

Quantum algorithm for learning secret strings and its experimental demonstration

Yongzhen Xu*, Shihao Zhang*[†] and Lvzhou Li [†]

Institute of Quantum Computing and Computer Theory,

School of Computer Science and Engineering,

Sun Yat-sen University, Guangzhou 510006, China

June 23, 2022

Abstract

In this paper, we consider the secret-string-learning problem in the teacher-student setting: the teacher has a secret string $s \in \{0,1\}^n$, and the student wants to learn the secret s by question-answer interactions with the teacher, where at each time, the student can ask the teacher with a pair $(x, q) \in \{0,1\}^n \times \{0,1,\dots,n-1\}$ and the teacher returns a bit given by the oracle $f_s(x, q)$ that indicates whether the length of the longest common prefix of s and x is greater than q or not. Our contributions are as follows.

- (i) We prove that any classical deterministic algorithm needs at least n queries to the oracle f_s to learn the n -bit secret string s in both the worst case and the average case, and also present an optimal classical deterministic algorithm learning any s using n queries.
- (ii) We obtain a quantum algorithm learning the n -bit secret string s with certainty using $\lceil n/2 \rceil$ queries to the oracle f_s , thus proving a double speedup over classical counterparts.
- (iii) Experimental demonstrations of our quantum algorithm on the IBM cloud quantum computer are presented, with average success probabilities of 85.3% and 82.5% for all cases with $n = 2$ and $n = 3$, respectively.

1 Introduction

Strings are one of the most basic structures in mathematics and computer science. A string is an abstract data structure consisting of a sequence of zero or more letters over a non-empty finite alphabet. The study of string processing methods is a fundamental concern in computer science, which turns out to have a wide range of applications in areas such as information theory, artificial intelligence, computational biology and linguistics. String problems have been extensively studied in classical computing for decades, and numerous effective algorithms to deal with them have been

*These authors contributed equally to this work.

[†]zhangshh63@mail.sysu.edu.cn (S. Zhang), lilvzh@mail.sysu.edu.cn (L. Li)

proposed, including exact string matching [1, 2], finding patterns in a string [3, 4], and many others [5].

Naturally, string problems have also attracted much attention from the quantum computing community over the past two decades. In 2003, a quantum algorithm for exact string matching by Ramesh and Vinay provided a near-quadratic speedup over the fastest known classical algorithms [6]. In 2017 Montarano developed a quantum algorithm that is super-polynomially faster than the best possible classical ones for d -dimensional pattern matching problem on average-case inputs [7]. In 2021, Niroula and Nam designed a quantum pattern matching algorithm from the perspective of circuit implementation [8]. Recently, some other string problems have also been investigated by focusing on novel quantum algorithms [9, 10, 11].

As one of the most important string problems, string learning has attracted much interest in both the classical and quantum computing models since it has interesting applications in data mining and cyber security. Typically, there are two parties called a teacher and a student. The teacher has a secret bit string s of length n and the student wants to identify this secret string by asking a certain number of queries to an oracle that answers some piece of information of s . The goal is to learn s using as few queries as possible. To date, there have been many wonderful results in quantum and classical settings for this problem with different oracle types, including the index oracle [12], inner product oracle [13], substring oracle [14, 15], balance oracle in counterfeit coin problem [16], and subset OR oracle in combinatorial group testing [17, 18, 19].

In this paper, we explore the power of another oracle as the length of the Longest Common Prefix (LCP) of two strings, which is a well-known string similarity metric used in data structures and algorithms [20, 21]. Afshani et al. [22] used this oracle to consider the hidden permutation problem that comes from Mastermind game and evolutionary computation. To the best of our knowledge, there has not been any work related to string learning based on this oracle in the quantum setting. Hence, in this paper we consider quantum algorithms based on the LCP information to learn a secret string. More specifically, the problem is described in the teacher-student setting: the teacher has a secret bit string $s \in \{0, 1\}^n$, and the student wants to learn the secret s using as few queries to the teacher as possible, where at each time, the student can ask the teacher with a pair $(x, q) \in \{0, 1\}^n \times \{0, 1, \dots, n-1\}$ and the teacher returns a bit given by the oracle $f_s(x, q)$ that indicates whether the length of the LCP of s and x is greater than the integer q or not. We manage to present a quantum algorithm that can offer an advantage over the best classical algorithm for solving this problem.

The main results of this paper are stated as follows. First, we prove that any classical deterministic algorithm needs at least n queries to f_s to learn the n -bit secret string s in both the worst case and the average case, and also present a classical deterministic algorithm learning any s with exact n queries as an optimal one. Second, we propose a quantum algorithm that learns the n -bit secret string s with certainty using $\lceil n/2 \rceil$ queries to the oracle f_s , thus proving a double speedup over classical counterparts. Third, we demonstrate our quantum algorithm on the IBM cloud quantum computer using quantum circuit synthesis, compilation and optimization techniques. In the

experiment, the average success probabilities for all cases with $n = 2$ and $n = 3$ achieve 85.3% and 82.5%, respectively, which show the feasibility of implementing our quantum algorithm. Finally, we conclude this paper and put forward an interesting open problem that is worthy of further study.

2 Results

2.1 The problem: learning a secret string

We can describe the problem (learning a secret string) in the teacher-student setting: the teacher has a n -bit secret string $s \in \{0, 1\}^n$, and the student wants to learn the secret s by question-answer interactions with the teacher. In this paper, we suppose that at each time, the student can ask the teacher with a pair $(x, q) \in \{0, 1\}^n \times \{0, 1, \dots, n-1\}$, and the teacher returns a bit given by the oracle

$$f_s(x, q) := \begin{cases} 0, & lcp(s, x) \leq q; \\ 1, & lcp(s, x) > q, \end{cases} \quad (1)$$

where

$$lcp(s, x) := \max\{i \in \{0, 1, \dots, n\} \mid \forall j \leq i : x_j = s_j\} \quad (2)$$

represents the length of the Longest Common Prefix (LCP) of s and x .

When designing algorithms for solving this problem, we hope to query the oracle f_s as little as possible.

2.2 Optimal classical algorithm

In this section, we propose a classical deterministic algorithm for the secret string learning problem defined above, and further prove its optimality in terms of both the worst-case and the average-case query complexity.

Algorithm 1: Classical algorithm for learning secret string $s \in \{0, 1\}^n$

Input: A query oracle f_s defined in Eq. (1).

Output: The secret string s .

```

1  $x \leftarrow 0^n$ .
2 for  $q = 0$  to  $n - 1$  do
3   if  $f_s(x, q) = 0$  then
4      $x_{q+1} \leftarrow x_{q+1} \oplus 1$ ;
5   end
6 end
7 return  $s \leftarrow x$ ;
```

Theorem 1. (1) There is a classical deterministic algorithm learning any n -bit secret string s using n queries to the oracle f_s ; (2) Any classical deterministic algorithm needs at least n queries to learn the secret string s in both the worst case and the average case.

Proof. For solving the problem, we propose a classical deterministic algorithm named Algorithm 1, which starts from querying $(x = 0^n, q = 0)$ and then assigns the query data in each step depending on the results from previous queries by Eq.(1). In this way, each time one bit of the secret string s is identified using one query according to the previous query outcomes, and as a result the total query complexity is exactly n for any secret s . More intuitively, the whole process of Algorithm 1 can be described as a binary decision tree with height n indicating its query complexity, and we present the case with $n = 3$ in Figure 1 for illustration.

Now we prove the lower bound for the query complexity of this problem (i.e. in the worst case) by information-theoretic argument. In general, any classical deterministic algorithm for this problem can be described in terms of a binary decision tree that queries different (x, q) in each internal node and identifies secret s in each leaf node (i.e. external node), with its height being the query complexity of the algorithm. Considering the fact that any binary tree with height h has at most 2^h leaf nodes, we conclude that the height of any binary tree with total 2^n leaf nodes is at least n . That is, any classical deterministic algorithm for the secret learning problem needs at least n queries in the worst case.

Moreover, we can prove the average-case query complexity of any classical deterministic algorithm is no less than n . Note that the path length from the root node to a leaf node in a binary tree indicates the number of queries to identify a secret string s in an algorithm, our proof can be derived from a fact about binary trees revealed as follows. The external path length (EPL) of a tree is defined as the sum of path lengths of all its leaf nodes, and the minimum value of the EPL of a binary decision tree with N leaves is known as $N(\log_2 N + 1 + \gamma - 2^\gamma)$ with $\gamma = \lceil \log_2 N \rceil - \log_2 N \in [0, 1)$ [23]. Hence, the minimum average path length of all N leaf nodes is $\log_2 N + 1 + \gamma - 2^\gamma$. Here in our case we consider binary trees corresponding to classical query algorithms with $N = 2^n$ and $\gamma = 0$, where the minimum EPL is $n2^n$ and the minimum average-case query complexity is n .

