Quantum Closest-Pair Search for Biological Sequences via k-Mer Distribution Statistics

Zhezheng Xander Song
Ray and Stephanie Lane Computational Biology Department,
Carnegie Mellon University, Pittsburgh, PA, USA

Carl Kingsford[1]
Ray and Stephanie Lane Computational Biology Department,
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Finding highly similar pairs of biological sequences is a fundamental task in bioinformatics. For alignment-free k-mer distributional similarities, the induced feature space is high-dimensional and lacks the low-dimensional geometric structure used by classical exact closest-pair algorithms. Thus, for a collection of N sequences in the pairwise-score setting, exhaustive evaluation over the $\binom{N}{2}$ candidate pairs is the natural classical baseline.

We present the first quantum framework targeting alignment-free closest-pair search in biological sequence collections using distributional k-mer statistics. The central technical contribution is the construction of a coherent pairwise-score estimation circuit for this similarity measure. It encodes empirical k-mer distributions as square-root amplitude states and provides a sparse prefix-tree construction for preparing these states, under which the state overlap is exactly the Bhattacharyya coefficient. Standard SWAP-test and quantum-amplitude-estimation subroutines provide a coherent bounded-precision estimator for the squared Bhattacharyya overlap.

We analyze maximum finding under an explicit assumption that a fixed ε-resolved total order over all legal pairs admits an efficient clean coherent implementation. Under this assumption, the procedure returns, with probability at least 2/3, a pair whose squared Bhattacharyya score is within ε of the optimal score, using $O(N)$ expected comparison-oracle calls. If the optimal score is separated from every strictly suboptimal score by more than ε, the returned pair is exactly optimal.

Combining this comparison-order assumption with an idealized qRAM-style data-access model gives the conditional sequential gate complexity $\widetilde{O}\left(\frac{NL}{\varepsilon}\right)$, whereas explicit multiplexed indexed loading gives $\widetilde{O}\left(\frac{N^2L}{\varepsilon}\right)$. We also provide a proof-of-concept Q# implementation that integrates coherent indexed loading, SWAP-test-based score estimation, finite-precision marking, and Grover-style search, providing circuit-level validation of the main computational components.

2012 ACM Subject Classification Theory of computation → Quantum query complexity; Applied computing → Molecular sequence analysis; Theory of computation → Quantum computation theory; Theory of computation → Design and analysis of algorithms

Keywords and phrases quantum algorithms, closest-pair search, k-mer statistics, Bhattacharyya coefficient, Dürr–Høyer maximum finding, quantum amplitude estimation

Digital Object Identifier 10.4230/LIPIcs.WABI.2026.9

Supplementary Material *Software*: `https://github.com/Kingsford-Group/tieria`

Funding *Carl Kingsford*: This work was supported in part by the US National Science Foundation [III-2232121], the US National Institutes of Health [R01HG012470].

Acknowledgements We thank Shane Elder for his careful reading of a later version, for identifying several important issues, and for suggesting substantial improvements. We also thank Frank Britto Bisso, Emmanuel Mhrous, and Yu-Hsiang Chen for their helpful comments on an earlier version of the manuscript.

[1] corresponding author

1 Introduction

Given a collection of biological sequences, we consider the problem of identifying the most similar pair under a prescribed sequence-similarity score. This task appears naturally in redundancy reduction [17], clustering [10], database cleaning [23], comparative genomics [20], and metagenomic analysis [19], where highly similar sequences may indicate homologous records, repeated observations, closely related strains, or sequences that should be grouped together for downstream analysis.

Traditionally, highly similar sequence pairs have been found either by direct all-pairs comparison or by scalable approximate strategies such as clustering heuristics, seed-and-extend search, sparse chaining, and sketching or locality-sensitive hashing. Direct all-pairs comparison provides a natural exact baseline but becomes expensive for large collections. Practical bioinformatics tools therefore use a variety of scalable alternatives: CD-HIT uses greedy representative-based clustering for redundancy reduction [17], USEARCH/UCLUST accelerates search and clustering through fast heuristic matching [10], VSEARCH supports large-scale clustering, dereplication, and all-vs-all comparison workflows [23], Mash uses MinHash sketches to estimate genomic distances efficiently [20], and skani estimates ANI using sparse approximate alignments and sparse chaining [26]. Minimizer-based sketching methods provide another major class of scalable sequence-comparison primitives, selecting representative k-mers so that sequence similarity or alignability can be estimated from compact sketches [30, 18]. These methods have been highly successful in practice, but they typically trade exact closest-pair search for filtering, approximation, or threshold-based clustering.

We analyze exact closest-pair search in a black-box pairwise-score model. The input induces $P = \binom{N}{2}$ legal candidate pairs, and an algorithm accesses a candidate pair only by evaluating its score $F(S_i, S_j)$. Without additional structure relating the scores of different pairs, exhaustive evaluation is the natural classical exact baseline. In the worst case, if a pair is left unevaluated, that pair may have a score larger than every evaluated pair. Thus, the classical exact query complexity in this model is $\Theta(P) = \Theta(N^2)$.

This black-box model is motivated by the high-dimensional nature of alignment-free k-mer representations. The square-root k-mer representations considered here lie in a space whose dimension is the number M of observed k-mers, and we do not assume the fixed-dimensional geometric structure exploited by classical closest-pair algorithms. For example, closest-pair search in the Euclidean plane can be solved in $O(N \log N)$ time using geometric structure such as divide-and-conquer arguments or Voronoi diagrams [25]. In contrast, Karthik and Manurangsi showed that, assuming the Strong Exponential Time Hypothesis, exact monochromatic closest pair under the ℓ_p metric in certain high-dimensional regimes admits no strongly subquadratic classical algorithm [24]. These metric-specific hardness results provide context for why low-dimensional geometric speedups should not be assumed in our setting.

