ROBUST STRING MATCHING IN $O(\sqrt{N}+M)$ QUANTUM QUERIES

CHRIS LOMONT

Abstract. Finding a match of an M item pattern against an N item text is a common computing task, useful in string matching and image matching. The best classical (i.e., non-quantum) string matching algorithms, which are $\Theta(N+M)$ time and query complexity, are improved in many cases in this paper using Grover's quantum searching algorithm. Specifically, given quantum oracles that return the j^{th} symbol in the text or pattern in constant time, we demonstrate a quantum string matching algorithm which returns a pattern match if it exists with query complexity $O(\sqrt{N}+M)$ and time complexity $O(\sqrt{N}\,\log N + M\log(MN)\,)$. Variations of this algorithm allow wildcard matches, can return the number of matches, can return the nearest match in case of no exact matches, as well as the first or last such match, each with a slight change in complexity. We are also better than the best quantum string matching algorithms in many cases.

1. Introduction

This paper gives a new quantum algorithm solving the the following general problems: Given a string T of N symbols, and a pattern P of M symbols (with possible wildcard matching), find one of the $t \geq 0$ matches of P in the text string T. Or, if there is no match, find the nearest match. Also we can return the number t of matches, or the first or last matches or near matches. These algorithms have applications to traditional string matching, image template matching, and other pattern matching problems.

The best classical (non-quantum) string matching algorithms are the Knuth-Morris-Pratt (KMP) [10] algorithm with $\Theta(N+M)$ complexity, and the Boyer-Moore algorithm [4], which often performs in practice better then the KMP algorithm, but has worst case complexity $O((N-M+1)M+|\Sigma|)$, where $|\Sigma|$ is the alphabet size. Finally, there are randomized approximate algorithms like that of Atallah et. al. [1] that give the approximate score vector between the text string and pattern in time $O(N\log M)$ using Fast Fourier Transforms. Note we are only seeking a *single* match, while the classical algorithms return all matches.

The usual 2D image matching algorithm computes the correlation of the template and the image in the following manner. First, to make the template and image have non-overlapping periods, each is padded to form an S pixel image. The Fast Fourier Transform is then done in time $\Theta(S\log S)$, followed by S per entry multiplications, then an inverse Fourier Transform in $\Theta(S\log S)$ time, and the result is searched for

Date: Oct 30, 2003.

2000 *Mathematics Subject Classification.* 03D15, 68Q05, 68W40, 81P68.

Key words and phrases. algorithms, quantum computers, Grover Database Search, image matching, string matching, pattern matching.

Research supported by AFRL grant F30602-03-C-0064.

a max in time $O(S)$, resulting in an $O(S \log S)$ algorithm. This method finds near matches as well, but suffers from the false positives resulting from uniformly bright regions in the image.

Ramesh and Vinay's paper on quantum string matching [12] claims an algorithm with time complexity $O(\sqrt{N} \log \sqrt{N/M} \log M + \sqrt{M} \log^2 M)$, but note 1 below would imply a time complexity $O(\sqrt{N} \log N \log \sqrt{N/M} \log M + \sqrt{M} \log^3 M)$ and a *query* complexity of $O(\sqrt{N} \log \sqrt{N/M} \log M + \sqrt{M} \log^2 M)$.

This paper gives a new string matching algorithm, which avoids the probabilistic matching of [12], and by using a deterministic matching or counting algorithm, followed by a Grover search or maximum finding algorithm, results in an algorithm with time complexity $O(\sqrt{N/t} \log N + M \log(MN))$ and query complexity $O(\sqrt{N/t} + M)$, where $t = \max\{1, \# \text{ of matches}\}$. Thus our new algorithm has superior running time than Ramesh and Vinay's algorithm for fixed M, such as searching longer and longer texts for a fixed phrase. Theirs is better if $M = \alpha N$ for a fixed $\alpha > 0$, as N grows. However, their algorithm cannot do closest match or wildcard matches, while ours can, so each method has benefits. Variations of our algorithm can also return the number of matches, the first or last match or closest match. Returning the nearest match allows our algorithm to replace the convolution algorithms, since our algorithm would be faster (if or when quantum computers become common), and it does not have the false positive problem that convolution causes. Finally, using our algorithms to return all matches is discussed.