In summary, our classical Algorithm 1 achieves optimality in terms of both worst- and average-case query complexity. $\square$

2.3 Quantum algorithm with speedup

In this section, we present a quantum algorithm that shows a speedup over any classical deterministic algorithm for learning the secret string s .

The quantum oracle associated with $f_s(x, q)$ in Eq. (1) is a quantum unitary operator, O_s , defined by its action on the computational basis:

$$|x, q\rangle |y\rangle \xrightarrow{O_s} |x, q\rangle |y \oplus f_s(x, q)\rangle, \quad (3)$$

where $|x, q\rangle$ is the query register and $|y\rangle$ is a single oracle qubit. Therefore, when we initialize the

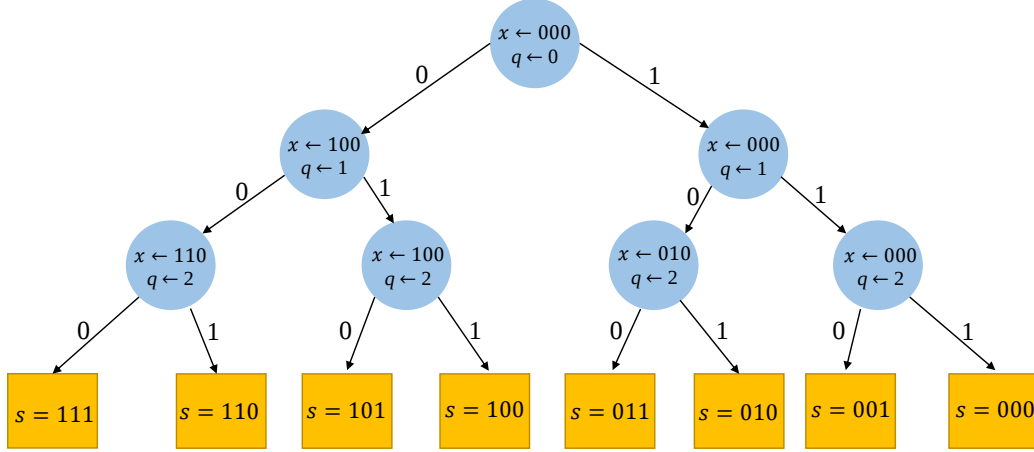


Figure 1: Binary decision tree for illustrating classical Algorithm 1 with $n = 3$. The value 0 or 1 along each edge is the outcome after querying (x, q) in blue circle nodes under a certain secret $s \in \{0, 1\}^3$ by Eq. (1), and any secret string s can be identified in an orange square node (i.e. leaf node) after three queries.

oracle qubit as $|y\rangle = |-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$, we have

$$|x, q\rangle |-\rangle \xrightarrow{O_s} \begin{cases} |x, q\rangle |-\rangle, & f_s(x, q) = 0; \\ -|x, q\rangle |-\rangle, & f_s(x, q) = 1, \end{cases} \quad (4)$$

which can be summarized as

$$|x, q\rangle |-\rangle \xrightarrow{O_s} (-1)^{f_s(x, q)} |x, q\rangle |-\rangle. \quad (5)$$

Note the state of the oracle qubit remains unchanged in this process, and thus can be omitted in the description of the action of oracle O_s for convenience as

$$|x, q\rangle \xrightarrow{O_s} (-1)^{f_s(x, q)} |x, q\rangle. \quad (6)$$

In fact, similar [phase kickback] effects [24] and associated oracle simplifications have been explored and exploited in adapting some well-known quantum query algorithms, including Deutsch-Jozsa algorithm [25, 26], Bernstein-Vazirani algorithm [27, 28], and Grover search algorithm [29, 30]. In the following, we can directly design quantum query algorithms using the quantum oracle O_s in Eq. (6) when all other involved operators only act on the query register.

Theorem 2. *There is a quantum algorithm learning the n -bit secret string s with certainty using $\lceil n/2 \rceil$ queries to the oracle f_s .*

Proof. We describe our quantum algorithm for addressing the case with even n as depicted in Figure 2, and the odd n cases can be handled in a similar way. The essence of our algorithm is to

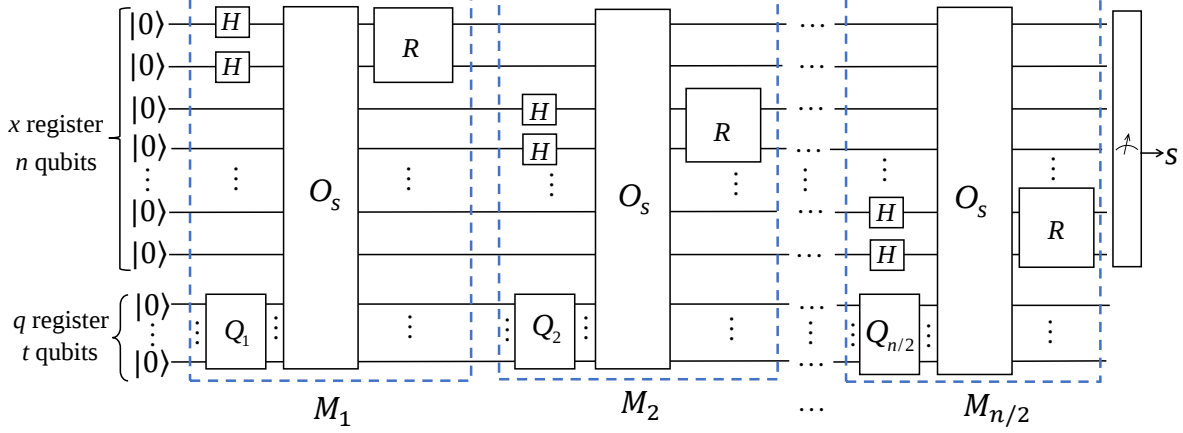


Figure 2: Schematic of the overall circuit for quantum learning algorithm with even n , which applies $n/2$ subroutines M_i ($i=1,2,\dots,n/2$) to the n -qubit x register together with the t -qubit q register initialized as $|0^{(n+t)}\rangle$. The working principle of each M_i that consists of a quantum oracle O_s in Eq. (6) and several fixed operators is illustrated in Lemma 1, and the output of the final measurement is exactly the target secret string s .

identify two bits of any n -bit secret string s at a time by applying one quantum oracle combined with other operators to a certain quantum superposition state, and thus the total query number is $n/2$ throughout the algorithm.

For even n and $t = \lceil \log_2(n) \rceil$, the input data (x, q) is encoded by a x register with n qubits combined with a q register with t qubits, and the whole quantum circuit consists of $n/2$ subroutines $\{M_i : i = 1, 2, \dots, n/2\}$, where each M_i can be broken into four steps:

- (1) Apply two Hadamard gates $H^{\otimes 2}$ to qubits $2i - 1$ and $2i$ in the x register.
- (2) Apply an operator Q_i to the q register that can transform a t -qubit state $|q^{(i-1)}\rangle$ into $|q^{(i)}\rangle$ with $q^{(0)} = 0$ and $q^{(i)} = 2i - 1$ for $i \geq 1$.
- (3) Apply the quantum oracle operator O_s described in Eq. (6) to all $(n + t)$ qubits.
- (4) Apply a 4×4 unitary operator

$$R = \frac{1}{2} \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} \quad (7)$$

to qubits $2i - 1$ and $2i$.

As a result, the application of $M_1, M_2, \dots, M_{n/2}$ on the initial input state $|0^{n+t}\rangle$ would produce the final output state $|x = s\rangle |q = n - 1\rangle$, and thus the target string s can be identified by measuring the n -qubit x register in the computational basis. To explicitly explain the working principle of the above quantum algorithm, we reveal the overall effect of each subroutine M_i in Lemma 1 in detail.

