mismatch Problem

by

Ruhan Habib
20141048

Thesis submitted to the Department of Mathematics and Natural Sciences
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Mathematics

Department of Mathematics and Natural Sciences
Brac University
April 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Ruhan Habib
20141048

Approval

The thesis titled " Simple Quantum Algorithms for k-mismatch problems" submitted by

1. Ruhan Habib(20141048)

Of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Mathematics on April 28, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Syed Hasibul Hassan Chowdhury
Professor
Department of Mathematics and Natural Sciences
Brac University

Co-Supervisor:
(Member)

Shadman Shahriar
Lecturer
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Md. Firoze H. Haque, PhD
Chairperson and Associate Professor
Department of Mathematics and Natural Sciences
Brac University

Abstract

For many problems, quantum computation has significant advantage compared to classical computers. One of the most surprising example of this is the Grover Search Algorithm, which can search through any unstructured data structure of size n with $O\left(\sqrt{n}\right)$ queries. This algorithm gives quadratic speed-up to many search-based problems. Another important example is the Shor's factorization algorithm, which solves the factorizatio problem with $\tilde{O}\left(n^3\right)$ operations. No polynomial-time classical algorithm for factorization is known to this date. In particular, quantum computing has given significant speed-ups to many string-based problems. String processing is an active and interesting field, with many applications in fields such as bioinformatics. It is also of significant theoretical interest: a subquadratic classical solution (or lack thereof) to string problems such as can shed light to the existence (or lack) of solutions to quite a few problems. If SETH (Strong Exponential Time Hypothesis) is true, string problems such as LCS do not have strongly subquadratic time ($O\left(n^{2-\epsilon}\right)$ for some $\epsilon > 0$) classical solutions. However, we can use quantum computing to get faster solutions to these problems: LCS has a quantum algorithm with $\tilde{O}\left(n^{2/3}\right)$ [13] query complexity in contrast to the classical known best of $\tilde{O}\left(n^2\right)$. In this thesis, we work on the k-mismatch problem. Here, given a pattern and a text, we have to find if the text has a substring with a Hamming distance of at most k from the pattern. This is a "fault-tolerant" version of the well-known string search problem. This important problem has been extensively studied in classical setting but had not been studied through quantum computation until 2022 by Jin and Nogler [16]. In that paper, they provided an $\tilde{O}\left(k\sqrt{n}\right)$-time quantum algorithm and showed that the problem has a quantum query lower-bound of $\Omega\left(\sqrt{kn}\right)$. They posed the question of whether there is a quantum algorithm with better query complexity than $\tilde{O}\left(k^{3/4}n^{1/2+o(1)}\right)$. In 2024, Kociumaka, Nogler, and Wellnitz [18] found an algorithm with optimal query complexity $\tilde{O}(\sqrt{kn})$ and time complexity $\tilde{O}\left(\sqrt{n/m}(\sqrt{km}+k^2)\right)$. This thesis provides simple quantum algorithms for two variants of the k-mismatch problems. The first is when $r = m - k$ is small. For this case, we have a $\tilde{O}(\sqrt{rmn})$-time quantum algorithm. The second algorithm solves the problem in an approximate manner, given a parameter $\epsilon > 0$. It returns an occurrence as a match only if it is a $(1+\epsilon)k$-mismatch. If it does not return any occurrence, then there is no k-mismatch. This algorithm has a time complexity of $\tilde{O}(\epsilon^{-1}\sqrt{mn/k})$.

Keywords: Quantum Computation; String Algorithms; Hamming Distance; k-Matching Problem

Acknowledgement

Thanks to my thesis supervisors Dr. Syed Hasibul Hassan Chowdhury and Mr. Shadman Shahriar and the many faculties of our BracU for their wonderful support.

Table of Contents

Chapter 1

Introduction

String algorithms are of fundamental importance to Computer Science both from a theoretical and practical point-of-views. They have numerous applications in bio-informatics, data-mining and so on. They are connected to both classical [10] and quantum fine-grained complexity theory [13], [17].
In this thesis, we show simple quantum algorithms for two variants of the k-mismatch problem.

1.1 Notations and Basic Definitions

We define $\mathbb{N}$ to be the set of positive integers. We define $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{C}$ as usual: set of integers, set of reals, and set of complex numbers. $\mathbb{R}_+$ denotes the set of positive real numbers. We define $\mathbb{B} = \{0, 1\}$ and $\mathcal{B} = \mathbb{C}^2$. Given vector space V, $\mathbf{L}(V)$ is the set of linear operators acting on V. Given Hilbert space V, $\mathbf{U}(V)$ is the set of unitary operators acting on V. Unless explicitly stated otherwise, all vector spaces are finite-dimensional.
Given $x, y \in \mathbb{B}$, we define $x \oplus y = (x + y) \mod 2$. Given $x = (x_0, \ldots, x_{n-1}), y = (y_0, \ldots, y_{n-1}) \in \mathbb{B}^n$, we define $x \oplus y$ as $z = (x_0 \oplus y_0, \ldots, x_{n-1} \oplus y_{n-1})$.

Definition 1 (Intervals)**.** *Given two integers $L \leq R$, we define*

$$\begin{aligned}
[L..R] &= \{x \in \mathbb{Z} : L \leq x \leq R\} \\
[L..R) &= [L..R] \setminus \{R\} \\
(L..R] &= [L..R] \setminus \{L\} \\
(L..R) &= [L..R] \setminus \{L, R\}
\end{aligned}$$

Given two real numbers $L \leq R$, we define

$$\begin{aligned}
[L, R] &= \{x \in \mathbb{R} : L \leq x \leq R\} \\
[L, R) &= [L, R] \setminus \{R\} \\
(L, R] &= [L, R] \setminus \{L\} \\
(L, R) &= [L, R] \setminus \{L, R\}
\end{aligned}$$

Given any real number $x \in \mathbb{R}$,

$$\begin{aligned}
\lfloor x \rfloor &= \max\{k \in \mathbb{Z} : k \leq x\} \\
\lceil x \rceil &= \min\{k \in \mathbb{Z} : k \geq x\}
\end{aligned}$$

Definition 2 (Big-O and Soft-O notation (O and $\tilde{O}$)**.** *Given functions $f : \mathbb{R}_+^n \times \mathbb{N}^m \to \mathbb{R}_+$, and $g : \mathbb{R}_+^n \times \mathbb{N}^m \to \mathbb{R}_+$, we say that $f(x_1, \ldots, x_n, k_1, \ldots, k_m) = O(g(x_1, \ldots, x_n, k_1, \ldots, k_m))$ if there exists $c > 0$, $\varepsilon_1, \ldots, \varepsilon_n > 0$, and $r_1, \ldots, r_m \in \mathbb{N}$ such that $x_1 \leq \epsilon_1 \wedge \cdots \wedge x_n \leq \epsilon_n$ and $k_1 \geq r_1 \wedge \cdots \wedge k_m \geq r_m$ implies that*

$$f(x_1, \ldots, x_n, k_1, \ldots, k_m) \leq c g(x_1, \ldots, x_n, k_1, \ldots, k_m)$$

.
We say that $f(x_1, \ldots, x_n, k_1, \ldots, k_m) = \tilde{O}(g(x_1, \ldots, x_n, k_1, \ldots, x_m))$ if

$$f(x_1, \ldots, x_n, k_1, \ldots, k_m) = O\left(g(x_1, \ldots, x_n, k_1, \ldots, k_m) \prod_{j=1}^{n} (\log x_j)^{p_j} \prod_{j=1}^{m} (\log k_j)^{q_j} \right)$$

for some $p_1, \ldots, p_n, q_1, \ldots, q_m \in \mathbb{N} \cup \{0\}$.

Our definition of Big-O and Soft-O notation may not be conventional. The intuition here is that for real parameters like error and failure probability, we want to know a function's behavior as they go to 0. For integer paraemeters like input length, we want to a function's behavior as they grow to infinity. The Soft-O ($\tilde{O}$) notation is used to hide and avoid logarithmic factors.

Definition 3 (o and $\tilde{o}$)**.** *Given functions $f : \mathbb{R}_+^n \times \mathbb{N}^m \to \mathbb{R}_+$, and $g : \mathbb{R}_+^n \times \mathbb{N}^m \to \mathbb{R}_+$, we say that $f(x_1, \ldots, x_n, k_1, \ldots, k_m) = o(g(x_1, \ldots, x_n, k_1, \ldots, k_m))$ if for all $c > 0$, there exists $\varepsilon_1, \ldots, \varepsilon_n > 0$, and $r_1, \ldots, r_m \in \mathbb{N}$ such that $x_1 \leq \epsilon_1 \wedge \cdots \wedge x_n \leq \epsilon_n$ and $k_1 \geq r_1 \wedge \cdots \wedge k_m \geq r_m$ implies that*

$$f(x_1, \ldots, x_n, k_1, \ldots, k_m) \leq c g(x_1, \ldots, x_n, k_1, \ldots, k_m)$$

.
We say that $f(x_1, \ldots, x_n, k_1, \ldots, k_m) = \tilde{o}(g(x_1, \ldots, x_n, k_1, \ldots, x_m))$ if

$$f(x_1, \ldots, x_n, k_1, \ldots, k_m) = o\left(g(x_1, \ldots, x_n, k_1, \ldots, k_m) \prod_{j=1}^{n} (\log x_j)^{p_j} \prod_{j=1}^{m} (\log k_j)^{q_j} \right)$$

for some $p_1, \ldots, p_n, q_1, \ldots, q_m \in \mathbb{N} \cup \{0\}$.

Definition 4 (Ω and $\tilde{\Omega}$)**.** *Given functions $f : \mathbb{R}_+^n \times \mathbb{N}^m \to \mathbb{R}_+$, and $g : \mathbb{R}_+^n \times \mathbb{N}^m \to \mathbb{R}_+$, we say that $f(x_1, \ldots, x_n, k_1, \ldots, k_m) = \Omega(g(x_1, \ldots, x_n, k_1, \ldots, k_m))$ if $g(x_1, \ldots, x_n, k_1, \ldots, k_m) = O(f(x_1, \ldots, x_n, k_1, \ldots, k_n))$.*
We say that $f(x_1, \ldots, x_n, k_1, \ldots, k_m) = \tilde{\Omega}(g(x_1, \ldots, x_n, k_1, \ldots, k_m))$ if $g(x_1, \ldots, x_n, k_1, \ldots, k_m) = \tilde{O}(f(x_1, \ldots, x_n, k_1, \ldots, k_n))$.

Definition 5 (ω and $\tilde{\omega}$)**.** *Given functions $f : \mathbb{R}_+^n \times \mathbb{N}^m \to \mathbb{R}_+$, and $g : \mathbb{R}_+^n \times \mathbb{N}^m \to \mathbb{R}_+$, we say that $f(x_1, \ldots, x_n, k_1, \ldots, k_m) = \omega(g(x_1, \ldots, x_n, k_1, \ldots, k_m))$ if $g(x_1, \ldots, x_n, k_1, \ldots, k_m) = o(f(x_1, \ldots, x_n, k_1, \ldots, k_n))$.*
We say that $f(x_1, \ldots, x_n, k_1, \ldots, k_m) = \tilde{\omega}(g(x_1, \ldots, x_n, k_1, \ldots, k_m))$ if $g(x_1, \ldots, x_n, k_1, \ldots, k_m) = \tilde{o}(f(x_1, \ldots, x_n, k_1, \ldots, k_n))$.

The following are a few examples of the notaion above:

$$\begin{aligned}
\frac{3}{2}n\log n + n &= O\left(n\log n\right)\\
\frac{3}{2}n\log n &= \tilde{O}\left(n\right)\\
n &= o(n\log n)\\
n\log n &\neq o(n\log n)\\
n\log n &= \Omega(n)\\
n\log n &= \omega(n)\\
\frac{n}{\log n} &= \tilde{\Omega}(n)
\end{aligned}$$

Note that the definition is somewhat redundant: for any pair of functions $f : \mathbb{N} \to \mathbb{R}$ and $g : \mathbb{N} \to \mathbb{R}$, $f(n) = \tilde{O}(g(n))$ if and only if $f(n) = \tilde{o}(g(n))$; $f(n) = \tilde{\Omega}(g(n))$ if and only if $f(n) = \tilde{\Omega}(g(n))$.

Definition 6 (Strings)**.** *Let Σ be a finite set. Define $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$. Note that $|\Sigma^0| = 1$. Given any $A \in \Sigma^*$, we say that A is a* string *with* alphabet Σ*. We define the* length *of A as the unique integer $n \geq 0$ such that $A \in \Sigma^n$. We let $|A|$ denote the length of A. Given any $j \in [0..|A|)$, we define A_j and $A[j]$ to be the j-th component of A. If $A = (a_0, \ldots, a_{|A|-1})$, then $A_j = A[j] = a_j$.*

We often place letters from the alphabet side by side to represent a string. Formally, given an alphabet Σ and $s_0, \ldots, s_{n-1} \in \Sigma$, $s_0 \ldots s_{n-1}$ represents $(s_0, \ldots, s_{n-1}) \in \Sigma^n$. Also, we often deal with binary strings: they are strings with alphabet $\mathbb{B}$. Examples of binary strings include: "" (the empty binary string), "0", "010101011".

Definition 7 (Hamming Distance)**.** *Given two strings $A, B \in \Sigma^*$ for some alphabet Σ, we define $\delta_H(A, B)$ as follows:*

$$\delta_H(A,B) = \begin{cases} |\{i \in [0..|A|) : A[i] \neq B[i]\}| & |A| = |B| \\ \infty & |A| \neq |B| \end{cases}$$

Definition 8 (Quantum Oracle)**.** *Let n be a positive integer. Given a boolean function $F : \mathbb{B}^n \to \mathbb{B}^m$, we say that a unitary $U \in \mathbf{U}\left(\mathcal{B}^{\otimes(n+m)}\right)$ is an* oracle *for F if, for all $x \in \mathbb{B}^n$ and $y \in \mathbb{B}^m$, we have $U|x, y\rangle = |x, y \oplus F(x)\rangle$. Here, $|0\rangle, \ldots, |2^n - 1\rangle$ form an orthonormal basis for $\mathcal{B}^{\otimes n}$ and $|0\rangle, \ldots, |2^m - 1\rangle$ form an orthonormal basis for $\mathcal{B}^{\otimes m}$. We say that we have* oracle access *to F if we have access to U.*

1.2 Problem Statement

Definition 9 (k-mismatch Problem)**.** *An algorithm decides the k-*mismatch matching problem *if, given oracle access a text string T of length n, a pattern string P of length m, and a positive integer k, the algorithm reports the existence of $i \in [0..n-m]$ such that $\delta_H(T[i..i+m), P) \leq k$. A quantum algorithm decides the problem if, given T, P, and k as defined above, it outputs a correct result (upon measurement) with probability of at least $2/3$.*

There may be slight variation to the definition depending on the paper. Some papers may define this problem by requiring the output of a list $d_0, \ldots, d_{n-1}$ of n integers, where $d_i = \min(k+1, \delta_H(T[i..i+m-1], P))$ for $i \in [0..n)$.
Unless otherwise stated, we assume that the alphabet size $|\Sigma|$ is polynomially bounded above by n.

1.3 Our Contribution

The following are the two main results of this thesis:

Theorem (Theorem 10)**.** *There exists a quantum algorithm that, given oracle access to a pattern P of length m and a text T of length n, and an integer threshold $r > 0$, reports the existence of $i \in [0..n-m]$ such that $\delta_H(T[i..i+m), P) \leq m - r$. Furthermore, this algorithm has time complexity $\tilde{O}(\sqrt{rmn})$.*

Theorem (Theorem 11)**.** *There exists a quantum algorithm that, given oracle access to a pattern P of length m and a text T of length n, an integer threshold $k > 0$, and $\epsilon \in (0, 1]$, such that:*

- *if there exists an $j \in [0..n-m]$ such that $\delta_H(T[j..j+m), P) \leq k$, then the algorithm, upon measurement, outputs $(j', 1)$ for some $j' \in [0..n-m]$ satisfying $\delta_H(T[j'..j'+m), P) \leq (1+\epsilon)k$ with a probability of at least $9/10$;*
- *if, for all $j \in [0..n-m]$, we have $\delta_H(T[j..j+m), P) > (1+\epsilon)k$, then the algorithm, upon measurement, outputs $(j', 0)$ for some $j' \in [0..2n-1]$ with probability of at least $9/10$.*

This algorithm has time complexity $\tilde{O}\left(\sqrt{\frac{mn}{k}}\right)$ (assuming that P and T can be accessed in $\tilde{O}(1)$ time).

An overview of the two theorems is provided in section 1.7. The theorems are actually proved in sections 3.2 and 3.3 respectively.

1.4 Classical Computation

First, we provide (very) brief overview of classical computation.
The exposition of this section is based on [5] and [8].

1.4.1 Turing Machines

First, we shall introduce Turing machines, which encapsulate what we *mean* by algorithms.

Definition 10 (Turing machine)**.** *A Turing Machine (TM) M is a tuple $(S, \sqcup, A, Q, q_0, \delta)$ where $\sqcup \in S$, $A \subseteq S \setminus \{\sqcup\}$, $q_0 \in Q$, and $\delta : Q \times S \to Q \times S \times \{-1, 0, +1\}$.*

- *S is a finite set and is called the* alphabet *of M.*
- *A is a finite set and is called the* external alphabet *of M.*
- *$\sqcup$ is called the* blank symbol *of M.*

- *Q is another finite set, elements of which are called* states *of M.*
- *δ is called the* transition state *of M.*

Note that this is not the only definition of a Turing machine (or even algorithms). There are other equivalent notions of computations: equivalent in the sense that any any algorithm in the first model can be simulated in the other. This is the informal assertion the "Church-Turing Thesis": that Turing Machines encapsulate the idea of algorithms.
Now, we describe how a Turing Machine M "computes".
Formally, we can describe the configuration of a TM at any step of the computation as a 3-tuple $(\sigma, p, q) \in S^\infty \times (\mathbb{N} \cup \{-1, 0\}) \times Q$. Here, S^∞ is defined as the set of all sequences $(s_j)_{j=0}^\infty$. We say that σ is the content of the *tape* of the Turing machine; p is the position of the *head* of the Turing machine, and; q is the *state* of the Turing machine. We say that the cell j (integer $j \geq 0$) of the tape contains s_j if the tape content is $\sigma = (s_j)_{j=0}^\infty$. We can refer to cell j as σ_j as well.
The tape of the Turing Machine M is initially in the state $\sigma = w_0 \dots w_{n-1} \sqcup \sqcup \dots$, where $w_0, \dots, w_{n-1} \in A$. More precisely, for $j \in [0..n-1]$, the cell j of the tape is given by w_j. For integer $j \geq n$, the cell j is given by $\sqcup$. Here, $w = w_0 \dots w_{n-1}$ is called the *input string* of the Turing Machine. Initially, there is a head (or pointer) on cell 0. The state of the TM is initially q_0. The initial configuration of the TM is thus $C_0 = (\sigma, 0, q_0)$.
If, at any point, we have $p = -1$, the computation halts. We shall describe what this formally means later.
At every step of the "computation", the transition function δ determines the next setup. Let $C_1 = (\sigma, p, q)$ be a valid configuration. If $p = -1$, then there is no next step. Otherwise, let $(q', s', \Delta_p) = \delta(q, \sigma_p)$. Define σ' by letting $\sigma'_j = \sigma_j$ if $j \neq p$ and $\sigma'_j = s'$ if $j = p$. The configuration of the TM in the next step is precisely $C_2 = (\sigma', q', p + \Delta_p)$. We say that the configuration C_1 *yields* the configuration C_2. Informally, the transition function rewrites cell j, updates the state, and moves the head (left, right, or not at all).
The *starting configuration* C_0 of M with input w is the configuration $(\sigma, 0, q_0)$ where $\sigma = w_0 \dots w_{n-1} \sqcup \sqcup \dots$. A valid *ending configuration* of M is a configuration (σ, p, q) where $p = -1$.
We say that the Turing machine M outputs $s = s_0 \dots s_{m-1} \in A^*$ on input w if there is a sequence of configurations $C_0, \dots, C_k$ such that

- C_0 is the starting configuration of M with input w;
- $C_k = (\sigma, p, q)$ is a valid ending configuration of M with $\sigma = s_0 \dots s_{m-1} \sqcup \sqcup \dots$;
- for each $j \in [0..k-1]$, C_j yields C_{j+1}.

Such sequence, if it exists, must be unique.

Definition 11 (Computability)**.** *Let Σ be some alphabet (that is, a finite set). We say that a function $f : \Sigma^* \to \Sigma^*$ is a* computable function *if there exists a Turing machine M which outputs $f(w)$ on input w for every $w \in \Sigma^*$. We say that M computes f.*

Definition 12 (Time Complexity of a Turing Machine)**.** *Suppose that a TM M computes a function $f : \Sigma^* \to \Sigma^*$. We define the* time complexity *of M as the function $T : \mathbb{N} \cup \{0\} \to \mathbb{N}$ such that $T(n)$ is the maximum number of steps that M uses on any input of length n. We say that M uses t steps on input x if the unique sequence of configuration $C_0, \ldots, C_k$ for output $f(x)$ given input x satisfies $k = t$. We say that M computes f in time $T(n)$.*

If M computes $f(x)$ for an input x in t steps, we can also say that M computes $f(x)$ in time t.

Definition 13 (Space Complexity of a Turing Machine)**.** *Suppose that a TM M computes a function $f : \Sigma^* \to \Sigma^*$. We define the* space complexity *of M as the function $f : \mathbb{N} \cup \{0\} \to \mathbb{N}$ such that $f(n)$ is the maximum number of space that M uses on any input of length n. We say that M uses s space on input x if the maximum of $|x|$ and the maximum value of $p + 1$ (the position of the head) is s.*

Informally, the space complexity of a TM M tells us the maximum amount of the tape that has actually been used.

1.4.2 RAM Turing Machines

Informally speaking, a RAM Turing Machine is a Turing Machine with random access: we can read from and write to any cell j of the TM's tape. We shall formalize this as follows[7].

Definition 14 (RAM Turing machine)**.** *A RAM Turing Machine (RAM TM) M is a tuple $M = (S, \sqcup, R, W, A, Q, q_0, q_{access}, \delta)$ where $\sqcup, R, W \in S$, $A \subseteq S \setminus \{\sqcup, R, W\}$, $|\{\sqcup, R, W\}| = 3$, $q_0 \neq q_{access} \in Q$, and $\delta : Q \times S^2 \to Q \times S^2 \times \{-1, 0, +1\}^2$. The $S, \sqcup, A, Q, q_0$ are defined in the same manner as Turing machines.*

Formally, we can describe the configuration of a RAM TM at any step of the computation as a 4-tuple $(\sigma, \sigma', p, p', q) \in (S^\infty))^2 \times (\mathbb{N} \cup \{-1, 0\})^2 \times Q$. Here, S^∞ is defined as the set of all sequences $(s_j)_{j=0}^\infty$. We say that σ is the content of the *main tape* of the RAM Turing machine; σ' is the content of the *address tape*; p is the position of the *head of the main tape* of the RAM Turing machine; p' is the position of the *head of the address tape*, and; q is the *state* of the RAM Turing machine. We say that the cell j (integer $j \geq 0$) of the main tape contains s_j if the main tape content is $\sigma = (s_j)_{j=0}^\infty$. Similarly, we say that the cell j of the address tape contains s_j if the address tape content is $\sigma' = (s_j)_{j=0}^\infty$. We can refer to cell j of the main tape as σ_j and cell j of the address tape as σ'_j as well.
The main tape of the RAM Turing Machine M is initially in the state $\sigma = w_0 \ldots w_{n-1} \sqcup \sqcup \ldots$, where $w_0, \ldots, w_{n-1} \in A$. More precisely, for $j \in [0..n-1]$, the cell j of the tape is given by w_j. For integer $j \geq n$, the cell j is given by $\sqcup$. Here, $w = w_0 \ldots w_{n-1}$ is called the *input string* of the Turing Machine. The address tape is intially $\sigma' = \sqcup \ldots$. Initially, there is a head (or pointer) on cell 0 of the main tape and the address tape. The state of the TM is initially q_0. The initial configuration of the TM is thus $C_0 = (\sigma, \sigma', 0, 0, q_0)$.
If, at any point, we have $p = -1$ or $p' = -1$, the computation halts. We shall describe what this formally means later.
At every step of the "computation", the transition function δ determines the next setup. Let $C_1 = (\sigma, \sigma', p, p', q)$ be a valid configuration. If $p = -1$ or $p' = -1$,

then there is no next step. Otherwise, let $(q', s', w', \Delta_p, \Delta_{p'}) = \delta(q, \sigma_p, \sigma'_{p'})$. Define $\sigma^{(0)}$ and $\sigma^{(1)}$ as follows If $q' = q_{\text{access}}$ and σ' is of the form $\langle j \rangle R$, where $\langle j \rangle$ is a representation of the non-negative integer j, then $\sigma^{(0)} = \sigma$ and $\sigma^{(1)} = \langle j \rangle R \sigma_j$. If $q' = q_{\text{access}}$ and σ' is of the form $\langle j \rangle W c$ where $\langle j \rangle$ is a representation of a non-negative integer j and $c \in S$, then $\sigma^{(0)}_k = \sigma_k$ for all $j \neq k$, and $\sigma^{(0)}_j = c$, and $\sigma^{(1)} = \sigma'$. If none of the previous two cases hold, then $\sigma^{(0)} = \sigma$ and $\sigma^{(1)} = \sigma'$.
Define $\sigma^{(2)}$ by letting $\sigma^{(2)}_j = \sigma^{(0)}_j$ if $j \neq p$ and $\sigma^{(2)}_j = s'$ if $j = p$. Similarly define $\sigma^{(3)}$ by letting $\sigma^{(3)}_j = \sigma^{(1)}_j$ if $j \neq p'$ and $\sigma^{(3)}_j = w'$ if $j = p$. The configuration of the RAM TM in the next step is precisely $C_2 = (\sigma^{(2)}, \sigma^{(3)}, q', p + \Delta_p, p' + \Delta_{p'})$. We say that the configuration C_1 *yields* the configuration C_2.
For the sake of clarity, let us informally redescribe a step of computation in RAM TM (how C_0 yields C_1):

- First, we handle the address tape
  - If the state is in q_{access} and the address tape is of the form $\langle j \rangle R$, then the content of main tape's cell j is placed to the right of R.
  - If the state is in q_{access} and the address tape is of the form $\langle j \rangle W c$, then the content of main tape's cell j is set to c.
  - If neither of the two cases mentioned above is satisfied, then do nothing.
- We use the transition function δ to determine the next state, cell contents of cells the heads are on, and whether to move the head (for the main tape and the address tape) to the left, right, or keep it in place. This step is essentially the same as that for a Turing machine.

