Response to Ramesh & Vinay, (2003)

String Matching in $\tilde{O}(\sqrt{n} + \sqrt{m})$ Quantum Time

Matthew Evans, Ariz Siddiqui, Nathan Puskuri

April 27, 2025

Outline

String Matching in $\tilde{O}(\sqrt{n} + \sqrt{m})$ Quantum Time

H. Ramesh & V. Vinay (IISc Bangalore, 2000)

Problem Statement

Given a text t of length n and a pattern p of length m, decide whether p occurs in t.

- **Classical bound:** $\Theta(n + m)$ via KMP, Boyer-Moore, etc.
- **Quantum goal:** Exploit amplitude amplification to beat linear time.
- **Main result:** A quantum algorithm running in

$$\widetilde{O}\left(\sqrt{n} + \sqrt{m}\right)$$

  with constant two-sided error probability.

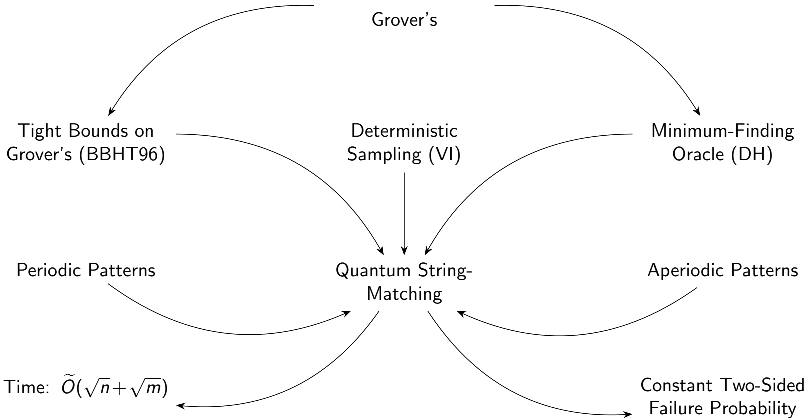

Figure: Conceptual dependencies in the $\widetilde{O}(\sqrt{n} + \sqrt{m})$ quantum string-matching algorithm.

Grover's Algorithm

- Problem: Given a database of N elements and oracle $f(x) = 1$ for marked items, find an x with $f(x) = 1$.
- Steps:
  1. Initialize n qubits into uniform superposition over $N = 2^n$ states.
  2. Apply the search oracle to flip the phase of marked states (multiply goal state amplitude by -1).
  3. Apply diffusion operator which reflects all states amplitude across the mean amplitude.
  4. Repeat oracle + diffusion $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$
  5. Measure to obtain the marked element with high probability.
- Time Complexity: $\mathcal{O}(\sqrt{N})$.
- Why it works: Oracle phase-flips mark targets; diffusion amplifies their amplitudes.
- Significance: Core building block for quantum search algorithms.

BBHT96: Tight bounds on quantum searching

What it is

- Even when the number of target $t \geq 1$ items in a database is unknown, you can still find a marked item in $\tilde{O}(\sqrt{N/t})$ oracle calls.

Why it matters

- BBHT96 modifies grovers algorithm slightly to function when the number of possible solutions is unknown like in this string matching problem.

Probablistic Oracles

What it is

- We will replace search oracles which flip target states with 100 percent probability with probablistic search oracles which flip target states with 75 percent accuracy
- How to preserve accuracy?
- We will apply the probabilistic search oracles log(n) times to the current superposition
- Afterwards each state will flip or not flip based on what it did in the majority of the log(n) times the probablistic oracle was applied

String Match in $O(\sqrt{m}\ \sqrt{n})$

Outer Grovers algorthm

- n = size of text, m = size of pattern. n - m + 1 = number of comparisons/states
- EX: text = abcdef, pattern = def, n = 6, m = 3.
- abc, bcd, cde, and def
- $|00\rangle$: index 0 ... index m-1 (abc)
- $|01\rangle$: index 1 ... index m (bcd)
- $|10\rangle$: index 2 ... index m+1 (cde)
- $|11\rangle$: index 3 ... index m+2 (def)
- Oracle f(i) applies phase kickback if match at position/state i
- EX: 1/sqrt(4) **f(i)** $(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$
- States provide index to search oracle
- n usually dominates so $O(\sqrt{n})$ iterations of grovers to find string match at desired state
- Goal: When we measure we want a state that maps to a