Lemma 1. Denote the integer $q^{(0)} = 0$, $q^{(i)} = 2i - 1$ ($i = 1, 2, \dots, n/2$), and $s^{(i)} = s_1 s_2 \dots s_{q^{(i)} - 1}$ as the

$(2i-2)$ -bit prefix of s ($s^{(1)} = \emptyset$). Then each subroutine M_i ($i=1,2,\dots,n/2$) consisting of $H_{2i-1}, H_{2i}, Q_i, O_s$, and R as shown in Figure 2 has the effect

$$M_i : |\psi_0\rangle = |x = s^{(i)}00^{n-2i}\rangle |q = q^{(i-1)}\rangle \rightarrow |x = s^{(i)}s_{2i-1}s_{2i}0^{n-2i}\rangle |q = q^{(i)}\rangle. \quad (8)$$

Proof. For brevity, we first illustrate some key facts related to our quantum algorithm. We denote a set of four specific n -bit strings as $T = \{x^{(k)} = s^{(i)}k0^{n-2i} : k = k_1k_2 \in \{0,1\}^2\}$, and according to the definitions of the functions in Eq. (2) and Eq. (1) we have

$$lcp(s, x^{(k)}) = \begin{cases} 2i-2, & k_1 \neq s_{2i-1}; \\ 2i-1, & k_1 = s_{2i-1}, k_2 \neq s_{2i}; \\ \geq 2i, & k_1k_2 = s_{2i-1}s_{2i} \end{cases} \quad (9)$$

by noting the $(2i-2)$ -bit prefix of each $x^{(k)}$ is $s^{(i)} = s_1s_2\dots s_{2i-2}$ and therefore

$$f_s(x^{(k)}, q^{(i)}) = 2i-1 = \begin{cases} 0, & k \neq s_{2i-1}s_{2i} \text{ (three different such } k); \\ 1, & k = s_{2i-1}s_{2i} \text{ (only one such } k) \end{cases} \quad (10)$$

for four $x^{(k)}$ in T . Besides, it can be directly verified that for any $k_0 \in \{0,1\}^2$, the operator R in Eq. (7) can transform a 2-qubit superposed state into the basis state $|k_0\rangle$ as

$$R : \sum_{k \in \{0,1\}^2} \alpha_k |k\rangle \rightarrow |k_0\rangle \quad (11)$$

when the coefficients are

$$\alpha_k = \begin{cases} \frac{1}{2}, & k \neq k_0; \\ -\frac{1}{2}, & k = k_0. \end{cases} \quad (12)$$

Based on these facts and notations, the overall effect of M_i in Eq. (8) is realized by following steps:

(1) Two Hadamard gates $H_{2i-1}H_{2i}$ and the operator Q_i together transform the state $|\psi_0\rangle$ on the left hand side of Eq. (8) into

$$|\psi_1\rangle = \frac{1}{2} \sum_{k \in \{0,1\}^2} |s^{(i)}k0^{n-2i}\rangle |q^{(i)}\rangle = \frac{1}{2} \sum_{k \in \{0,1\}^2} |x^{(k)}\rangle |q^{(i)}\rangle. \quad (13)$$

(2) By utilizing Eq. (10), it can be derived the quantum oracle operator O_s defined in Eq. (6) can transform the state $|\psi_1\rangle$ of Eq. (13) into

$$|\psi_2\rangle = \sum_{k \in \{0,1\}^2} \alpha_k |x^{(k)}\rangle |q^{(i)}\rangle = |s^{(i)}\rangle \left(\sum_{k \in \{0,1\}^2} \alpha_k |k\rangle \right) |0^{n-2i}\rangle |q^{(i)}\rangle \quad (14)$$

with the coefficients

$$\alpha_k = \begin{cases} \frac{1}{2}, & k \neq s_{2i-1}s_{2i}; \\ -\frac{1}{2}, & k = s_{2i-1}s_{2i}. \end{cases} \quad (15)$$

(3) In the end, the operator R in Eq. (7) acting on qubits $2i - 1$ and $2i$ can transform $|\psi_2\rangle$ of Eq. (14) into

$$|\psi_3\rangle = |s^{(i)}\rangle |s_{2i-1}s_{2i}\rangle |0^{n-2i}\rangle |q^{(i)}\rangle \quad (16)$$

by using Eq. (11) and Eq. (12), which thus proves Eq. (8) of Lemma 1. $\square$

According to Lemma 1, the subroutines $\{M_i : i = 1, 2, \dots, n/2\}$ in Figure 2 transforms the initial input state as:

$$\begin{aligned} |x = 0^n\rangle |q = 0\rangle &\xrightarrow{M_1} |x = s_1 s_2 0^{n-2}\rangle |q = 1\rangle \\ &\xrightarrow{M_2} |x = s_1 s_2 s_3 s_4 0^{n-4}\rangle |q = 3\rangle \\ &\xrightarrow{M_3} \dots \xrightarrow{M_{n/2}} |x = s\rangle |q = n - 1\rangle, \end{aligned}$$

and thus the secret string s can be identified by measuring the x register of the output state. Therefore, the whole quantum algorithm totally employs $n/2$ quantum oracle queries for identifying a secret string s , which outperforms classical deterministic algorithms that use at least n queries in both the worst and average cases (see Theorem 1). This double speedup of our quantum algorithm comes from the intrinsic quantum parallelism, that is, the ability to evaluate four function values $\{f_s(x^{(k)}, q^{(i)}) : k \in \{0, 1\}^2\}$ by each quantum oracle query O_s (i.e. from Eq. (13) to Eq. (14)) and then to extract two bits of desired information $s_{2i-1}s_{2i}$ by interference via the operator R (i.e. from Eq. (14) to Eq. (16)) in each subroutine M_i .

Finally, we address the case with odd n . The idea is to first obtain the $(n - 1)$ -bit prefix of the secret s by $(n - 1)/2$ quantum queries following the idea in Lemma 1 and then identify the last bit of s by a classical query. More precisely, a sequence of $(n - 1)/2$ subroutines $\{M_i : i = 1, 2, \dots, (n - 1)/2\}$ constructed in the same way as those in Figure 2 are applied to an input state $|0^{n+t}\rangle$ with $t = \lceil \log_2(n - 1) \rceil$, and then the x register of the output state is measured to obtain a string $x \in \{0, 1\}^n$ revealing $s_1 s_2 \dots s_{n-1} = x_1 x_2 \dots x_{n-1}$. Next, one classical query on $(x, q = n - 1)$ is performed to identify the last bit of s as

$$s_n = \begin{cases} x_n \oplus 1, & f_s(x, q = n - 1) = 0; \\ x_n, & f_s(x, q = n - 1) = 1 \end{cases} \quad (17)$$

according to Eq. (1). Therefore, we employ $(n - 1)/2$ quantum queries and 1 classical query together to identify s for odd n . Note that there are totally $\lceil n/2 \rceil$ queries. Thus, we have completed the proof of Theorem 2. $\square$

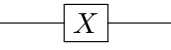
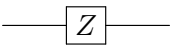
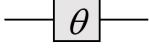
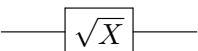
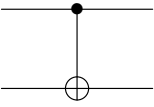
In the next section, we demonstrate our quantum algorithm on an IBM cloud quantum computer

for several problem instances and evaluate the experimental performances.

2.4 Experimental demonstrations on the IBM quantum computer

For demonstrating our quantum algorithm in the commonly used circuit model of quantum computation, we would employ a simple set of quantum gates consisting of the single-qubit Pauli X and Z gates, Hadamard gate H , Z -axis rotation gate $R_Z(\theta)$, square-root of X gate $\sqrt{X}$, together with the two-qubit controlled-NOT (CNOT) gate. Unitary matrices and circuit symbols for these gates are listed in Table 1.

Table 1: Unitary matrices and circuit symbols for quantum gates used in this paper.

Quantum Gate	Unitary Matrix	Circuit Symbol
Pauli- X	$X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	
Pauli- Z	$Z := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	
Hadamard	$H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	
Z -axis rotation	$R_Z(\theta) := \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$	
square-root of X	$\sqrt{X} := \frac{1}{2} \begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}$	
controlled-NOT	$CNOT := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	

In order to experimentally demonstrate our quantum algorithm on real quantum hardware, e.g., IBM cloud superconducting quantum computers [31], a series of key issues need to be taken into account:

(1) First, the unitary operators Q_i , O_s , and R shown in Figure 2 are supposed to be explicitly constructed using basic gates, which is called quantum circuit synthesis. In particular, Q_i can be built from a series of Pauli X gates for converting a basis state $|q^{(i-1)}\rangle$ into $|q^{(i)}\rangle$, the quantum oracle O_s defined in Eq. (6) can be expressed as

$$O_s = \sum_{(x,q) \in \{0,1\}^{n+t}} (-1)^{f_s(x,q)} |x, q\rangle \langle x, q|, \quad (18)$$

which is actually a diagonal unitary matrix with diagonal elements ± 1 and can be realized by $\{\text{CNOT}, R_Z(\theta)\}$ gate set [32, 33], and the operator R in Eq. (7) can be decomposed into $R = (H \otimes I)(Z \otimes X)\text{CNOT}(H \otimes I)$ as shown in Fig. 3(b).