In this black-box setting, quantum maximum-finding algorithms can reduce the number of pair-score queries. The Dürr–Høyer framework reduces maximum finding to repeated quantum searches for candidates that improve on the current threshold [9]. The resulting searches use Grover-style amplitude amplification, with the BBHT procedure handling the case in which the number of improving pairs is unknown [14, 6].

This unstructured-search viewpoint is also consistent with known quantum complexity results for closest-pair problems. Aaronson et al. gave a $\widetilde{O}(N^{2/3})$ quantum algorithm for constant-dimensional Euclidean closest pair, while showing that in certain higher-dimensional

regimes a substantially better-than-Grover speedup would contradict the Quantum Strong Exponential Time Hypothesis [1]. As above, this result concerns metric closest-pair problems and is included as complexity-theoretic context rather than as a lower bound for the squared Bhattacharyya score studied here.

Many quantum algorithms that operate on large classical inputs assume a coherent data-access primitive, often modeled as quantum random access memory (qRAM). At a high level, qRAM allows an address register in superposition to access the corresponding stored data coherently, e.g., $\sum_i \alpha_i |i\rangle|0\rangle \mapsto \sum_i \alpha_i |i\rangle|\text{data}_i\rangle$. In our analysis, the stored data are indexed state-preparation data for the empirical k-mer distribution states, and we assume that such data can be queried coherently with polylogarithmic overhead. Such assumptions are common in high-level quantum algorithm analyses, but scalable qRAM is a strong architectural assumption and remains challenging in practice [12, 11, 3].

To apply these quantum search and estimation ideas to biological sequences, we use a distributional k-mer representation and a similarity measure that can be evaluated naturally within a quantum circuit. For each sequence, the empirical frequencies of its observed k-mers define a probability distribution over a global k-mer basis $\mathcal{K}$. Given two sequences A and B with empirical k-mer distributions P_A and P_B, we compare them using the Bhattacharyya coefficient [5] $\mathrm{BC}(A, B) = \sum_{x \in \mathcal{K}} \sqrt{P_A(x) P_B(x)}$. This coefficient is large when two sequences assign similar weights to the same k-mers, so it measures the overlap between empirical k-mer distributions and captures alignment-free compositional similarity. Bhattacharyya-type distances have also appeared in previous alignment-free sequence-comparison work; for example, Zhang and Wang used the Bhattacharyya distance for alignment-free phylogenetic analysis of protein sequences [29].

Under square-root amplitude encoding, define the corresponding quantum states by $|\psi_A\rangle = \sum_{x \in \mathcal{K}} \sqrt{P_A(x)} |x\rangle$, $|\psi_B\rangle = \sum_{x \in \mathcal{K}} \sqrt{P_B(x)} |x\rangle$. Since the amplitudes are real and nonnegative, their inner product is $\langle \psi_A | \psi_B \rangle = \sum_{x \in \mathcal{K}} \sqrt{P_A(x) P_B(x)} = \mathrm{BC}(A, B)$. We refer to this inner product $\langle \psi_A | \psi_B \rangle$ as the state overlap between the two encoded sequence states. Thus the squared Bhattacharyya coefficient becomes the squared state overlap $|\langle \psi_A | \psi_B \rangle|^2$, which can be estimated by the SWAP test [8]. This connection is also related to quantum fingerprinting and quantum hashing, where short quantum states are used to represent classical inputs for comparison and communication tasks [2].

Moreover, quantum amplitude estimation (QAE) can estimate the probabilities produced by the SWAP test more efficiently than repeated sampling [7]: whereas naïve sampling requires $O(1/\varepsilon^2)$ repetitions to achieve additive error ε, QAE reduces this dependence to $O(1/\varepsilon)$ coherent queries.

Quantum nearest-neighbor and similarity-search algorithms have been studied previously in the quantum machine-learning literature. Wiebe et al. [28] developed coherent inner-product- and Euclidean-distance-estimation procedures for nearest-neighbor classification and clustering. Basheer et al. [4] considered a fidelity-based quantum k-nearest-neighbor algorithm and reduced the selection of the nearest states to quantum k-maxima finding. These works primarily address query-to-database search, in which a fixed test item is compared with a collection of training items. In contrast, our setting is an all-pairs closest-pair problem over a collection of classical biological sequences, with no distinguished query sequence.

Quantum search with imperfect predicates has also been studied in several related models. Høyer, Mosca, and de Wolf showed that bounded-error verifiers can be incorporated into quantum search without losing the quadratic query speedup, using a dedicated construction that interleaves amplitude amplification and error reduction [16]. Quek et al. [21] studied quantum minimum finding with an imprecise comparator, in which comparisons are guaran-

teed to be correct only when the compared values are separated by more than a prescribed resolution parameter. The latter model is particularly relevant to our QAE-based comparison circuit, because additive-error score estimates may not reliably distinguish pair scores that lie within the estimation precision.

In this work, we present a quantum framework for closest-pair search in biological sequence collections using the squared Bhattacharyya coefficient as the similarity score. Our central technical contribution is a coherent pairwise-score estimation circuit, together with a sparse prefix-tree construction for preparing the corresponding distribution states.

At the oracle-analysis level, we combine coherent indexed loading with Dürr–Høyer/BBHT maximum finding. Because a single bounded-error QAE execution does not by itself induce a fixed execution-wide comparison order, we state the main theorem under an explicit consistent ε-resolved coherent-comparison-oracle assumption. This assumption requires a fixed total order over the legal pairs that preserves every true score difference larger than ε and admits an efficient clean coherent implementation. Under this assumption, the procedure achieves additive-ε recovery using $O(N)$ expected comparison-oracle calls.

With idealized qRAM-style data access, the resulting conditional sequential gate complexity is $\widetilde{O}\left(\frac{NL}{\varepsilon}\right)$. For constant precision, this gives a quadratic reduction in the candidate-pair search component relative to the $O(N^2L)$ classical exhaustive baseline, conditional on the stated oracle and data-access assumptions.