The *starting configuration* C_0 of M with input w is the configuration $(\sigma, \sigma', 0, 0, q_0)$ where $\sigma = w_0 \dots w_{n-1} \sqcup \sqcup \dots$ and $\sigma' = \sqcup \dots$. A valid *ending configuration* of M is a configuration $(\sigma, \sigma', p, p', q)$ where $p = -1$ or $p' = -1$.
We say that the Turing machine M outputs $s = s_0 \dots s_{m-1} \in A^*$ on input w if there is a sequence of configurations $C_0, \dots, C_k$ such that

- C_0 is the starting configuration of M with input w;
- $C_k = (\sigma, \sigma', p, p', q)$ is a valid ending configuration of M with $\sigma = s_0 \dots s_{m-1} \sqcup \sqcup \dots$;
- for each $j \in [0..k-1]$, C_j yields C_{j+1}.

Such sequence, if it exists, must be unique.

1.4.3 Boolean circuits

Boolean circuits are another model of computation.
A *Boolean function* is defined to be a function $f : \mathbb{B}^n \to \mathbb{B}$, where $n \geq 1$ is an integer.
Let $\mathcal{A}$ be some fixed set of Boolean functions. We shall call $\mathcal{A}$ a *basis* and its elements *gates*. A *circuit* C over $\mathcal{A}$ has n input variables $x_0, \dots, x_{n-1}$, m auxiliary variables $y_0, \dots, y_{m-1}$, and m assignments of the form $y_j := f_j(u_1, \dots, u_r)$ for $j \in [0..m-1]$ (done in ordered manner with assignment of y_j done before $y_{j'}$ if and only if $j < j'$),

where $f_j \in \mathcal{A}$ and each of $u_1, \ldots, u_r$ is either an input variable or an auxiliary variable y_k with $k < j$.
We can select $1 \leq o \leq m$ of the auxiliary variables as output variables. Then the values of those auxiliary variable is the *result* of the computation. A circuit with n input variables and o output variables computes a function $F : \mathbb{B}^n \to \mathbb{B}^o$.
For example, the following is a circuit over $\mathcal{A} = \{\wedge, \oplus\}$: input variables x_0, x_1, x_2, auxiliary variables y_0, y_1, $y_0 := x_0 \oplus x_1$, $y_1 := y_0 \oplus x_2$, and output variable y_1. Note that $\wedge$ represents multiplication modulo 2 and $\oplus$ represents addition modulo 2. This circuit computes the parity of a length 3 Boolean string. If the input is 011, then $x_0 = 0, x_1 = x_2 = 1$, $y_0 = 0 \oplus 1 = 1$, $y_1 = 1 \oplus 1 = 0$, and thus the output is 0.
We shall use the basis $\mathcal{A} = \{\neg, \vee, \wedge, \oplus\}$ for our Boolean circuits, though any complete basis will do. A basis $\mathcal{A}$ is *complete* if, for any Boolean function f, there is a circuit over $\mathcal{A}$ that computes f. The boolean functions in $\mathcal{A}$ are defined as follows: for $x, y \in \mathbb{B}$,

$$\begin{aligned} \neg x &= (1 + x) \mod 2 \\ x \wedge y &= xy \\ x \vee y &= \neg(\neg x \wedge \neg y) \\ \oplus &= (x \wedge \neg y) \vee (\neg x \wedge y) \end{aligned}$$

Note that $\neg$ has higher operator precedence than other Boolean functions (i.e. $\wedge, \vee, \oplus$). So $x \wedge \neg y = x \wedge (\neg y)$.

Theorem 1. *The basis $\{\neg, \wedge\}$ is complete.*

Proof. Note that $\vee$ can be realized using $\wedge$ and $\neg$ gates.
Let $f : \mathbb{B}^n \to \mathbb{B}$ be a Boolean function. Let X be the set of subsets of $[0..n-1]$. Let $I : \mathbb{B}^n \to X$ and $J : \mathbb{B}^n \to X$ be defined as follows: given $x_0, \ldots, x_{n-1} \in \mathbb{B}$,

$$\begin{aligned} I(x_0, \ldots, x_{n-1}) &= \{j \in [0..n-1] : x_j = 1\} \\ J(x_0, \ldots, x_{n-1}) &= \{j \in [0..n-1] : x_j = 0\} \end{aligned}$$

. Then, for $x_0, \ldots, x_{n-1} \in \mathbb{B}$,

$$f(x_0, \ldots, x_{n-1}) = \bigvee_{u \in f^{-1}(\{1\})} \left(\left(\bigwedge_{j \in I(u)} x_j \right) \wedge \left(\bigwedge_{j \in J(u)} \neg x_j \right) \right)$$

Clearly, $\{\neg, \wedge\}$ is a complete basis. □

The number of assignments m in a circuit C is called its *size*.
From the proof of the preceding theorem, we see that any Boolean valued function $F : \mathbb{B}^n \to \mathbb{B}$ can be computed using a $O(n2^n)$-size circuit over the basis $\{\neg, \wedge\}$.
Given any complete bases $\mathcal{A}$ and $\mathcal{A}'$ and a circuit C of size m over $\mathcal{A}$, there is a circuit C' of size $O(m)$ over $\mathcal{A}'$ that computes the same function $F : \mathbb{B}^n \to \mathbb{B}^o$. This is because for any Boolean function $f \in \mathcal{A}$, there is a Boolean circuit C'' over $\mathcal{A}$ that computes f. Since $\mathcal{A}$ is finite, the size of these circuits is bounded by a constant. So we can replace every assignment in C with their corresponding Boolean circuits over $\mathcal{A}'$ to get a circuit over $\mathcal{A}'$ that computes the same function F with at most a constant scaling in the size of the circuit.
This is why the exact basis is not important, as long as it is complete.
Next we show that any Turing Machine can be simulated by a sequence of Boolean circuits.

Theorem 2. *Let $M = (S, \sqcup, A, Q, q_0, \delta)$ be a TM that computes $F : A^* \to A^*$ in time upper bounded by $T(n)$ and space upper bounded by $s(n)$ on input size n. Then there is a sequence of Boolean circuits $(C_n)_{n=0}^{\infty}$ of size $O(T(n)s(n))$ such that the output of the Boolean circuit for input (which is a binary representation of $x \in A^*$) is a binary representation of $F(x)$.*

Proof. Let $n \in \mathbb{N}$ We shall construct the Boolean circuit C_n.
We can represent the configuration Γ_t of M at step t using $(m_S + m_Q)(s(n) + 1)$ Booleans, where $m_S = \lceil \log_2 |S| \rceil$ and $m_Q = \lceil \log_2 (|Q| + 1) \rceil$. Let us partition and name this Booleans as $\Gamma_{t,0}, \ldots, \Gamma_{t,s(n)}$. For each $j \in [0..s(n)]$, $\Gamma_{t,j}$ is a binary representation of $S \times (\{\Lambda\} \cup Q)$, where $\Lambda \notin Q$.
For $j \geq 1$, the first m_S bits (Booleans) of $\Gamma_{t,j}$ to represent the content of the cell $j-1$. The next m_Q bits represent the state if the head is in cell $j-1$ and otherwise represent Λ. The first m_S bits of $\Gamma_{j,0}$ represent $\sqcup$, while the next m_Q bits represent the state if the head has gone to the left of cell 0 (that is, $p = -1$) and otherwise those bits represent Λ. Since the space used is upper bounded by $s(n)$, we thus need $(m_S + m_Q)(s(n) + 1)$ bits to represent Γ_t.
Now, we observe that given $t \in [0..T(n))$ and $j \in [0..s(n)]$, $\Gamma_{t+1,j}$ only depends on $\Gamma_{t,j-1}$, $\Gamma_{t,j}$, and $\Gamma_{t,j+1}$ if they are well-defined. This is because at each step, the head of a TM can only change its position by at most 1, and at most one cell's value can change. Suppose that at step t, the head of the TM is at cell $j-1$. Then the previous head must be at cell $j-1-1$, $j-1$ or $j+1-1$. Then $\Gamma_{t+1,j}$ will depend on either $\Gamma_{t,j-1}$, $\Gamma_{t,j}$, or $\Gamma_{t,j+1}$. If the value of cell $j-1$ changes from step t to step $t+1$, then the head must have been on $\Gamma_{t,j}$ and thus $\Gamma_{t+1,j}$ depends on $\Gamma_{t,j}$. If cell $j-1$ has not changed and is not the current head position, then $\Gamma_{t+1,j} = \Gamma_{t,j}$.
Of course, we need to handle $j = 0$ specially: if the head is in $\Gamma_{t,0}$, then $\Gamma_{t+1} = \Gamma_t$.
In other words, given $t \in [0..T(n))$ and $j \in (0..s(n))$, $\Gamma_{t+1,j} \in \mathbb{B}^{m_S+m_Q}$ is a function of at most $3(m_S + m_Q)$ Boolean variables. Thus there is a constant size Boolean circuit (that is, the size does not depend on n) that computes $\Gamma_{t+1,j}$. Using $s(n)$ of these circuits, we can compute Γ_{t+1} from Γ_t for any $t \in [0..T(n))$.
We can compute Γ_0 from the $m_S \cdot n$-bit representation of the input $x \in \mathbb{B}^n$ very easily. For each $j \in [0..n-1]$, the first m_S bits of $\Gamma_{0,j+1}$ are copied from jm_S-th through $(j+1)m_S - 1$-th bits of the input. For $j \in [0..s(n)] \setminus [1..n]$, set the first m_S bits of $\Gamma_{0,j}$ to the binary representation of $\sqcup$. Set the last m_Q bits of $\Gamma_{0,1}$ to the binary representation of q_0. For $j \in [0..s(n)] \setminus \{1\}$, set the last m_Q bits of $\Gamma_{t,j}$ to the binary representation of Λ. This computation can be done using an $O(n)$-sized Boolean circuit.
Since we can compute Γ_0 and Γ_{t+1} from Γ_t for $t \in [0..T(n))$, we can compute $T(n)$ using an $O(T(n)s(n))$ sized circuit.
Denote the first m_Q bits of each of $\Gamma_{T(n),1}, \ldots, \Gamma_{T(n),s(n)}$ as output variables. □

This proof uses the fact that computation (via Turing Machines) is "local". The content of a cell at any steps depends only on the state of the TM and the cell's immediate neighbors.
We define the *fan-out* of a Boolean circuit as the number of times an input variable or an auxiliary variable is used. We shall assume henceforth that the fan-out is bounded by some constant.
We shall now give a brief example of Boolean circuits:

Lemma 1. *There is a sequence C_n of Boolean circuits such that for $n \in \mathbb{N}$, C_n has size $O(n)$ and compares two n-bit non-negative integers.*

Proof. Let $x_0, \ldots, x_{n-1}$ and $y_0, \ldots, y_{n-1}$ be the input variables. Let $x = 2^0 x_0 + \cdots + 2^{n-1} x_{n-1}$ and $y = 2^0 y_0 + \cdots + 2^{n-1} y_{n-1}$. We wish to compute (f, g), where $x = y$ implies that $f = 1$, $x < y$ implies that $f = 0 \wedge g = 0$, and $x > y$ implies that $f = 0 \wedge g = 1$. Let $a_0, \ldots, a_n$ and $b_0, \ldots, b_n$ denote auxiliary variables. Let $a_n = b_n = 1$. For $j \in [0..n-1]$,

$$(a_j, b_j) := \begin{cases} (a_{j+1}, b_{j+1}) & a_j = 0 \\ (1, 0) & x_j = y_j \\ (0, 0) & x_j < y_j \\ (0, 1) & x_j > y_j \end{cases}$$

Let $f := a_0$ and $g := b_0$. Thus, we have our Boolean circuit. If $x = y$, then we get $f = g = 1$. Otherwise, letting j be the largest such that $x_j \neq y_j$, we have $a_{j+1} = 1$ but $a_j = 0$ and $b_j = 0 \Leftrightarrow x_j < y_j$. So if $x \neq y$, we also get the correct result. Clearly, we have an $O(n)$ size circuit for comparing two n-bit numbers. □

We shall now discuss reversible boolean circuits.

Definition 15. *Let k and n be integers such that $k \leq n$. Let $A = \{a_0, \ldots, a_{k-1}\} \subseteq [0..n-1]$ be an ordered set and let $G : \mathbb{B}^k \to \mathbb{B}^k$ be bijective. We define $G[A]_n$ as a bijective function $G[A]_n : \mathbb{B}^n \to \mathbb{B}^n$ such that given $(x_0, \ldots, x_{n-1}) \in \mathbb{B}^n$ and $(y_0, \ldots, y_{n-1}) = G[A]_n(x_0, \ldots, x_{n-1})$,*

$$\begin{aligned} (y_{a_0}, \ldots, y_{k-1}) &= G\left(x_{a_0}, \ldots, x_{a_{k-1}}\right) \\ y_j &= x_j \qquad\qquad \forall j \notin A \end{aligned}$$

For example, let $G : \mathbb{B}^2 \to \mathbb{B}^2$ be defined as $G(x, y) = (x, y \oplus x)$ for all $(x, y) \in \mathbb{B}^2$. Then $G[\{2, 1\}]_3(x, y, z) = (x, y \oplus z, z)$ for all $(x, y, z) \in \mathbb{B}^3$.
Now, let us define reversible Boolean ciruits.

Definition 16 (Reversible Boolean Circuits)**.** *Let $\mathcal{A}$ be a set of bijective functions of the form $G : \mathbb{B}^k \to \mathbb{B}^k$, where k is not fixed (different elements of $\mathcal{A}$ may be permutations on $\mathbb{B}^k$ for different values of k). A* reversible classical circuit over the basis $\mathcal{A}$ *is a sequence of permutations $G_0[A_0]_n, \ldots, G_{l-1}[A_{l-1}]_n$ where $G_j \in \mathcal{A}$ and $A_j \subseteq [0..n-1]$ for all $j \in [0..l-1]$. Of course, we must have $n \geq k_j$ for all $j \in [0..l-1]$ as well, where k_j is the integer such that G_j is a permutation of $\mathbb{B}^{k_j}$. We say that this reversible circuit realizes the permutation $G = G_{l-1}[A_{l-1}] \cdots G_0[A_0]_n$.*

Definition 17 (Reversible Circuit using Ancillas)**.** *Suppose that we have a reversible circuit $G_0[A_0]_N, \ldots, G_{l-1}[A_{l-1}]_N$. This circuits realizes the permutation $W = G_{l-1}[A_{l-1}]_N \cdots G_0[A_0]_N$. We say that this circuit realizes a permutation $G : \mathbb{B}^n \to \mathbb{B}^n$ using ancillas if $N \geq n$ and $W(x, 0, \ldots, 0) = (Gx, 0, \ldots, 0)$ for all $x \in \mathbb{B}^n$.*

While reversible Boolean circuits may feel more restrictive than Boolean circuits as reversible circuits only compute permutations, any Boolean circuit has an "equivalent" reversible Boolean circuits. We shall express this "equivalence" in Lemma 2.

First, note that for any function $F : \mathbb{B}^n \to \mathbb{B}^m$, there exists a permutation $F_\oplus : \mathbb{B}^{n+m} \to \mathbb{B}^{n+m}$ defined by $F_\oplus(x, y) = (x, y \oplus F(x))$ for all $x \in \mathbb{B}^n$ and $y \in \mathbb{B}^m$. Here, $(y_0, \ldots, y_{m-1}) \oplus (y'_0, \ldots, y'_{m-1}) = (y_0 \oplus y'_0, \ldots, y_{m-1} \oplus y'_{m-1})$ for all $(y_0, \ldots, y_{m-1}), (y'_0, \ldots, y'_{m-1}) \in \mathbb{B}^m$. This definition is convenient, because then we have $F(x, 0) = (x, F(x))$ for all $x \in \mathbb{B}^n$.

Lemma 2. *Suppose that the function $F : \mathbb{B}^n \to \mathbb{B}^m$ is realized by a Boolean circuit of size L over some basis $\mathcal{A}$. Then we can realize $F_\oplus$ by a reversible circuit of size $O(L + n + m)$ using ancillas over the basis $\mathcal{A}_\oplus$. The basis $\mathcal{A}_\oplus$ is defined as $\mathcal{A}_\oplus = \{f_\oplus : f \in \mathcal{A}\} \cup I_\oplus$, where $I : \mathbb{B} \to \mathbb{B}$ is the identity function.*

Proof. First, we shall build a reversible circuit C that computes F along with additional "garbage". More precisely, C realizes a permutation that maps $(x, 0) \mapsto (F(x), G(x))$,
where G is the "garbage" that we shall later remove.
Let C_0 be the Boolean circuit that computes F. Let $x_0, \ldots, x_{n-1}$ be the input variables and let $x_n, \ldots, x_{n+L-1}$ be the auxiliary variables (including the output variables). We construct a reversible circuit C' by replacing, for each $j \in [0..L-1]$, the assignment $x_{n+j} := f_j(x_{k_{j,0}}, x_{k_{j,l_j}})$ with $x_{n+j} := x_{n+j} \oplus f_j(x_{k_{j,0}}, \ldots, x_{k_{j,l_j}})$. So the reversible circuit C' is given by

$$(f_0)_\oplus[\{k_{0,0}, \ldots, k_{0,l_0}, n+0\}]_{n+L}, \ldots, (f_{L-1})_\oplus[\{k_{L-1,0}, \ldots, k_{L-1,l_{L-1}}, n+L_1\}]_{n+L}$$

.
On input $(x, 0)$, the variables $x_n, \ldots, x_{n+L-1}$ are initially 0 and thus their final value in C' is the same as that of C_0.
We note that we can swap two variables using the function $(\leftrightarrow) : \mathbb{B}^2 \to \mathbb{B}^2$ defined by $(\leftrightarrow)(a, b) = (b, a)$ for all $(a, b) \in \mathbb{B}^2$. We can realize $(\leftrightarrow)$ as follows:

$$(\leftrightarrow) = I_\oplus[\{0,1\}]_2 I_\oplus[\{1,0\}]_2 I_\oplus[\{0,1\}]_2$$

This can be seen by direct application: given $(a, b) \in \mathbb{B}^2$,

$$\begin{aligned}
(\leftrightarrow)(a, b) &= I_\oplus[\{0,1\}]_2 I_\oplus[\{1,0\}]_2 I_\oplus[\{0,1\}]_2(a, b) \\
&= I_\oplus[\{0,1\}]_2 I_\oplus[\{1,0\}]_2(a, b \oplus a) \\
&= I_\oplus[\{0,1\}]_2(a \oplus (b \oplus a), b \oplus a) \\
&= I_\oplus[\{0,1\}]_2(b, b \oplus a) \\
&= (b, (b \oplus a) \oplus b) \\
&= (b, a)
\end{aligned}$$

Thus, using at most L application of $(\leftrightarrow)$ at the end of C', we can bring the output variables to the first m bits of the result. We thus have our circuit C which realizes the map of the form $(x, 0) \mapsto (F(x), G(x))$ for all $x \in \mathbb{B}^n$.
For any $f : \mathbb{B}^k \to \mathbb{B}^r$, $f_\oplus$ is its own inverse: given $x \in \mathbb{B}^k$ and $y \in \mathbb{B}^r$,

$$f_\oplus(f_\oplus(x, y)) = f_\oplus(x, y \oplus f(x)) = (x, (y \oplus f(x)) \oplus f(x)) = (x, y)$$

Therefore, we can construct the inverse circuit C^{-1} of C by simply applying the gates applied in C in reverse order.
We know construct reversible circuit $\tilde{C}$ that realizes, with ancillas, the map $(x, y) \mapsto (x, y \oplus F(x))$ for $x \in \mathbb{B}^n, y \in \mathbb{B}^m$. More precisely, the circuit realizes a permutation F' on $\mathbb{B}^{n+L+m}$ such that $F'(x, 0, y) = (x, 0, y \oplus F(x))$ for all $x \in \mathbb{B}^n$ and $y \in \mathbb{B}^m$. The circuit works as follows:

- Apply $C[\{0, \ldots, n-1, n, \ldots, n+L-1\}]$.
- Apply $I_\oplus[\{0, n+L\}], \ldots, I_\oplus[\{m-1, n+L+m-1\}]$.
- Apply $C^{-1}[\{0, \ldots, n-1, n, \ldots, n+L-1\}]$.

We are applying C, "copying" over its results, and finally applying C^{-1}. The computation can be seen as the following sequence of maps:

$$(x, 0, y) \mapsto (F(x), G(x), y) \mapsto (F(x), G(x), F(x) \oplus y) \mapsto (x, 0, F(x) \oplus y)$$

□

Lemma 3. *Let $F : \mathbb{B}^n \to \mathbb{B}^n$ be a permutation. Suppose that there exists Boolean circuits C and C' over basis $\mathcal{A}$ that computes F and F^{-1} such that the size of C and C' are both at most L. Then F can be realized by a reversible Boolean circuit of size $O(L+n)$ using ancillas.*

Proof. By Lemma 2, there exists reversible circuits C_0 and C_1 of size $O(L+n)$ over $\mathcal{A}_\oplus$ realizing F and F^{-1} using ancillas. Suppose that C_0 uses m_0 ancillas and C_1 using m_1 ancillas.
We shall use the first n bits for the input and output. We shall also use $n+m_0+m_1$ ancilliary bits (which are initially 0). Then our circuit works as follows:

- Apply $C_0[\{0, \ldots, n-1, 2n, \ldots, 2n+m_0-1, n, \ldots, 2n-1\}]$.
- Apply $I_\oplus[\{0, n\}]$, ..., $I_\oplus[\{n-1, 2n-1\}]$.
- Apply $C_1[\{0, \ldots, n-1, 2n+m_0, \ldots, 2n+m_0+m_1-1, n, \ldots, 2n-1\}]$.

The computation can be seen as the following sequence of maps:

$$(x, 0, 0, 0) \mapsto (x, F(x), 0, 0) \mapsto (F(x), x, 0, 0) \mapsto (F(x), x \oplus F^{-1}(F(x)), 0, 0) = (F(x), 0, 0, 0)$$

□

Lastly, we want to show that $\{\neg, \wedge_\oplus\}$ form a complete basis for reversible circuits.