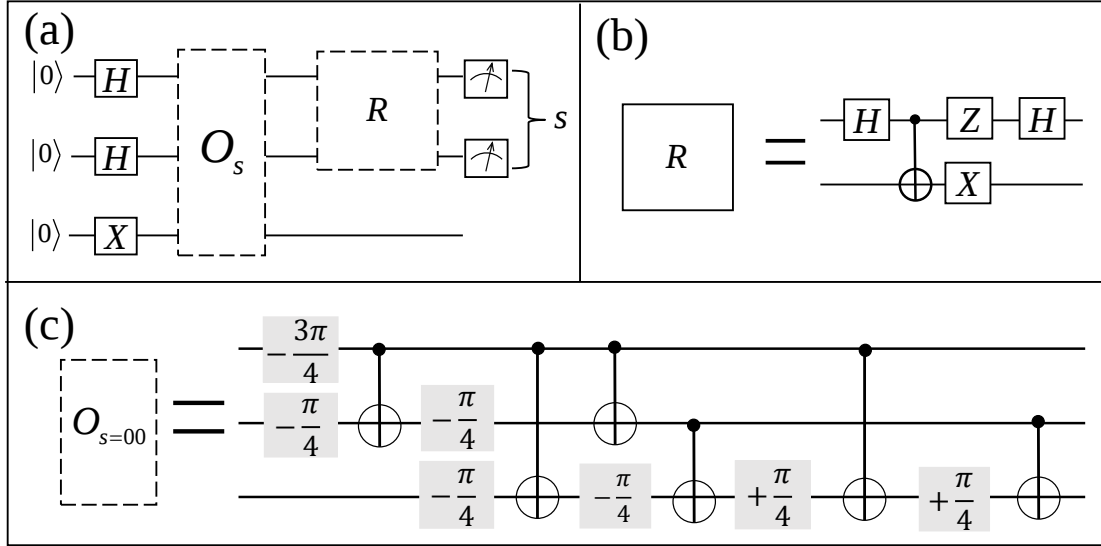


Figure 3: (a) Schematic of the circuit to demonstrate our quantum algorithm for cases with $n = 2$, with the fixed operator R of Eq. (7) shown in (b) and the implementation of an oracle operator example $O_{s=00} = \text{diag}(-1,-1,-1,1,1,1,1,1)$ shown in (c). The details of the employed quantum gates are listed in Table. 1.

(2) Second, considering both available physical gates and restricted qubit connectivity of employed quantum hardware, any synthesized circuit needs to be transpiled into a feasible circuit that can be executed in the practical hardware platform with acceptable outcomes. This essential processing procedure, including gate conversion and qubit mapping strategies, is usually called quantum circuit compilation [34, 35] and has attracted a great deal of attention in the NISQ era [36, 37, 38, 39, 40, 41]. Here we implement customized circuit compilation procedures aimed at running each synthesized circuit on the hardware named *ibmq_quito* with its topology structure shown in Fig. 4(a) and available gate set $G_{ibmq} = \{\text{CNOT}, R_Z(\theta), \sqrt{X}, X\}$, including a Hadamard gate decomposition $H \simeq R_Z(\pi/2)\sqrt{X}R_Z(\pi/2)$ (differing only by an unimportant global phase factor) and a CNOT identity as shown in Fig. 4(b) for implementing a CNOT between two unconnected qubits via four usable CNOT gates.

(3) Finally, we consider to take specific optimization techniques as shown in Fig. 4(c) for further

reducing the gate counts as well as circuit depth of compiled circuits, including:

$$\begin{aligned}
R_Z \text{ gate merging : } R_Z(\theta_2)R_Z(\theta_1) &= R_Z(\theta_1 + \theta_2), \\
\text{CNOT cancellation : } \text{CNOT} \cdot \text{CNOT} &= I, \\
\text{Commutation rules : } \text{CNOT}(R_Z(\theta) \otimes I) &= (R_Z(\theta) \otimes I)\text{CNOT}, \\
&\text{CNOT}(I \otimes R_Z(\theta_2))\text{CNOT}(I \otimes R_Z(\theta_1)) \\
&= (I \otimes R_Z(\theta_1))\text{CNOT}(I \otimes R_Z(\theta_2))\text{CNOT}.
\end{aligned} \tag{19}$$

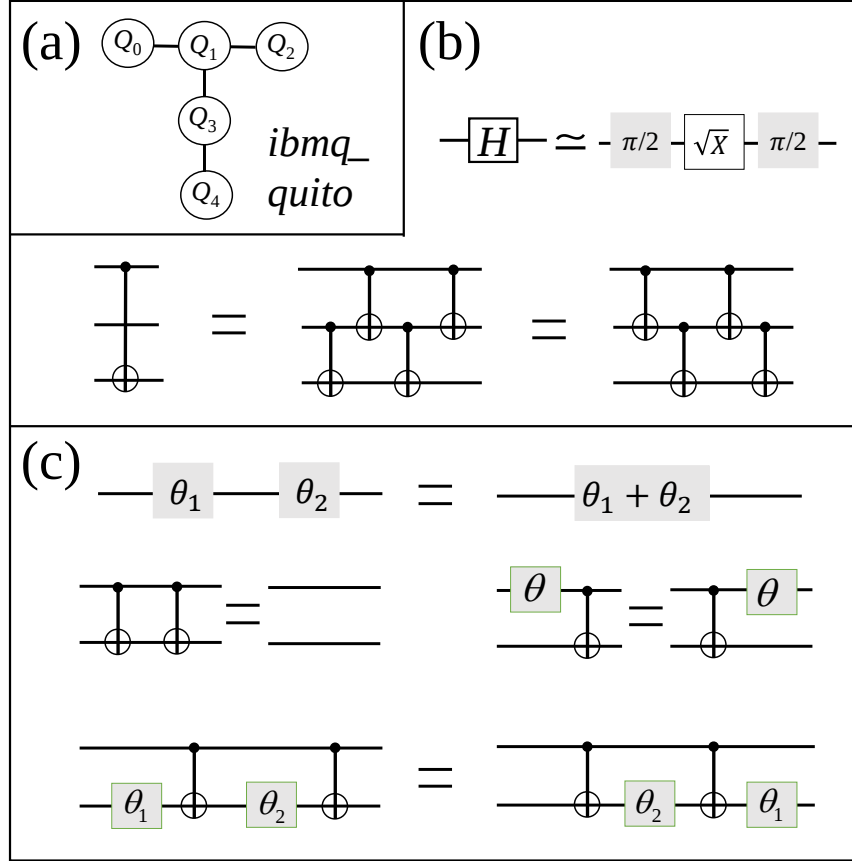


Figure 4: (a) Topology structure of the IBM quantum hardware *ibmq_quito*. (b) Gate transformations used for quantum circuit compilation. (c) Some useful quantum circuit optimization techniques in Eq. (19).

Based on the above process, we can bridge the gap between our quantum algorithm described at an abstract level and its practical implementation on real quantum hardware for any problem size in principle. In the following, we demonstrate the even case with $n = 2$ and the odd case with $n = 3$ on the IBM quantum hardware *ibmq_quito*.

For $n = 2$, we use a three-qubit circuit with $n = 2$ and $t = 1$ for performing our quantum

algorithm as shown in Fig. 3(a), where the operator Q_1 acting on the q register is a X gate and each problem instance s determines a specific oracle operator O_s in Eq. (18). For example, in Fig. 3(c) we present a construction of an oracle example $O_{s=00} = \text{diag}(-1, -1, -1, 1, 1, 1, 1, 1)$ using six CNOT gates and seven $R_Z(\theta)$ gates. Next, we compile this circuit into a new one that fits the $Q_0 - Q_1 - Q_2$ linear structure and gate set G_{ibmq_quito} of *ibmq_quito* using the equivalent gate transformations in Fig. 4(b) followed by the application of optimization techniques in Fig. 4(c), which would lead to a final transpiled circuit consisting of 9 CNOT, 10 $R_Z(\theta)$, 4 $\sqrt{X}$, and 2 X gates with a circuit depth 15 as shown in Fig. 5(a). Besides, the oracle constructions and transpiled circuits for other three instances $s = 01, 10$, and 11 have similar forms as provided in Supplementary Information A.

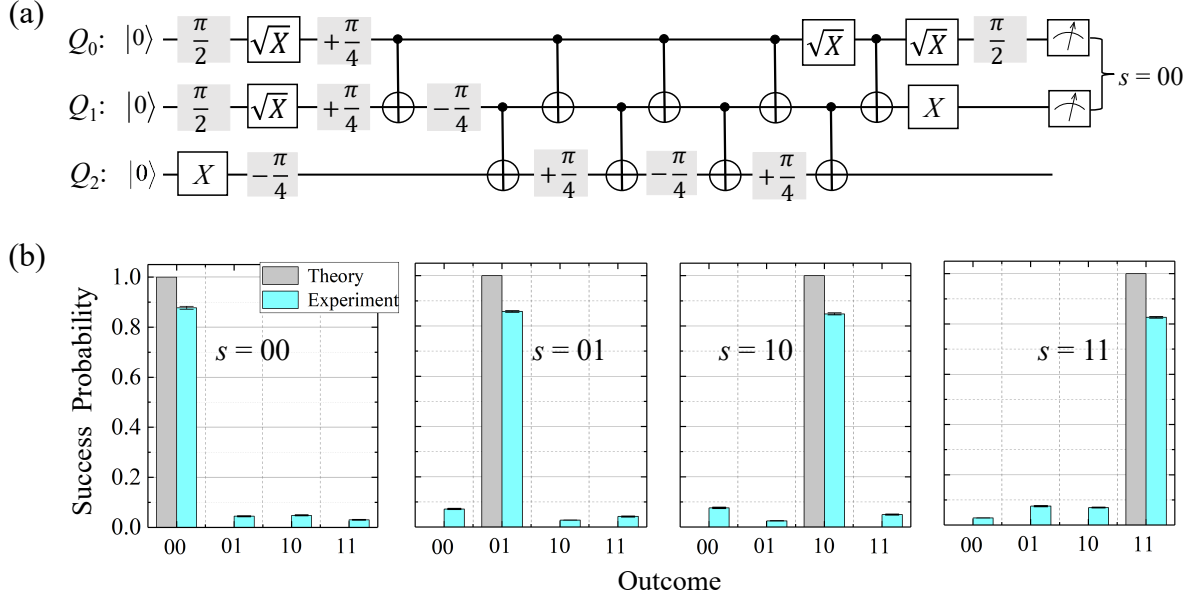


Figure 5: (a) Transpiled circuit acting on three physical qubits $\{Q_0, Q_1, Q_2\}$ of *ibmq_quito* for identifying secret $s = 00$ as an example, which is obtained from the original circuit in Fig. 3 through compilation and optimization procedures in Fig. 4. (b) Experimental results for demonstrating all instances $s \in \{0, 1\}^2$ via their associated transpiled circuits. The success probabilities are 87.7(6)%, 85.8(3)%, 84.8(4)%, and 82.6(3)% for $s = 00, 01, 10$, and 11 , respectively, with 5 trials and the sample size 8192 (5×8192 realizations for each s).