We also implement a proof-of-concept version of the method in Q# [27]. The implementation uses finite-precision QAE, explicit multiplexed indexed loading, Grover-style search, and a Python controller that classically verifies measured candidates before updating the current candidate. The loading construction is related to probability-tree state-preparation methods [15, 13, 22]. The resulting workflow provides circuit-level validation of the implemented loading, estimation, marking, and search components.

Combining explicit multiplexed loading with the efficient consistent-order implementation assumed in the oracle-level analysis gives the conditional bound $\widetilde{O}\left(\frac{N^2L}{\varepsilon}\right)$, which, for constant precision, matches the $O(N^2L)$ classical exhaustive scaling up to polylogarithmic factors. The proof-of-concept workflow uses classically verified candidate updates and therefore differs from the fixed-order oracle procedure analyzed in the recovery theorem.

This is the first quantum framework specifically targeting alignment-free closest-pair search over biological sequence collections, and it provides both an oracle-level analysis and a circuit-level implementation platform for k-mer-distribution-based quantum sequence search.

2 Method

The method consists of four main components: (i) encode each sequence using the square roots of its empirical k-mer distribution; (ii) prepare pair-index superpositions and coherently load the corresponding sequence states; (iii) estimate pairwise similarities using a SWAP-test-based quantum amplitude estimation procedure; and (iv) apply Dürr–Høyer/BBHT maximum finding under a consistent ε-resolved coherent-comparison-oracle model.

2.1 Problem formulation and quantum state encoding

Let $S_1, S_2, \ldots, S_N$ be a collection of biological sequences over an alphabet Σ. Let $\mathcal{K} = \bigcup_{i=1}^{N} \{k\text{-mers appearing in } S_i\}$ be the set of distinct k-mers appearing in the input sequences, and write $M = |\mathcal{K}|$ for the size of this global observed k-mer basis.

For a sequence X of length $L_X \geq k$, let $c_X(x)$ denote the number of occurrences of $x \in \mathcal{K}$ among the $L_X - k + 1$ consecutive length-k windows of X. The empirical k-mer distribution of X is

$$P_X(x) = \frac{c_X(x)}{L_X - k + 1}, \qquad x \in \mathcal{K}.$$

For two sequences A and B, their Bhattacharyya coefficient is

$$\mathrm{BC}(A, B) = \sum_{x \in \mathcal{K}} \sqrt{P_A(x) P_B(x)}.$$

We use the squared Bhattacharyya coefficient as the similarity score, $F(A, B) = \mathrm{BC}(A, B)^2$. The closest-pair problem is therefore $(i^*, j^*) = \mathrm{argmax}_{1 \leq i < j \leq N} F(S_i, S_j)$.

We use a quantum register to mean a named collection of qubits that stores one part of the algorithmic state. To each sequence X, we associate the quantum state $|\psi_X\rangle = \sum_{x \in \mathcal{K}} \sqrt{P_X(x)}\,|x\rangle$. Since all amplitudes are real and nonnegative, the inner product between two such states is

$$\langle \psi_A | \psi_B \rangle = \sum_{x \in \mathcal{K}} \sqrt{P_A(x) P_B(x)} = \mathrm{BC}(A, B).$$

Therefore, $F(A, B) = |\langle \psi_A | \psi_B \rangle|^2$. Thus, closest-pair search under F is equivalent to finding the encoded sequence pair with maximum squared state overlap.

2.2 Coherent indexed state loading

The maximum-finding procedure requires coherent access to the encoded sequence states. We denote the corresponding indexed-loading operation by $U_{\mathrm{load}} : |i\rangle|0\rangle \longmapsto |i\rangle|\psi_{S_i}\rangle$, and write C_{load} for its sequential gate cost.

The operation U_{load}, its inverse $U_{\mathrm{load}}^\dagger$, and their controlled versions must act coherently when the index register is in superposition. Below, we give an explicit multiplexed construction realizing this operation. For the improved qRAM-based complexity bound, we separately introduce an idealized indexed-access assumption.

Given an index register in a superposition $\sum_i \alpha_i |i\rangle$, linearity gives $\sum_i \alpha_i |i\rangle|0\rangle \longmapsto \sum_i \alpha_i |i\rangle|\psi_{S_i}\rangle$. For a candidate pair (i, j), the loading operation is applied twice:

$$|i, j\rangle|0\rangle|0\rangle \longmapsto |i, j\rangle|\psi_{S_i}\rangle|\psi_{S_j}\rangle.$$

2.2.1 Pair-index search state

Our method searches over candidate sequence pairs by representing the pair indices in quantum registers. We prepare a uniform superposition over the ordered index space $\mathcal{R} = \{1, \ldots, N\} \times \{1, \ldots, N\}$. This gives the pair-index state

$$|\Psi_{\mathcal{R}}\rangle = \left(\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i\rangle \right) \otimes \left(\frac{1}{\sqrt{N}} \sum_{j=1}^{N} |j\rangle \right) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} |i, j\rangle.$$

2.2.2 Sparse prefix-tree preparation of a single k-mer distribution

For each sequence X, the implementation prepares the state $|\psi_X\rangle = \sum_{x \in \mathcal{K}} \sqrt{P_X(x)}\,|x\rangle$ from its empirical k-mer distribution P_X using a sparse binary prefix-tree construction.

Let $d = \lceil \log_2 M \rceil$, and identify each k-mer basis element in $\mathcal{K}$ with a d-bit label, padding the basis to dimension 2^d if necessary. The preparation circuit is defined over the binary prefix tree of these d-bit labels.