Lemma 4. *The basis $\{\neg, \wedge_\oplus\}$ forms a complete basis for reversible Boolean circuits. That is, for any permutation $F : \mathbb{B}^n \to \mathbb{B}^n$, there exists a reversible circuit over the basis $\{\neg, \wedge_\oplus\}$ that realizes F with ancillas.*

Proof. Let $I : \mathbb{B} \to \mathbb{B}$ denote the identity function. $I_\oplus$ can be realized with ancillas by the circuit $\neg[\{2\}] \wedge_\oplus [\{0, 2, 1\}]\neg[\{2\}]$. To see this, we apply the given circuit on $(x, y, 0)$, where $x, y \in \mathbb{B}^n$:

$$\begin{aligned} \neg[\{2\}] \wedge_\oplus [\{0, 2, 1\}]\neg[\{2\}](x, y, 0) &= \neg[\{2\}] \wedge_\oplus [\{0, 2, 1\}](x, y, 1) \\ &= \neg[\{2\}](x, y \oplus (x \cdot 1), 1) \\ &= (x, y \oplus x, 0) \end{aligned}$$

Also, for $x, y \in \mathbb{B}$,

$$\begin{aligned} \neg[\{0\}]I_\oplus[\{0, 1\}]\neg[\{0\}](x, y) &= \neg[\{0\}]I_\oplus[\{0, 1\}](\neg x, y) \\ &= \neg[\{0\}](\neg x, y \oplus (\neg x)) \\ &= (x, y \oplus (\neg x)) \end{aligned}$$

Therefore, every permutation in $\{\neg_\oplus, \wedge_\oplus, I_\oplus\}$ can be realized with ancillas over the basis $\{\neg, \wedge_\oplus\}$.
Since $\{\neg, \wedge\}$ form a complete basis for Boolean circuits, there are Boolean circuits C_0 and C_1 of size $O(n^2 2^n)$ that realize F and F^{-1} respectively. By Lemma 3, there exists reversible Boolean circuits C_2 of size $O(n2^n)$ over the basis $\{\neg_\oplus, \wedge_\oplus, I_\oplus\}$ that realizes F using ancillas. Since each of $\{\neg_\oplus, \wedge_\oplus, I_\oplus\}$ can be realized over the basis $\{\neg, \wedge_\oplus\}$, there exists a reversible Boolean circuit C_2 of size $O(n^2 2^n)$ over the basis $\{\neg, \wedge_\oplus\}$ that computes F using ancillas. □

1.4.4 Probabilistic algorithms

Randomized algorithms are simply algorithms that are also given a sequence of random Booleans.

Definition 18 (Randomized Algorithm)**.** *Let $F : A^* \to A^*$ be computable function. For $m \in \mathbb{N}$, let W_m be a uniformly random binary string of length m. We say that a randomized algorithm computes F in time $T(n)$ (where n represents the size of the input) if it describes a Turing Machine M such that M computes a function $G : A^* \times \mathbb{B}^* \to A^*$ in time $T(n)$, where*

$$\mathbb{P}\left[G(x, W_{T(|x|)}) = F(x)\right] \geq p(|x|) \qquad \forall x \in A^*$$

and

$$p(n) \geq \frac{2}{3} \qquad \forall n \in \mathbb{N}$$

.
The function $p : \mathbb{B} \to [0, 1]$ is any such that for $n \in \mathbb{N}$, $p(n)$ gives a lower bound on the probability of success *of the algorithm for inputs of length n.*

So for an input of size n, given uniformly random binary string, a randomized algorithm has a probability of at least $p(n)$ of outputting correct result. We can easily amplify (or boost) $p(n)$ as well. The following lemma shows that for randomized algorithms that compute Boolean functions.

Lemma 5. *Suppose that we have a randomized algorithm $\mathcal{A}$ that computes a Boolean function $F : \mathbb{B}^* \to \mathbb{B}$ with success probability of at least $2/3$. Furthermore, $\mathcal{A}$ has time complexity of $O(T(n))$, where $T : \mathbb{N} \to \mathbb{N}$ is some computable function. Let $c \geq 1$ be some integer. Then there exists a randomized algorithm with time complexity $O(T(n) \log n)$ that computes F with success probability of at least $1 - n^{-c}$, where n is the size of the input.*

Proof. The algorithm required is Algorithm 1.
For any given input x, the success probability of $\mathcal{A}$ is given by $p = 2/3 + \epsilon$, where $\epsilon \in [0, 1/3]$. Then $1 - p = 1/3 - \epsilon < p$. Therefore, the success probability q of

Algorithm 1 Success_Boosting$(\mathcal{A}, c, x)$

Set $n \leftarrow |x|$.
if $n \leq 1$ **then**
    Set $y \leftarrow \mathcal{A}(x)$.
    return y.
end if
Set $m \leftarrow \lceil \frac{c \log n}{\log (9/8)} \rceil$.
Set $c_1 \leftarrow 0$.
for $j \in [0..2m-1]$ **do**
    Set $y \leftarrow \mathcal{A}(x)$.
    if $y = 1$ **then**
        Set $c_1 \leftarrow c_1 + 1$.
    end if
end for
if $c_1 \geq m$ **then**
    return 1.
else
    return 0.
end if

Algorithm 1 is given by

$$
\begin{aligned}
1 - q &\leq \sum_{j=0}^{m} \binom{2m}{j} p^j (1-p)^{2m-j} \\
&= \sum_{j=0}^{m} \binom{2m}{j} p^j (1-p)^{m-j} (1-p)^m \\
&= \sum_{j=0}^{m} \binom{2m}{j} p^m (1-p)^m \\
&= (p(1-p))^m \sum_{j=0}^{m} \binom{2m}{j} \\
&\leq (p(1-p))^m \sum_{j=0}^{2m} \binom{2m}{j} \\
&\leq (p(1-p))^m 2^{2m} \\
&= 4^m (p(1-p))^m \\
&= 4^m \left(\left(\frac{2}{3} + \epsilon \right) \left(\frac{1}{3} - \epsilon \right) \right)^m \\
&= 4^m \left(\frac{2}{9} - \frac{1}{3}\epsilon - \epsilon^2 \right)^m \\
&= \left(\frac{8}{9} \right)^m \left(1 - \frac{3}{2}\epsilon - \frac{9}{2}\epsilon^2 \right)^m \\
&\leq \left(\frac{8}{9} \right)^m \\
&\leq \left(\frac{8}{9} \right)^{\frac{c \log n}{\log (9/8)}} \\
&= \left(\frac{8}{9} \right)^{\frac{-c \log n}{\log (8/9)}} 14 \\
&= n^{-c}
\end{aligned}
$$

Thus, $q \geq 1 - n^{-c}$. □

1.5 Quantum Computation

Unless explicitly cited (or stated otherwise), the contents of this section are from [3] or [5].

1.5.1 In Praise of Quantum Computation

There is a limit to what classical computers can achieve. For example, many classical cryptographic systems rely on the fact that there is no (as far we know) polynomial-time classical algorithm to factorise large integers. To solve problems untenable by classical computing, we need new forms of computing devices. Unfortunately, classical physics can be simulated by classical computers in polynomial time. This means that classical physics can not give us computing devices that can quickly solve problems that require exponential time in classical computing.
However, classical computers seem to be need exponential time to simulate quantum mechanics. This is because the spin-state of an n-particle system needs 2^n complex numbers to be expressed and we need to compute multiplication by a $2^n \times 2^n$ matrix to compute the change of state over some interval of time. But a quantum system can simulate itself efficiently and thus seem to be more powerful than classical computers. [5]
Clearly, quantum computers can be used to efficiently simulate quantum systems that classical computers have been unable to.
Shor's celebrated $\tilde{O}(n^3)$-time quantum algorithm for factorisation is a celebrated example of the power of quantum computing. It can break cryptosystems such as RSA.
Another crucial example is the Grover search algorithm and its many variants. Through the Grover Search algorithm, we can find an element among n elements in $O(\sqrt{n})$ queries. This leads to powerful results such as algorithms for string matching in $\tilde{O}(\sqrt{n})$ and 3SUM in $O(n)$. The fastest classical algorithms for these problems work in $O(n+m)$ and $\tilde{O}(n^2)$ time respectively (the second one is conditional on SETH) [14], [15]. The Grover can also be used to speed up matrix multiplication over semirings to $\tilde{O}(n^{2.5})$ which finds application in getting graph problems as shown in [6], [9]. Grover may also be used to optimise classical Dynamic Programming algorithms.
There also has been a significant body of work on Quantum Machine Learning, such as Quantum Convolutional Neural Networks [12].

1.5.2 Some Quantum Mechanics

To begin, we need the following axioms of quantum mechanics:

1. A quantum system is completely specified by a unit vector $|\psi\rangle \in \mathcal{H}$, where $\mathcal{H}$ is a complex Hilbert-space. For our purposes, we can assume that $\mathcal{H}$ is isomorphic to $\mathcal{B}^{\otimes n}$ for some $n \in \mathbb{N}$. In other words, $\mathcal{H} \cong \mathbb{C}^{2^n}$.

2. Every observable (energy, momentum, etc.) is a Hermitian operator which acts on $\mathcal{H}$. We can only "observe" the eigenvalues of these operators. After

the "observation", the state collapses to an eigenvector corresponding to the observed eigenvalue. The state becomes "classical". This means that measurement of a quantum system alters it.

3. If $|\phi\rangle$ is a unit eigenvector of some observable then the probability of the state $|\psi\rangle$ collapsing to it upon measurement is given by $|\langle\psi|\phi\rangle|^2$.

4. A closed quantum system evolves unitarily. In other words, the new state $|\phi\rangle$ from state $|\psi\rangle$ in a closed system is given by $|\phi\rangle = U|\psi\rangle$ for some unitary operator U. Since unitary matrices are invertible, this means that much of quantum computation (the part before measurement) has to be invertible.

5. If we have two quantum systems with states $|\psi\rangle \in \mathcal{H}_1$ and $|\phi\rangle \in \mathcal{H}_2$, then the state of the combined system is given by $|\psi\rangle \otimes |\phi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$.

It should be noted that classical computers can solve any problem that quantum computers can and vice-versa. A classical computer can simulate an n-qubit quantum computer using $O(2^n)$ memory and time. Similarly, any classical algorithm has a sequence of Boolean circuits, which can be converted into a sequence of reversible Boolean circuits, which in turn can be converted into a sequence of quantum circuits. In other words, in terms of computability, both classical and quantum computers have the same power. [5]

1.5.3 Qubits

Classical computers (and Boolean circuits) uses bits, where a bit is represented by $x \in \mathbb{B}$ ($\mathbb{B} = \{0, 1\}$), On the other hand, quantum computers (and quantum circuits) uses quantum bits. A quantum bit (aka qubit) is represented by a unit vector $|\phi\rangle \in \mathcal{B} = \mathbb{C}^2$. We call some orthonormal pair of vectors $|0\rangle, |1\rangle \in \mathcal{B}$ the *computational basis*. A qubit $|\phi\rangle$ can be $|0\rangle$, $|1\rangle$, or a superposition of both (such as $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$). When measuring, however, we typically have to let go of the superposition.
Another difference is that in classical computers, n bits are represented by a binary string $x_1 x_2 \dots x_n$ of length n. So any n-bit classical system can be represented by $x = (x_1, x_2, \dots, x_n) \in \mathbb{B}^n$. An n-qubit quantum system, on the other hand, is represented by $|\phi\rangle \in \mathcal{B} \otimes \dots \otimes \mathcal{B} = \mathcal{B}^{\otimes n}$, where $|\phi\rangle = \sum_{x \in \mathbb{B}^n} c_x |x\rangle = \sum_{x_1,\dots,x_n \in \mathbb{B}} c_{x_1,\dots,x_n} |x_1, \dots, x_n\rangle$ and $1 = \||\phi\rangle\|^2 = \sum_{x \in \mathbb{B}^n} |c_x|^2$. . For example, the state of a 1-qubit system is given by $|\phi\rangle = c_0|0\rangle + c_1|1\rangle$ where $|c_0|^2 + |c_1|^2 = 1$. And the state of a 2-qubit system is given by $|\phi\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle$ where $|c_{00}|^2 + |c_{01}|^2 + |c_{10}|^2 + |c_{11}|^2 = 1$.
Another thing to note is that the computatonal basis of an n-qubit system is given by $|0\rangle_1 \otimes \dots \otimes |0\rangle_1, |0\rangle_1 \otimes \dots \otimes |1\rangle_1, \dots, |1\rangle_1 \otimes \dots \otimes |1\rangle_1$, where $|0\rangle_1, |1\rangle_1 \in \mathcal{B}$. We may also refer to the basis elements as $|0\rangle, |1\rangle, |2\rangle, \dots, |2^n - 1\rangle \in \mathcal{B}^{\otimes n}$. For example, in a 4-qubit system, $|10\rangle = |1\rangle_1 \otimes |0\rangle_1 \otimes |1\rangle_1 \times |0\rangle_1$ because 10 is given by 1010 in binary).

1.5.4 Quantum Circuits

A quantum system changes through unitary transformations and measurements. We can define a quantum algorithm as a sequence of unitary transformations possibly

followed by a measurement operator. Even if we have measurements in between unitary transformations, we can always defer said measurements to the end without affecting the outcome of the computation, as we shall show in Theorem 4. [3].
Quantum circuits are defined in a similar manner to reversible Boolean circuits. We can define quantum circuits over a fixed gate set $\mathcal{A}$ of unitary operators of bounded input size. That is, there exists an integer n such that $A \in \mathcal{A}$ implies that $A \in \mathbf{U}\left(\mathcal{B}^{\otimes k}\right)$ for some integer $k \leq n$.
The size of the quantum circuit is defined as the number of basis gates used in the circuit.

Definition 19. *Let k and n be integers such that $k \leq n$. Let $A = \{a_0, \ldots, a_{k-1}\} \subseteq [0..n-1]$ be an ordered set and let $U \in \mathbf{U}\left(\mathcal{B}^{\otimes k}\right)$ be an unitary transformation. Let $P \in \mathbf{U}\left(\mathcal{B}^{\otimes n}\right)$ be a unitary defined by*

$$P = \sum_{(x_0,\ldots,x_{n-1})\in\mathbb{B}^n} \left(\left(\bigotimes_{j=0}^{k-1} |x_{a_j}\rangle \right) \otimes \left(\bigotimes_{j\notin A} |x_j\rangle \right) \right) \langle x_0, \ldots, x_{n-1}|$$

Let $I_{n-k} \in \mathbf{U}\left(\mathcal{B}^{\otimes(n-k)}\right)$ be the identity unitary transformation.
We define $U[A]_n$ as a unitary transformation $U[A]_n \in \mathbf{U}\left(\mathcal{B}^{\otimes n}\right)$ given by $U[A]_n = P^{-1}(U \otimes I_{n-k})P$.

To put it simply, $U[A]_n$ is a unitary that "acts" on the qubits $a_0, \ldots, a_{k-1}$.
For example, if $U : \mathcal{B}^{\otimes 2} \to \mathcal{B}^{\otimes 2}$ is defined by $U|x_0, x_1\rangle = |x_0 \oplus x_1, x_1 \oplus 1\rangle$ for $(x_0, x_1) \in \mathbb{B}^2$, then $U[\{1,3\}]|x_0, x_1, x_2, x_3\rangle = |x_0, x_1 \oplus x_3, x_2, x_3 \oplus 1\rangle$ for $(x_0, x_1, x_2, x_3) \in \mathbb{B}^4$.

Definition 20 (Quantum Circuit)**.** *A quantum circuit over a basis $\mathcal{A}$ is a sequence $U_0[A_0]_n, \ldots, U_{L-1}[A_{L-1}]_n$ where, for all $j \in [0..L-1]$, $U_j \in \mathcal{A}$ and $A_j \subseteq [0..n-1]$ is an ordered set. This quantum ciruit realizes the unitary operator $U = U_L[A_L] \ldots U_1[A_1]$.*

Of course, we can use additional qubits (more than the input size n) as well to help the computation.

Definition 21 (Quantum Circuits using ancillas)**.** *We say that a unitary $U \in \mathbf{U}\left(\mathcal{B}^{\otimes n}\right)$ is realized by a quantum circuit $U_0[A_0]_N, \ldots, U_{L-1}[A_{L-1}]_N$ ($N \geq n$) using ancillas if we have*

$$W\left(|\xi\rangle \otimes |0^{N-n}\rangle\right) = (U|\xi\rangle) \otimes |0^{N-n}\rangle$$

for any $|\xi\rangle \in (\mathcal{B})^{\otimes n}$.

Note that the notation $|0^{N-n}\rangle$ means a tensor product of $N-n$ of $|0\rangle$.
We now note that quantum computers are able to perform classical computations. In particular, any reversible classical circuits has its corresponding quantum circuits. Let $G : \mathbb{B}^k \to \mathbb{B}^k$ be a permutation. It corresponds to a unitary operator $\widehat{G} \in \mathbf{U}\left(\mathcal{B}^{\otimes k}\right)$ defined by $\widehat{G}|x\rangle = |G(x)\rangle$ for all $x \in \mathbb{B}^k$. The unitarity of $\widehat{G}$ can be seen by noting that for $x, y \in \mathbb{B}^k$, $\langle \widehat{G}|x\rangle | \widehat{G}|y\rangle\rangle = \langle G(x)|G(y)\rangle$. Since G is bijective, $x \neq y \Leftrightarrow G(x) \neq G(y)$ and thus $\langle \widehat{G}|x\rangle | \widehat{G}|y\rangle\rangle = \langle x|y\rangle$.
For our purposes, we shall define our basis $\mathcal{A}$ as the set of all one and two qubit unitary operators. That is, $\mathcal{A} = \mathbf{U}\left(\mathcal{B}^{\otimes 1}\right) \cup \mathbf{U}\left(\mathcal{B}^{\otimes 2}\right)$. We shall show that this basis is complete in the sense that any unitary $U \in \mathbf{U}\left(\mathcal{B}^{\otimes n}\right)$ can be realized by this basis. First, we shall prove a useful lemma.

Definition 22. *Let $\mathcal{N}$ be a subspace of the Hilbert space $\mathcal{M}$. Let $\mathcal{N}_\perp$ denote the orthogonal complement of $\mathcal{N}$. So $\mathcal{M} = \mathcal{N} \oplus \mathcal{N}_\perp$. Let $U \in \mathbf{U}(\mathcal{M})$. We say that U* acts on the subspace *$\mathcal{N}$ if there exists $V \in \mathbf{U}(\mathcal{N})$ such that $U = V \oplus I$, where $I \in \mathbf{U}(\mathcal{N}_\perp)$ is an identity map.*

Lemma 6. *Let $U \in \mathbf{U}\left(\mathbb{C}^M\right)$, where $M \geq 2$ is a positive integer. Let $|1\rangle, \ldots, |M-1\rangle$ be an orthonormal basis of $\mathbb{C}^M$. There exists a finite sequence of unitary operators $\left(U^{(j,k)}\right)_{j\in[1..M-1],k\in[j..M-1]}$ such that*

$$U = U^{(M-1,M-1)} \left(U^{(M-2,M-2)} U^{(M-2,M-1)}\right) \cdots \left(U^{(1,1)} \cdots U^{(1,M-1)}\right)$$

and for each $j \in [1..M-1], k \in [j..M-1]$, the unitary operator $U^{(j,k)} \in \mathbf{U}\left(\mathbb{C}^M\right)$ acts on the subspace $\mathbb{C}(|s\rangle, |s+1\rangle)$.

Proof. First, suppose that we have $c_1, c_2 \in \mathbb{C}$. We claim that there exists a unitary operator $V \in \mathbf{U}(\mathcal{B})$

$$V \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \sqrt{|c_1|^2 + |c_2|^2} \\ 0 \end{bmatrix}$$

.
If $c_1 = c_2 = 0$ then any unitary operator $V \in \mathbf{U}(\mathcal{B})$ satisfies our requirement.
Otherwise, let $|\xi\rangle \in \mathcal{B}$ be a unit vector orthogonal to $(c_1, c_2) \in \mathcal{B}$. Therefore, $|\eta\rangle = \frac{1}{\sqrt{|c_1|^2+|c_2|^2}}(c_1, c_2)$ and $|\xi\rangle$ form an orthonormal basis for $\mathcal{B}$. Therefore, there exists a unitary operator V that maps $|\eta\rangle$ to $(1, 0)$ and $|\xi\rangle$ to $(0, 1)$. By computation, we see that V satisfies our requirement.
Now, suppose that we have a unit vector $|\xi\rangle = \sum_{j=1}^{M} a_j|j\rangle \in \mathbb{C}^M$. We shall inductively define $|\xi_1\rangle, \ldots, |\xi_M\rangle$ and $V^1, \ldots, V^{(M-1)}$. Define $|\xi_M\rangle = |\xi\rangle$. Suppose that $s \in [1..M-1]$. Suppose that we have defined $|\xi_{s+1}\rangle$ such that $|\xi_{s+1}\rangle$ is a unit vector, $\sum_{j=s+2}^{M-1} |\langle j|\xi_{s+1}\rangle|^2 = 0$, and $\langle j|\xi_{s+1}\rangle = a_j$ for all $j \in [1..s]$. As we have already shown, there exists a unitary $V^{(s)}$ such that $\langle s+1|V^{(s)}|\xi_{s+1}\rangle = 0$ and $\langle j|V^{(s)}|\xi_s\rangle = a_j$ for $j \in [1..s-1]$. Define $|\xi_s\rangle = V^{(s)}|\xi_{s+1}\rangle$.
Now, suppose that have a $M \times M$ matrix representing U^{-1} in the basis $|1\rangle, \ldots, |M\rangle$. We shall, again, use induction. Let $j \in [1..M-1]$ be given. Assume that we have a unitary matrix V that satisfies the following for all $r, c \in [1..M]$ with $r < j \vee c < j$:

$$V_{r,c} = \begin{cases} 1 & r = c \\ 0 & r \neq c \end{cases}$$

As we have already shown, there exists $U^{(j,j)}, \ldots, U^{(j,M-1)}$ such that $V' = U^{(j,1)} \cdots U^{(j,M-1)}V$ is a unitary matrix such that the column j of V' is a column vector with 0 in columns $j+1, \ldots, M$. This finite sequence of matrices act on $|j\rangle, \ldots, |M\rangle$ and $V_{r,c} = 0$ when $c < j \wedge r \geq j$, we have $V'_{r,c} = V_{r,c}$ for all $r < j \vee c < j$. Furthermore, since $V'_{j,1} = V_{j,1} = \cdots = V'_{j,j-1} = V_{j,j-1} = 0$, we must have $V'_{j,j} = 1$ (as V' is a unitary matrix).
Thus, we have operators $U^{(M-1,M-1)}, U^{(M-2,M-2)}, U^{(M-2,M-1)}, \ldots, U^{(1,1)}, \ldots, U^{(1,M-1)}$ such that

$$U^{(M-1,M-1)} \left(U^{(M-2,M-2)} U^{(M-2,M-1)}\right) \cdots \left(U^{(1,1)} \cdots U^{(1,M-1)}\right) U^{-1} = I$$

, where I is the identity operator on $\mathbb{C}^M$.
In other words,

$$U = U^{(M-1,M-1)} \left(U^{(M-2,M-2)} U^{(M-2,M-1)}\right) \cdots \left(U^{(1,1)} \cdots U^{(1,M-1)}\right) U^{-1}$$

. □

Definition 23. *Let $U \in \mathbf{U}(\mathcal{B}^{\otimes n})$. Then, for any integer $k \geq 1$, $\Lambda^k(U) \in \mathbf{U}\left(\mathcal{B}^{\otimes(n+k)}\right)$ is defined by, for $(x_0, \ldots, x_{k-1}) \in \mathbb{B}^k$ and $|\xi\rangle \in \mathcal{B}^{\otimes n}$,*

$$\Lambda^k(U)\left(|x_0, \ldots, x_{k-1}\rangle \otimes |\xi\rangle\right) = \begin{cases} |x_0, \ldots, x_{k-1}\rangle \otimes (U|\xi\rangle) & x_0 = \ldots x_{k-1} = 1 \\ |x_0, \ldots, x_{k-1}\rangle \otimes |\xi\rangle \end{cases}$$

. Also, define $\Lambda(U) = \Lambda^1(U)$. Given a complex number z with $|z| = 1$, we define $\Lambda^k(z) \in \mathcal{B}^{\otimes k}$ by letting, for $(x_0, \ldots, x_{k-1}) \in \mathbb{B}^k$,

$$\Lambda^k(z)|x_0, \ldots, x_{k-1}\rangle = \begin{cases} z|x_0, \ldots, x_{k-1}\rangle & x_0 = \ldots x_{k-1} = 1 \\ |x_0, \ldots, x_{k-1}\rangle \end{cases}$$

Finally, define $\Lambda(z) = \Lambda^1(z)$.

We also define $\sigma^x = |1\rangle\langle 0| + |0\rangle\langle 1 \in \mathbf{U}(\mathcal{B}^{\otimes 1})$. Clearly, $\sigma^x = \widehat{\neg}$, $\Lambda(\sigma^x) = \widehat{I_\oplus}$, and $\Lambda^2(\sigma^x) = \widehat{\wedge_\oplus}$. Here, $I : \mathbb{B} \to \mathbb{B}$ denotes the identity mapping on a single Boolean.