All four instances for $n = 2$ are tested in our experiment, where we employ a figure of merit called the algorithm success probability (ASP) to denote the probability of correctly identifying the target string as the experiment outcome. For demonstrating an instance $s \in \{0, 1\}^2$ on the device, we experimentally implement 5 trials and in each trial we consider 8192 repetitions of the transpiled circuit to record outcome data for calculating the ASP. At the time the device *ibmq_quito* was accessed, the average CNOT error was 1.080×10^{-2} and average readout error was 4.424×10^{-2} , average $T_1 = 98.55\mu s$ (decay time from the excited state to ground state), average $T_2 = 101.74\mu s$ (decoherence time). As the distributions over possible measurement outcomes show in Fig. 5(b),

the success probabilities for identifying secret $s = 00, 01, 10$, and 11 in the experiment are 87.7(6)%, 85.8(3)%, 84.8(4)%, and 82.6(3)%, respectively. As an overall evaluation, the average success probability defined over all problem instances is 85.3% in the above experimental demonstration.

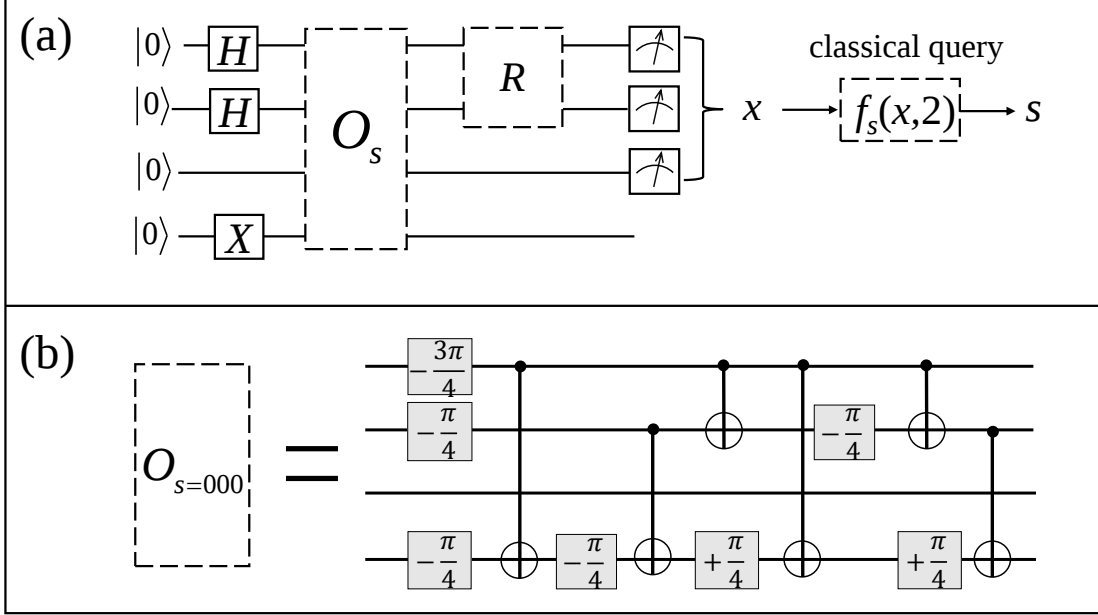


Figure 6: (a) Schematic of the quantum circuit followed by a classical query to demonstrate our quantum algorithm for cases with $n = 3$, with the fixed operator R shown in Fig. 3(b) and implementation of an oracle operator example $O_{s=000} = \text{diag}(-1, -1, -1, -1, -1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1)$ shown in (b).

For $n = 3$, we use a four-qubit circuit with $n = 3$ and $t = 1$ followed by a classical query for performing our quantum algorithm as shown in Fig. 6(a). For example, in Fig. 6(b) we present a construction of an oracle example $O_{s=000} = \text{diag}(-1, -1, -1, -1, -1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1)$ using six CNOT gates and seven $R_Z(\theta)$ gates. Next, we compile this circuit by assigning four qubits of Fig. 6(a) to physical qubits $\{Q_0, Q_1, Q_2, Q_3\}$ in the T-shape structure of *ibmq_quito* in Fig. 4(a) and then using circuit transformation techniques in Figs. 4(b) and 4(c), which would lead to a final transpiled circuit consisting of 9 CNOT, 10 $R_Z(\theta)$, 4 $\sqrt{X}$, and 2 X gates with a circuit depth 15 as shown in Fig. 7(a). Also, the oracle constructions and transpiled circuits for other instances s have similar forms as provided in Supplementary Information B. For completeness, we experimentally perform our algorithm by taking 5×8192 repetitions for each one of the eight instances $s \in \{0, 1\}^3$, and the success probabilities for identifying each instance are 85.3(7)%, 85.3(4)%, 83.0(9)%, 82.2(6)%, 82.3(5)%, 82.4(1)%, 79.7(4)% and 79.5(5)% as recorded in Fig. 7(b), respectively. As an overall evaluation, the average success probability defined over all problem instances is 82.5% in our experiment.