For a prefix $u \in \{0,1\}^\ell$, define the probability mass $\mu_X(u) = \sum_{x:\, x \text{ has prefix } u} P_X(x)$. Thus $\mu_X(u)$ is the total probability mass of all k-mers whose binary labels begin with u. At an internal prefix node u, the mass splits between its two children u_0 and u_1. Conditional on the register already having prefix u, the next qubit should be prepared with probabilities

$$\Pr(0 \mid u) = \frac{\mu_X(u_0)}{\mu_X(u)}, \qquad \Pr(1 \mid u) = \frac{\mu_X(u_1)}{\mu_X(u)}.$$

Using the convention $R_y(\theta)|0\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)|1\rangle$, this conditional split is implemented by a prefix-controlled R_y rotation on the next qubit with angle $\theta_u = 2 \arcsin \sqrt{\frac{\mu_X(u_1)}{\mu_X(u)}}$. After this rotation, the branch u_0 receives amplitude factor $\sqrt{\frac{\mu_X(u_0)}{\mu_X(u)}}$, and the branch u_1 receives amplitude factor $\sqrt{\frac{\mu_X(u_1)}{\mu_X(u)}}$.

Applying these rotations from the root to the leaves prepares the target state. For a leaf $x = x_1 x_2 \cdots x_d$, the amplitude accumulated along its root-to-leaf path is

$$\prod_{\ell=0}^{d-1} \sqrt{\frac{\mu_X(x_1 \cdots x_{\ell+1})}{\mu_X(x_1 \cdots x_\ell)}} = \sqrt{\mu_X(x)} = \sqrt{P_X(x)},$$

where the product telescopes and $\mu_X(\emptyset) = 1$. Therefore the prepared state has exactly the desired square-root amplitudes.

▶ **Assumption 1** (Idealized qRAM-style indexed loading)**.** *For a sequence index i and a position in the sparse state-preparation schedule of S_i, the corresponding rotation data can be queried coherently with overhead*

$$\eta_{\text{idx}} = \text{polylog}(N, L, M).$$

Under this access model, the indexed-loading operation U_{load}, its inverse, and their controlled versions can be implemented with sequential gate cost

$$C_{\text{load}}^{\text{qRAM}} = O\big(L \log M \, (\eta_{\text{idx}} + \log M)\big) = \widetilde{O}(L).$$

2.2.3 Explicit indexed loading from sparse state preparations

We now give an explicit non-qRAM construction of the coherent indexed-loading operation U_{load} used in the Q# implementation. This construction does not require Assumption 1.

Let P_t denote the sparse prefix-tree preparation circuit for sequence S_t. The indexed loader iterates over $t \in \{1, \ldots, N\}$, reversibly checks whether the index register equals t, and conditionally applies P_t to the sequence-state register. Thus, $|t\rangle|0\rangle \longmapsto |t\rangle|\psi_{S_t}\rangle$ for every computational-basis index t, and the same operation acts coherently on superpositions by linearity.

For a candidate pair, the construction is applied independently to the two sequence indices: $|i,j\rangle|0\rangle|0\rangle \longmapsto |i,j\rangle|\psi_{S_i}\rangle|\psi_{S_j}\rangle$.

Because the construction explicitly multiplexes over all stored preparation circuits, its indexed-loading cost depends on the total preparation-schedule size over all N sequences. This cost is derived in Section 3.

2.3 Coherent pairwise-score estimation

After indexed loading, each pair-index basis state $|i,j\rangle$ is associated with the sequence states $|\psi_{S_i}\rangle$ and $|\psi_{S_j}\rangle$. By the square-root amplitude encoding, $\langle\psi_{S_i} \mid \psi_{S_j}\rangle = \mathrm{BC}(S_i,S_j)$, and hence $F(S_i,S_j) = \mathrm{BC}(S_i,S_j)^2 = \left|\langle\psi_{S_i} \mid \psi_{S_j}\rangle\right|^2$.

To access this score coherently, we apply a SWAP test to the two loaded sequence-state registers. For arbitrary pure states $|\phi\rangle$ and $|\varphi\rangle$, the probability of measuring the SWAP-test ancilla in state $|0\rangle$ is $p = \frac{1+|\langle\phi|\varphi\rangle|^2}{2}$. Therefore, for pair (i,j), $p_{ij} = \frac{1+\left|\langle\psi_{S_i}|\psi_{S_j}\rangle\right|^2}{2} = \frac{1+F(S_i,S_j)}{2}$, or equivalently, $F(S_i,S_j) = 2p_{ij} - 1$.

Let A_{ij} denote the coherent circuit that loads the two sequence states and applies the SWAP test. It prepares a state of the form $A_{ij}|0\rangle = \sqrt{p_{ij}}\,|\mathrm{good}_{ij}\rangle + \sqrt{1-p_{ij}}\,|\mathrm{bad}_{ij}\rangle$, where the good subspace is defined by the SWAP-test ancilla being in state $|0\rangle$.

Over a superposition of pair indices, the corresponding controlled operation is $A = \sum_{i,j} |i,j\rangle\langle i,j| \otimes A_{ij}$.

Let $\varepsilon \in (0,1)$ denote the desired final additive accuracy of the returned pair score, and define the internal score-estimation accuracy $\xi = \frac{\varepsilon}{2}$. Quantum amplitude estimation is used to estimate p_{ij} to additive error at most $\xi/2$. A measurement of the resulting estimate register gives a score estimate $\widetilde{F}_{ij} = 2\widetilde{p}_{ij} - 1$ satisfying $\left|\widetilde{F}_{ij} - F(S_i,S_j)\right| \le \xi$ with constant success probability.

Obtaining this precision requires $O\left(\frac{1}{\xi}\right) = O\left(\frac{1}{\varepsilon}\right)$ coherent applications of A, $A^\dagger$, and the associated reflections.

A single bounded-error QAE execution does not by itself define the execution-wide deterministic comparison order required by the standard Dürr–Høyer analysis. In particular, repeated coherent executions need not induce the same ordering among candidate pairs whose scores are close, and the resulting marking operation is not automatically a clean Boolean phase oracle on the pair-index register.