Theorem 3. *The basis $\mathcal{A} = \mathbf{U}(\mathcal{B}^{\otimes 1}) \cup \mathbf{U}(\mathcal{B}^{\otimes 2})$ consisting of the set of all one-qubit and two-qubit unitary operators allow the realization of an arbitrary $U \in \mathbf{U}(\mathcal{B}^{\otimes n})$.*

Proof. First, we show that $\widehat{\wedge_\oplus} = \Lambda^2(\sigma^x)$ can be realized using ancillas. Let $X, Y \in \mathbf{U}(\mathcal{B})$ be defined by

$$X = \frac{1}{\sqrt{2}} \begin{bmatrix} -i & -1 \\ 1 & i \end{bmatrix}$$
$$Y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Through computation, we see that $iXYX^{-1}Y^{-1} = \sigma^x$.
The following quantum circuit realizes $\wedge_\oplus$:

$$\Lambda^2(i)[\{0,1\}]\Lambda(X)[\{1,2\}]\Lambda(Y)[\{0,2\}]\Lambda(X^{-1})[\{1,2\}]\Lambda(Y^{-1})[\{0,2\}]$$

Let U denote the unitary operator realized by the circuit above. To see that the circuit actually realizes $\wedge_\oplus$, we select $a, b, c \in \mathbb{B}$. If $a = b = 1$, then

$$\begin{aligned} U|a,b,c\rangle &= \Lambda^2(i)[\{0,1\}]\Lambda(X)[\{1,2\}]\Lambda(Y)[\{0,2\}]\Lambda(X^{-1})[\{1,2\}]\Lambda(Y^{-1})[\{0,2\}]|a,b,c\rangle \\ &= \Lambda^2(i)[\{0,1\}]\Lambda(X)[\{1,2\}]\Lambda(Y)[\{0,2\}]\Lambda(X^{-1})[\{1,2\}]\left(|a,b\rangle \otimes (Y^{-1}|c\rangle)\right) \\ &= \Lambda^2(i)[\{0,1\}]\Lambda(X)[\{1,2\}]\Lambda(Y)[\{0,2\}]\left(|a,b\rangle \otimes (X^{-1}Y^{-1}|c\rangle)\right) \\ &= \Lambda^2(i)[\{0,1\}]\Lambda(X)[\{1,2\}]\left(|a,b\rangle \otimes (YX^{-1}Y^{-1}|c\rangle)\right) \\ &= \Lambda^2(i)[\{0,1\}]\left(|a,b\rangle \otimes (XYX^{-1}Y^{-1}|c\rangle)\right) \\ &= |a,b\rangle \otimes (iXYX^{-1}Y^{-1}|c\rangle) \\ &= |a,b\rangle \otimes (\sigma^x|c\rangle) \\ &= |a,b\rangle \otimes |c \oplus 1\rangle \\ &= |a,b,c \oplus ab\rangle \end{aligned}$$

If $a = 0$ but $b = 1$,

$$\begin{aligned}
U|a,b,c\rangle &= \Lambda^2(i)[\{0,1\}]\Lambda(X)[\{1,2\}]\Lambda(Y)[\{0,2\}]\Lambda(X^{-1})[\{1,2\}]\Lambda(Y^{-1})[\{0,2\}]|a,b,c\rangle \\
&= \Lambda^2(i)[\{0,1\}]\Lambda(X)[\{1,2\}]\Lambda(Y)[\{0,2\}]\Lambda(X^{-1})[\{1,2\}]\left(|a,b\rangle \otimes (|c\rangle)\right) \\
&= \Lambda^2(i)[\{0,1\}]\Lambda(X)[\{1,2\}]\Lambda(Y)[\{0,2\}]\left(|a,b\rangle \otimes (X^{-1}|c\rangle)\right) \\
&= \Lambda^2(i)[\{0,1\}]\Lambda(X)[\{1,2\}]\left(|a,b\rangle \otimes (X^{-1}|c\rangle)\right) \\
&= \Lambda^2(i)[\{0,1\}]\left(|a,b\rangle \otimes (XX^{-1}|c\rangle)\right) \\
&= |a,b\rangle \otimes |c\rangle \\
&= |a,b\rangle \otimes |c \oplus 0\rangle \\
&= |a,b,c \oplus ab\rangle
\end{aligned}$$

If $b = 0$,

$$\begin{aligned}
U|a,b,c\rangle &= \Lambda^2(i)[\{0,1\}]\Lambda(X)[\{1,2\}]\Lambda(Y)[\{0,2\}]\Lambda(X^{-1})[\{1,2\}]\Lambda(Y^{-1})[\{0,2\}]|a,b,c\rangle \\
&= \Lambda(Y)[\{0,2\}]\Lambda(Y^{-1})[\{0,2\}]|a,b,c\rangle \\
&= |a,b,c\rangle \\
&= |a,b\rangle \otimes |c \oplus 0\rangle \\
&= |a,b,c \oplus ab\rangle
\end{aligned}$$

So we can realize $\Lambda^2(\sigma^x)$.
Now, we shall show that we can realize $\Lambda^k(U)$ for any $U \in \mathbf{U}(\mathcal{B})$.
Let $F : \mathbb{B}^k \to \mathbb{B}$ be defined by $F(x_0, \ldots, x_{k-1}) = x_0 \wedge \cdots \wedge x_{k-1}$. Clearly, there exists an $O(k)$ size Boolean circuit P' over the basis $\{\neg, \wedge\}$ that realizes F. So there exists a reversible Boolean circuit P of size $O(k)$ over the basis $\{\neg, \wedge_\oplus\}$ that realizes $F_\oplus$ using ancillas. Since all one-qubit and two-qubit gates are in $\mathcal{A}$ and $\widehat{\Lambda_\oplus}$ can be realized, we have a quantum circuit $\widehat{P}$ of size $O(k)$ over $\mathcal{A}$ that realizes $\widehat{F_\oplus}$. Suppose that $\widehat{P}$ uses m ancillas and maps $|x_0, \ldots, x_{k-1}, y, 0, \ldots, 0\rangle$ to $|x_0, \ldots, x_{k-1}, y \oplus F(x_0, \ldots, x_{k-1}), 0, \ldots, 0\rangle$ for all $x_0, \ldots, x_{k-1}, y \in \mathbb{B}$. Then the following circuit realizes $\Lambda^k(U)$ using ancillas:

$$\widehat{P}^{-1}[\{0, \ldots, k+m\}]\Lambda(U)[\{k, k+m+1\}]\widehat{P}[\{0, \ldots, k+m\}]$$

Let $x_0, \ldots, x_{k-1} \in \mathbb{B}$. Also let $|\xi\rangle \in \mathcal{B}$. Then

$$\begin{aligned}
&\widehat{P}^{-1}[\{0, \ldots, k+m\}]\Lambda(U)[\{k, k+m+1\}]\widehat{P}[\{0, \ldots, k+m\}]\left(|x_0, \ldots, x_{k-1}, 0, 0, \ldots, 0\rangle \otimes |\xi\rangle\right) \\
&= \widehat{P}^{-1}[\{0, \ldots, k+m\}]\Lambda(U)[\{k, k+m+1\}]\left(|x_0, \ldots, x_{k-1}, (x_0 \wedge \ldots x_{k-1}), 0, \ldots, 0\rangle \otimes |\xi\rangle\right) \\
&= \widehat{P}^{-1}[\{0, \ldots, k+m\}]\left(|x_0, \ldots, x_{k-1}, x_0 \wedge \ldots x_{k-1}, 0, \ldots, 0\rangle \otimes (U^{x_0 \cdots x_{k-1}}|0\rangle)\right) \\
&= |x_0, \ldots, x_{k-1}, 0, 0, \ldots, 0\rangle \otimes (U^{x_0 \wedge \cdots \wedge x_{k-1}}|0\rangle)
\end{aligned}$$

So we see that the circuit above realizes $\Lambda^k(U)$ using ancillas. The circuit's size is $O(k)$.
Note that $\Lambda^k(U)$ acts on the subspace $\mathbb{C}(|1, \ldots, 1, 0\rangle, |1, \ldots, 1, 1\rangle)$.
Now, let $x = (x_0, \ldots, x_{n-1}), y = (y_0, \ldots, y_{n-1}) \in \mathbb{B}^n, x \neq y$. Suppose that we wish to realize the unitary $V \in \mathbf{U}(\mathcal{B}^{\otimes n})$ that acts on the subspace $\mathbb{C}(|x\rangle, |y\rangle)$. Let U be a unitary matrix that describes $V|_{\mathbb{C}(|x\rangle,|y\rangle)}$ on basis $|x\rangle, |y\rangle$. Let $U_0 \in \mathbf{U}(\mathcal{B})$ denote the unitary operator defined by U on the basis $|0\rangle, |1\rangle \in \mathcal{B}$. That is, for all $p, q \in \mathbb{C}$,

$$U \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} r \\ s \end{bmatrix}.$$

gives $V(p|x\rangle+q|y\rangle) = r|x\rangle+s|y\rangle$, $U_0(p|0\rangle+q|1\rangle) = r|0\rangle+s|1\rangle$, and $\Lambda^{(n-1)}(U_0)(p|1,\ldots,1,0\rangle+q|1,\ldots,1,1\rangle) = r|1,\ldots,1,0\rangle + s|1,\ldots,1,1\rangle$.
Let $f : \mathbb{B}^n \to \mathbb{B}^n$ be any permutation such that $f(x) = (1,\ldots,1,0)$ and $f(y) = (1,\ldots,1,1)$. Clearly, $V = \widehat{f}^{-1}\Lambda^{(n-1)}(U_0)\widehat{f}$.
Since there exists a $O(n^2 2^n)$-size reversible Boolean circuit that realizes f with ancillas, there exists a quantum circuit over $\mathcal{A}$ of size $O(n^2 2^n)$ that realizes $\widehat{f}$ using ancillas. And since we can also realize $\Lambda^{(n-1)}(U_0)$ using a quantum circuit of size $O(n)$, we can thus realize V using a quantum circuit of size $O(n^2 2^n)$.
Finally for any $U \in \mathbf{U}(\mathcal{B}^{\otimes n})$, we use Lemma 6 to get U as a composition of $(2^n)^2 = 2^{2n}$ unitary operators that act on subspaces of the form $\mathbb{C}(|x\rangle, |y\rangle)$, where $x \neq y \in \mathbb{B}^n$. Since each of these operators can be realized using quantum circuits of size $O(n^2 2^n)$, the operator U can be realized by a quantum circuit of size $O(n^2 2^{3n})$. □

1.5.5 Density Matrices and Measurements

To discuss measurement, we need to introduce density matrices.

Definition 24 (Density Matrix)**.** *Let $\mathcal{N}$ be a Hilbert space. A* density matrix *$\rho \in \mathbf{L}(\mathcal{N}) = \mathcal{N} \otimes \mathcal{N}^*$ is a linear transformation of the form $\rho = \sum_{j=0}^{m-1} p_j |\xi_j\rangle\langle\xi_j|$ where $p_0,\ldots,p_{m-1}$ are non-negative real numbers satisfying $p_0 + \cdots + p_{m-1} = 1$ and $|\xi_0\rangle,\ldots,|\xi_{m-1}\rangle \in \mathcal{N}$.*

Suppose that we have a density matrix $\rho = \sum_{j=0}^{m-1} p_j|\xi_j\rangle\langle\xi_j|$. Then this density matrix represents a quantum system where its state is $|\xi_j\rangle$ with probability p_j.
Given an arbitrary subspace $\mathcal{M}$ of $\mathcal{N}$, we define the probability $\mathbb{P}(|\Psi\rangle, \mathcal{M})$ of "collapsing" to an element of $\mathcal{M}$ is given by

$$\mathbb{P}(|\psi\rangle, \mathcal{M}) = \langle\psi|\Pi_{\mathcal{M}}|\psi\rangle$$

where $\Pi_{\mathcal{M}} \in \mathbf{L}(\mathcal{N})$ is the projection onto $\mathcal{M}$.
So, for a system represented by ρ, the probability $\mathbb{P}(\rho, \mathcal{M})$ of the system "collapsing" to an element of $\mathcal{M}$ is

$$\begin{aligned}
\mathbb{P}(\rho, \mathcal{M}) &= \sum_{j=0}^{m-1} p_j \mathbb{P}(|\xi_j\rangle, \mathcal{M}) \\
&= \sum_{j=0}^{m-1} p_j \langle\xi_j|\Pi_{\mathcal{M}}|\xi_j\rangle \\
&= \sum_{j=0}^{m-1} p_j \mathrm{Tr}\left(|\xi_j\rangle\langle\xi_j|\Pi_{\mathcal{M}}\right) \\
&= \sum_{j=0}^{m-1} \mathrm{Tr}\left(p_j|\xi_j\rangle\langle\xi_j|\Pi_{\mathcal{M}}\right) \\
&= \mathrm{Tr}\left(\rho\Pi_{\mathcal{M}}\right)
\end{aligned}$$

Given two density matrices $\rho_1 = \sum_{j=0}^{m-1} p_j|\xi_j\rangle\langle\xi_j| \in \mathbf{L}(\mathcal{N}_1)$ and $\rho_2 = \sum_{j=0}^{n-1} q_j|\eta_j\rangle\langle\eta_j| \in \mathbf{L}(\mathcal{N}_2)$, we see that $\rho_1 \otimes \rho_1 \in \mathbf{L}(\mathcal{N}_1) \otimes \mathbf{L}(\mathcal{N}_2) = \mathbf{L}(\mathcal{N}_1 \otimes \mathcal{N}_2)$ is given by

$$\rho_1 \otimes \rho_2 = \sum_{j=0}^{m-1}\sum_{k=0}^{n-1} p_j q_k |\xi_j\rangle\langle\xi_j| \otimes |\eta_k\rangle\langle\eta_j| = \sum_{j=0}^{m-1}\sum_{k=0}^{n-1} p_j q_k |\xi_j,\eta_k\rangle\langle\xi_j,\eta_j|$$

It is clear that $\rho_1 \otimes \rho_2$ represents the combined system of ρ_1 and ρ_2.

Definition 25 (Partial Trace)**.** *The* partial trace *is a transformation* $\mathrm{Tr}_{\mathcal{N}_2} : \mathbf{L}(\mathcal{N}_1 \otimes \mathcal{N}_2) \to \mathbf{L}(\mathcal{N}_1)$ *defined by, for all* $\rho \otimes \rho' \in \mathbf{L}(\mathcal{N}_1 \otimes \mathcal{N}_2)$*, as*

$$\mathrm{Tr}_{\mathcal{N}_2}(\rho \otimes \rho') = \rho \mathrm{Tr}(\rho')$$

Suppose that we have a density matrix $\rho \in \mathbf{L}(\mathcal{N}_1 \otimes \mathcal{N}_2)$ and a subspace $\mathcal{M}_1$ of $\mathcal{N}_1$. Then $\rho = \sum_{j=0}^{m-1} A_j \otimes B_j$ for some $A_0, \ldots, A_{m-1} \in \mathbf{L}(\mathcal{N}_1)$ and $B_0, \ldots, B_{m-1} \in \mathbf{L}(\mathcal{N}_2)$. Therefore,

$$\begin{aligned}
\mathbb{P}(\rho, \mathcal{M}_1 \otimes \mathcal{N}_2) &= \mathrm{Tr}(\rho \Pi_{\mathcal{M}_1 \otimes \mathcal{N}_2}) \\
&= \mathrm{Tr}\left(\sum_{j=0}^{m-1} (A_j \otimes B_j) \Pi_{\mathcal{M}_1 \otimes \mathcal{N}_2} \right) \\
&= \mathrm{Tr}\left(\sum_{j=0}^{m-1} (A_j \Pi_{\mathcal{M}_1}) \otimes B_j \right) \\
&= \sum_{j=0}^{m-1} \mathrm{Tr}\left((A_j \Pi_{\mathcal{M}_1}) \otimes B_j \right) \\
&= \sum_{j=0}^{m-1} \mathrm{Tr}(A_j \Pi_{\mathcal{M}_1}) \mathrm{Tr}(B_j) \\
&= \sum_{j=0}^{m-1} \mathrm{Tr}(A_j \Pi_{\mathcal{M}_1} \mathrm{Tr}(B_j)) \\
&= \mathrm{Tr}\left(\sum_{j=0}^{m-1} A_j \Pi_{\mathcal{M}_1} \mathrm{Tr}(B_j) \right) \\
&= \mathrm{Tr}\left(\mathrm{Tr}_{\mathcal{N}_2}(\rho) \Pi_{\mathcal{M}_1} \right) \\
&= \mathbb{P}(\mathrm{Tr}_{\mathcal{N}_2} \rho, \Pi_{\mathcal{M}_1})
\end{aligned}$$

Furthermore, suppose that we have $U \otimes Y \in \mathbf{U}(\mathcal{N}_1 \otimes \mathcal{N}_2)$. Then,

$$\begin{aligned}
\mathrm{Tr}_{\mathcal{N}_2}\left((U \otimes Y) \rho (U \otimes Y)^\dagger \right) &= \mathrm{Tr}_{\mathcal{N}_2}\left(\sum_{j=0}^{m-1} (U A_j U^\dagger) \otimes (Y B_j Y^\dagger) \right) && (1.1) \\
&= \sum_{j=0}^{m-1} (U A_j U^\dagger) \mathrm{Tr}(Y B_j Y^\dagger) && (1.2) \\
&= \sum_{j=0}^{m-1} (U A_j U^\dagger) \mathrm{Tr}(B_j) && (1.3) \\
&= \sum_{j=0}^{m-1} (U (\mathrm{Tr}(B_j) A_j) U^\dagger) && (1.4) \\
&= U \left(\sum_{j=0}^{m-1} \mathrm{Tr}(B_j) A_j \right) U^\dagger && (1.5) \\
&= U \mathrm{Tr}_{\mathcal{N}_2}(\rho) U^\dagger && (1.6) \\
& && (1.7)
\end{aligned}$$

For any subspace $\mathcal{M}$ of $\mathcal{N}_1$, regardless of the unitary Y applied on the second subsystem, we have

$$\begin{aligned}\mathbb{P}\left((U\otimes Y)\rho(U\otimes Y)^\dagger, \mathcal{M}\otimes\mathcal{N}_2\right) &= \mathbb{P}\left(\mathrm{Tr}_{\mathcal{N}_2}\left((U\otimes Y)\rho(U\otimes Y)^\dagger\right), \mathcal{M}\right)\\ &= \mathbb{P}\left(U\mathrm{Tr}_{\mathcal{N}_2}(\rho)U^\dagger, \mathcal{M}\right)\end{aligned}$$

This means that we can *discard* or forget the second system of ρ by taking partial trace. That is, $\rho' = \mathrm{Tr}_{\mathcal{N}_2}\rho$ gives a density matrix representing the first system.
Let $U \in \mathbf{U}(\mathcal{N})$ be a unitary operator. Suppose that we apply U to the system represented by ρ. For each $j \in [0..m-1]$, since the system has state $|\xi_j\rangle$ with probability p_j, then the new state is $U|\xi_j\rangle$ with probability p_j. In other words, the density matrix ρ' representing the system after U is applied is given by

$$\rho' = \sum_{j=0}^{m-1} |U\xi_j\rangle\langle U\xi_j| = U\rho U^\dagger$$

.
Now, let us discuss measurement. Suppose that we wish to measure some state $|\psi\rangle = \sum_{x\in\mathbb{B}^n} a_x|x\rangle \in \mathcal{B}^{\otimes n}$ in the computational basis. The density matrix ρ representing the state ψ is

$$|\psi\rangle\langle\psi| = \left(\sum_{x\in\mathbb{B}^n} a_x|x\rangle\right)\left(\sum_{x\in\mathbb{B}^n} a_x^*\langle x|\right) = \sum_{x\in\mathbb{B}^n} |a_x|^2 |x\rangle\langle x| + \sum_{x\neq y\in\mathbb{B}^n} a_x a_y^*|x\rangle\langle y|$$

Now, the probability of getting $x \in \mathbb{B}^n$ when measuring $|\psi\rangle$ in the computational basis is given by $|a_x|^2$. In other words, the density matrix ρ' representing the system after the measurement must be $\sum_{x\in\mathbb{B}^n} |a_x|^2 |x\rangle\langle x|$.
So, we define measurement superoperator $M_{\mathcal{N}} : \mathbf{L}(\mathcal{N}) \to \mathbf{L}(\mathcal{N})$ ($\mathcal{N} = \mathcal{B}^{\otimes n}$) as follows: given $x, y \in \mathbb{B}^n$,

$$M_{\mathcal{N}}|x\rangle\langle y| = \begin{cases} |x\rangle\langle y| & x = y \\ 0 & x \neq y \end{cases}$$

For a density matrix $\rho = \sum_{j=0}^{m-1} p_j|\xi_j\rangle\langle\xi_j| \in \mathbf{L}(\mathcal{N})$ ($\mathcal{N} = \mathcal{B}^{\otimes n}$), where $\xi_j = \sum_{x\in\mathbb{B}^n} a_{j,x}|x\rangle$, $\rho' = M_{\mathcal{N}}\rho$ is given by

$$\begin{aligned}\rho' &= M_{\mathcal{N}}\rho\\ &= \sum_{j=0}^{m-1} p_j M_{\mathcal{N}}|\xi_j\rangle\\ &= \sum_{j=0}^{m-1} p_j \sum_{x\in\mathbb{B}^n} |a_{j,x}|^2 |x\rangle\langle x|\\ &= \sum_{x\in\mathbb{B}^n} \left(\sum_{j=0}^{m-1} p_j |a_{j,x}|^2\right) |x\rangle\langle x|\end{aligned}$$

Clearly, ρ' represents a system which is $|x\rangle \in \mathbb{B}^n$ with probability $\sum_{j=0}^{m-1} p_j |a_{j,x}|^2$, which is exactly the probability of state of ρ being $|\xi_j\rangle$ and then collapsing to $|x\rangle$. So our definition of M makes sense.
Now, we shall show that we do not need our defintion of $M_{\mathcal{N}}$.

Theorem 4 (Measurement Deferral)**.** *Let $\mathcal{N} = \mathcal{N}_1 = \mathcal{B}^{\otimes n}$. There is a unitary circuit of size n that realizes a unitary $U \in \mathbf{U}(\mathcal{N} \otimes \mathcal{N}_1)$ such that for any density matrix $\rho \in \mathbf{L}(\mathcal{N})$, we have*

$$\rho' = M_{\mathcal{N}}\rho = \mathrm{Tr}_{\mathcal{N}_1}\left(U(\rho \otimes |0\rangle\langle 0|)U^\dagger\right)$$

In other words, we can replace measurement by applying U with n ancillas and then forgetting about the n ancillas.

Proof. Let $|\psi\rangle = \sum_{x\in\mathbb{B}^n} a_x|x\rangle \in \mathcal{N}$ and $\rho = |\psi\rangle\langle\psi|$. Then $\rho' = M\rho = \sum_{x\in\mathbb{B}^n} |a_x|^2 |x\rangle\langle x|$. Let $X = \Lambda(\sigma^x)$. Our circuit is then given by $X[\{0,n\}], \ldots, X[\{n-1, 2n-1\}]$. So $U = X[\{0,n\}]\cdots X[\{n-1,2n-1\}]$ is the unitary realized by the circuit. Given $x, y \in \mathbb{B}^n$, we have

$$U|x\rangle|y\rangle = |x\rangle|y \oplus x\rangle$$

Now,

$$\begin{aligned}
\rho &= \sum_{x\in\mathbb{B}^n} |a_x|^2|x\rangle\langle x| + \sum_{x\neq y\in\mathbb{B}^n} a_x a_y^*|x\rangle\langle y| \\
U\left(\rho\otimes|0\rangle\langle 0|\right)U^\dagger &= \sum_{x\in\mathbb{B}^n} |a_x|^2|x,x\rangle\langle x,x| + \sum_{x\neq y\in\mathbb{B}^n} a_x a_y^*|x,x\rangle\langle y,y| \\
\mathrm{Tr}_{\mathcal{N}_1}\left(U\left(\rho\otimes|0\rangle\langle 0|\right)U^\dagger\right) &= \sum_{x\in\mathbb{B}^n} |a_x|^2|x\rangle\langle x|\mathrm{Tr}\left(|x\rangle\langle x|\right) + \sum_{x\neq y\in\mathbb{B}^n} a_x a_y^*|x\rangle\langle y|\mathrm{Tr}\left(|x\rangle\langle y|\right) \\
&= \sum_{x\in\mathbb{B}^n} |a_x|^2|x\rangle\langle x| \\
&= \rho'
\end{aligned}$$

□

Note that in unitary circuits, we do not actually discard any of the qubits. So rather than discarding the n ancilliary qubits, we ignore them (pretend they do not exist). This means that our unitary circuits may still use "measurements" in the computational basis. The circuit size increases by at most a constant factor through this approach.