2. Background

For an introduction to quantum computing see [2] or the very good book by Chuang and Nielsen [6]. For coverage of algorithms, including the classical string matching algorithms, see [7].

2.1. Notation.

We assume that P and T are binary strings, since this makes the explanation and proofs easier to understand. Any fixed finite alphabet can be recoded in binary and our algorithms still have the same time and query complexities as with a larger alphabet. Let M be the length of P and N be the length of T, with $M \leq N$. We assume $N = 2^n$ and $M = 2^m$ for integers n and m, in order to simplify the analysis later. This restriction can be removed as in [3], and is only made to allow the qubit-wise Hadamard operators $H^{\otimes n}$, which would need to be replaced with Fourier Transforms for the general case. T_j is the j^{th} symbol in T, for $j = 0, 1, \ldots, N-1$. Similarly for P_k in P, $k = 0, 1, \ldots, M-1$. We also assume there are $t \geq 0$ matches of the pattern P to the text T. All values $0, 1, \ldots, N-1$ can be stored in an n qubit register, and similarly for $0, 1, \ldots, M-1$ in an m qubit register. log always denotes log base 2. Finally we define the constant $\lambda = \frac{1}{\sqrt{N}}$, since it cleans up some notation.

We also define for $j \in \{0, 1, \ldots, N-1\}$ the functions

$$f(j) = \begin{cases} 0 \text{ if } P \text{ does NOT match } T \text{ starting at position j} \\ 1 \text{ if } P \text{ matches } T \text{ starting at position j} \end{cases} \tag{1}$$

$$h(j) = \text{number of pattern matches starting at text position } j \tag{2}$$

where the pattern is matched cyclically to the text, that is, with wraparound. f is a boolean function marking a pattern match, and $h(j)$ counts the number of pattern symbols matching T starting at index j. We will construct oracles implementing these functions efficiently in lemmas 16 and 17.

2.2. **Previous work.** Before reviewing previous work and recalling necessary theorems, we want to make the following point.

Note 1. It is common throughout the quantum computing literature to assume Grover's algorithm searches $N = 2^n$ items in time $O(\sqrt{N}\,)$, but this is not quite true. Grover's algorithm requires $\Theta(\sqrt{N}\,)$ *oracle calls*. Grover's algorithm needs to apply $H^{\otimes n}$ (or it's equivalent, see [3]) to a quantum state $O(\sqrt{N}\,)$ times, where H is the Hadamard matrix. Most reasonable models of computation (classical or quantum) require an operation only to affect a fixed, finite number of bits or qubits. So as n grows, $H^{\otimes n}$ cannot be computed in $O(1)$ time as an $O(\sqrt{N}\,)$ algorithm would require. This makes the running time of Grover's algorithm $O(\sqrt{N}\,\log N\,)$. This misconception makes many results based on Grover's algorithm off by a factor of $\log N$. In this paper we take the more cautious route, and assume Grover's algorithm requires $O(\sqrt{N}\,\log N\,)$ *time* complexity and $\Theta(\sqrt{N}\,)$ *query* complexity. The downside is that our time complexities are therefore burdened with log factors other authors omit.

We reworked the necessary papers to make the following theorems agree with note 1, in order to make our algorithm complexities precise.

An extension of Grover's algorithm [9] to an unknown number of marked items is given in [3], and the main result in light of note 1 is

Theorem 2 (Modified Grover Search)**.** *Given N items with $t > 0$ "marked" items and an $O(1)$ time complexity oracle that identifies marked items, there is a quantum algorithm* **QSEARCH** *that locates one of the t marked items with probability $p \geq \frac{3}{4}$ using $\Theta(\sqrt{N/t}\,)$ query complexity and $O(\sqrt{N/t}\,\log N\,)$ time complexity. If $t = 0$, the algorithm returns a random element in time $O(\sqrt{N}\,\log N\,)$ using $O(\sqrt{N}\,)$ oracle calls. The algorithm does NOT need to know the value of t in order to operate.*