We therefore separate the coherent score-estimation construction from the oracle-level maximum-finding analysis. The QAE construction shows that additive score estimation uses $O\left(\frac{1}{\varepsilon}\right)$ applications of the overlap-estimation block and motivates the cost assumed for one coherent comparison. The main theorem, however, is stated under the additional efficient-consistent-order assumption of Assumption 3. That assumption requires one fixed deterministic comparison order, a clean coherent implementation of that order, and the stated implementation cost.

Establishing Assumption 3 directly from the bounded-error QAE circuit would require an additional construction ensuring execution-wide consistency and clean uncomputation. Such a construction is outside the scope of the present work and is not implemented in the Q# proof of concept. An alternative route would be to analyze the QAE-based comparison procedure within a margin-based imprecise-comparator model and apply robust quantum minimum- or maximum-finding techniques. Such an analysis may introduce an additional dependence on the number of candidate scores lying within the comparison margin and is likewise left for future work.

2.4 Consistent comparison oracle and quantum maximum finding

Within the ordered pair-index space $\mathcal{R}$ defined above, the legal closest-pair candidates are the canonically oriented unordered pairs $\mathcal{P} = \{(i,j) : 1 \le i < j \le N\}$, $|\mathcal{P}| = \binom{N}{2}$. The legality condition $i < j$ is checked reversibly inside the marking circuit, thereby excluding self-pairs and reversed duplicate pairs.

For a current classical candidate $y \in \mathcal{P}$, the maximum-finding procedure requires a coherent comparison operation that marks pairs ranked above y. We formalize this requirement in the following definition.

▶ **Definition 2** (Consistent ε-resolved comparison oracle). *A family of coherent phase oracles* $\left\{O_y^{(\varepsilon)} : y \in \mathcal{P}\right\}$ *is a consistent ε-resolved comparison-oracle family if there exists a strict total order $\succ_\varepsilon$ on $\mathcal{P}$ such that $O_y^{(\varepsilon)}|x\rangle = (-1)^{[x \succ_\varepsilon y]}|x\rangle$ for every $x, y \in \mathcal{P}$, with all score-estimation, loading, and comparison work registers returned to their initial states.*

The order must preserve every score difference larger than ε: $F(x) > F(y) + \varepsilon \implies x \succ_\varepsilon y$. Pairs whose scores differ by at most ε may be ordered arbitrarily, provided that the same total order is used consistently throughout the execution.

▶ **Assumption 3** (Efficient coherent implementation of a consistent ε-resolved order). *There exists a fixed deterministic ordering key $K_\varepsilon : \mathcal{P} \longrightarrow \{0,1\}^b$ whose lexicographic order induces a strict total order $\succ_\varepsilon$ on the legal candidate pairs. The same key and the same tie-breaking rule are used throughout the entire execution.*

Without loss of generality, the key is injective and has $b = O(\log N + \log(1/\varepsilon))$ bits, including a deterministic pair-index tie-breaking component. We additionally assume that U_{K_ε} and the reversible comparison circuit use $q_K = \mathrm{polylog}(N, L, M, 1/\varepsilon)$ work qubits.

The induced order preserves every score difference larger than ε:

$$F(x) > F(y) + \varepsilon \quad \Longrightarrow \quad K_\varepsilon(x) >_{\mathrm{lex}} K_\varepsilon(y),$$

and hence $x \succ_\varepsilon y$.

We assume that the key can be evaluated coherently by a reversible circuit $U_{K_\varepsilon} : |x\rangle|0\rangle \longmapsto |x\rangle|K_\varepsilon(x)\rangle$, and that U_{K_ε}, its inverse, and their controlled versions have sequential gate cost $\widetilde{O}\left(\frac{C_{\mathrm{load}} + \log M}{\varepsilon}\right)$. Using a reversible lexicographic comparison followed by uncomputation, this circuit implements the clean phase oracle $O_y^{(\varepsilon)}|x\rangle = (-1)^{[x \succ_\varepsilon y]}|x\rangle$, with all key, loading, score-estimation, and comparison work registers returned to their initial states.

This assumption includes both the execution-wide consistency of the comparison order and the existence of an efficient clean coherent implementation of that order. It is not implied by the bounded-error guarantee of a single QAE execution. The present Q# proof-of-concept implementation does not construct U_{K_ε} or establish this assumption.

Under Assumption 3, the Dürr–Høyer procedure is applied to the fixed total order $\succ_\varepsilon$. For the current classical candidate y, the comparison oracle marks precisely the legal pairs x satisfying $x \succ_\varepsilon y$. The number of marked pairs is not known in advance, so each search step uses a BBHT-style randomized Grover-iteration schedule [6].

Whenever the oracle-level search returns a marked pair x, the current candidate is updated to x. Thus the candidate rank improves strictly according to the fixed order $\succ_\varepsilon$, as required by the standard Dürr–Høyer analysis.

The Q# proof-of-concept workflow described in Section 4 instead evaluates the measured pair classically and accepts it only when its true score improves. That hybrid verification rule is useful experimentally, but it defines a different candidate-update procedure and is not covered by the oracle-level recovery guarantee below.

▶ **Theorem 4** (Main result). *Under the efficient coherent comparison-order implementation assumed in Assumption 3, let $S_1, \ldots, S_N$ be biological sequences of length at most L, and define $F^\star = \max_{1 \le i < j \le N} F(S_i, S_j)$.*

The Dürr–Høyer maximum-finding procedure returns, with probability at least $2/3$*, a legal pair* $\widehat{y} = (\widehat{i}, \widehat{j})$ *satisfying* $F(S_{\widehat{i}}, S_{\widehat{j}}) \geq F^\star - \varepsilon$.

Under the implementation-cost component of Assumption 3, the expected conditional sequential gate complexity is $\widetilde{O}\left(\frac{N(C_{\mathrm{load}}+\log M)}{\varepsilon}\right)$.