1.5.6 Quantum Algorithm

Now, suppose that we want to compute the function $F : \mathbb{B}^n \to \mathbb{B}^m$. We say that the circuit $U = U_L \cdots U_1$ computes F with ancillas if we have

$$\sum_{z\in\mathbb{B}^{N-m}} \left|\langle F(x), z|U|x, 0^{N-n}\rangle\right|^2 \geq 2/3 \qquad\qquad \forall x \in \mathbb{B}^n$$

.
Just as for randomized algorithms, the number 2/3 is somewhat arbitrary. We can use a quantum circuit version of **Success_Boosting**.
Finally, we formally define a quantum algorithm. A quantum algorithm that computes a function $F : \mathbb{B}^* \to \mathbb{B}^*$ is a classical algorithm (Turing machine) that, given input x, computes the description of a quantum circuit C_x which computes $F(x)$ on

an empty input. The language defined by function $F : \mathbb{B}^* \to \mathbb{B}$ is said to belong to **BQP** if there is a quantum algorithm that computes F in polynomial time [5].
The *query complexity* of a quantum algorithm with oracle access to some quantum oracles is a function (depending on parameters such as input length) giving the maximum number of times the oracles are queried or used.
The *time complexity* of a quantum algorithm is given by a function (depending on parameters such as input length) giving the maximum size of the quantum circuit for given inputs. It is assumed that oracles can be accessed by a $\tilde{O}(1)$-sized circuit. We also assume that the time complexity of the algorithm (implemented via RAM TM) is at most a constant (or logarithmic, when we are working with $\tilde{O}(\cdot)$ notation) factor of the size of the circuit. This is necessary for the circuit to make sense for non-oracle problems, as otherwise the classical algorithm might take exponential time to compute the result and just output a quantum algorithm that outputs the result; this would mean that all non-oracle computable problems have a constant time complexity quantum algorithm. Note that this is may not be a standard definition.

1.6 Related Works

In this section, we shall give a brief history of works related to the k-mismatch problem. Let T denote the text string of length $n \geq 1$, P denote the pattern string of length $1 \leq m \leq n$, and let k be a positive integer such that $1 \leq k \leq m$.
In 1986 [2], Landau and Vishkin gave a $O(k(m \log m + n))$-time and $O(nk)$-space classical algorithm for the k-mismatch problem. The algorithm also reported all occurences of substrings T' of T such that $\delta_H(T', P) \leq k$. This paper mentions another paper which solves the problem in $O(f(k)(n + m))$, where $f(k) \geq 2^k$ for all integer $k \geq 1$.
In the paper [1] from 1986, Galil and Giancarlo present a "compact version" of Landau and Vishkin's algorithm [2]. This algorithm uses $O(m)$ space, unlike its predecessor with $O(nk)$ space complexity. Its time complexity is $O(m \log m + kn)$ for general alphabets and $O(m + kn)$ for fixed alphabets.
The paper [20] from 2016 presents multiple results related to the k-mismatch problem.
The first is a deterministic $O(nk^2 \log(k)/m)$-time offline algorithm, which is a k-factor improvement of some other papers from SODA 2000. In the same year, a $O\left(n\sqrt{k \log k}\right)$ time algorithm was published.
The authors then present a randomised and online algorithm that uses $\tilde{O}(k^2)$ extra space and has the same time complexity (as the deterministic $O(nk^2 \log(k)/m)$-time offline algorithm). In this problem, the text of length n is streamed in one symbol at a time. The probability of error is very small as well: at most m^{-2}.
Then the authors solve an approximate version of the streaming k-mismatch problem. In this version, there is no length limit n to the text. The $(1 + \epsilon)$-approximate k-mismatch problem is defined as follows. Let T' be an m-length substring of T; let x be the algorithms output; and let $y = \delta_H(T', P)$. Then algorithm must satisfy $y \leq x \leq (1 + \epsilon)y$ with high probability or report that $x > k$ when $y > k$ with high probability. Here, $y = \delta_H(T', P)$. The algorithm presented by this paper's authors solves this in randomised $O(k^2 \text{polylog}\, m/\epsilon^2)$ space and $O(\text{polylog}\, m/\epsilon^2)$ worst-case

time per arriving symbol (this is a streaming problem). The probability of error is ony $1/m^2$ at most.
The final result of this paper is an algorithm for the streaming k-mismatch problem, where the streaming text arrives on symbol at a time without any limit on the text's final length. The algorithm is a randomised one with $O\left(k^2 \text{polylog}\, m\right)$ space complexity and worst-case $O\left(\sqrt{k}\log k + \text{polylog}\, m\right)$ time per arriving symbol. The probability of error, like for the previous two algorithms, is at most $1/m^2$.
In the paper [11] from 2018, Gawrychowski and Uznański exhibit an $\tilde{O}\left(n + nk/\sqrt{m}\right)$ time algorithm for the k-mismatch problem. This variation is almost the same as the previous one: if $\delta_H(T', P) > k$ then the algorithm should report ∞. The paper mentions Abrahamson's contribution of $O\left(n\sqrt{m\log m}\right)$ time algorithm to the text-to-pattern Hamming distance problem (k-mismatch problem where $k = m$). The authors also mention the work of Porat and Kopelowitz, where they solve the $(1\pm\epsilon)$-approximate text-to-pattern Hamming distance problem in $O\left(\epsilon^{-1} n \log n \log m\right)$ time complexity.
This paper's main contribution for the k-mismatch problem is a deterministic algorithm with time-complexity $O\left((m\log^2 m \log|\Sigma| + k\sqrt{m\log m})\cdot n/m\right)$.
The authors also show that unless the combinatorial matrix multiplication conjecture fails, for any $\epsilon, \alpha, \kappa > 0$ such that $\alpha/2 \leq \kappa \leq \alpha \leq 1$, there is no combinatorial algorithm solving pattern matching with $k = \Theta\left(n^\kappa\right)$ mismatches in time $O\left((k\sqrt{m})^{1-\epsilon}\cdot n/m\right)$ for a text of length n and pattern of length $m = \Theta\left(n^\alpha\right)$. Essentially, the algorithm presented in this paper is optimal up to logarithmic factors.
In the paper [19] from 2019, the authors find a pseudolinear algorithm for the k-mismatch problem. For small k and text compressed by a Straight Line Path (SLP) of size n with pattern of length m, the authors contribute an $\tilde{O}\left((n+m)k^c\right)$ time algorithm, where $c > 0$ is some fixed constant.
Before the paper [16] from 2022, the k-mismatch problem had not been studied from a quantum computational perspective. In this paper (and ours), the k-mismatch problem asks us to find an algorithm that can locate any (if it exists) k-mismatch of P in T.
The authors contributed a quantum algorithm with $\tilde{O}\left(k\sqrt{n}\right)$ time complexity and $\tilde{O}\left(k^{3/4}n^{1/2}m^{o(1)}\right)$ query complexity.
The latest relevant paper [18] is from 2024. This paper published a quantum algorithm with $\tilde{O}(\sqrt{kn})$ query complexity and $\tilde{O}\left(\sqrt{\frac{n}{m}}\left(\sqrt{km}+k^2\right)\right)$ time complexity.

1.7 Overview of Our Main Results

The pseudocodes for the algorithms and their proof of correctness (and time complexity) are presented in Sections 3.2 and 3.3 respectively.

1.7.1 Bounded Hamming Distance Pattern Matching with Large Bound

The algorithm for Theorem 10 is very simple. We use a modification of Amplitude Amplification [4], [18] (via Theorem 6) along with quantum counting [4] (via Corollary 2, which in turn is a combination of Theorem 9 and Theorem 8). Theorem 6 is used to search for $j \in [0..n-m]$ such that $\delta_H(T[j..j+m-1], P) \leq m - r$.

We can check a candidate $j \in [0..n-m]$ by using Theorem 9 or by directly using Theorem 8 depending on a certain corner case ($r = 1$). Since verifying a candidate j takes $\tilde{O}(\sqrt{rm})$ time and queries, the overall time complexity and query complexity becomes $\tilde{O}(\sqrt{rmn})$.
Compared to [18] and [16]'s algorithms, this is faster when $r = o(m)$.
Let $k = m - r$.
When we consider their time complexity upper bounds, it is faster than [16]'s quantum algorithm by a factor of $f(r, m, n)$ that satisfies $f(r, m, n) = \tilde{\Omega}\left(\frac{k\sqrt{n}}{\sqrt{rmn}}\right)$ and thus $f(r, m, n) = \tilde{\Omega}\left(\sqrt{m/r} - \sqrt{r/m}\right)$. If $r = o(m)$, then our algorithm is faster than [16] by a factor of $\tilde{\Omega}\left(\sqrt{m/r}\right)$.
Again, when we assume that $r = o(m)$ and consider their time complexity upper bounds, it is faster than [18]'s quantum algorithm by a factor of $f(r, m, n)$ that satisfies

$$
\begin{aligned}
f(r, m, n) &= \tilde{\Omega}\left(\frac{\sqrt{n/m}\left(k^2 + \sqrt{km}\right)}{\sqrt{rmn}}\right) \\
f(r, m, n) &= \tilde{\Omega}\left(\frac{k^2}{m\sqrt{r}} + \sqrt{\frac{k}{r}}\right) \\
f(r, m, n) &= \tilde{\Omega}\left(\frac{k^2}{m\sqrt{r}}\right) \\
f(r, m, n) &= \tilde{\Omega}\left(\sqrt{m}\left(\sqrt{m/r} - 2\sqrt{r/m}\right)\right) \\
f(r, m, n) &= \tilde{\Omega}\left(m/\sqrt{r}\right) \\
f(r, m, n) &= \omega\left(\sqrt{m}\right)
\end{aligned}
$$

However, we are not claiming that Theorem 10's quantum algorithm is better than the ones cited. Just as this algorithm is faster when $r = o(m)$, theirs are better when $k = o(m)$.

1.7.2 Approximate Bounded Hamming Distance Pattern Matching

The algorithm for Theorem 11 is also quite simple. Just as before, we use a modification of Amplitude Amplification [4], [18] (via Theorem 6) to search for $j \in [0..n-m]$ such that $\delta_H(T[j..j+m-1], P) \leq k$. We check a candidate j using the quantum algorithm provided by Theorem 9, which takes $\tilde{O}\left(\sqrt{m/k}\right)$ time and queries. So the overall time complexity and query complexity becomes $\tilde{O}(\sqrt{mn/k})$.
Theorem 9 is a simple adaptation of Lemma 3.12 from [18]. We state the theorem here for convenience:

Theorem (Theorem 9)**.** *There is a quantum algorithm that, given oracle access to two strings X and Y of equal length $|X| = |Y| = m$, an integer threshold $k > 0$, and $\epsilon \in (0, 1]$, outputs YES (1) or NO (0) so that*

- *If $\delta_H(X, Y) \leq k$, then the algorithm outputs YES with probability of at least 9/10.*

- *If* $\delta_H(X,Y) > (1+\epsilon)k$*, then the algorithm outputs NO with probability of at least* $9/10$*.*

This algorithm takes $\tilde{O}\left(\epsilon^{-1}\sqrt{m/k}\right)$ *quantum time.*

Let $f(\epsilon,k,m,n)$ denote the ratio (disregarding polylogarithmic factors) between the upper bound of the algorithm from [16] and the algorithm from Theorem 9. Let $g(\epsilon,k,m,n)$ denote the ratio (disregarding polylogarithmic factors) between the upper bound of the algorithm from [18] and the algorithm from Theorem 9.
Suppose that $k = \omega(m^{2/3}\epsilon^{-2/3})$. Then $k^{3/2} = \omega(m\epsilon^{-1})$ and

$$
\begin{aligned}
f(\epsilon,k,m,n) &= \tilde{\Omega}\left(\frac{k\sqrt{n}}{\epsilon^{-1}\sqrt{mn/k}}\right)\\
f(\epsilon,k,m,n) &= \tilde{\Omega}\left(\frac{\epsilon k\sqrt{k}}{\sqrt{m}}\right)\\
f(\epsilon,k,m,n) &= \omega\left(\frac{\epsilon(m\epsilon^{-1})}{\sqrt{m}}\right)\\
f(\epsilon,k,m,n) &= \omega\left(\sqrt{m}\right)\\
g(\epsilon,k,m,n) &= \tilde{\Omega}\left(\frac{\sqrt{n/m}\left(\sqrt{km}+k^2\right)}{\epsilon^{-1}\sqrt{mn/k}}\right)\\
g(\epsilon,k,m,n) &= \tilde{\Omega}\left(\frac{\epsilon k}{\sqrt{m}}+\frac{\epsilon k^2\sqrt{k}}{m}\right)\\
g(\epsilon,k,m,n) &= \tilde{\Omega}\left(\frac{\epsilon k^2\sqrt{k}}{m}\right)\\
g(\epsilon,k,m,n) &= \tilde{\Omega}\left(\frac{\epsilon k\sqrt{k}\cdot k}{m}\right)\\
g(\epsilon,k,m,n) &= \omega\left(\frac{\epsilon(m\epsilon^{-1})\cdot k}{m}\right)\\
g(\epsilon,k,m,n) &= \omega\left(k\right)
\end{aligned}
$$

Thus, the algorithm from Theorem 11 is faster than that of [16] and [18] when $k = \tilde{\omega}\left(m^{2/3}\epsilon^{-2/3}\right)$.
In particular, suppose that $p \in (0,1)$ and $\epsilon \in (0,1)$ are fixed parameters and further suppose that the parameter k is given by $k = pm$. In this case, the time complexity becomes $\tilde{O}\left(\epsilon^{-1}\sqrt{n/p}\right) = \tilde{O}\left(\sqrt{n}\right)$.

Chapter 2

Important Quantum Algorithms

The content of this chapter is based on [4].

2.1 Quantum Amplitude Amplification

Let $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$. Let $\chi : \mathbb{B}^n \to \mathbb{B}$ be a Boolean function.
We say that $x \in \mathbb{B}^n$ is *good* if and only if $\chi(x) = 1$. Otherwise, we say that x is *bad*. The subspace of $\mathcal{H}$ spanned by the set of all $|x\rangle \in \mathcal{H}$ where $x \in \mathbb{B}^n$ is good is called the *good subspace* of $\mathcal{H}$. Similarly, the subspace of $\mathcal{H}$ spanned by the set of all $|x\rangle \in \mathcal{H}$ where $x \in \mathbb{B}^n$ is bad is called the *bad subspace*.
Given $\mathcal{A} \in \mathbf{U}\left((\mathbb{C}^2)^{\otimes n}\right)$, we define,

$$\mathbf{Q}(\mathcal{A}, \chi) = -\mathcal{A}\mathbf{S}_0\mathcal{A}^{-1}\mathbf{S}_\chi$$

where $\mathbf{S}_0$ and $\mathbf{S}_\chi$ are defined as follows: for all $x \in \mathbb{B}^n$,

$$\mathbf{S}_\chi|x\rangle = \begin{cases} -|x\rangle & \chi(x) = 1 \\ |x\rangle & \chi(x) = 0 \end{cases} \tag{2.1}$$

$$\mathbf{S}_0|x\rangle = \begin{cases} -|x\rangle & x = 0 \\ |x\rangle & x \neq 0 \end{cases} \tag{2.2}$$

Clearly, $\mathbf{Q}(\mathcal{A}, \chi)$ is unitary. Furthermore, if we are given unitary circuits for $\mathcal{A}$ and $\mathbf{S}_\chi$, then we can compose them to get a unitary circuit for $\mathbf{Q}(\mathcal{A}, \chi)$.
Let $|\Psi\rangle = \mathcal{A}|0\rangle$, let $|\Psi_1\rangle$ and $|\Psi_0\rangle$ be the projection of $|\Psi\rangle$ on the good and bad subspaces of $\mathcal{H}$ respectively. Let $a = \langle\Psi_1|\Psi_1\rangle$.
For the rest of this paper, we shall abuse notation to write $\mathbf{Q}$ to denote $\mathbf{Q}(\mathcal{A}, \chi)$.

Lemma 7.

$$\mathbf{Q}|\Psi_1\rangle = (1-2a)|\Psi_1\rangle - 2a|\Psi_0\rangle$$
$$\mathbf{Q}|\Psi_0\rangle = 2(1-a)|\Psi_1\rangle + (1-2a)|\Psi_0\rangle$$

Proof. First, note that $\mathbf{S}_0 = I - 2|0\rangle\langle 0|$. So,

$$\mathcal{A}\mathbf{S}_0\mathcal{A}^{-1} = \mathcal{A}I\mathcal{A}^{-1} - 2\mathcal{A}|0\rangle\langle 0|\mathcal{A}^{-1} = I - 2|\Psi\rangle\langle\Psi|.$$

Here, I is the identity transformation acting on $\mathcal{H}$.

Since $|\Psi_1\rangle$ is the projection of $|\Psi\rangle$ on the good subspace, $\mathbf{S}_\chi|\Psi_1\rangle = -|\Psi_1\rangle$. Similarly, since $|\Psi_2\rangle$ is the projection of $|\Psi\rangle$ on the bad subspace, $\mathbf{S}_\chi|\Psi_0\rangle = |\Psi_0\rangle$. Therefore,

$$\mathbf{Q}|\Psi_1\rangle = \mathcal{A}\mathbf{S}_0\mathcal{A}^{-1}|\Psi_1\rangle = (I - 2|\Psi\rangle\langle\Psi|)\,|\Psi_1\rangle = (1-2a)|\Psi_1\rangle - 2a|\Psi_0\rangle$$
$$\mathbf{Q}|\Psi_0\rangle = -\mathcal{A}\mathbf{S}_0\mathcal{A}^{-1}|\Psi_0\rangle = -\,(I - 2|\Psi\rangle\langle\Psi|)\,|\Psi_0\rangle = 2(1-a)|\Psi_1\rangle + (1-2a)|\Psi_0\rangle$$

□

Let $\mathcal{H}_\Psi$ be the subspace of $\mathcal{H}$ spanned by $|\Psi_1\rangle$ and $|\Psi_0\rangle$. It is clear that $\mathbf{Q}$ maps $\mathcal{H}_\Psi$ to itself.
If $a = 0$ or $a = 1$, then $\mathcal{H}_\Psi$ is 1 dimensional. Otherwise, $\mathcal{H}_\Psi$ is 2 dimensional.
Assuming that $0 < a < 1$, $\mathbf{Q}$ has the following eigenvectors in $\mathcal{H}_\Psi$, along with the following eigenvalues:

$$|\Psi_+\rangle = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a}}|\Psi_1\rangle + \frac{i}{\sqrt{1-a}}|\Psi_0\rangle\right)$$
$$|\Psi_-\rangle = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a}}|\Psi_1\rangle - \frac{i}{\sqrt{1-a}}|\Psi_0\rangle\right)$$
$$\lambda_+ = e^{+i2\theta_a}$$
$$\lambda_- = e^{-i2\theta_a}$$

where $i = \sqrt{-1}$, and $\theta_a \in [0, \pi/2]$ is such that $(\sin\theta_a)^2 = a$.
To see this, we simply apply $\mathbf{Q}$ to get

$$\begin{aligned}
&\mathbf{Q}|\Psi_\pm\rangle \\
&= \mathbf{Q}\left(\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a}}|\Psi_1\rangle \pm \frac{i}{\sqrt{1-a}}|\Psi_0\rangle\right)\right) \\
&= \frac{1}{\sqrt{2a}}\left((1-2a)|\Psi_1\rangle - 2a|\Psi_0\rangle\right) \pm \frac{i}{\sqrt{2(1-a)}}\left(2(1-a)|\Psi_1\rangle + (1-2a)|\Psi_0\rangle\right) \\
&= \left(\frac{1-2a}{\sqrt{2a}} \pm i\sqrt{2(1-a)}\right)|\Psi_1\rangle \pm \left(\frac{(1-2a)i}{\sqrt{2(1-a)}} \mp \sqrt{2a}\right)|\Psi_0\rangle \\
&= \frac{1}{\sqrt{2a}}\left(1-2a \pm i\sqrt{4a(1-a)}\right)|\Psi_1\rangle \pm \frac{i}{\sqrt{2(1-a)}}\left(1-2a \pm i\sqrt{4a(1-a)}\right)|\Psi_0\rangle \\
&= \left(1-2a \pm i\sqrt{4a(1-a)}\right)\cdot\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{a}}|\Psi_1\rangle \pm \frac{i}{\sqrt{1-a}}|\Psi_0\rangle\right) \\
&= \left(1-2(\sin\theta_a)^2 \pm i\sqrt{4(\sin\theta_a)^2(1-(\sin\theta_a)^2)}\right)\left(\frac{1}{\sqrt{a}}|\Psi_1\rangle \pm \frac{i}{\sqrt{1-a}}|\Psi_0\rangle\right) \\
&= \left(1-2(\sin\theta_a)^2 \pm i2\sin\theta_a\cos\theta_a\right)\left(\frac{1}{\sqrt{a}}|\Psi_1\rangle \pm \frac{i}{\sqrt{1-a}}|\Psi_0\rangle\right) \\
&= (\cos 2\theta_a \pm i\sin 2\theta_a)\left(\frac{1}{\sqrt{a}}|\Psi_1\rangle \pm \frac{i}{\sqrt{1-a}}|\Psi_0\rangle\right) \\
&= e^{\pm i2\theta_a}\left(\frac{1}{\sqrt{a}}|\Psi_1\rangle \pm \frac{i}{\sqrt{1-a}}|\Psi_0\rangle\right) \\
&= \lambda_\pm\left(\frac{1}{\sqrt{a}}|\Psi_1\rangle \pm \frac{i}{\sqrt{1-a}}|\Psi_0\rangle\right)
\end{aligned}$$

Lemma 8. *Let j be a non-negative integer. Then, if $0 < a < 1$,*

$$\begin{aligned}\mathbf{Q}^j|\Psi\rangle &= \frac{-i}{\sqrt{2}}\left(e^{(2j+1)i\theta_a}|\Psi_+\rangle - e^{-(2j+1)i\theta_a}|\Psi_-\rangle\right)\\ &= \frac{1}{\sqrt{a}}\sin\left((2j+1)\theta_a\right)|\Psi_1\rangle + \frac{1}{\sqrt{1-a}}\cos\left((2j+1)\theta_a\right)|\Psi_0\rangle\end{aligned}$$

Proof. Since $|\Psi_+\rangle \in \mathcal{H}_\Psi$ and $|\Psi_-\rangle \in \mathcal{H}_\Psi$ are eigenvectors with distinct eigenvalues of the unitary $\mathbf{Q}$, they form an orthonormal basis for $\mathcal{H}_\Psi$ (assuming that $0 < a < 1$). So we can express $|\Psi\rangle$ in terms of $|\Psi_\pm\rangle$. More precisely, we have

$$\begin{aligned}\frac{-i}{\sqrt{2}}\left(e^{i\theta_a}|\Psi_+\rangle - e^{-i\theta_a}|\Psi_-\rangle\right) &= \frac{-i}{2}\left(\left(\frac{e^{i\theta_a}}{\sqrt{a}} - \frac{e^{-i\theta_a}}{\sqrt{a}}\right)|\Psi_1\rangle + \left(\frac{ie^{i\theta_a}}{\sqrt{1-a}} + \frac{ie^{-i\theta_a}}{\sqrt{1-a}}\right)|\Psi_0\rangle\right)\\ &= \frac{1}{\sqrt{a}}\left(\frac{e^{i\theta_a} - e^{-i\theta_a}}{2i}\right)|\Psi_1\rangle + \frac{1}{\sqrt{1-a}}\left(\frac{e^{i\theta_a} + e^{-i\theta_a}}{2}\right)|\Psi_0\rangle\\ &= \frac{1}{\sqrt{a}}\sin\theta_a|\Psi_1\rangle + \frac{1}{\sqrt{1-a}}\cos\theta_a|\Psi_0\rangle\\ &= |\Psi_1\rangle + |\Psi_0\rangle\\ &= |\Psi\rangle\end{aligned}$$

Note that we have $\sin\theta_a = \sqrt{a}$ and $\cos\theta_a = \sqrt{1-a}$ because $(\sin\theta_a)^2 = a$ and $0 \le \theta_a \le \pi/2$.
We already know that the statement is true for $j = 0$. Assume that $j \ge 1$ and that the statement is true for $j-1$. Then,

$$\begin{aligned}\mathbf{Q}^j|\Psi\rangle &= \mathbf{Q}\mathbf{Q}^{j-1}|\Psi\rangle\\ &= \mathbf{Q}\left(\frac{-i}{\sqrt{2}}\left(e^{(2j-1)i\theta_a}|\Psi_+\rangle - e^{-(2j-1)i\theta_a}|\Psi_-\rangle\right)\right)\\ &= \frac{-i}{\sqrt{2}}\left(e^{(2j+1)i\theta_a}|\Psi_+\rangle - e^{-(2j+1)i\theta_a}|\Psi_-\rangle\right)\\ &= \frac{-i}{2}\left(\left(\frac{e^{(2j+1)i\theta_a}}{\sqrt{a}} - \frac{e^{-(2j+1)i\theta_a}}{\sqrt{a}}\right)|\Psi_1\rangle + \left(\frac{ie^{(2j+1)i\theta_a}}{\sqrt{1-a}} + \frac{ie^{-(2j+1)i\theta_a}}{\sqrt{1-a}}\right)|\Psi_0\rangle\right)\\ &= \frac{1}{\sqrt{a}}\left(\frac{e^{(2j+1)i\theta_a} - e^{-(2j+1)i\theta_a}}{2i}\right)|\Psi_1\rangle + \frac{1}{\sqrt{1-a}}\left(\frac{e^{i(2j+1)\theta_a} + e^{-(2j+1)i\theta_a}}{2}\right)|\Psi_0\rangle\\ &= \frac{1}{\sqrt{a}}\sin(2j+1)\theta_a|\Psi_1\rangle + \frac{1}{\sqrt{1-a}}\cos(2j+1)\theta_a|\Psi_0\rangle\end{aligned}$$

□

Corollary 1. *For any integer $m \ge 0$, if we compute $\mathbf{Q}^m|\Psi\rangle$ and then measure it, the probability of getting a good state is $\left(\sin\left((2m+1)\theta_a\right)\right)^2$.*

Proof. If $0 < a < 1$, then Lemma 8 shows that

$$\mathbf{Q}^m|\Psi\rangle = \frac{1}{\sqrt{a}}\sin(2m+1)\theta_a|\Psi_1\rangle + \frac{1}{\sqrt{1-a}}\cos(2m+1)\theta_a|\Psi_0\rangle$$

. Clearly, noting that $\||\Psi_1\rangle\| = \sqrt{a}$, the probability of measuring a good state is $\left(\sin\left((2m+1)\theta_a\right)\right)^2$.