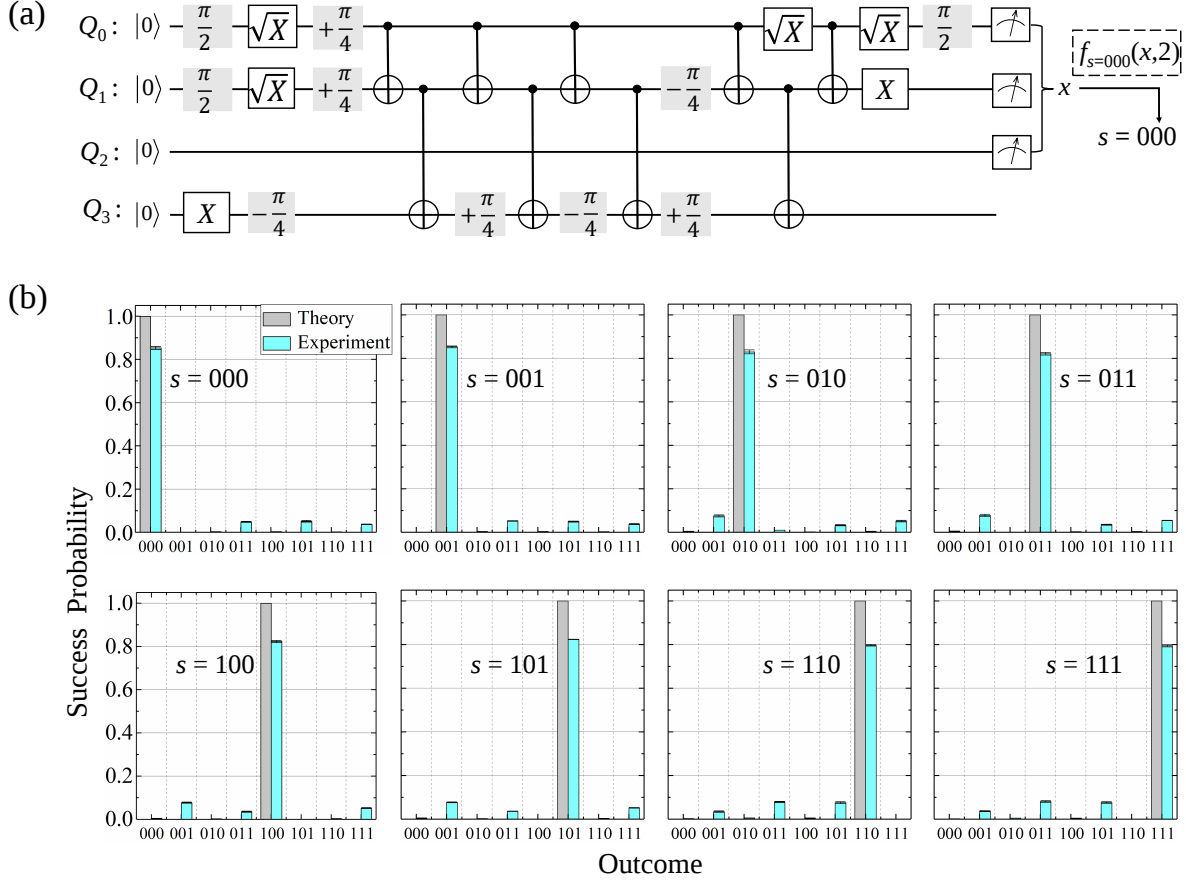


Figure 7: (a) Transpiled circuit acting on four physical qubits $\{Q_0, Q_1, Q_2, Q_3\}$ of *ibmq_quito* followed by a classical query $f_{s=000}(x, 2)$ for identifying secret $s = 000$ as an example, which is obtained from the original circuit in Fig. 6 through compilation and optimization procedures in Fig. 4. (b) Experimental results for demonstrating all eight instances $s \in \{0, 1\}^3$ via their associated transpiled circuits together with a classical query, where the corresponding success probabilities are 85.3(7)%, 85.3(4)%, 83.0(9)%, 82.2(6)%, 82.3(5)%, 82.4(1)%, 79.7(4)% and 79.5(5)%, respectively, with 5 trials and the sample size 8192 for each s .

3 Discussion

In this study, we discussed how to learn a secret string $s \in \{0, 1\}^n$ by querying the oracle $f_s(x, q)$ that indicates whether the length of the longest common prefix of s and x is greater than q or not. We first proved that the classical query complexity of this problem is n in both the worst and average case as well as gave an optimal classical deterministic algorithm. Then, we proposed an exact quantum algorithm with $\lceil n/2 \rceil$ query complexity for solving any problem instance, which thus provided a double speedup over its classical counterparts. Finally, we experimentally demonstrated our quantum algorithm on an IBM cloud quantum device by utilizing specific circuit design and compilation techniques, where the average success probability for all instances with $n = 2$ or $n = 3$

reached 85.3% or 82.5%, respectively. Also, experiments on our quantum algorithm for larger-scale secret string problems are likely to be performed by following a similar way. Besides, the q register used in our quantum algorithm can be alternatively represented by higher-dimensional qudit ($d > 2$) systems for experimental demonstrations, e.g., using quantum photonic [42] or nuclear magnetic resonance (NMR) systems [43]. Since it is not yet clear whether our quantum algorithm is optimal, in future work we would consider an interesting open problem about the lower bound of exact or bounded-error quantum query complexity for this secret-string-learning problem.

Data Availability

All the data supporting the results of this work are available from the authors upon reasonable request.

Code Availability

The codes for designing circuits in this work are available from the authors upon reasonable request.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62102464, 61772565), the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2020B1515020050), the Key Research and Development project of Guangdong Province (Grant No. 2018B030325001), and Project funded by China Postdoctoral Science Foundation (Grant Nos. 2020M683049, 2021T140761).

Author Contributions

L. Li conceived this problem. Y. Xu and S. Zhang designed the quantum algorithm together. Y. Xu put forward the classical algorithm and proved its optimality, S. Zhang performed the quantum experiments and data processing. L. Li and S. Zhang supervised the project, and all authors wrote the manuscript. Y. Xu and S. Zhang contributed equally to this work.

Competing Interests

The authors declare no competing interests.

References

- [1] Knuth, D. E., Morris, Jr, J. H. & Pratt, V. R. Fast Pattern Matching in Strings. *SIAM J. Comput.* **6**, 323–350 (1977).

- [2] Boyer, R. S. & Moore, J. S. A Fast String Searching Algorithm. *Commun. ACM* **20**(10), 762–772 (1977).
- [3] Manacher, G. A New Linear-Time “On-Line” Algorithm for Finding the Smallest Initial Palindrome of a String. *J. ACM* **22**(3), 346–351 (1975).
- [4] Apostolico, A., Breslauer, D. & Galil, Z. Parallel Detection of all Palindromes in a String. *Theor. Comput. Sci.* **141**(1&2), 163–173 (1995).
- [5] Crochemore, M., Hancart, C. & Lecroq, T. Algorithms on strings. Cambridge University Press, 2007.
- [6] Ramesh, H. & Vinay, V. String matching in $\tilde{O}(\sqrt{n} + \sqrt{m})$ quantum time. *J. Discrete Algorithms* **1**, 103–110 (2003).
- [7] Montanaro, A. Quantum Pattern Matching Fast on Average. *Algorithmica* **77**, 16–39 (2017).
- [8] Niroula, P. & Nam, Y. A quantum algorithm for string matching. *npj Quantum Inf* **7**, 37 (2021).
- [9] Boroujeni, M., Ehsani, S., Ghodsi, M., HajiAghayi, M. & Seddighin, S. Approximating Edit Distance in Truly Subquadratic Time: Quantum and MapReduce. *J. ACM* **68**(3), 19:1–19:41 (2021).
- [10] Le Gall, F. & Seddighin, S. Quantum Meets Fine-Grained Complexity: Sublinear Time Quantum Algorithms for String Problems. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference*, 97:1–97:23 (2022).
- [11] Akmal, S.S. & Jin, C. Near-Optimal Quantum Algorithms for String Problems. In *Proceedings of the 33rd ACM-SIAM Symposium on Discrete Algorithms*, 2791–2832 (2022).
- [12] Van Dam, W. Quantum Oracle Interrogation: Getting All Information for Almost Half the Price. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, 362–367 (1998).
- [13] Bernstein, E. & Vazirani, U. Quantum complexity theory. *SIAM J. Comput.* **26**, 1411–1473 (1997).
- [14] Skiena, S. S. & Sundaram, G. Reconstructing strings from substrings. *J. Comput. Biol.* **2**, 333–353 (1995).
- [15] Cleve, R., Iwama, K., Gall, F. L., Nishimura, H., Tani, S., Teruyama, J. & Yamashita, S. Reconstructing Strings from Substrings with Quantum Queries. In *Proceedings of the 13th Scandinavian Symposium and Workshops on Algorithm Theory*, 388–397 (2012).
- [16] Iwama, K., Nishimura, H., Raymond, R. & Teruyama, J. Quantum counterfeit coin problems. *Theor. Comput. Sci.* **456**, 51–64 (2012).

- [17] Du, D., Hwang, F. K. & Hwang, F. Combinatorial group testing and its applications. volume 12. World Scientific, 2000.
- [18] Ambainis, A. & Montanaro, A. Quantum algorithms for search with wildcards and combinatorial group testing. *Quantum Inf. Comput.* **14**(5-6), 439–453 (2014).
- [19] Belovs, A. Quantum Algorithms for Learning Symmetric Juntas via the Adversary Bound. *Comput. Complex.* **124**, 255–293 (2015).
- [20] Kasai, T., Lee, G., Arimura, H., Arikawa, S. & Park, K. Linear-time longest-common-prefix computation in suffix arrays and its applications. In *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching*, 181–192 (2001).
- [21] Bonizzoni, P., De Felice, C., Zaccagnino, R. & Zizza, R., On the longest common prefix of suffixes in an inverse Lyndon factorization and other properties. *Theor. Comput. Sci.* **862**, 24–41 (2021).
- [22] Afshani, P., Agrawal, M., Doerr, B., Doerr, C., Larsen, K. G. & Mehlhorn, K. The query complexity of a permutation-based variant of Mastermind. *Discret. Appl. Math.* **260**, 28–50 (2019).
- [23] Knuth, D. E. The art of computer programming, volume 3: (2nd edn.) sorting and searching. Addison Wesley, see pages 192–193, 1998.
- [24] Cleve, R., Ekert, A., Macchiavello, C. & Mosca, M. Quantum algorithms revisited. *Proc. R. Soc. Lond. A.* **439**, 553–558 (1998).
- [25] Deutsch, D. & Jozsa, R. Rapid solution of problems by quantum computation. *Proc. R. Soc. Lond. A.* **454**, 339–354 (1992).
- [26] Collins, D., Kim, K. W. & Holton, W. C. Deutsch-Jozsa algorithm as a test of quantum computation. *Phys. Rev. A* **58**, R1633 (1998).
- [27] Bernstein, E. & Vazirani, U. Quantum complexity theory. *SIAM J. Comput.* **26**(5), 1411–1473 (1997).
- [28] Du, J., Shi, M., Zhou, X., Fan, Y., Ye, B., Han, R. & Wu, J. Implementation of a quantum algorithm to solve the Bernstein-Vazirani parity problem without entanglement on an ensemble quantum computer. *Phys. Rev. A* **64**, 042306 (2001).
- [29] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, 212–219 (1996).
- [30] Figgatt, C., Maslov, D., Landsman, K. A., Linke, N. M., Debnath, S. & Monroe, C. Complete 3-qubit Grover search on a programmable quantum computer. *Nature communications* **8**, 1–9 (2017).