Combining Assumption 3 with the idealized qRAM-style loading Assumption 1 gives the conditional bound $\widetilde{O}\left(\frac{NL}{\varepsilon}\right)$. *Combining the same comparison-order implementation assumption with the explicit multiplexed loading cost gives* $\widetilde{O}\left(\frac{N^2 L}{\varepsilon}\right)$.

Proof. Because each successful update replaces the current candidate by a pair that is strictly larger under $\succ_\varepsilon$, the candidate-update process is exactly the Dürr–Høyer procedure applied to this fixed total order. By Assumption 3, the comparison operations used throughout the execution are induced by one fixed strict total order $\succ_\varepsilon$. The standard Dürr–Høyer maximum-finding analysis therefore applies to this order. Since $|\mathcal{P}| = \binom{N}{2} = \Theta(N^2)$, the procedure finds a maximal element $\widehat{y}$ of $\succ_\varepsilon$ with constant success probability using $O\left(\sqrt{|\mathcal{P}|}\right) = O(N)$ comparison-oracle applications in expectation. A constant number of independent repetitions raises the success probability to at least $2/3$.

Let $x^\star$ be a true optimal pair. If $F(\widehat{y}) < F^\star - \varepsilon$, then $F(x^\star) > F(\widehat{y}) + \varepsilon$. Definition 2 would therefore imply $x^\star \succ_\varepsilon \widehat{y}$, contradicting the maximality of $\widehat{y}$. Hence $F(\widehat{y}) \geq F^\star - \varepsilon$.

By Assumption 3, one comparison-oracle application costs

$$O\left(\frac{C_{\mathrm{load}} + \log M}{\varepsilon}\right)$$

up to polylogarithmic overheads. Multiplying by the $O(N)$ expected oracle applications gives $\widetilde{O}\left(\frac{N(C_{\mathrm{load}}+\log M)}{\varepsilon}\right)$. Substituting the two indexed-loading costs gives the stated qRAM and explicit-loading bounds. ◀

2.4.1 Exact recovery under a score-gap condition

If a strictly suboptimal score exists, let $F^{(2)}$ denote the largest score strictly below $F^\star$. If every legal pair is optimal, exact recovery is immediate.

▶ **Assumption 5** (Optimal-score gap). *The optimal and strictly suboptimal scores satisfy* $F^\star - F^{(2)} > \varepsilon$.

▶ **Corollary 6** (Exact recovery). *Under the assumptions of Theorem 4 and Assumption 5, the returned pair is exactly optimal whenever the maximum-finding procedure succeeds.*

Proof. Let $x^\star$ be any optimal pair and let y be strictly suboptimal. Then $F(x^\star) - F(y) \geq F^\star - F^{(2)} > \varepsilon$. Definition 2 therefore gives $x^\star \succ_\varepsilon y$. No strictly suboptimal pair can be maximal in $\succ_\varepsilon$, so every maximal element returned by the maximum-finding procedure is optimal. ◀

3 Complexity analysis and comparison

The complexity bounds in this section are conditional on Assumption 3. We first express the quantum cost in terms of the abstract indexed-loading cost C_{load}, and then instantiate it under the idealized qRAM-style and explicit multiplexed loading models. We also derive the classical preprocessing cost and logical qubit count.

Throughout this section, the alphabet Σ is treated as fixed, while the k-mer length k, and hence the observed basis size $M \leq |\Sigma|^k$, may vary with the input. Thus extracting the overlapping k-mers of a length-L sequence takes $O(L)$ word operations. We retain the dependence on the observed k-mer basis size $M = |\mathcal{K}|$, with $M \leq |\Sigma|^k$, because M determines the dimension of the encoded states and the size of the k-mer basis register.

Let s_i be the number of distinct observed k-mers in sequence S_i. If all sequences have length at most L, then $s_i \leq L_i - k + 1 \leq L - k + 1 = O(L)$.

3.1 Classical exhaustive baseline

An exact classical closest-pair search evaluates the similarity score for all $\binom{N}{2} = \Theta(N^2)$ candidate pairs. Using a hashmap for the k-mer counts, the Bhattacharyya coefficient between two sequences can be computed by iterating over the nonzero k-mer counts of one sequence and looking up the corresponding counts in the other. Thus one pairwise comparison costs $O(L)$ time in the worst case, and the classical exhaustive baseline is $T_{\text{classical}} = O(N^2 L)$.

3.2 Classical preprocessing

Before running the quantum circuit, we compute k-mer counts for each input sequence, construct the global observed k-mer basis $\mathcal{K}$, normalize each sequence into an empirical distribution, and construct the square-root amplitude representation. Extracting k-mer counts takes $O(NL)$ time.

In addition, the implementation constructs a sparse state-preparation schedule for each sequence. By a state-preparation schedule, we mean the classical list of prefix-controlled R_y rotations generated by the sparse prefix-tree construction. Let R_i denote the number of rotations in the schedule for sequence S_i, and let $R_{\text{tot}} = \sum_{i=1}^{N} R_i$ be the total schedule size over all sequences. The schedule construction uses cached prefix probability masses in the binary preparation tree. For sequence S_i, the schedule size satisfies $R_i = O(s_i \log M)$, and the schedule can be constructed in $O(s_i \log M)$ time. Therefore, $R_{\text{tot}} = O\left(\sum_{i=1}^{N} s_i \log M\right)$. Since $s_i = O(L)$, we have

$$R_{\text{tot}} = O(NL \log M). \tag{1}$$

Thus the preprocessing time is $O(NL \log M)$.

3.3 Indexed loading cost

Let C_{load} denote the gate cost of coherently loading one indexed sequence state, $U_{\text{load}} : |i\rangle|0\rangle \mapsto |i\rangle|\psi_{S_i}\rangle$. We keep C_{load} abstract first, and later substitute the cost under a qRAM-style model and under our concrete implementation.