If $a = 0$, then $|\Psi\rangle = |\Psi_0\rangle$ and $\mathbf{Q}|\Psi\rangle = \mathbf{Q}|\Psi_0\rangle = |\Psi_0\rangle$ (by Lemma 7). In this case, we have $\sin\theta_a = 0$, $\theta_a = 0$ and thus the probability of measuring a good state is $0 = (\sin 0)^2 = (\sin((2j+1)\cdot 0))^2$.
If $a = 1$, then $|\Psi\rangle = |\Psi_1\rangle$ and $\mathbf{Q}|\Psi\rangle = \mathbf{Q}|\Psi_1\rangle = |\Psi_1\rangle$ by Lemma 7. In this case, we have $\sin\theta_a = 1$, $\theta_a = \pi/2$, and thus the probability of measuring a good state is $1 = \left(\sin\frac{\pi}{2}\right)^2 = \left(\sin\left((2j+1)\cdot\frac{\pi}{2}\right)\right)^2$. □

Finally, we arrive at the first theorem of this chapter: Quadratic speedup without knowing a [4]. Our presentation of the theorem is slightly different from [4] to correspond with our definition of quantum algorithms. More precisely, while their quantum algorithm has guaranteed success but random (and unbounded) amount of application of $\mathcal{A}$ and $\mathcal{A}^{-1}$, the presentation here gives a quantum algorithm with finite applications of $\mathcal{A}$ and $\mathcal{A}^{-1}$ but only a guaranteed 2/3 probability of success. While the two are essentially the same, this difference is to account for our quantum circuit model, where the size of the circuit is fixed.

Theorem 5 (Quadratic speedup without knowing a [4])**.** *There exists a quantum algorith* **QSearch** *with the following property. Let $\mathcal{A}$ be any quantum algorithm (that uses no measurements), and let $\chi : \mathbb{B}^n \to \mathbb{B}$ be a Boolean computable function. Also suppose that we are given oracle access or a quantum circuit for computing χ. Let a denote the success probability of $\mathcal{A}$. Let T be a positive parameter such that $a = 0$ or $T \geq 1/a$. If $a = 0$ then* **QSearch** *reports no answer. Otherwise,* **QSearch** *reports an answer in $O(\sqrt{T})$ applications of $\mathcal{A}$ and $\mathcal{A}^{-1}$. with probability greater than or equal to* 2/3*.*

Proof. First, note the pseudocode of the algorithm 2, where α and C are some fixed constants to be determined later.
Note that using principle of deferred measurement, we can postpone the measurements to the end of the algorithm and then ignore them.
Since χ is computed deterministically, if we measure f and get 1, then o is guaranteed to be a good solution.
Let t_f be a random variable representing the final value of t if we ignore the $l < L$ condition in line 3. Also let l_f be a random variable representing the final value of l if we ignore the $l < L$ condition. So, ignore this condition ($l < L$) for now.
At the lth iteration of the outerloop, t increases by at most $1 + \lceil c^l \rceil \leq 2 + c^l$. Therefore, if $a \geq 3/4$, then

$$\begin{aligned}\mathbb{E}[t_f] &\leq \sum_{l=1}^{\infty} \left(c^l + 2\right)\left(\frac{1}{4}\right)^{l-1} \\ &= 4\sum_{l=1}^{\infty}\left(\frac{c}{4}\right)^l + 8\sum_{l=1}^{\infty}\left(\frac{1}{4}\right)^l \\ &< \infty\end{aligned}$$

In other words, if $a \geq 3/4$, $\mathbb{E}[t_f]$ is bounded above by a constant $C/3$. And obviously, $l_f \leq t_f$ and thus $\mathbb{E}[l_f] \leq C/3$. So $\mathbb{P}[l_f \geq C] \leq \frac{1}{3}$ and $\mathbb{P}[l_f \leq C] \geq 2/3$.
Now, suppose that $0 < a < 3/4$.

Algorithm 2 QSearch$(\mathcal{A}, \chi, T)$

1: Set $l \leftarrow 0, t \leftarrow 0, f \leftarrow 0, o \leftarrow 0$ and let c be any constant such that $1 < c < 2$.
2: Set constant $L \leftarrow \max\left(C, \left\lceil \log\left(4\alpha\sqrt{T}\right) / \log c \right\rceil\right)$.
3: **while** $l < L$ and $f = 0$ **do**
4: Set $l \leftarrow l + 1$ and set $M \leftarrow \lceil c^l \rceil$.
5: Set $t \rightarrow t + 1$.
6: Apply $\mathcal{A}$ on the intial state of appropriate size $|0\rangle$.
7: Measure the system, let $|z\rangle$ denote the outcome of the register on which $\mathcal{A}$ acts.
8: **if** $\chi(z) = 1$ **then**
9: Set $o \leftarrow z$ and $f \leftarrow 1$
10: **else**
11: Initialize a register of appropriate size to $|\Psi\rangle = \mathcal{A}|0\rangle$.
12: Pick an integer j between 1 and M uniformly at random.
13: Set $t \leftarrow t + j$.
14: Apply $\mathbf{Q}(\mathcal{A}, \chi)^j$ to the register.
15: Measure the register, let $|z\rangle$ denote the outcome.
16: **if** $\chi(z) = 1$ **then**
17: Set $o \leftarrow z$ and $f \leftarrow 1$.
18: **end if**
19: **end if**
20: **end while**
21: **return** (o, f).

Then $1 > 1 - a > 1/4$ and $1 < \frac{1}{\sqrt{1-a}} < 2$. Also, $0 < \theta_a < \arcsin\sqrt{3/4} = \pi/3$ and thus $0 < 2\theta_a < 2\pi/3$. So $\sin 2\theta_a > 0$. Furthermore, for any $M > 0$, we have

$$\begin{aligned}
2\sin 2\theta_a \sum_{j=1}^{M} \cos\left((2j+1)2\theta_a\right) &= \sum_{j=1}^{M} 2\cos\left((2j+1)2\theta_a\right)\sin 2\theta_a \\
&= \sum_{j=1}^{M} \sin\left((2j+1)2\theta_a + 2\theta_a\right) - \sin\left(2j \cdot 2\theta_a\right) \\
&= \sum_{j=1}^{M} \sin\left(\left(2(j+1)\cdot 2\theta_a\right)\right) - \sin\left(2j \cdot 2\theta_a\right) \\
&= \sin\left(2(M+1)\cdot 2\theta_a\right) - \sin\left(2 \cdot 2\theta_a\right) \\
\sum_{j=1}^{M} \cos\left((2j+1)2\theta_a\right) &= \frac{\sin\left(2(M+1)\cdot 2\theta_a\right) - \sin\left(2 \cdot 2\theta_a\right)}{2\sin 2\theta_a}
\end{aligned}$$

Now, at each iteration, the probabilty p_M of getting $\chi(z) = 1$ at line 16 satisfies the following

$$
\begin{aligned}
p_M &= \frac{1}{M}\sum_{j=1}^{M}\left(\sin\left((2j+1)\theta_a\right)\right)^2 \\
&= \frac{1}{M}\sum_{j=1}^{M}\frac{1-\cos\left(2(2j+1)\theta_a\right)}{2} \\
&= \frac{1}{2} - \frac{1}{2M}\sum_{j=1}^{M}\cos\left((2j+1)2\theta_a\right) \\
&= \frac{1}{2} - \frac{1}{2M}\left(\frac{\sin\left(4(M+1)\theta_a\right)-\sin\left(4\theta_a\right)}{2\sin 2\theta_a}\right) \\
&\geq \frac{1}{2} - \frac{1}{2M}\left(\frac{1}{\sin 2\theta_a}\right) \\
&= \frac{1}{2} - \frac{1}{2M}\left(\frac{1}{2\sin\theta_a\cos\theta_a}\right) \\
&= \frac{1}{2} - \frac{1}{2M}\left(\frac{1}{2\sqrt{a(1-a)}}\right) \\
&> \frac{1}{2} - \frac{1}{2M}\frac{2}{2\sqrt{a}} \\
&= \frac{1}{2} - \frac{1}{2M\sqrt{a}}
\end{aligned}
$$

Let $c_0 > 0$ be such that $c = 2(1-c_0)$ and let $M_0 = 1/(c_0\sqrt{a})$. Since $1 < c < 2$, we have $1 < 2(1-c_0) < 2$ and thus $0 < c_0 < 1/2$. We have $\mathbb{E}[t_f] \leq T_1 + T_2$, where T_1 is the maximum number value of t while $M < M_0$ and T_2 is the expected increase of t while $M \geq M_0$.
Let L_0 denote $\lfloor \frac{\log M_0}{\log c} \rfloor$. We must have $l \leq L_0$ while $M < M_0$. So,

$$
\begin{aligned}
T_1 &\leq \sum_{l=1}^{L_0}(c^l + 2) \\
&= 2L_0 + \sum_{l=1}^{L_0} c^l \\
&= 2L_0 + \frac{c^{L_0+1} - c}{c-1} \\
&\leq 2L_0 + c^{L_0+1} \\
&\leq \frac{\log M_0}{\log c} + cM_0 \\
&= \frac{\log M_0}{\log c} + \frac{c}{c_0\sqrt{a}} \\
&= O(1/\sqrt{a})
\end{aligned}
$$

Now, $M \geq M_0$ implies that $-1/M \geq -1/M_0 \geq c_0\sqrt{a}$ and thus $p_M \geq \frac{1}{2}(1-c_0)$. Therefore, the probability of $\chi(z) = 0$ at line 16 when $M \geq M_0$ is at most $q_0 =$

$1 - \frac{1}{2}(1 - c_0) = \frac{1}{2}(1 + c_0)$. Thus,

$$\begin{aligned} T_2 &\leq \sum_{j=0}^{\infty} (M_0 c^j + 2) q_0^j \\ &= \sum_{j=0}^{\infty} M_0 (c q_0)^j + 2 q_0^j \\ &= M_0 \sum_{j=0}^{\infty} (c q_0)^j + 2 \sum_{j=0}^{\infty} q_0^j \\ &= \frac{1}{c_0 \sqrt{a}} \sum_{j=0}^{\infty} (c q_0)^j + 2 \sum_{j=0}^{\infty} q_0^j \\ &= O\left(1/\sqrt{a}\right) \end{aligned}$$

Note that $\sum_{j=0}^{\infty} (cq_0)^j$ and $\sum_{j=0}^{\infty} q_0^j$ are some finite and well-defined constants because $0 \leq q_0 < 1$ and $0 \leq cq_0 = 2(1 - c_0) \cdot \frac{1}{2}(1 + c_0) = 1^2 - c_0^2 < 1$.
Therefore, $\mathbb{E}[t_f] \leq T_1 + T_2 = O(1/\sqrt{a})$. So there exists a constant $\beta > 0$ such that $\mathbb{E}[t_f] \leq \beta/\sqrt{a}$. Using Markov's inequality, we see that

$$\mathbb{P}\left[t_f \geq \frac{4\beta}{\sqrt{a}}\right] \leq \frac{1}{4}$$

So by letting $\alpha = 4\beta$ and assuming that $T \geq 1/\alpha$, we have $\mathbb{P}[t_f \leq \alpha\sqrt{T}] \geq 3/4$. This means that if our algorithm is restricted to $\alpha\sqrt{T}$ applications of $\mathcal{A}$ and $\mathcal{A}^{-1}$, the algorithm succeeds with probability of at least 3/4.
Again, ignoring the $l < L$ condition, we know that $\mathbb{P}[t_f \geq \alpha/\sqrt{a}] \leq 1/4$. So,

$$\begin{aligned} \frac{1}{4} \geq \mathbb{P}[t_f \geq \alpha/\sqrt{a}] &\geq \mathbb{P}\left[l_f \geq \frac{\log\left(4\alpha/\sqrt{a}\right)}{\log c}\right] \cdot \frac{3}{4} \\ \frac{1}{3} &\geq \mathbb{P}\left[l_f \geq \frac{\log\left(4\alpha/\sqrt{a}\right)}{\log c}\right] \end{aligned}$$

Since $L = \left\lceil \log\left(4\alpha\sqrt{T}\right) / \log c \right\rceil \geq \log\left(4\alpha/\sqrt{a}\right) / \log c$, we have a success rate of at least 2/3.
Let A and B denote the size of the circuit for $\mathcal{A}$ and χ respectively. Then the size of the circuit for **QSearch** must be $O\left(\sum_{l=1}^{L} \left(c^l A + B\right)\right) = O\left(c^L A + LB\right)$.
So, the final size of the circuit becomes

$$O\left(c^L A + LB\right) = O\left(\sqrt{T} A + B \log \sqrt{T}\right)$$

with the circuit for $\mathcal{A}$ and $\mathcal{A}^{-1}$ being used $O\left(\sqrt{T}\right)$ times. □

The next result is from [18], though our presentation may differ.

Theorem 6. *Let $n \geq 1$ be an integer and let $N = 2^n$. Let $F : [0..N-1] \to \{0, 1, 2\}$ be a function. Let $\mathcal{D}$ be a quantum circuit such that for any $j \in [0..N-1]$, if $F(j) = 0$ or $F(j) = 1$ then*

$$|\langle j, F(j), 0^x | \mathcal{D} | j, 0, 0^x \rangle| \geq 2/3.$$

Here, x refers to the number of auxiliary qubits used by $\mathcal{D}$. Then there is a quantum circuit $\mathcal{B}$ such that $F^{-1}(\{1\}) = \varnothing$ or

$$\sum_{j \in F^{-1}(\{1,2\})} |\langle j, 1, 0^y | \mathcal{B} | 0, 0, 0^y \rangle|^2 \geq \frac{2}{3}.$$

In any case,

$$\sum_{j \in F^{-1}(\{0\})} |\langle j, 1, 0^y | \mathcal{B} | 0, 0, 0^y \rangle|^2 \leq \frac{1}{3}.$$

Here, y is the number of auxiliary qubits used by $\mathcal{B}$. Furthermore, $\mathcal{B}$ queries $\mathcal{D}$ at most $\tilde{O}\left(\sqrt{N}\right)$ times. And $\mathcal{B}$ increases the circuit size of $\mathcal{D}$ by a factor of $\tilde{O}\left(\sqrt{N}\right)$.

Proof. First, we define the **QSearch′** (Algorithm 3), which is a slight variation if Algorithm 2 where everytime we check $\chi(z) = 1$, we also check if **Success_Boosting**$(\mathcal{D}, 4,)$. outputs 1. Note that C and α are as defined in Algorithm 2.

Algorithm 3 QSearch′$(\mathcal{A}, \mathcal{D}, \chi, T)$

Set $l \leftarrow 0, t \leftarrow 0, f \leftarrow 0, o \leftarrow 0$ and let c be any constant such that $1 < c < 2$.
Set constant $L \leftarrow \max\left(C, \left\lceil \log\left(4\alpha\sqrt{T}\right) / \log c \right\rceil\right)$.
while $l < L$ and $f = 0$ **do**
    Set $l \leftarrow l + 1$ and set $M \leftarrow \lceil c^l \rceil$.
    Set $t \rightarrow t + 1$.
    Apply $\mathcal{A}$ on the intial state of appropriate size $|0\rangle$.
    Measure the system, let $|z, b\rangle$ denote the outcome of the register on which $\mathcal{A}$ acts.
    if $\chi(z, b) = 1 \wedge \mathcal{D}(z)$ **then**
        Set $o \leftarrow (z, b)$ and $f \leftarrow 1$
    else
        Initialize a register of appropriate size to $|\Psi\rangle = \mathcal{A}|0\rangle$.
        Pick an integer j between 1 and M uniformly at random.
        Set $t \leftarrow t + j$.
        Apply $\mathbf{Q}(\mathcal{A}, \chi)^j$ to the register.
        Measure the register, let $|z\rangle$ denote the outcome.
        if $\chi(z) = 1 \wedge \mathcal{D}(z)$ **then**
            Set $o \leftarrow (z, b)$ and $f \leftarrow 1$.
        end if
    end if
end while
return (o, f).

We then define algorithm $\mathcal{A}$ as in Algorithm 4.
Note that we are using principle of deferred measurement on Algorithm 4 to postpone any measurement and then ignore the measurement. So $\mathcal{A}$ outputs a superposition of states $|j\rangle|b\rangle$ for $j \in [0..N-1]$ and $b \in \mathbb{B}$.
For $n \in \mathbb{N}$, let $N = 2^n$ and define $\chi_N : [0..N-1] \times \mathbb{B} \rightarrow \mathbb{B}$ by

$$\chi(j, b) = b \qquad \forall j \in [0..N-1], b \in \mathbb{B}.$$

Algorithm 4 Weak_Search_Auxiliary$(\mathcal{D}, N)$

Require: $N = 2^n$ for some integer $n \geq 1$.

1: Sample j uniformly randomly from $[0..N-1]$.
2: Set $b \leftarrow$ **Success_Boosting**$(\mathcal{D}, 4, j)$.
3: **return** (j, b).

Algorithm 5 Weak_Search$(\mathcal{D}, N)$

Require: $N = 2^n$ for some integer $n \geq 1$.

1: Let $\mathcal{A}$ denote **Weak_Search_Auxiliary**.
2: Let $\mathcal{D}'$ denote **Success_Boosting**
3: Let $\gamma \in \mathbb{N}$ be a fixed constant such that **QSearch′** uses at most $\gamma\sqrt{N}$ applications of $\mathcal{A}(\mathcal{D}, N)$.
4: Let $\lambda \geq 4$ be a fixed integer such that $4\gamma 2^{-\lambda+\frac{1}{2}} \leq \frac{1}{9}$.
5: **for** $t \in [0..1]$ **do**
6: Set $((j, b), f) \leftarrow$ **QSearch′** $(\mathcal{A}(\mathcal{D}, N), \mathcal{D}'(\mathcal{D}, \lambda, \_), \chi_N, 2N)$.
7: **if** $f = 1$ **then**
8: **return** (j, f).
9: **end if**
10: **end for**
11: **return** $(0, 0)$.

Let $\mathcal{B}$, N, F be given.
Let L be the constant defined in line 2 of Algorithm 3. Let $Z_0, \ldots, Z_{4L-1}$ and $B_0, \ldots, B_{4L-1}$ be random variables for each measured z, b (line 7 and 15). Note that there are $2 \cdot 2L$ indices for the random variables, because we are computing **QSearch′** twice.
Let F' and J' be the random variable for the final output of **Weak_Search**. Then,

$$\begin{aligned}
\mathbb{P}[F' = 1 \wedge F(J') = 0] &\leq \sum_{j=0}^{4L-1} \mathbb{P}[F(Z_j) = 0 \wedge B_j = 1] \\
&\leq \sum_{j=0}^{4L-1} \mathbb{P}[F(Z_j) \neq B_j] \\
&\leq \sum_{j=0}^{4L-1} N^{-\lambda} \\
&= (4L-1)N^{-\lambda}
\end{aligned}$$

To simplify calculation, we use the fact that $L \leq \gamma\sqrt{N}$ For large enough $N \geq 2$, we have

$$\mathbb{P}[F' = 1 \wedge F(J') = 0] \leq 4LN^{-\lambda} \leq \gamma 4N^{-\lambda+\frac{1}{2}} \leq 4\gamma 2^{-\lambda+\frac{1}{2}} \leq \frac{1}{9}$$

In other words, we have shown that

$$\sum_{j \in F^{-1}(\{0\})} |\langle j, 1, 0^y | \mathcal{B} | 0, 0, 0^y \rangle|^2 \leq \frac{1}{9} \leq \frac{1}{3}.$$

Now, suppose that $F^{-1}(\{1\}) \neq \varnothing$. Let J and B be random variables denoting the j and b from Algorithm 4. Then

$$\begin{aligned}\mathbb{P}[B=1] &\geq \mathbb{P}[F(J)=1 \wedge B=1] \\ &= \mathbb{P}[F(J)=1] \cdot \mathbb{P}[B=1|F(J)=1] \\ &\geq \frac{1}{N} \cdot \left(1 - N^{-\lambda}\right)\end{aligned}$$

For $N \geq 2$, $N^{-\lambda} \leq N^{-4} \leq \frac{1}{16}$ and thus

$$\mathbb{P}[B=1] \geq \frac{1}{N}\left(1 - N^{-\lambda}\right) \geq \frac{15}{16N} \geq \frac{1}{2N}$$

In other words, the a for **Weak_Search_Auxiliary** is bounded below by $\frac{1}{2N}$ (when $N \geq 2$). Since we are repeating **QSearch′** twice, we have $\mathbb{P}[F'=1] \geq 1 - \frac{1}{3} \cdot \frac{1}{3} = \frac{8}{9}$ For any $j \in [0..4L-1]$,

$$\begin{aligned}\mathbb{P}[F(Z_j)=0 \wedge B_j=1] &= \mathbb{P}[F(Z_j)=0] \cdot \mathbb{P}[B_j=1|F(Z_j)=0] \\ &\leq 1 \cdot N^{-\lambda} \\ &= N^{-\lambda}\end{aligned}$$

Using the fact that $\mathbb{P}[F'=1 \wedge F(J')=0] \leq \frac{1}{9}$, we get

$$\begin{aligned}\frac{8}{9} &\leq \mathbb{P}[F'=1] \\ &= \mathbb{P}\left[F'=1 \wedge F(J')=0\right] + \mathbb{P}[F'=1 \wedge F(J') \in \{1,2\}] \\ &\leq \frac{1}{9} + \mathbb{P}\left[F'=1 \wedge F(J') \in \{1,2\}\right] \\ \frac{7}{9} &\leq \mathbb{P}[F'=1 \wedge F(J') \in \{1,2\}]\end{aligned}$$

In other words, if $F^{-1}\left(\{1\}\right) \neq \varnothing$, then

$$\sum_{j \in F^{-1}(\{1,2\})} \left|\langle j, 1, 0^y | \mathcal{B} | 0, 0, 0^y \rangle\right|^2 \geq \frac{7}{9} \geq \frac{2}{3}.$$

□

The function F in Theorem 6 partitions $[0..N-1]$ into 3 sets: [18] notes them as good ($I^+ = F^{-1}(\{1\})$), bad ($I^- = F^{-1}(\{0\})$), and neutral ($I^\sim = F^{-1}(\{2\})$). They named this "Quantum Search on Bounded-Error Inputs and Neutral Inputs", while we are calling this "Weak Search" because the presence of the neutral elements make the search less "predictable". This is because there is no restriction on how likely it is for $\mathcal{D}$ to accept or reject neutral elements: an element may very well have exactly 1/2 chance of being accepted by $\mathcal{D}$. Our algorithm is unlikely (with less than 1/3 chance) to report a bad element. If a good element exists, then the algorithm is guaranteed (with at least 2/3 chance) to report a good or a neutral element. If no good element exists, but there are neutral elements, the algorithm may or may not report one of the neutral elements. At least, it will not report any bad element (with at least 2/3 chance).

2.2 Quantum Amplitude Estimation

First, we establish a circuit for quantum Fourier Transform. Given an integer $N \geq 1$, the Quantum Fourier Transform (QFT) $F_N \in \mathbf{U}\left(\mathbb{C}^N\right)$ is defined by, for all $x \in [0..N-1]$,

$$F_N|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \exp\left(2\pi i x y/N\right)|y\rangle.$$

Here, we're assuming that $|0\rangle, \ldots, |N-1\rangle$ form a basis for $\mathbb{C}^N$.