Using this theorem, Dürr and Høyer construct a quantum algorithm to find a *unique* minimum (or maximum) element out of an N element ordered set [8]. Again, using note 1 and the modified theorem 2, and extending their analysis to $t > 0$ equal minimum (or maximum) values, this results in

Theorem 3 (Quantum Min/Max)**.** *Given N integers, assume there are $t > 0$ minimal (maximal) ones. Then there is a quantum algorithm* **QMIN** *(respectively,* **QMAX** *) that locates a minimal (maximal) one in $O(\sqrt{N}\,)$ query complexity and $O(\sqrt{N}\,\log N\,)$ time complexity with probability of success $p \geq \frac{3}{4}$. The algorithm does NOT need to know the value of t in order to operate.*

Note 4. In theorem 3 there are no N/t factors as in the others, since the way the algorithm is designed needs the full number of queries. It would be interesting and useful if this could be improved.

Brassard, Høyer, and Tapp give a quantum algorithm to find the number of elements meeting some boolean criteria [5]. Following note 1 and the modified theorem 2, this results in

Theorem 5 (Quantum Counting)**.** *Given N items with $t > 0$ "marked" items and an $O(1)$ time complexity oracle that identifies marked items, there is a quantum* *algorithm* **QCOUNT** *that determines t with probability $p \geq \frac{3}{4}$ using $\Theta(\sqrt{tN}\,)$ query complexity and $O(\sqrt{tN}\log N\,)$ time complexity, requiring only space linear in $\log N$. The algorithm does NOT need to know the value of t in order to operate.*

Note 6. Theorem 5 could be extended to handle the case $t = 0$, but it is not needed in this paper.

The only quantum string matching algorithm we are aware of is the one given by Ramesh and Vinay [12]. Their algorithm does not allow wildcard matches, nor does it return nearest matches, two problems our algorithms remedy. Furthermore our algorithm has better query and time complexity in many cases, especially when $M << N$. The main result of their paper, amended according to note 1 and theorems 2, 3, and 5, is

Theorem 7. *There is a quantum string matching algorithm* **QSTRMATCHRV** *requiring $O(\sqrt{N}\log\sqrt{\frac{N}{M}}\log M + \sqrt{M}\log^2 M)$ oracle calls, with a time complexity of $O(\sqrt{N}\log N\log\sqrt{\frac{N}{M}}\log M + \sqrt{M}\log^3 M)$, finding a match if it exists with probability $p \geq \frac{3}{4}$.*

Note 8. Finally, if note 1 is ignored, then the $\log N$ terms in theorems 2, 3, and 5 can be dropped, and Ramesh and Vinay's time complexity in theorem 7 can be replaced with their better oracle complexity, i.e., each term loses a log factor.

2.3. **Open problems answered.** We also address two open problems in Ramesh and Vinay [12], namely:

(1) **Q:** Can string matching be done with wildcards with complexity as good as theorem 7?
A: Our algorithm does string matching with wildcards, and has complexity better than theirs in some cases. Finding a quantum string matching algorithm allowing wildcards that is better than both of our algorithms in all cases is still an open problem, as is proving lower bounds on time and query complexity.

(2) **Q:** Can convolution be implemented in time $\tilde{O}(\sqrt{N})$ using Fast Fourier Transforms?
A: We answer this in the negative (at least if the coefficients are stored as quantum states) in the paper [11]. The rough idea is that quantum evolution is linear, but convolution and correlation are not, thus no quantum process can compute correlation or convolution *on quantum states*. This work was done trying to make an image matching algorithm. Thus it seems unlikely to find a good matching algorithm based on correlation, so Grover's algorithm seems to be the only known tool for constructing better than classical matching algorithms.

3. The algorithms

In this section we present three algorithms. The first, **QSTRMATCH**, finds a match of the pattern P to the text T if it exists, and variations can return the first or last such match in T. **QSTRMATCHCOUNT** counts the number of matches. And the third, **QSTRNEAR**, finds the nearest match to the pattern. If there is an exact match, **QSTRNEAR** will return it, effectively performing **QSTRMATCH**.