For a queried pair (i, j), the SWAP-test block applies U_{load} twice, once for i and once for j, and then performs a controlled-SWAP between the two sequence-state registers. The sequence-state register has $d = \lceil \log_2 M \rceil$ qubits, so the controlled-SWAP part costs $O(\log M)$. Therefore one coherent overlap-estimation block costs $O(C_{\text{load}} + \log M)$, where the factor of two from loading two states is absorbed into the big-O notation.

3.4 Cost of one comparison-oracle application

The following oracle cost is an explicit component of Assumption 3.

The coherent score estimator uses the internal precision $\xi = \frac{\varepsilon}{2}$. Estimating the SWAP-test probability to additive error $\xi/2$ requires $O\left(\frac{1}{\xi}\right) = O\left(\frac{1}{\varepsilon}\right)$ coherent applications of the overlap-estimation block and its inverse. Thus the score-estimation component of one threshold-comparison query costs $O\left(\frac{C_{\text{load}}+\log M}{\varepsilon}\right)$.

The threshold-search circuit also prepares and reflects about the ordered pair-index search state $|\Psi_{\mathcal{R}}\rangle = |u_N\rangle \otimes |u_N\rangle$, $|u_N\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$. Here the computational-basis label $i \in \{0, \ldots, N-1\}$ corresponds to sequence S_{i+1}.

We prepare $|u_N\rangle$ exactly by a recursive binary decomposition. Let $m = \lceil \log_2 N \rceil$. If $N = 2^m$, then $|u_N\rangle = H^{\otimes m}|0^m\rangle$. Otherwise, write $N = 2^{m-1} + r$, $1 \le r < 2^{m-1}$. The most significant qubit is first prepared as $\sqrt{\frac{2^{m-1}}{N}}\,|0\rangle + \sqrt{\frac{r}{N}}\,|1\rangle$ by applying $R_y\left(2 \arcsin \sqrt{\frac{r}{N}}\right)$. Conditioned on the most significant qubit being 0, Hadamard gates are applied to the remaining $m-1$ qubits. Conditioned on it being 1, the same construction is applied recursively to prepare $|u_r\rangle$ on the remaining qubits. The resulting state is

$$\sqrt{\frac{2^{m-1}}{N}}\,|0\rangle \left(\frac{1}{\sqrt{2^{m-1}}} \sum_{x=0}^{2^{m-1}-1} |x\rangle\right) + \sqrt{\frac{r}{N}}\,|1\rangle \left(\frac{1}{\sqrt{r}} \sum_{x=0}^{r-1} |x\rangle\right) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle.$$

At each of the $m = O(\log N)$ recursion levels, the construction uses at most $O(m)$ controlled one- and two-qubit operations. Therefore, using standard controlled-gate decompositions, one preparation of $|u_N\rangle$ requires $O(\log^2 N)$ elementary gates. Preparing the two-register product state $|\Psi_{\mathcal{R}}\rangle$ has the same asymptotic cost.

Let $U_N|0^m\rangle = |u_N\rangle$. Reflection about the pair-index search state is implemented as

$$(U_N \otimes U_N)\left(2|0^{2m}\rangle\langle 0^{2m}| - I\right)(U_N^\dagger \otimes U_N^\dagger).$$

The middle operation is a multi-controlled phase on $2m = O(\log N)$ qubits and has polynomial-in-$\log N$ gate cost under standard decompositions. Thus both preparation of and reflection about $|\Psi_{\mathcal{R}}\rangle$ cost $O(\log^2 N)$ gates, up to the chosen ancilla–gate tradeoff. The reversible validity test $i < j$ can be implemented by a standard m-bit comparator using $O(\log N)$ gates and work qubits.

Hence, under Assumption 3, one comparison-oracle application has cost $C_{\text{query}} = O\left(\frac{C_{\text{load}}+\log M}{\varepsilon}\right)$ up to logarithmic overheads from pair-index preparation, reflection, and reversible comparisons. Accordingly, throughout the oracle-level complexity analysis we assume $C_{\text{query}} = \widetilde{O}\left(\frac{C_{\text{load}}+\log M}{\varepsilon}\right)$. This bound is conditional on the efficient coherent implementation postulated in Assumption 3.

3.5 Number of comparison-oracle queries

By Theorem 4, maximum finding over $|\mathcal{P}| = \Theta(N^2)$ legal pairs uses $O(N)$ expected comparison-oracle applications. Combining this with $C_{\text{query}} = \widetilde{O}\left(\frac{C_{\text{load}}+\log M}{\varepsilon}\right)$ gives

$$T_{\text{abstract}} = \widetilde{O}\left(\frac{N(C_{\text{load}} + \log M)}{\varepsilon}\right).$$

3.6 Number of qubits

Let $m = \lceil \log_2 N \rceil, d = \lceil \log_2 M \rceil, r = \lceil \log_2(1/\varepsilon) \rceil$ be the sequence-index register width, the k-mer basis register width, and the QAE evaluation-register width, respectively. The circuit uses two index registers, two sequence-state registers, one QAE evaluation register, and a

constant number of ancilla qubits for the SWAP-test and marking logic. Ignoring temporary work qubits used to decompose arithmetic and multi-controlled gates, the logical qubit count is $Q_{\text{logical}} = 2m + 2d + r + O(1)$. Including reversible comparison and gate-decomposition workspace, the qubit count is $Q = O(\log N + \log M + \log(1/\varepsilon))$. The sparse state-preparation schedules are classical data supplied to the circuit, and are not counted as qubits.

3.7 Total complexity under idealized qRAM-style data access

Under Assumption 1, the indexed-loading operation has sequential gate cost $C_{\text{load}}^{\text{qRAM}} = O\big(L \log M \left(\eta_{\text{idx}} + \log M\right)\big) = \widetilde{O}(L)$.