Lemma 9. *Given $N = 2^n$, where $n \geq 1$. There is a $O(n^2)$ size quantum circuit that computes F_N.*

Proof. This proof (and the circuit) is from [3]. Any integer $x, y \in [0..2^n - 1]$ can be represented in binary by $x_0 \ldots x_{n-1}$ and $y_0 \ldots y_{n-1}$, where $x = x_{n-1}2^{n-1} + \cdots + x_0$ and $y = y_{n-1}2^{n-1} + \cdots + y_0$.
Now, given $x \in [0..N-1]$,

$$\begin{aligned}
F_N|x_0 \ldots x_{n-1}\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \exp\left(2\pi i x y/N\right)|y\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \bigotimes_{k=0}^{n-1} \left[\exp\left(2\pi i x y_k 2^k/N\right)|y_k\rangle\right] \\
&= \frac{1}{\sqrt{N}} \bigotimes_{k=0}^{n-1} \left[|0\rangle + \exp\left(2\pi i x 2^k/N\right)|1\rangle\right] \\
&= \frac{1}{\sqrt{N}} \bigotimes_{k=0}^{n-1} \left[|0\rangle + \exp\left(2\pi i x 2^{n-k}\right)|1\rangle\right] \\
&= \frac{1}{\sqrt{N}} \bigotimes_{l=n}^{1} \left[|0\rangle + \exp\left(2\pi i x 2^{-l}\right)|1\rangle\right]
\end{aligned}$$

Now, note that $e^{2\pi i k} = 1$ for any integer k. Therefore,

$$\begin{aligned}
F_N|x\rangle &= \frac{1}{\sqrt{N}} \bigotimes_{l=n}^{1} \left[|0\rangle + \exp\left(2\pi i \sum_{j=0}^{n} x_j 2^{j-l}\right)|1\rangle\right] \\
&= \frac{1}{\sqrt{N}} \bigotimes_{l=n}^{1} \left[|0\rangle + \exp\left(2\pi i \sum_{j=0}^{l-1} x_j 2^{j-l}\right)|1\rangle\right]
\end{aligned}$$

Define, for all positive integers k,

$$R_k = |0\rangle\langle 0| + \exp\left(2\pi i/2^k\right)|1\rangle\langle 1|$$

Applying H (Hadamard) to j-th qubit produces $\frac{1}{\sqrt{2}}\left(|0\rangle + \exp\left(2\pi i x_j 2^{-1}\right)|1\rangle\right)$ because $\exp\left(2\pi i x_j 2^{-1}\right)$ is -1 when $x_j = 1$ and 1 when $x_j = 0$. We then apply R_2, …, R_{j+1} gates on the jth qubit, conditioned on $j-1, \ldots,$ 0-th qubits respectively.
Since we apply at most n gates to each qubit, the total size of the circuit becomes $O(n^2)$. □

In the Quantum Amplitude Estimation problem, given $\mathcal{A}$ and χ as before, we need to estimate a, the probability of getting a good solution. To do so, we find an estimate for θ_a, where θ_a was defined as the real number satisfying $0 \leq \theta_a \leq \pi/2$ and $a = (\sin\theta_a)^2$. We first show that an estimation of θ_a gives an estimation for a. It is more precisely stated in the next lemma.

Lemma 10. *Let $a = (\sin\theta_a)^2$ and $\tilde{a} = \left(\sin\tilde{\theta_a}\right)^2$, with $0 \leq \theta_a, \tilde{\theta_a} \leq 2\pi$ then, for any ϵ such that $0 \leq \epsilon \leq \pi/2$,*

$$\left|\theta_a - \tilde{\theta_a}\right| \leq \epsilon \Rightarrow |\tilde{a} - a| \leq 2\epsilon\sqrt{a(1-a)} + \epsilon^2$$

Proof. For any δ such that $0 \leq \delta \leq \pi/2$,

$$\begin{aligned}
(\sin(\theta_a+\delta))^2 - (\sin(\theta_a))^2 &= (\sin\theta_a\cos\delta + \cos\theta_a\sin\delta)^2 - a \\
&= \left(\sqrt{a}\cos\delta + \sqrt{1-a}\sin\delta\right)^2 - a \\
&= a(\cos\delta)^2 + \sqrt{a(1-a)}\cdot 2\cos\delta\sin\delta + (1-a)(\sin\delta)^2 - a \\
&= a\left(1-(\sin\delta)^2\right) - a + (1-a)(\sin\delta)^2 + \sqrt{a(1-a)}\cdot\sin 2\delta \\
&= \sqrt{a(1-a)}\sin 2\delta + (1-2a)(\sin\delta)^2 \\
&\leq \sqrt{a(1-a)}\sin 2\delta + (\sin\delta)^2 \\
-(\sin(\theta_a+\delta))^2 + (\sin(\theta_a))^2 &= -\sqrt{a(1-a)}\sin 2\delta + (2a-1)(\sin\delta)^2 \\
&\geq -\sqrt{a(1-a)}\sin 2\delta - (\sin\delta)^2 \\
(\sin\theta_a)^2 - (\sin(\theta_a-\delta))^2 &= a - (\sin\theta_a\cos\delta - \cos\theta_a\sin\delta)^2 \\
&= a - \left(\sqrt{a}\cos\delta - \sqrt{1-a}\sin\delta\right)^2 \\
&= a - \left(a(\cos\delta)^2 - \sqrt{a(1-a)}\cdot 2\cos\delta\sin\delta + (1-a)(\sin\delta)^2\right) \\
&= a - \left(a\left(1-(\sin\delta)^2\right) + (1-a)(\sin\delta)^2 - \sqrt{a(1-a)}\sin 2\delta\right) \\
&= \sqrt{a(1-a)}\sin 2\delta + (2a-1)(\sin\delta)^2 \\
&\leq \sqrt{a(1-a)}\sin 2\delta + (2\cdot 1 - 1)(\sin\delta)^2 \\
&\leq \sqrt{a(1-a)}\sin 2\delta + (\sin\delta)^2 \\
-(\sin\theta_a)^2 + (\sin(\theta_a-\delta))^2 &= -\sqrt{a(1-a)}\sin 2\delta + (1-2a)(\sin\delta)^2 \\
&\geq -\sqrt{a(1-a)}\sin 2\delta - \left(\sin\delta^2\right)
\end{aligned}$$

For all $x \geq 0$, we have $\sin x \leq x$. This is because $0 - \sin 0 = 0$ and $\frac{d}{dt}(t - \sin t) = 1 - \cos t \geq 0$. So,

$$\begin{aligned}
\left|(\sin(\theta_a+\delta))^2 - (\sin(\theta_a))^2\right| &\leq \sqrt{a(1-a)}\sin 2\delta + (\sin\delta)^2 \quad &\leq 2\delta\sqrt{a(1-a)} + \delta^2 \\
\left|(\sin\theta_a)^2 - (\sin(\theta_a-\delta))^2\right| &\leq \sqrt{a(1-a)}\sin 2\delta + (\sin\delta)^2 \quad &\leq 2\delta\sqrt{a(1-a)} + \delta^2
\end{aligned}$$

If $\left|\tilde{\theta_a} - \theta_a\right| \leq \epsilon$, for some ϵ satisfying $0 \leq \epsilon \leq \pi/2$, then $\left|\tilde{\theta_a} - \theta_a\right| = \delta$ for some $\delta \leq \epsilon$. So, $\tilde{\theta_a} = \theta_a + \delta$ or $\tilde{\theta_a} = \theta_a - \delta$ and thus,

$$|\tilde{a} - a| \leq 2\delta\sqrt{a(1-a)} + \delta^2 \leq 2\epsilon\sqrt{a(1-a)} + \epsilon^2$$

□

This means that an estimation for θ_a with a maximum error of ϵ will give an estimation for a with a maximum error of $\epsilon + \epsilon^2$ (note that $a(1-a)$ is maximised when $a = 1/2$).
Since $\lambda_\pm = e^{\pm i2\theta_a}$ are the eigenvalues of $\mathbf{Q}$, it is enough to estimate the eigenvalues of $\mathbf{Q}$ to get an estimate for θ_a and thus a.
For any positive integer M and any unitary operator U, we define $\Lambda_M(U)$ by

$$\Lambda_M(U)|j\rangle|y\rangle = |j\rangle\left(U^j|y\rangle\right) \qquad\qquad 0 \leq j < M$$

Definition 26. *For any integer $M > 0$ and a real number $0 \leq \omega < 1$, define*

$$|\mathcal{S}_M|\omega\rangle\rangle = \frac{1}{\sqrt{M}} \sum_{y=0}^{M-1} \exp\left(2\pi i\omega y\right)|y\rangle$$

So for all integers $x \in [0..M-1]$, we have $F_M|x\rangle = |\mathcal{S}_M(x/M)\rangle$.

Definition 27. *For any two real numbers $\omega_0, \omega_1 \in \mathbb{R}$, define*

$$d_{\mathcal{S}}(\omega_0, \omega_1) = \min_{z\in\mathbb{Z}} \{|z + \omega_1 - \omega_0|\}$$

Lemma 11. *Let $\omega_0, \omega_1 \in [0, 1)$ and $\Delta = d_{\mathcal{S}}(\omega_0, \omega_1)$. If $\Delta = 0$ we have $|\langle\mathcal{S}_M(\omega_0)|\mathcal{S}_M(\omega_1)\rangle|^2 = 1$. Otherwise,*

$$|\langle\mathcal{S}_M(\omega_0)|\mathcal{S}_M(\omega_1)\rangle|^2 = \frac{\left(\sin\left(M\pi\Delta\right)\right)^2}{M^2\left(\sin\left(\pi\Delta\right)\right)^2}$$

Proof.

$$\begin{aligned}|\langle\mathcal{S}_M(\omega_0)|\mathcal{S}_M(\omega_1)\rangle|^2 &= \left|\left(\frac{1}{\sqrt{M}}\sum_{y=0}^{M-1}\exp\left(-2\pi i\omega_0 y\right)\langle y|\right)\left(\frac{1}{\sqrt{M}}\sum_{y=0}^{M-1}\exp\left(2\pi i\omega_0 y\right)|y\rangle\right)\right|^2 \\ &= \frac{1}{M^2}\left|\sum_{y=0}^{M-1}\exp\left(2\pi i(\omega_1-\omega_0)y\right)\right|^2\end{aligned}$$

Now, for any integer $z \in \mathbb{Z}$, we have $e^{2\pi z} = 1$. So, for any integer $y \in [0..M-1]$, $\exp\left(2\pi i(\omega_1-\omega_0)y\right) = \exp\left(\pm 2\pi i y\Delta\right)$.
If $\exp\left(2\pi i(\omega_1-\omega_0)\right) = \exp\left(-2\pi i y\Delta\right)$, then

$$\begin{aligned}\frac{1}{M^2}\left|\sum_{y=0}^{M-1}\exp\left(2\pi i(\omega_1-\omega_0)y\right)\right|^2 &= \frac{1}{M^2}\left|\sum_{y=0}^{M-1}\exp\left(-2\pi i\Delta y\right)\right|^2 \\ &= \frac{1}{M^2}\left|\sum_{y=0}^{M-1}\overline{\exp\left(2\pi i\Delta y\right)}\right|^2 \\ &= \frac{1}{M^2}\left|\overline{\sum_{y=0}^{M-1}\exp\left(2\pi i\Delta y\right)}\right|^2 \\ &= \frac{1}{M^2}\left|\sum_{y=0}^{M-1}\exp\left(2\pi i\Delta y\right)\right|^2\end{aligned}$$

So, assuming $\Delta \neq 0$,

$$\begin{aligned}
|\langle \mathcal{S}_M(\omega_0)|\mathcal{S}_M(\omega_1)\rangle|^2 &= \frac{1}{M^2}\left|\sum_{y=0}^{M-1}\exp\left(2\pi i\Delta y\right)\right|^2 \\
&= \frac{1}{M^2}\left|\frac{e^{i2\pi\Delta M}-1}{e^{i2\pi\Delta}-1}\right|^2 \\
&= \frac{1}{M^2}\left|\frac{e^{i\pi\Delta M}\left(e^{i\pi\Delta M}-e^{-i\pi\Delta M}\right)}{e^{i\pi\Delta}\left(e^{i\pi\Delta}-e^{-i\pi\Delta}\right)}\right|^2 \\
&= \frac{1}{M^2}\left|e^{i\pi\Delta(M-1)}\left(\frac{e^{i\pi\Delta M}-e^{-i\pi\Delta M}}{e^{i\pi\Delta}-e^{-i\pi\Delta}}\right)\right|^2 \\
&= \frac{1}{M^2}\left|\frac{\left(e^{i\pi\Delta M}-e^{-i\pi\Delta M}\right)/2i}{\left(e^{i\pi\Delta}-e^{-i\pi\Delta}\right)/2i}\right|^2 \\
&= \frac{1}{M^2}\left|\frac{\sin\left(M\Delta\pi\right)}{\sin\left(\Delta\pi\right)}\right|^2 \\
&= \frac{\left(\sin\left(M\pi\Delta\right)\right)^2}{M^2\left(\sin\left(\pi\Delta\right)\right)^2}
\end{aligned}$$

If $\Delta = 0$, then

$$|\langle \mathcal{S}_M(\omega_0)|\mathcal{S}_M(\omega_1)\rangle|^2 = \frac{1}{M^2}\left|\sum_{y=0}^{M-1}\exp\left(2\pi i\Delta y\right)\right|^2 = \frac{1}{M^2}\left|M\right|^2 = 1$$

□

Lemma 12. *Let $\omega \in [0,1)$, let $x \in [0..M-1]$ be an integer, and let X be a discrete random variable corresponding to measuring $F_M^{-1}|\mathcal{S}_M(\omega)\rangle$ in the computational basis. If $M\omega \in \mathbb{Z}$ then $\mathbb{P}[X = M\omega] = 1$. Else letting $\Delta = d_{\mathcal{S}}(\omega, x/M)$,*

$$\mathbb{P}[X = x] = \frac{\left(\sin\left(M\pi\Delta\right)\right)^2}{M^2\left(\sin\left(\pi\Delta\right)\right)^2} \leq \frac{1}{\left(2M\Delta\right)^2}.$$

For any integer $k > 1$, we also have

$$\mathbb{P}\left[d_{\mathcal{S}}(X/M, \omega) \leq k/M\right] \geq 1 - \frac{1}{2(k-1)}.$$

Proof.

$$\begin{aligned}
\mathbb{P}[X = x] &= \left|\langle x|F_M^{-1}|\mathcal{S}_M(\omega)\rangle\right|^2 \\
&= \left|\langle F_M x|\mathcal{S}_M(\omega)\rangle\right|^2 \\
&= \left|\langle \mathcal{S}_M(x/M)|\mathcal{S}_M(\omega)\rangle\right|^2
\end{aligned}$$

If $\Delta = 0$, then $x = M\omega$ and thus we have shown that $\mathbb{P}[X = M\omega] = 1$.
Note that we must have $\Delta \leq 1/2$. This is because if $|\omega_0 - \omega_1| > 1/2$, then assuming (without loss of generality) that $\omega_1 - \omega_0 > 1/2$ gives us $0 \leq 1 + \omega_0 - \omega_1 < 1/2$.

Furthermore, the function $f : (0, \pi/2] \to \mathbb{R}$ defined by $f(\theta) = \theta/\sin\theta$ for all $\theta \in (0, \pi/2]$ has maximum value $\pi/2$. To see that, we calculate $f'(\theta) = (\sin\theta - \theta\cos\theta)/(\sin\theta)^2$ and $\frac{d}{d\theta}(\sin\theta - \theta\cos\theta) = \theta\sin\theta$. Since $0\sin 0 = 0$ and $\theta\sin\theta \geq 0$ for $\theta \in [0, \pi/2]$, we have $f'(\theta) \geq 0$ for all $\theta \in (0, \pi/2]$. Since $\lim_{\theta\to 0} f(\theta) = 1$, this means that f is a non-decreasing function. In other words, $f(\pi/2) = \pi/2$ is its maximum value. Since $f(\theta)$ is non-negative on $[0, \pi/2]$, this means that $(\theta/\sin\theta)^2$ has maximum value $\pi^2/4$ for $\theta \in [0, \pi/2]$.
Assuming $\Delta \neq 0$,

$$\mathbb{P}[X = x] = \frac{(\sin(M\pi\Delta))^2}{M^2(\sin(\pi\Delta))^2}$$

We claim that $\frac{(\sin(M\pi\Delta))^2}{M^2(\sin(\pi\Delta))^2} \leq \frac{1}{(2M\Delta)^2}$. Let $\theta = \pi\Delta$. Clearly, $0 \leq \theta \leq \pi/2$. Then

$$\begin{aligned}
\frac{4\theta^2}{\pi^2}\left(\frac{\sin(\theta M)}{\sin\theta}\right)^2 &= \frac{4}{\pi^2}\left(\frac{\theta}{\sin\theta}\right)^2(\sin(\theta M))^2 \\
&\leq \frac{4}{\pi^2}\frac{\pi^2}{4}\cdot 1 \\
&= 1 \\
\left(\frac{\sin(\theta M)}{\sin\theta}\right)^2 &\leq \frac{\pi^2}{4\theta^2} \\
\frac{(\sin(M\pi\Delta))^2}{M^2(\sin(\pi\Delta))^2} &\leq \frac{1}{(2M\Delta)^2}
\end{aligned}$$

For any integer $j \geq 1$, noting that there are at most two x such that $j/M < d_{\mathcal{S}}(\omega, x/M) \leq (j+1)/M$, we have

$$\mathbb{P}\left[j/M < d_{\mathcal{S}}(X/M, \omega) \leq (j+1)/M\right] \leq \frac{2}{4M^2(j/M)^2} = \frac{1}{2j^2}$$

For any integer $k > 1$,

$$\begin{aligned}
\mathbb{P}\left[d_{\mathcal{S}}(X/M, \omega) \leq k/M\right] &= 1 - \mathbb{P}\left[d_{\mathcal{S}}(X/M, \omega) > k/M\right] \\
&= 1 - \sum_{j=k}^{\infty}\mathbb{P}\left[j/M < d_{\mathcal{S}}(X/M, \omega) \leq (j+1)/M\right] \\
&\geq 1 - \sum_{j=k}^{\infty}\frac{1}{2j^2} \\
&\geq 1 - \sum_{j=k}^{\infty}\int_{j-1}^{j}\frac{1}{2y^2}\,dy \\
&\geq 1 - \int_{k-1}^{\infty}\frac{1}{2j^2}\,dj \\
&= 1 - \frac{1}{2(k-1)}
\end{aligned}$$

□

Algorithm 6 Amplitude_Estimation$(\mathcal{A}, \chi, M)$

1: Let $n \geq 1$ be the smallest integer such that $M \leq 2^n$ and let $N = 2^n$.
2: Set $x \leftarrow |0^n\rangle$.
3: Set $y \leftarrow \mathcal{A}|0^m\rangle$.
4: Apply F_N on the first n qubits (that is, on x).
5: Apply $\Lambda_N(\mathbf{Q})$, where $\mathbf{Q} = -\mathcal{A}\mathbf{S}_0\mathcal{A}^{-1}\mathbf{S}_\chi$.
6: Apply F_N^{-1} on x.
7: Measure the first register and denote the outcome $|z\rangle$.
8: Output $\tilde{a} = \left(\sin\left(\pi z/N\right)\right)^2$.

Finally, we state the algorithm for estimating a.
The part before the measurement can be expressed as a unitary transformation $\left(\left(F_N^{-1} \otimes I\right) \Lambda_N(\mathbf{Q}) \left(F_N \otimes I\right)\right)$, on $|0\rangle\mathcal{A}|0\rangle$, where I refers an identity transformation on $\mathbf{U}\left(\left(\mathbb{C}^2\right)^{\otimes m}\right)$, where m is the number of qubits that $\mathcal{A}$ acts on (in other words, $\mathcal{A} \in \mathbf{U}\left(\left(\mathbb{C}^2\right)^{\otimes m}\right)$).
Note that $\Lambda_N(\mathbf{Q})$ can be implemented using controlled $\mathbf{Q}^{2^0}, \ldots, \mathbf{Q}^{2^{n-1}}$ gates. In other words, the algorithm queries $\mathcal{A}$ and χ $\Theta(M)$ times (as $2^0 + \cdots + 2^{n-1} = 2^n - 1 \leq 2M$).

Theorem 7 (Amplitude Estimation [4])**.** *For any positive integer $k > 1$, the algorithm* **Amplitude_Estimation**$(\mathcal{A}, \chi, M)$ *outputs $\tilde{a}) \in [0, 1]$ such that*

$$|\tilde{a} - a| \leq 2\pi k \frac{\sqrt{a(1-a)}}{M} + k^2 \frac{\pi^2}{M^2}$$

with probability at least $1 - \frac{1}{2(k-1)}$.

Proof. After line 3 from Algorithm 6, the state of the system is given by

$$|0\rangle\mathcal{A}|0\rangle = \frac{-i}{\sqrt{2}}|0\rangle \left(e^{i\theta_a}|\Psi_+\rangle - e^{-i\theta_a}|\Psi_-\rangle\right).$$

After line 4, we have (ignoring global phase)

$$\frac{1}{\sqrt{2N}} \sum_{j=0}^{N-1} |j\rangle \left(e^{i\theta_a}|\Psi_+\rangle - e^{-i\theta_a}|\Psi_-\rangle\right)$$

where n and N are as defined in Algorithm 6.

After line 5, we have (again ignoring global phase)

$$\begin{aligned}
&\frac{1}{\sqrt{2N}}\sum_{j=0}^{N-1}|j\rangle\left(e^{i\theta_a}e^{2ij\theta_a}|\Psi_+\rangle - e^{-i\theta_a}e^{-2ij\theta_a}|\Psi_-\rangle\right)\\
=&\frac{e^{i\theta_a}}{\sqrt{2N}}\sum_{j=0}^{N-1}e^{2ij\theta_a}|j\rangle|\Psi_+\rangle - \frac{e^{-i\theta_a}}{\sqrt{2N}}\sum_{j=0}^{N-1}e^{-2ij\theta_a}|j\rangle|\Psi_-\rangle\\
=&\frac{e^{i\theta_a}}{\sqrt{2N}}\sum_{j=0}^{N-1}e^{2\pi i(\theta_a/\pi)\cdot j}|j\rangle|\Psi_+\rangle - \frac{e^{-i\theta_a}}{\sqrt{2N}}\sum_{j=0}^{N-1}e^{2\pi ij-2ij\theta_a}|j\rangle|\Psi_-\rangle\\
=&\frac{e^{i\theta_a}}{\sqrt{2N}}\sum_{j=0}^{N-1}e^{2\pi ij(\theta_a/\pi)j}|j\rangle|\Psi_+\rangle - \frac{e^{-i\theta_a}}{\sqrt{2N}}\sum_{j=0}^{N-1}e^{2\pi i(1-\theta_a/\pi)j}|j\rangle|\Psi_-\rangle\\
=&\frac{e^{i\theta_a}}{\sqrt{2}}|\mathcal{S}_N\left(\theta_a/\pi\right)\rangle|\Psi_+\rangle - \frac{e^{-i\theta_a}}{\sqrt{2}}|\mathcal{S}_N\left(1-\theta_a/\pi\right)\rangle|\Psi_-\rangle
\end{aligned}$$

Assuming $j \neq 0$, for any $z \in \mathbb{Z}$,

$$z + \frac{\theta_a}{\pi} - \frac{j}{N} = -\left(-z + \left(1 - \frac{\theta_a}{\pi}\right) - \left(1 - \frac{j}{N}\right)\right)$$

If $j = 0$ then for any $z \in \mathbb{Z}$,

$$z + \frac{\theta_a}{\pi} - 0 = -\left((-z-1) + \left(1 - \frac{\theta_a}{\pi}\right) - 0\right)$$

In either case, we find that

$$d_{\mathcal{S}}\left(\frac{\theta_a}{\pi}, \frac{j}{N}\right) = d_{\mathcal{S}}\left(1 - \frac{\theta_a}{\pi}, \frac{(N-j) \mod N}{N}\right)$$

Therefore, for any $j \in [0..N-1]$, the probability of measuring $|j\rangle$ given $F_N^{-1}|\mathcal{S}_N(1-\theta_a/\pi)\rangle$ is the same as the probability of measuring $|(N-j) \mod N\rangle$ given $F_N^{-1}|\mathcal{S}_N(\theta_a/\pi)\rangle$. For $j \in [1..N-1]$,

$$\left(\sin\left(\pi\frac{N-j}{N}\right)\right)^2 = \left(\sin\left(\pi - \frac{\pi j}{N}\right)\right)^2 = \left(\sin\left(\frac{\pi j}{N}\right)\right)^2$$

If $j = 0$, then $(N-j) \mod N = 0$.
Let X be the discrete random variable representing the measurement result $|z\rangle$. Define Y to be the random variable with value θ_a/π if the second register is $|\Psi_+\rangle$ and value $1 - \theta_a/\pi$ otherwise. Then, for positive integers $k \geq 2$,

$$\mathbb{P}[d_{\mathcal{S}}(X/M, Y) \leq k/M] \geq a\left(1 - \frac{1}{2(k-1)}\right) + (1-a)\left(1 - \frac{1}{2(k-1)}\right) = 1 - \frac{1}{2(k-1)}$$

This is due to lemma 12.
Now, if $d_{\mathcal{S}}(X/M, Y) \leq k/M$, we have an approximation $\tilde{\theta_a}/\pi$ for θ_a/π or $1 - \theta_a/\pi$ satisfying $\left|\tilde{\theta_a}/\pi - \theta_a/\pi\right| \leq k/M$ or $\left|\tilde{\theta_a}/\pi - \theta'_a/\pi\right| \leq k/M$, where $\theta'_a = \pi - \theta_a$. So, we have $\left|\tilde{\theta_a} - \theta_a\right| \leq \frac{k\pi}{M}$ or $\left|\tilde{\theta_a} - \theta'_a\right| \leq \frac{k\pi}{M}$. By lemma 10, this means that

$$|\tilde{a} - a| \leq 2\pi k\frac{\sqrt{a(1-a)}}{N} + k^2\frac{\pi^2}{N^2} \leq 2\pi k\frac{\sqrt{a(1-a)}}{M} + k^2\frac{\pi^2}{M^2}$$

□

Using Amplitude Estimation, we have the following algorithm.