3.1. **Assumptions.** We make the following assumptions:

(1) the existence of an *oracle* U_T, which computes in $O(1)$ time the unitary transformation on quantum states

$$|j\rangle|y\rangle \quad \xrightarrow{U_T} |j\rangle|T_j \oplus y\rangle$$

where $|y\rangle$ is a single qubit, T_j is a binary digit, and addition is mod 2.

(2) the existence of a similar oracle U_P that operates on the pattern string.

(3) there is an $O(1)$ time quantum operation S_k that swaps the first and k^{th} qubits.

Note 9. The first two assumptions are fairly standard in the quantum literature; see for example [3, 5, 8, 9, 12]. If the oracles U_T and U_P take time polylog in N and M, our results still give good algorithms. Since classically we assume a character can be indexed in $O(1)$ time, and we have no reason yet to believe this will be impossible on quantum computers, we make the $O(1)$ assumptions. Similarly we assume we can swap two (qu)bits in constant time.

3.2. **The algorithms.**

3.2.1. *Matching Algorithm.* The algorithm **QSTRMATCH**

(1) Start with two registers in the state $|\psi_1\rangle = \lambda \sum_{i=0}^{N-1} |i\rangle|0\rangle$

(2) Apply f to get $|\psi_2\rangle = \lambda \sum |i\rangle|f(i)\rangle$

(3) Use **QSEARCH** from theorem 2 to find a $|1\rangle$ in the second register if it exists, giving the state $|\psi_3\rangle = \frac{1}{\sqrt{t}} \sum_{i,f(i)=1} |i\rangle|1\rangle$, else it returns $|\psi_3\rangle = \lambda \sum |i\rangle|0\rangle$

(4) Measure the second register to see if there is a solution, that is, if the register contains 1. If so, measure first register to get an index i, and check it is a solution. Otherwise step 3 failed or there is no match.

3.2.2. *Match Counting Algorithm.* The algorithm **QSTRMATCHCOUNT**

(1) Use **QSTRMATCH** to ensure there is at least one match by running it twice. If no match is found on either run, set $ret = 0$ and exit.

(2) Start with two registers in the state $|\psi_1\rangle = \lambda \sum_{i=0}^{N-1} |i\rangle|0\rangle$

(3) Apply f to get $|\psi_2\rangle = \lambda \sum |i\rangle|f(i)\rangle$

(4) Use **QCOUNT** from theorem 5 to count states with $|1\rangle$ in the second register. Set ret to the number of matches.

(5) Repeat steps 2–4 three times to boost success probability to $p \geq \frac{3}{4}$, then return the most common ret value if there is one, else fail.

3.2.3. *Nearest Match Algorithm.* The algorithm **QSTRNEAR**

(1) Start with two registers in the state $|\psi_1\rangle = \lambda \sum_{i=0}^{N-1} |i\rangle|0\rangle$

(2) Apply h to get $|\psi_2\rangle = \lambda \sum |i\rangle|h(i)\rangle$

(3) Use **QMAX** from theorem 3 to find one of the $t > 0$ maximum values v in the second register, giving the state $|\psi_3\rangle = \frac{1}{\sqrt{t}} \sum_{i,h(i)=v} |i\rangle|v\rangle$

(4) Measure the first register to get an index where the pattern matches most closely to the text.

Note 10. In **QSTRMATCH**, for a slight change in complexity we could use theorem 3 to find the index of the first or last match right after step 3. Similarly we could find the first or last best match in case of ties in algorithm **QSTRNEAR**.