- [31] IBM Quantum, <https://quantum-computing.ibm.com>.
- [32] Bullock, S. S. & Markov, I. L. Asymptotically optimal circuits for arbitrary n-qubit diagonal comutations. *Quantum Inf. Comput.* **4**, 27–47 (2004).
- [33] Welch, J., Greenbaum, D., Mostame, S. & Aspuru-Guzik, A. Efficient quantum circuits for diagonal unitaries without ancillas. *New J. Phys.* **16**, 033040 (2014).
- [34] Leymann, F. & Barzen, J. The bitter truth about gate-based quantum algorithms in the NISQ era. *Quantum Sci. Technol* **5**, 044007 (2020).
- [35] Kusyk, J., Saeed, S. M., and Uyar, M. U. Survey on quantum circuit compilation for noisy intermediate-scale quantum computers: Artificial intelligence to heuristics. *IEEE Transactions on Quantum Engineering* **2**, 1–16 (2021).
- [36] Martinez, E. A., Monz, T., Nigg, D., Schindler, P. & Blatt, R. Compiling quantum algorithms for architectures with multi-qubit gates. *New J. Phys.* **18**, 063029 (2016).
- [37] Itoko, T., Raymond, R., Imamichi, T., Matsuo, A. & Cross, A. W. Quantum circuit compilers using gate commutation rules. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 191–196 (2019).
- [38] Alam, M., Ash-Saki, A. & Ghosh, S. Circuit compilation methodologies for quantum approximate optimization algorithm. In *Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture*, 215–228 (2020).
- [39] Nash, B., Gheorghiu, V. & Mosca, M. Quantum circuit optimizations for NISQ architectures. *Quantum Sci. Technol* **5**, 025010 (2020).
- [40] Itoko, T., Raymond, R., Imamichi, T. & Matsuo, A. Optimization of quantum circuit mapping using gate transformation and commutation. *Integration* **70**, 43–50 (2020).
- [41] Bandic, M., Feld, S. & Almudever, C. G. Full-stack quantum computing systems in the NISQ era: algorithm-driven and hardware-aware compilation techniques. In *2022 Design, Automation & Test in Europe Conference & Exhibition*, 1–6 (2022).
- [42] Erhard, M., Krenn, M. & Zeilinger, A. Advances in high-dimensional quantum entanglement. *Nat. Rev. Phys.* **2**, 365–381 (2020).
- [43] Gedik, Z., Silva, I., ?akmak, B. et al. Computational speed-up with a single qudit. *Sci. Rep.* **5**, 14671 (2015).

Supplementary Information

As mentioned in the main text, we experimentally demonstrate our quantum algorithm for all cases with $n = 2$ and $n = 3$ on the IBM quantum device named *ibmq_quito*, and the calibration parameters for the day the device was accessed are shown in Table S1. For completeness, in the following we give detailed descriptions of all employed quantum circuits.

Table S1: Calibration parameters of *ibmq_quito* from the IBM quantum website. In the CNOT error column, i_j indicates that the physical qubits i and j are the control and target for the CNOT gate, respectively.

Qubit	T1 (us)	T2 (us)	Freq. (GHz)	Anharm. (GHz)	Readout error	ID error	$\sqrt{x}$ (sx) error	Pauli-X error	CNOT error
Q0	79.19	126.78	5.301	-0.33148	3.81×10^{-2}	3.23×10^{-4}	3.23×10^{-4}	3.23×10^{-4}	$0_1 : 7.401 \times 10^{-3}$
Q1	117.96	132.4	5.081	-0.31925	4.11×10^{-2}	2.90×10^{-4}	2.90×10^{-4}	2.90×10^{-4}	$1_3 : 1.044 \times 10^{-2};$ $1_2 : 6.435 \times 10^{-3};$ $1_0 : 7.401 \times 10^{-3}$
Q2	95.79	115.86	5.322	-0.33232	7.17×10^{-2}	2.74×10^{-4}	2.74×10^{-4}	2.74×10^{-4}	$2_1 : 6.435 \times 10^{-3}$
Q3	107.55	22.83	5.164	-0.33508	3.41×10^{-2}	3.44×10^{-4}	3.44×10^{-4}	3.44×10^{-4}	$3_4 : 1.890 \times 10^{-2};$ $3_1 : 1.044 \times 10^{-2}$
Q4	92.27	110.84	5.052	-0.31926	3.62×10^{-2}	4.57×10^{-4}	4.57×10^{-4}	4.57×10^{-4}	$4_3 : 1.890 \times 10^{-2}$

A Details of quantum circuits for demonstrating cases with $n = 2$

As introduced in the main text, the designed quantum oracle and the whole transpiled circuit for $s = 00$ have been presented in Fig. 3(c) and Fig. 5(a) of main text, respectively. Similarly, the oracle constructions of O_s in Eq. (18) of the main text and corresponding transpiled circuits for demonstrating other three cases $s=01$, 10, and 11 are explicitly shown in Fig. S1 and Fig. S2, respectively. The experimental results for testing these cases are plotted in Fig. 5 (b) of the main text.

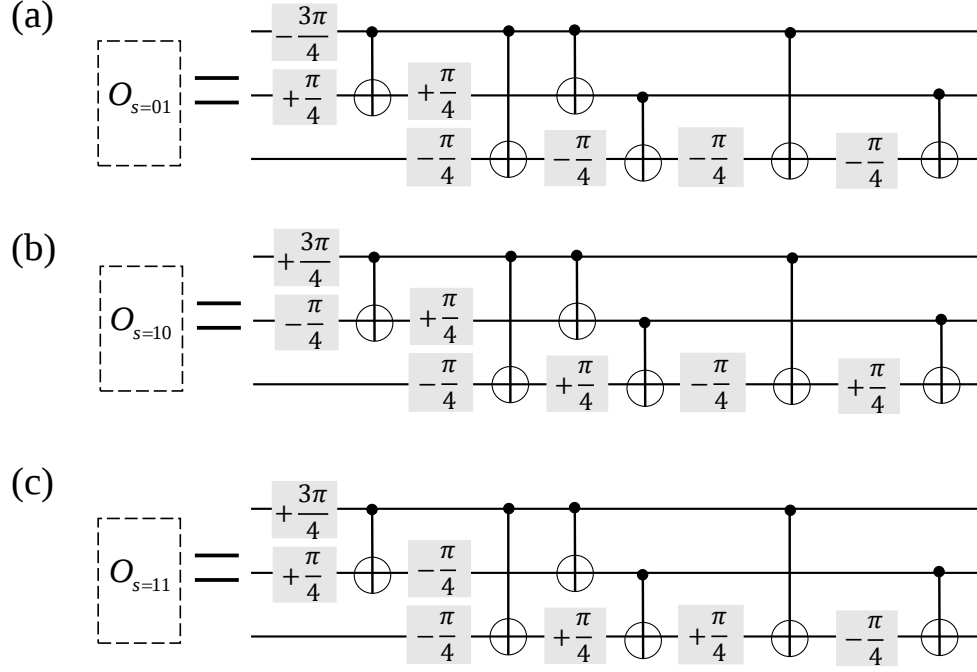


Figure S1: Quantum oracle constructions of (a) $O_{s=01} = \text{diag}(-1,1,-1,-1,1,1,1,1)$, (b) $O_{s=10} = \text{diag}(1,1,1,1,-1,-1,-1,-1)$, and (c) $O_{s=11} = \text{diag}(1,1,1,1,-1,1,-1,-1)$, respectively.