Theorem 8 (Counting [4])**.** *Suppose that we are given positive integers M and k, and boolean (computable) function $f : [0..N-1] \to \mathbb{B}$, where $N = 2^n$ for some integer $n \geq 1$. There is a quantum algorithm* **Count**(f, M) *that outputs an estimate t' to $t = |f^{-1}(1)|$ such that*

$$|t' - t| \leq 2\pi k \frac{\sqrt{t(N-t)}}{M} + \pi^2 k^2 \frac{N}{M^2}$$

with probability greater than $1 - \frac{1}{2(k-1)}$ for $k > 1$. Furthermore, this algorithm uses f $\Theta(M)$ times.

Proof. Now, set $\mathcal{A} = F_N$. Clearly, $\mathcal{A}$ maps $|0\rangle$ to $\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle$. We use the algorithm from Theorem 7 with $\mathcal{A}$, f, and M as parameters to get an estimation $\tilde{a}$ for the success probability a of $\mathcal{A}$.
Since t is the number of "good" solutions, the probability a of getting a good solution is exactly t/N. That is, $t = aN$. Let $t' = \tilde{a}N$. So, with probabiliy of $1 - \frac{1}{2(k-1)}$, we get t' such that

$$\begin{aligned}
|t' - t| &= N\,|\tilde{a} - a| \\
&\leq N \left(2\pi k \frac{\sqrt{a(1-a)}}{M} + k^2 \frac{\pi^2}{M^2} \right) \\
&= 2\pi k \frac{\sqrt{N^2 a(1-a)}}{M} + \pi^2 k^2 \frac{N}{M^2} \\
&= 2\pi k \frac{\sqrt{(Na)(N-Na)}}{M} + \pi^2 k^2 \frac{N}{M^2} \\
&= 2\pi k \frac{\sqrt{t(N-t)}}{M} + \pi^2 k^2 \frac{N}{M^2}
\end{aligned}$$

That the algorithm uses f $\Theta(M)$ times is directly from theorem 7. □

Chapter 3

Main Theorems

3.1 Approximate Bounded Hamming Distance Decision

The following is a generalization of Lemma 3.12 from [18] and its proof

Theorem 9. *There is a quantum algorithm that, given oracle access to two strings X and Y of equal length $|X| = |Y| = m$, an integer threshold $k > 0$, and $\epsilon \in (0, 1]$, outputs YES (1) or NO (0) so that*

- *If $\delta_H(X, Y) \leq k$, then the algorithm outputs YES with probability of at least $9/10$.*
- *If $\delta_H(X, Y) > (1 + \epsilon)k$, then the algorithm outputs NO with probability of at least $9/10$.*

This algorithm takes $\tilde{O}\left(\epsilon^{-1}\sqrt{m/k}\right)$ quantum time.

Proof. First, we present the quantum algorithm (Algorithm 7).

Algorithm 7 ApproxBoundedHammingDecider(X, Y, k, ϵ)

Set $m \leftarrow |X|$.
Set $N \leftarrow \min\{2^j : j \in \mathbb{N} \wedge 2^j \geq m\}$.
procedure F(j)
    return $j < m \wedge X_j \neq Y_j$.
end procedure
Set $M \leftarrow \left\lceil \frac{6\pi\sqrt{N/k}}{\sqrt{1+3\epsilon/2}-\sqrt{\epsilon}} \right\rceil$.
if $k \geq m$ **then**
    return 1
else
    Set $t' \leftarrow$ **Count**(F, M).
    return $t' \leq \left(1 + \frac{\epsilon}{2}\right) k$.
end if

If $k \geq m$, then the algorithm correctly returns YES (or 1, to be precise). Otherwise, the algorithm outputs YES if and only if $t' \leq \left(1 + \frac{\epsilon}{2}\right) k$.

For the rest of the proof, assume that $k < m$.
Instead of using Theorem 8 with parameters $\left(\left\lceil 48\pi\sqrt{N/k}\right\rceil, 6\right)$ as done in [18], we use parameters

$$\left(\left\lceil \frac{6\pi\sqrt{N/k}}{\sqrt{1+3\epsilon/2}-\sqrt{1+\epsilon}}\right\rceil, 6\right)$$

and with F as the Boolean predicate.
Let $\beta = \sqrt{1+3\epsilon/2}-\sqrt{1+\epsilon}$ and $\alpha = 6\pi/\beta$. Then our first parameter is $M = \left\lceil \alpha\sqrt{N/k}\right\rceil$. Calculating, we get

$$\begin{aligned}
\beta + \sqrt{1+\epsilon} &= \sqrt{1+\frac{3\epsilon}{2}} \\
\left(\beta+\sqrt{1+\epsilon}\right)^2 &= 1+\frac{3\epsilon}{2} \\
\beta^2 + 2\beta\sqrt{1+\epsilon} &= \frac{\epsilon}{2}
\end{aligned}$$

Let t denote the actual number of mismatches and let t' be a possible output by the counting algorithm. By Theorem 8, we have

$$|t'-t| \le 12\pi\frac{\sqrt{t(N-t)}}{M} + \frac{36\pi^2 N}{M^2} \le 12\pi\frac{\sqrt{tN}}{M} + \frac{36\pi^2 N}{M^2}$$

We shall show that if $t \le k$ then $t' \le (1+\epsilon/2)k$ and if $t > (1+\epsilon)k$ then $t' > (1+\epsilon/2)k$. First, suppose that $t \le k$. Then,

$$\begin{aligned}
t' &\le t + 12\pi\frac{\sqrt{tN}}{\left\lceil \alpha\sqrt{\frac{N}{k}}\right\rceil} + 36\pi^2\frac{N}{\left\lceil\frac{\alpha N}{k}\right\rceil^2} \\
&\le t + 12\pi\frac{\sqrt{tN}}{\alpha\sqrt{\frac{N}{k}}} + 36\pi^2\frac{N}{\left(\frac{\alpha^2 N}{k}\right)} \\
&= k + 12\pi\frac{\sqrt{tN}}{\alpha\sqrt{\frac{N}{k}}} + (6\pi/\alpha)^2 k \\
&= k + 2\beta\sqrt{kt} + \beta^2 k \\
&\le \left(1+2\beta+\beta^2\right)k \\
&= \left(1+2\beta\sqrt{1+\epsilon}+\beta^2\right)k - 2\beta\left(\sqrt{1+\epsilon}-1\right)k \\
&< \left(1+2\beta\sqrt{1+\epsilon}+\beta^2\right)k \\
&= (1+\epsilon/2)k
\end{aligned}$$

Now, suppose that $t > (1+\epsilon)k$. Then,

$$\begin{aligned}
t' &\geq t - \left(12\pi\frac{\sqrt{tN}}{M} + \frac{36\pi^2 N}{M^2}\right) \\
&\geq t - \left(12\pi\frac{\sqrt{tN}}{\left\lceil\alpha\sqrt{\frac{N}{k}}\right\rceil} + 36\pi^2\frac{N}{\left\lceil\frac{\alpha N}{k}\right\rceil^2}\right) \\
&\geq t - \left(12\pi\frac{\sqrt{tN}}{\alpha\sqrt{\frac{N}{k}}} + 36\pi^2\frac{N}{\left(\frac{\alpha^2 N}{k}\right)}\right) \\
&\geq t - \left(2\beta\sqrt{kt} + \beta^2 k\right) \\
&= \sqrt{kt}\left(\sqrt{\frac{t}{k}} - 2\beta\right) - \beta^2 k \\
&> \sqrt{k^2(1+\epsilon)}\left(\sqrt{1+\epsilon} - 2\beta\right) - \beta^2 k \\
&= (1+\epsilon)k - 2\beta k\sqrt{1+\epsilon} - \beta^2 k \\
&= (1+\epsilon)k - \left(2\beta\sqrt{1+\epsilon} - \beta^2\right)k \\
&= (1+\epsilon)k - \frac{\epsilon}{2}k \\
&\geq (1+\epsilon/2)k
\end{aligned}$$

So, using Theorem 8 with parameters $(M, 6)$ gives correct result with probability of at least $1 - 1/(2(6-1)) = 9/10$.
Finally, we analyze the time complexity of this algorithm. From Theorem 8, we know that our algorithm queries X and Y at most $\tilde{O}(M)$ times.

$$\begin{aligned}
M &= \left\lceil\frac{6\pi\sqrt{N/k}}{\sqrt{1+3\epsilon/2} - \sqrt{1+\epsilon}}\right\rceil \\
&\leq 1 + \frac{6\pi\sqrt{N/k}}{\sqrt{1+3\epsilon/2} - \sqrt{1+\epsilon}} \\
&\leq 1 + \frac{6\pi\sqrt{2m/k}}{\sqrt{1+3\epsilon/2} - \sqrt{1+\epsilon}} \\
&\leq 1 + \frac{6\pi\sqrt{2}\sqrt{m/k}}{\sqrt{1+3\epsilon/2} - \sqrt{1+\epsilon}} \\
&= O\left(\frac{\sqrt{\frac{m}{k}}}{\sqrt{1+\frac{3}{2}\epsilon} - \sqrt{1+\epsilon}}\right)
\end{aligned}$$

A little algebra shows that $\frac{1}{\beta} \leq 6\epsilon^{-1}$ because $0 < \epsilon \leq 1$:

$$\begin{aligned}
\frac{1}{2}\epsilon &= \left(1 + \frac{3}{2}\epsilon\right) - (1+\epsilon) \\
&= \left(\sqrt{1+\frac{3}{2}\epsilon} + \sqrt{1+\epsilon}\right)\left(\sqrt{1+\frac{3}{2}\epsilon} - \sqrt{1+\epsilon}\right) \\
&\leq \left(\sqrt{\frac{5}{2}} + \sqrt{2}\right)\left(\sqrt{1+\frac{3}{2}\epsilon} - \sqrt{1+\epsilon}\right) \\
&\leq 3\left(\sqrt{1+\frac{3}{2}\epsilon} - \sqrt{1+\epsilon}\right) \\
\frac{\epsilon}{6} &\leq \sqrt{1+\frac{3}{2}\epsilon} - \sqrt{1+\epsilon} \\
6\epsilon^{-1} &\geq \frac{1}{\sqrt{1+\frac{3}{2}\epsilon} - \sqrt{1+\epsilon}}
\end{aligned}$$

Thus, the complexity of the overall algorithm becomes $\tilde{O}\left(\epsilon^{-1}\sqrt{\frac{m}{k}}\right)$. □

The following corollary can be handled (and has been handled in [18]) without Theorem 9, but we use Theorem 9 for the sake of convenience.

Corollary 2. *There exists a quantum algorithm that, given oracle access to strings X and Y with $|X| = |Y| = m$ for some integer $m \geq 1$, and an integer threshold $r > 0$, decides if $\delta_H(X,Y) \leq m - r$ in time $\tilde{O}(\sqrt{rm})$*

Proof. First, we present the quantum algorithm (Algorithm 8):

Algorithm 8 LargeBoundedHammingDecider(X, Y, r)

Set $m \leftarrow |X|$.
Set $N \leftarrow \min\{2^j : j \in \mathbb{N} \wedge 2^j \geq m\}$.
procedure A'(j)
    return 0.
end procedure
procedure B'(j)
    return $j < m \wedge X_j = Y_j$.
end procedure
if $r = 1$ **then**
    Set $M \leftarrow \left\lceil 18\pi\sqrt{N} \right\rceil$.
    Set $t' \leftarrow$ **Count**(B', M).
    return $t' \geq \frac{2}{9}$.
else
    return ¬**ApproxBoundedHammingDecider** $\left(\text{A'}, \text{B'}, r-1, \frac{1}{r}\right)$.
end if

Through procedures A' and B', we are providing an oracle for binary strings A and B of length N respectively.

Since $A_j \neq B_j \Leftrightarrow X_j = Y_j$ for all $j \in [0..m-1]$,

$$\delta_H(X,Y) = m - \delta_H(A,B)$$

So $m - \delta_H(A,B) = \delta_H(X,Y) \leq m - r$ if and only if $\delta_H(A,B) \geq r$.
If $r = 1$ then we apply the algorithm from 8 with B' and parameters $\left(\left\lceil 18\pi\sqrt{N}\right\rceil, 6\right)$.
Let $t = \delta_H(A,B)$ and let t' denote a possible output of the counting algorithm. Then if $t = 0$, with probability of at least $1 - \frac{1}{2(6-1)} = \frac{9}{10}$,

$$\begin{aligned}
t' &\leq t + \left(\frac{2\pi \cdot 6\sqrt{0(N-0)}}{\left\lceil 18\pi\sqrt{N}\right\rceil} + 6^2\pi^2\frac{N}{\left\lceil 18\pi\sqrt{N}\right\rceil^2}\right) \\
&\leq 0 + \left(6^2\pi^2\frac{N}{\left(18\pi\sqrt{N}\right)^2}\right) \\
&\leq \frac{1}{9}
\end{aligned}$$

Otherwise, if $t \geq 1$, with probability of at least $\frac{9}{10}$,

$$\begin{aligned}
t' &\geq t - \left(\frac{2\pi \cdot 6\sqrt{t(N-t)}}{\left\lceil 18\pi\sqrt{N}\right\rceil} + 6^2\pi^2\frac{N}{\left\lceil 18\pi\sqrt{N}\right\rceil^2}\right) \\
&\geq t - \left(\frac{2\pi \cdot 6\sqrt{t(N-t)}}{18\pi\sqrt{N}} + 6^2\pi^2\frac{N}{\left(18\pi\sqrt{N}\right)^2}\right) \\
&\geq t - \left(\frac{2\pi \cdot 6\sqrt{tN}}{18\pi\sqrt{N}} + 6^2\pi^2\frac{N}{\left(18\pi\sqrt{N}\right)^2}\right) \\
&\geq t - \left(\frac{2}{3}\sqrt{t} + \frac{1}{9}\right) \\
&= \sqrt{t}\left(\sqrt{t} - \frac{2}{3}\right) - \frac{1}{9} \\
&\geq \frac{1}{3} - \frac{1}{9} \\
&= \frac{2}{9}
\end{aligned}$$

If $t' \leq \frac{1}{9}$, then $t = 0 < 1$. Otherwise, $t' \geq \frac{2}{9}$ and $t \geq 1$. So when $r = 1$, our algorithm correctly decides the result with probability of at least 9/10.
Now, assume that $r \geq 2$. We apply and return the negation of the result of Algorithm 7 from Theorem 9 with strings A, B, integer threshold $r-1$, and $\epsilon = \frac{1}{r}$. If $\delta_H(A,B) \leq r-1$ then **ApproxBoundedHammingDecider** returns YES with probability of at least 9/10. If $\delta_H(A,B) \geq r > r - 1 + \frac{1}{r} = \left(1 + \frac{1}{r}\right)(r-1)$ then **ApproxBoundedHammingDecider** returns NO with probability of at least 9/10.

So with a probability of at least 9/10, **ApproxBoundedHammingDecider** outputs NO if and only if $\delta_H(A, B) \geq r$. Since Algorithm 8 returns the negation of the result from **ApproxBoundedHammingDecider** in this case, Algorithm 8's result is correct with a probability of at least 9/10. □

3.2 Bounded Hamming Distance Pattern Matching with Large Bound

Theorem 10. *There exists a quantum algorithm that, given oracle access to a pattern P of length m and a text T of length n, and an integer threshold $r > 0$, reports the existence of $i \in [0..n-m]$ such that $\delta_H(T[i..i+m), P) \leq m - r$. Furthermore, this algorithm has time complexity $\tilde{O}(\sqrt{rmn})$.*

Proof. First, we present the quantum algorithm (Algorithm 9):

Algorithm 9 LargeBoundedDistanceMatching(T, P, r)

```
1: Set n ← |T|, m ← |P|.
2: Set N ← min{2^j : j ∈ ℕ ∧ 2^j ≥ n − m + 1}.
3: procedure DECIDER(j)
4:     if j > n − m then
5:         return 0.
6:     else
7:         return LargeBoundedHammingDecider(T[j..j + m), P, r).
8:     end if
9: end procedure
10: return Weak_Search(DECIDER, N).
```

Define $F : [0..N-1] \to \{0, 1, 2\}$ by letting, for $j \in [0..N-1]$,

$$F(j) = \begin{cases} 1 & j \leq n - m \wedge \delta_H(T[j..j+m), P) \leq m - r \\ 0 & \text{otherwise} \end{cases}$$

.
Applying Theorem 6 with DECIDER and F, we get a $\tilde{O}\left(\sqrt{N} \cdot \sqrt{rm}\right) = \tilde{O}\left(\sqrt{rmn}\right)$ time algorithm (assuming queries to P and T takes $\tilde{O}(1)$ time).
This works because DECIDER outputs 1 with probability of at least $\frac{2}{3}$ when $j \leq n - m \wedge \delta_H(T[j..j+m-1], P) \leq m - r$, and outputs 0 with probability of at least 2/3 otherwise. □

3.3 Approximate Bounded Hamming Distance Pattern Matching

Now, we arrive at the final result of this thesis.

Theorem 11. *There exists a quantum algorithm that, given oracle access to a pattern P of length m and a text T of length n, an integer threshold $k > 0$, and $\epsilon \in (0, 1]$, such that:*

- *if there exists an* $j \in [0..n-m]$ *such that* $\delta_H(T[j..j+m), P) \leq k$*, then the algorithm, upon measurement, outputs* $(j', 1)$ *for some* $j' \in [0..n-m]$ *satisfying* $\delta_H(T[j'..j'+m), P) \leq (1+\epsilon)k$ *with a probability of at least* $9/10$*;*
- *if, for all* $j \in [0..n-m]$*, we have* $\delta_H(T[j..j+m), P) > (1+\epsilon)k$*, then the algorithm, upon measurement, outputs* $(j', 0)$ *for some* $j' \in [0..2n-1]$ *with probability of at least* $9/10$*.*

This algorithm has time complexity $\tilde{O}\left(\sqrt{\frac{mn}{k}}\right)$ *(assuming that* P *and* T *can be accessed in* $\tilde{O}(1)$ *time).*

Proof. First, we present the quantum algorithm (Algorithm 10):

Algorithm 10 ApproxBoundedDistMatching(T, P, k, ϵ)

```
Set n ← |T|, m ← |P|.
Set N ← min{2^j : j ∈ ℕ ∧ 2^j ≥ n − m + 1}.
procedure DECIDER(j)
    if j > n − m then
        return 0.
    else
        return ApproxBoundedHammingDecider(T[j..j + m), P, k, ϵ).
    end if
end procedure
return Weak_Search(DECIDER, N).
```

Define $F : [0..N-1] \to \{0, 1, 2\}$ by letting, for $j \in [0..N-1]$,

$$F(j) = \begin{cases} 0 & j > n-m \vee \delta_H(T[j..j+m-1], P) > (1+\epsilon)k \\ 1 & \delta_H(T[j..j+m-1], P) \leq k \\ 2 & \text{otherwise} \end{cases}$$

.

From Theorem 9, it is clear that for $j \in [0..N-1]$, $F(j) = 1$ implies that DECIDER returns 1 with probability of at least $2/3$ and $F(j) = 0$ implies that $\mathcal{D}$ returns 0 with probability of at least $2/3$.

Thus, applying Theorem 6 with DECIDER and F, we get our desired quantum algorithm with time complexity $\tilde{O}\left(\sqrt{\frac{mn}{k}}\right)$. □

Bibliography

[1] Z. Galil and R. Giancarlo, "Improved string matching with k mismatches," *SIGACT News*, vol. 17, no. 4, pp. 52–54, Mar. 1986, ISSN: 0163-5700. DOI: 10.1145/8307.8309. [Online]. Available: https://doi.org/10.1145/8307.8309.

[2] G. M. Landau and U. Vishkin, "Efficient string matching with k mismatches," *Theoretical Computer Science*, vol. 43, pp. 239–249, 1986, ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(86)90178-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304397586901787.

[3] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information* (Cambridge Series on Information and the Natural Sciences). Cambridge University Press, 2000, ISBN: 9780521635035. [Online]. Available: https://books.google.com.bd/books?id=65FqEKQOfP8C.

[4] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, *Quantum amplitude amplification and estimation*, 2002. DOI: 10.1090/conm/305/05215. [Online]. Available: http://dx.doi.org/10.1090/conm/305/05215.

[5] A. Kitaev, A. Shen, and M. Vyalyi, *Classical and Quantum Computation* (Graduate studies in mathematics). American Mathematical Society, 2002, ISBN: 9780821832295. [Online]. Available: https://books.google.com.bd/books?id=08vZYhafYEAC.

[6] C. Dürr, M. Heiligman, P. HOyer, and M. Mhalla, "Quantum query complexity of some graph problems," *SIAM Journal on Computing*, vol. 35, no. 6, pp. 1310–1328, Jan. 2006, ISSN: 1095-7111. DOI: 10.1137/050644719. [Online]. Available: http://dx.doi.org/10.1137/050644719.

[7] S. Arora and B. Barak, *Computational Complexity: A Mordern Approach*. 2007.

[8] M. Sipser, *Introduction to the Theory fo Computation*. CENGAGE Learning, 2013.

[9] A. Nayebi and V. V. Williams, *Quantum algorithms for shortest paths problems in structured instances*, 2014. arXiv: 1410.6220 [quant-ph].

[10] A. Abboud, A. Backurs, and V. V. Williams, *Quadratic-time hardness of lcs and other sequence similarity measures*, 2015. arXiv: 1501.07053 [cs.CC].

[11] P. Gawrychowski and P. Uznanski, "Towards Unified Approximate Pattern Matching for Hamming and L1 Distance," in *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, Eds., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 107, Dagstuhl, Germany: Schloss

Dagstuhl – Leibniz-Zentrum für Informatik, 2018, 62:1–62:13, ISBN: 978-3-95977-076-7. DOI: 10.4230/LIPIcs.ICALP.2018.62. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2018.62.

[12] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, pp. 1273–1278, Aug. 2019, ISSN: 1745-2481. DOI: 10.1038/s41567-019-0648-8. [Online]. Available: http://dx.doi.org/10.1038/s41567-019-0648-8.

[13] S. Akmal and C. Jin, *Near-optimal quantum algorithms for string problems*, 2021. arXiv: 2110.09696 [cs.DS].

[14] P. Niroula and Y. Nam, "A quantum algorithm for string matching," *npj Quantum Information*, vol. 7, 37, p. 37, Jan. 2021. DOI: 10.1038/s41534-021-00369-3.

[15] H. Buhrman, B. Loff, S. Patro, and F. Speelman, "Limits of quantum speed-ups for computational geometry and other problems: Fine-grained complexity via quantum walks," en, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. DOI: 10.4230/LIPICS.ITCS.2022.31. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2022.31.

[16] C. Jin and J. Nogler, *Quantum speed-ups for string synchronizing sets, longest common substring, and k-mismatch matching*, 2022. arXiv: 2211.15945 [cs.DS].

[17] F. Le Gall and S. Seddighin, "Quantum Meets Fine-Grained Complexity: Sublinear Time Quantum Algorithms for String Problems," in *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, M. Braverman, Ed., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 215, Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 97:1–97:23, ISBN: 978-3-95977-217-4. DOI: 10.4230/LIPIcs.ITCS.2022.97. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2022.97.

[18] T. Kociumaka, J. Nogler, Philip, and Wellnitz, *Near-optimal-time quantum algorithms for approximate pattern matching*, 2024. eprint: 2410.06808 (cs.DS).

[19] K. Bringmann, M. Künnemann, and P. Wellnitz, "Few matches or almost periodicity: Faster pattern matching with mismatches in compressed texts," in *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1126–1145. DOI: 10.1137/1.9781611975482.69. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611975482.69. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611975482.69.

[20] R. Clifford, A. Fontaine, E. Porat, B. Sach, and T. Starikovskaya, "The <italic>k</italic>-mismatch problem revisited," in *Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2039–2052. DOI: 10.1137/1.9781611974331.ch142. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611974331.ch142. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611974331.ch142.