3.3. **The main results.** From these algorithms we have the main results of this paper:

Theorem 11. *Given the assumptions in section 3.1 and using the notation of section 2.1, algorithm* **QSTRMATCH** *(section 3.2.1) requires $O(\sqrt{N/t} + M)$ oracle calls, $O(M)$ auxiliary space, and has running time $O(\sqrt{N/t}\ \log N + M\log(MN))$ when $t > 0$. If $t = 0$, these are $O(\sqrt{N} + M)$, $O(M)$, and $O(\sqrt{N}\ \log N + M\log(MN))$ respectively. If P matches T in $t \geq 1$ distinct starting indices, then the algorithm returns a random matching index with probability $p \geq \frac{3}{4}$. If there are no matches, the algorithm returns a random item. The algorithm does NOT need to know t to operate.*

Theorem 12. *Given the assumptions in section 3.1 and using the notation of section 2.1, algorithm* **QSTRMATCHCOUNT** *(section 3.2.2) breaks into two cases. If $t = 0$, the algorithm has the same requirements as* **QSTRMATCH***. If $t > 0$, the algorithm requires $O(\sqrt{tN} + M)$ oracle calls, $O(M + \log N)$ auxiliary space, and has running time $O(\sqrt{tN}\ \log N + M\log(MN))$. The algorithm returns the number $t \geq 0$ of matches with success probability $p \geq \frac{3}{4}$. The algorithm does NOT need to know t to operate.*

Theorem 13. *Given the assumptions in section 3.1 and using the notation of section 2.1, algorithm* **QSTRNEAR** *(section 3.2.3) requires $O(\sqrt{N} + M)$ oracle calls, has running time $O(\sqrt{N}\ \log N + M\log(M^2N))$, and uses $O(\log M)$ auxiliary space. This algorithm returns the index of the closest match of the pattern to the text, where distance is the number of mismatching symbols. In case of multiple answers, one is returned at random. The algorithm succeeds with probability $p \geq \frac{3}{4}$.*

Note 14. In theorem 13, the complexity $O(M\log(M^2N)) = O(2M\log(M\sqrt{N}))$, so $O(M\log(M^2N)$ could be replaced with $O(M\log(MN))$, since the log and O allow changing exponents.

Note 15. **QSTRMATCH** (**QSTRNEAR**) can be used to try to find all of the matches (nearest matches) as follows. Run **QSTRCOUNT** several times to get the proper number of matches with very high probability, and then use **QSTRMATCH** enough times to find all the matches. The running time of this can be improved if the found matches are marked invalid for subsequent runs. Numerical simulations of this algorithm imply fast convergence, but nice analytical bounds would be nice.

4. Proofs of the theorems

To do pattern matching, we need oracles that can compute the functions f and h from section 2.1. Ramesh and Vinay [12] use a probabilistic matching oracle to match the pattern to the text with two sided probability. We use deterministic oracles, constructed in the following lemmas from the assumptions in section 3.1.

Lemma 16. *Given the assumptions in section 3.1 and using the notation of section 2.1, then there is a quantum operation* **COUNT** *performing* [1]

$$|j\rangle|z\rangle \xrightarrow{\textbf{COUNT}} |j\rangle|h(j)+z\rangle \tag{3}$$

where $j \in \{0, 1, \ldots, N-1\}$*, the function* h *is that in equation 2, and the sum is* mod 2^m*.* **COUNT** *is performed using* $4M$ *oracle calls and* $2 + \log M$ *ancillary qubits in time* $O(M \log(M^2 N))$*.*

Proof. The idea is to count matches, and add them to the accumulator register $|z\rangle$. Start with the 5 register state

$$|j\rangle|z\rangle|0_1 0_2 \ldots 0_m\rangle|0\rangle|0\rangle \tag{4}$$

using $2 + m$ ancillary qubits. Write these 5 registers as $|a\rangle|b\rangle|c\rangle|d\rangle|e\rangle$, and define the following quantum operations on them and note time complexities.

operation	notation	output	complexity
increment a	A	$\|a+1 \text{ mod } 2^n\rangle\|b\rangle\|c\rangle\|d\rangle\|e\rangle$	$O(\log N)$
increment c	C	$\|a\rangle\|b\rangle\|c+1 \mod 2^m\rangle\|d\rangle\|e\rangle$	$O(\log M)$
text oracle	U_T	$\|a\rangle\|b\rangle\|c\rangle\|T_a \oplus d\rangle\|e\rangle$	$O(1)$
patt oracle	U_P	$\|a\rangle\|b\rangle\|c\rangle\|d\rangle\|P_c \oplus e\rangle$	$O(1)$
$\oplus$ 2 bits	R	$\|a\rangle\|b\rangle\|c\rangle\|d \oplus e\rangle\|e\rangle$	$O(1)$
add d to b	θ	$\|a\rangle\|b+d \mod 2^m\rangle\|c\rangle\|d\rangle\|e\rangle$	$O(\log M)$