String Match in $O(\sqrt{m}\ \sqrt{n})$ continued

Inner Grovers algorithm

- How does the search oracle f(i) detect matches? f(i) performs Grovers algorithm
- Goal: Another grovers algorithm that maps to an integer j such that i+j is a mismatch
- After the search oracle f(i) acts on a state it knows where the starting position of the substring in the text, i
- Now the oracle f(i) needs to detect if there exists a text[i+j] thats not equal to pattern[j] where j is an integer from [0,m]
- This can be done with another grovers search with
- Our states in the superposition $\sum_{j=0}^{m} a$
- Inside f(i) we have search oracle g(i,j) which acts on states i to i+m-1
- g(i,j) applies inverts phase of state when text[i+j] != pattern[j]
-

Deterministic Sampling

What it is

- Given an aperiodic pattern p of length m, and a text block of length $m/2$, there are $m/2$ possible alignments.
- DS picks $O(\log m)$ indices in the pattern so that at most one alignment can match all sampled positions.

Steps

1. Form $m/2$ "copies" of the pattern, each shifted by one position.
2. Find a column where at least two copies differ.
3. Select one of the symbols at that column as the sample and discard copies that don't match.
4. After $O(\log m)$ rounds, one copy remains; its chosen columns form the sample S.

Deterministic Sampling

Why it works

- Instead of a full pattern check per alignment ($O(m)$ classically, $\tilde{O}(\sqrt{m})$ quantum), only $O(\log m)$ sampled positions are tested.
- Only the single surviving alignment requires the expensive full check.

Why it matters

- Enables the overall quantum string matching to run in $\tilde{O}(\sqrt{n} + \sqrt{m})$ by reducing costly $\sqrt{m}$ checks to one per text block.

DH96: Minimum Finding Oracle

What it is

- Given a database of size n and a comparison oracle that, for any two indices i, j, indicates which element is smaller.
- Finds the index of the minimum element in $O(\sqrt{n})$ time.
- Serves as the backbone for all "pick the smallest (or leftmost) index satisfying a condition" steps.

Steps

1. Pick a random starting position k.
2. Use Grover's search to find any index i with database$[i] <$ database$[k]$.
3. If such an i is found, set $k \leftarrow i$ and repeat.
4. Otherwise, k is the index of the minimum element.

DH96: Minimum Finding Oracle

Why it matters

- Building the deterministic sampling set: Repeatedly eliminate half of the $m/2$ pattern copies by finding the leftmost and rightmost survivor via DH96 in $O(\sqrt{m}\log m)$ time.
- After locating the matching text-block with the $h(i)$ oracle, invoke DH96 over block indices to pinpoint the earliest occurrence, preserving the overall $\tilde{O}(\sqrt{n}+\sqrt{m})$ bound.

The Algorithm (Part 1)

1. **Deterministic-Sampling Preprocessing.**
   - Run Vishkin's deterministic-sampling on p of length m to obtain an $O(\log m)$-sized sample set S.
   - Cost: $\widetilde{O}(\sqrt{m}\log^2 m)$.
2. **Partition the text.**
   - Divide the text t into

     $$B = \left\lceil \frac{2(n-m+1)}{m} \right\rceil$$

     blocks, each of size $\approx m/2$.
3. **Quantum search for a "hit" block.**
   - Define oracle $h(i)$: tests if block i has at least one alignment matching on all positions in S in $\widetilde{O}(\sqrt{m}\log m)$ time.
   - Use Grover search over $i = 1, \ldots, B$ with oracle h; time $\widetilde{O}(\sqrt{n}\log m)$. If none found, conclude "no occurrence."

The Algorithm (Part 2)

4. **Locate surviving alignment in block.**
   - Define oracle $k(i^*, j)$: checks alignment at shift j in block i^* on sample S in $O(\log m)$ time.
   - Use Grover search over $j = 0, \ldots, \lfloor m/2 \rfloor$ with oracle k; time $\widetilde{O}(\sqrt{m} \log m)$. If none survives, conclude "no occurrence."
5. **Full quantum verification.**
   - Run Grover search over $\ell = 1, \ldots, m$ with oracle "$t[i^* + j^* + \ell] \neq p[\ell]$" to find any mismatch; time $\widetilde{O}(\sqrt{m})$.
   - If no mismatch is found, report occurrence at $i^* + j^*$; otherwise, conclude "no occurrence."

Concerning Periodicity

Constant Two-Sided Failure Probability

Achieving $\tilde{O}(\sqrt{n} + \sqrt{m})$

Conclusion