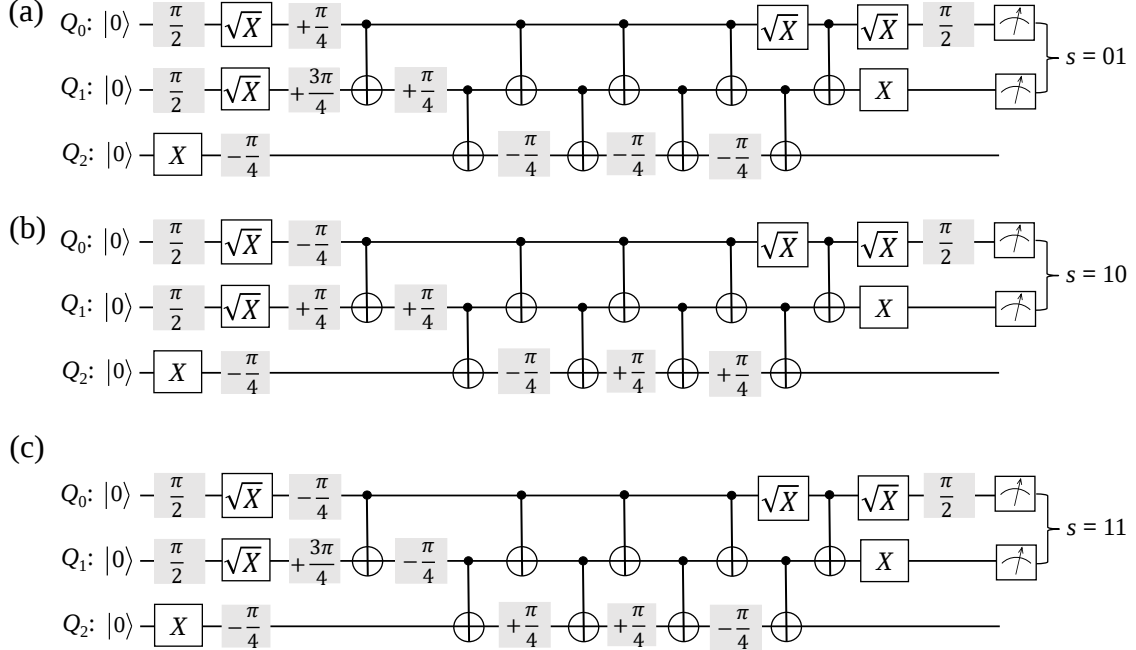


Figure S2: Overall transpiled circuits for demonstrating (a) $s = 01$, (b) $s = 10$, and (c) $s = 11$ on *ibmq_quito*, respectively.

B Details of quantum circuits for demonstrating cases with $n = 3$

For demonstrating cases with $n = 3$, two features of our algorithm are noteworthy. First, it can be verified that the quantum oracles are the same for any two instances s_1s_20 and s_1s_21 as shown in Fig. S3. Therefore, we only need to design and compile four different quantum circuits in total, that is, Fig. 7(a) in the main text as well as Fig. S4 below to identify two bits $s_1s_2 \in \{0,1\}^2$, supplemented with a final classical query $f_s(x, 2)$ to identify s_3 of secret s . For example, the transpiled quantum circuit inside Fig. 7(a) of the main text together with a query $f_{s=001}(x, 2)$ on the measured outcome string x can be used for identifying secret $s = 001$. Second, according to Eq. (17) of the main text, our algorithm can correctly identify the secret string s when the outcome measured from the quantum circuit in imperfect experiments is $s_1s_2s_3$ or $s_1s_2(s_3 \oplus 1)$, leading to the experimental success probabilities recorded in Fig. 7(b) of main text for identifying each s .

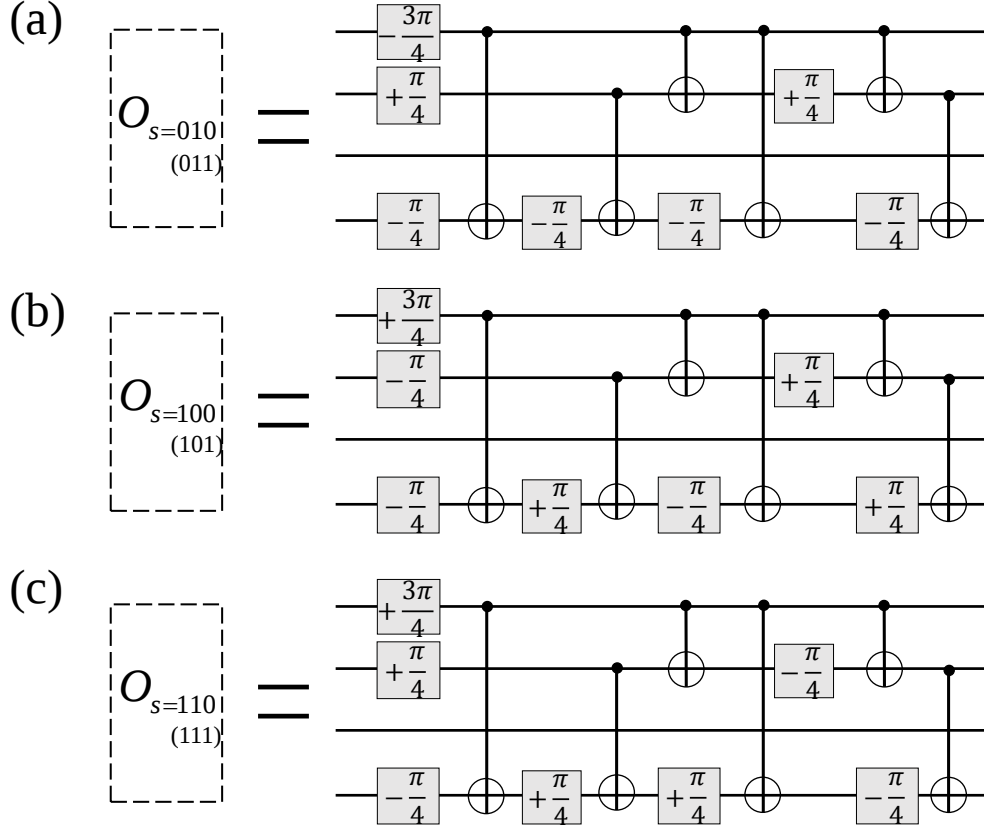


Figure S3: Quantum oracle constructions of (a) $O_{s=010(011)} = \text{diag}(-1, +1, -1, +1, -1, -1, -1, -1, +1, +1, +1, +1, +1, +1, +1, +1)$, (b) $O_{s=100(101)} = \text{diag}(+1, +1, +1, +1, +1, +1, +1, +1, -1, -1, -1, -1, -1, -1, -1, -1)$, and (c) $O_{s=110(111)} = \text{diag}(+1, +1, +1, +1, +1, +1, +1, +1, -1, +1, -1, +1, -1, -1, -1, -1)$, respectively.

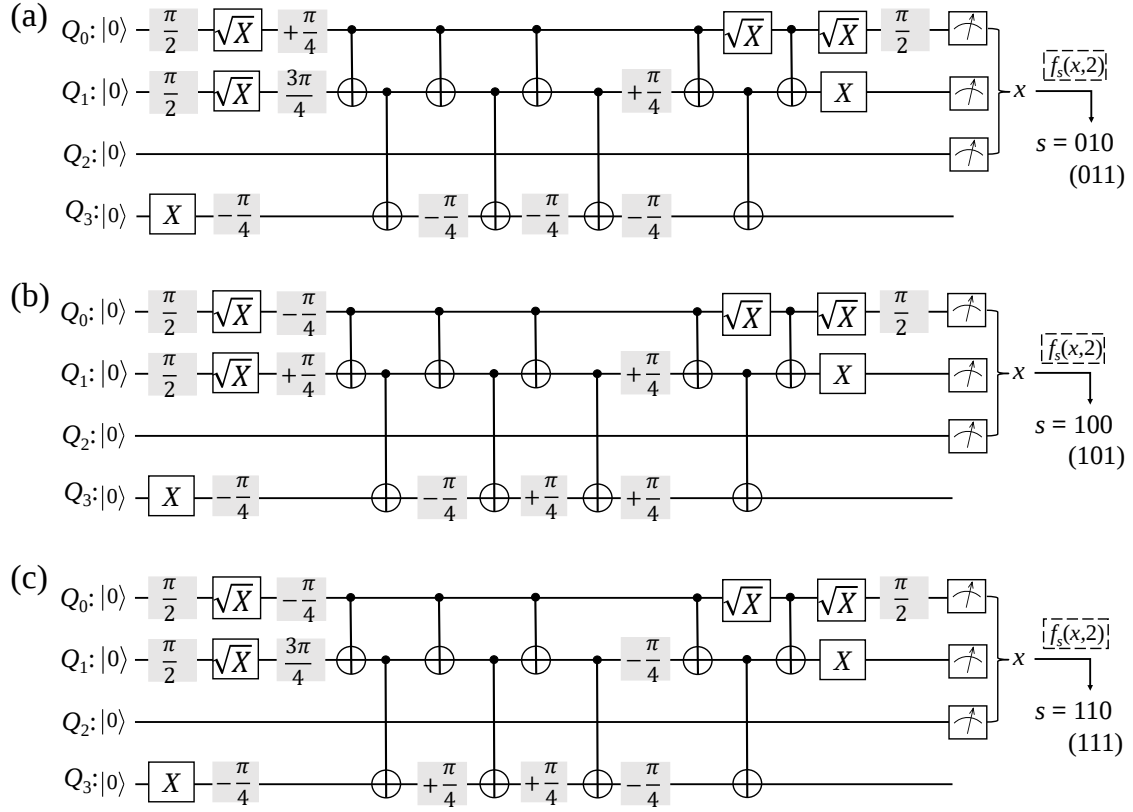


Figure S4: Overall transpiled quantum circuit acting on four physical qubits $\{Q_0, Q_1, Q_2, Q_3\}$ of *ibmq_quito* followed by a classical query $f_s(x, 2)$ to identify (a) $s=010$ or 011 , (b) $s=100$ or 101 , and (c) $s=110$ or 111 .