Define the combination $L = ACU_P U_T R \theta R U_T U_P$. Then in time $O(2\log M + \log N)$ and 4 oracle queries L performs

$$|a\rangle|b\rangle|c\rangle|d\rangle|e\rangle \xrightarrow{L} |a+1\rangle|b + (T_a \oplus P_c)\rangle|c+1\rangle|d\rangle|e\rangle \tag{5}$$

where the mod has been dropped in the first three registers for clarity. Applying this M times to the state in equation 4 counts all the matches between the text and the pattern at the start position j, using $O(2M\log M + M\log N) = O(M\log(M^2N))$ time and $4M$ oracle queries. Finally, in $O(\log N + \log M)$ time subtract M from the first and third registers, resetting the index counters. This proves the lemma. □

Lemma 17. *Given the assumptions in section 3.1 and using the notation of section 2.1, then there is a quantum operation* **MATCH** *performing* [2]

$$|j\rangle|y\rangle \xrightarrow{\textbf{MATCH}} |j\rangle|f(j) \oplus y\rangle \tag{6}$$

where $j \in \{0, 1, \ldots, N-1\}$ *and the function* f *is that in equation 1.* **MATCH** *is performed using* $4M$ *oracle calls and* $M + \log M + 1$ *ancillary qubits in time* $O(M \log(MN))$*.*

Proof. The idea is to use the M ancillary qubits as flags denoting matches between the text starting at position j and the pattern. Start with the 5 register state

$$|j\rangle|z\rangle|0_1 0_2 \ldots 0_m\rangle|0_1 0_2 \ldots 0_M\rangle|0\rangle \tag{7}$$

[1] The first register is n qubits, and the second is m qubits.

[2] The first register is n qubits, and the second is 1 qubit.

using $m+M+1$ ancillary qubits. Write these 5 registers as $|a\rangle|b\rangle|c\rangle|d\rangle|e\rangle$, and define the following quantum operations on them and note time complexities.

operation	notation	output	complexity
increment a	A	$\lvert a+1 \mod 2^n\rangle\lvert b\rangle\lvert c\rangle\lvert d\rangle\lvert e\rangle$	$O(\log N)$
increment c	C	$\lvert a\rangle\lvert b\rangle\lvert c+1 \mod 2^m\rangle\lvert d\rangle\lvert e\rangle$	$O(\log M)$
text oracle	U_T	$\lvert a\rangle\lvert b\rangle\lvert c\rangle\lvert T_a \oplus d_1, d_2, \ldots, d_M\rangle\lvert e\rangle$	$O(1)$
patt oracle	U_P	$\lvert a\rangle\lvert b\rangle\lvert c\rangle\lvert d\rangle\lvert P_c \oplus e\rangle$	$O(1)$
$\oplus$ 2 bits	R	$\lvert a\rangle\lvert b\rangle\lvert c\rangle\lvert d_1 \oplus e, d_2, \ldots, d_M\rangle\lvert e\rangle$	$O(1)$
qubits $1 \leftrightarrow k$	S_k	$\lvert a\rangle\lvert b\rangle\lvert c\rangle\lvert d_k, d_2, \ldots, d_1, \ldots, d_M\rangle\lvert e\rangle$	$O(1)$

S_k swaps the first and k^{th} qubits in the fourth register. Define the combination $J_k = ACS_kU_PU_TRU_TU_PS_k$. Then in time $O(\log M + \log N)$ and 4 oracle queries J_k performs

$$|a\rangle|b\rangle|c\rangle|d\rangle|e\rangle \xrightarrow{J_k} |a+1\rangle|b\rangle|c+1\rangle|d\oplus\beta\rangle|e\rangle \tag{8}$$

where the mod has been dropped in the first and third registers for clarity, and $\beta = 2^{k-1}$ if $T_a \neq P_c$, else $\beta = 0$. Applying M of the J_k concatenated as $\prod_{k=1}^{M} J_k$ to the state in equation 7 computes the matches between the text and the pattern at the start position j, using $O(M\log M + M\log N) = O(M\log(MN)\,)$ time and $4M$ oracle queries, and sets exactly one qubit in the 4th register to 0 for each matching position, and one qubit to 1 for each mismatch. To reset the index counters, subtract M from the first and third registers in $O(\log N + \log M)$ time. Finally, in $O(\log M)$ apply to the second register a **NOT** controlled by all the bits in the 4th register, replacing $|b\rangle$ with $|b\oplus 1\rangle \Leftrightarrow |d\rangle = |0\rangle$. This proves the lemma. □

Note 18. Wildcard matching can be done in lemmas 16 and 17 by ignoring symbols in P with no change in complexity.

Finally, we prove the main theorems 11, 12, and 13.

Proof of theorem 11. Assume $t > 0$. The $t = 0$ case has the same proof, with care taken not to divide N by $t = 0$. Step 1 of **QSTRMATCH** requires time $O(\log N)$ to prepare using the Hadamard operator on $\log N$ of the qubits. By lemma 17 step 2 takes $O(M\log(MN)\,)$ time, $O(M)$ auxiliary space, and uses $4M$ queries. By theorem 2 step 3 requires $O(\sqrt{N/t}\ \log N\,)$ time, $\Theta(\sqrt{N/t}\,)$ queries, and succeeds with probability $p \geq \frac{3}{4}$. Finally, we assume the measurement is $O(1)$. Thus the algorithm requires $O(\sqrt{N/t} + M\,)$ queries, $O(M)$ auxiliary space, and $O(\sqrt{N/t}\ \log N + M\log(MN)\,)$ time, with $p \geq \frac{3}{4}$ probability of success. □

Proof of theorem 12. Step 1 applies **QSTRMATCH** so uses the resources from theorem 11, and succeeds with probability $p \geq \frac{3}{4}$. If $t = 0$, step 1 never passes, so the complexity is a constant multiple of the requirement of **QSTRMATCH**. Otherwise we can assume $t > 0$. Step 2 requires time $O(\log N)$ using the Hadamard operator on $\log N$ of the qubits. By lemma 17 step 3 takes $O(M\log(MN)\,)$ time and uses $4M$ queries, with $O(M\,)$ auxiliary space. By theorem 5 step 4 requires $O(\sqrt{tN}\log N\,)$ time, $\Theta(\sqrt{tN}\,)$ queries, $O(\log N\,)$ space, and succeeds with probability $p \geq \frac{3}{4}$.

So far the algorithm fails if step 1 or step 4 fails. Set $q = \frac{1}{4}$. If step 1 finds a match, it is with certainty, since the match is checked classically. So step 1 fails only if $t > 0$ and **QSTRMATCH** returns no match on each run; thus failure has probability $p_1 \leq q^2$. Running step 4 three times and taking the majority answer fails with probability $p_2 \leq \binom{3}{2}q^2(1-q) + \binom{3}{3}q^3$. So the probability of the entire algorithm failing is $\leq p_1 + (1-p_1)p_2 < \frac{1}{4}$. The overall algorithm succeeds with probability $p \geq \frac{3}{4}$.

Thus if $t = 0$ the algorithm requires the same resources as **QSTRMATCH**, otherwise the algorithm uses $O(\sqrt{tN}\ \log N + M\log(MN)\,)$ time and $O(\sqrt{tN} + M\,)$ queries, using $O(M + \log N\,)$ auxiliary space. It succeeds with probability $p \geq \frac{3}{4}$ for any t value. □

Proof of theorem 13. Step 1 requires time $O(\log N)$ using the Hadamard operator on at most $\log N$ of the qubits. By lemma 16 step 2 takes $O(M\log(M^2N)\,)$ time and uses $4M$ queries, with $O(\log M\,)$ auxiliary space. By theorem 3 finding a maximum value in step 3 requires $O(\sqrt{N}\ \log N\,)$ time and $O(\sqrt{N}\,)$ queries, and succeeds with probability $p \geq \frac{3}{4}$. Finally, we assume the measurement is $O(1)$. Thus the algorithm requires $O(\sqrt{N} + M\,)$ queries, $O(\sqrt{N}\ \log N + M\log(M^2N)\,)$ time, and $O(\log M\,)$ auxiliary space, and succeeds with $p \geq \frac{3}{4}$ probability of success. □

5. Conclusion and open problems

We have demonstrated a new quantum string matching algorithm which finds a pattern match against a text if it exists in $O(\sqrt{N} + M\,)$ queries and using $O(\sqrt{N/t}\ \log N + M\log(MN)\,)$ time, where $t = \max\{1, \#\text{ of matches}\}$. Variations are shown that return the number of matches, the first or last match, the nearest match, or the first or last nearest match, all allowing wildcard matching. Finally, an algorithm was explained to find all matches.

Our oracles that implement f and h are not very sophisticated, and we think this algorithm could be improved by finding better oracles implementing them. However we were unable to do so at this time.

Finally, note that our complexities are estimated very conservatively. For example, if we assume we can increment an m qubit quantum index register or do addition in constant time, instead of $O(m\,)$, our complexities would all replace the terms like $M\log(MN)$ with M, improving our results. However it is unlikely that this is physically possible for quantum registers as M and N grow.

As a last open problem, it would be good to get quantum lower bounds for string matching, similar to the classical $\Omega(N+M)$ bounds. It would be impressive indeed if quantum string matching turns out to be $\Theta(\sqrt{N} + \sqrt{M}\,)$ time and query complexity!

References

1. Mikhail Atallah, Frederic Chyzac, and Philippe Dumas, *A randomized algorithm for approximate string matching*, Algorithmica **29** (2001), 468–486.
2. Adriano Barenco, *Quantum physics and computers*, Contemporary Physics **38** (1996), 357–389.
3. M. Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp, *Tight bounds on quantum searching*, Proc. 4th Workshop on Physics and Computation-PhysComp, 1996, quant-ph/9605034, pp. 36–43.
4. Robert S. Boyer and J. Strother Moore, *A fast string-searching algorithm*, Communications of the ACM **20** (1977), no. 10, 762–772.
5. Gilles Brassard, Peter Høyer, and Alain Tapp, *Quantum counting*, Lecture Notes in Computer Science **1443** (1998), 820+, quant-ph/9805082.
6. I. L. Chuang and M. A. Nielsen, *Quantum computation and quantum information*, Cambridge University Press, Cambridge, 2000.

7. Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein, *Introduction to Algorithms, Second Edition*, MIT Press, 2001, ISBN: 0262032937.
8. C. Dürr and P. Høyer, *A quantum algorithm for finding the minimum*, quant-ph/9607014.
9. L. K. Grover, *A fast quantum mechanical algorithm for database search*, Proc. 28th Ann. ACM Symp. on Theory of Comput., 1996, pp. 212–219.
10. Donald Knuth, James Morris, and Vaughan Pratt, *Fast pattern matching in strings*, SIAM Journal on Computing **6** (1977), no. 2, 323–350.
11. Chris Lomont, *Quantum convolution and quantum correlation algorithms are physically impossible*, (2003), quant-ph/0309070.
12. H. Ramesh and V. Vinay, *String matching in* $\tilde{O}(\sqrt{n}+\sqrt{m})$ *qauntum time*, Journal of Discrete Algorithms **2** (2001), no. 1, quant-ph/0011049.

E-mail address: clomont@cybernet.com
E-mail address: clomont@math.purdue.edu
URL: www.math.purdue.edu/~clomont
URL: www.cybernet.com
Current address: Cybernet Systems Corporation, 727 Airport Blvd., Ann Arbor, MI, 48108-1639 USA.