Substituting this loading cost into the comparison-oracle cost bound gives $C_{\text{query}}^{\text{qRAM}} = O\left(\frac{C_{\text{load}}^{\text{qRAM}} + \log M}{\varepsilon}\right)$. Hence, $C_{\text{query}}^{\text{qRAM}} = O\left(\frac{L \log M \left(\eta_{\text{idx}} + \log M\right)}{\varepsilon}\right) = \widetilde{O}\left(\frac{L}{\varepsilon}\right)$.

Under Assumption 3, the Dürr–Høyer procedure uses $O(N)$ expected comparison-oracle applications over the $\Theta(N^2)$ legal candidate pairs. Therefore, the total sequential gate complexity is

$$T_{\text{qRAM}} = O\big(N C_{\text{query}}^{\text{qRAM}}\big) = O\left(\frac{NL \log M \left(\eta_{\text{idx}} + \log M\right)}{\varepsilon}\right) = \widetilde{O}\left(\frac{NL}{\varepsilon}\right).$$

For constant precision, the leading dependence on the number of sequences is linear rather than quadratic. Compared with the classical exhaustive baseline $T_{\text{classical}} = O(N^2 L)$, this reduces the number of comparison-oracle applications from $\Theta(N^2)$ classically to $O(N)$ quantumly, conditional on the availability of the assumed comparison oracle.

3.8 Total complexity under explicit non-qRAM loading

For the explicit multiplexed construction of Section 2.2, one indexed loading operation applies the stored preparation schedules conditionally over all sequence indices. Its cost is therefore $C_{\text{load}}^{\text{impl}} = O(R_{\text{tot}}(\log M + \log N))$.

Using Eq. (1), this becomes $C_{\text{load}}^{\text{impl}} = O(NL \log M(\log M + \log N))$.

Substituting this loading cost into the comparison-oracle cost bound gives $C_{\text{query}}^{\text{impl}} = O\left(\frac{NL \log M(\log M + \log N)}{\varepsilon}\right)$.

Under Assumption 3, the Dürr–Høyer procedure uses $O(N)$ expected comparison-oracle applications. Therefore, the total sequential gate complexity is

$$T_{\text{impl}} = O\big(N C_{\text{query}}^{\text{impl}}\big) = O\left(\frac{N^2 L \log M(\log M + \log N)}{\varepsilon}\right) = \widetilde{O}\left(\frac{N^2 L}{\varepsilon}\right).$$

For constant precision, this matches the classical exhaustive baseline $T_{\text{classical}} = O(N^2 L)$ up to polylogarithmic factors. Thus the explicit non-qRAM construction makes the coherent indexed-loading cost explicit, but does not provide an end-to-end asymptotic speedup over exhaustive classical comparison under the stated comparison-oracle model.

4 Implementation and simulation

We implemented a proof-of-concept version of the method in Q# with a Python driver to validate the circuit-level construction. The source code, experiment scripts, and data-processing utilities are available at `https://github.com/Kingsford-Group/tieria`.

4.1 Implementation details

The implementation realizes the circuit-level components described in Section 2. A Python driver performs the classical preprocessing: it extracts observed k-mers, constructs the global basis $\mathcal{K}$, computes empirical k-mer distributions, and generates the sparse prefix-tree preparation schedules used by the Q# circuit.

The Q# component implements pair-index state preparation, multiplexed indexed loading from sparse schedules, SWAP-test-based QAE, threshold marking, and Grover-style search. The indexed loading operation uses the explicit non-qRAM construction analyzed in Section 3; that is, the circuit multiplexes over the stored sequence schedules rather than assuming qRAM access.

The outer candidate-update loop is controlled by the Python driver. At each iteration, the driver stores the current candidate pair and computes its threshold score from the preprocessed amplitude vectors. The Q# circuit then performs a finite-precision BBHT-style search, and the driver accepts a measured pair only when its classically evaluated score is strictly larger than that of the current candidate.

This hybrid update rule is not the oracle-order update rule used in Theorem 4. The oracle-level theorem updates according to a fixed total order $\succ_\varepsilon$, whereas the proof-of-concept implementation updates according to classically verified true-score improvement.

The implementation therefore exercises the explicit loading, SWAP-test-based estimation, finite-precision marking, and Grover-style search components, but it neither constructs the deterministic key K_ε nor establishes Assumption 3. The simulation results are reported only as small-scale empirical validation of the implemented workflow and do not establish the theorem's additive-ε or exact-recovery guarantee.

4.2 Simulation setup

We evaluate the proof-of-concept implementation in two complementary ways. First, we use the Q# simulator to test whether the search procedure can recover an exact closest pair found by classical exhaustive enumeration on small random DNA instances. Second, we estimate logical circuit resources for the generated coherent search circuits.

We generated random DNA sequence collections over the alphabet $\Sigma = \{A, C, G, T\}$. The experiments vary the k-mer length k, sequence length L, and number of sequences N. We use one-parameter sweeps designed to isolate the effects of k, L, and N:

$$
\begin{aligned}
&k\text{-sweep:} && (k, L, N) \in \{(2, 12, 3), (3, 12, 3), (4, 12, 3)\},\\
&L\text{-sweep:} && (k, L, N) \in \{(2, 5, 3), (2, 8, 3), (2, 12, 3), (2, 16, 3)\},\\
&N\text{-sweep:} && (k, L, N) \in \{(2, 8, 3), (2, 8, 4), (2, 8, 5), (2, 8, 6), (2, 8, 7), (2, 8, 8)\}.
\end{aligned}
$$

We also include the additional settings $(k, L, N) \in \{(3, 16, 3), (4, 16, 3)\}$ to test larger k-mer representations at the longer sequence length $L = 16$.

For the Q# simulator runs, we use $r = 3$ QAE precision qubits, a BBHT growth factor of 1.2, and a maximum cumulative Grover-iteration budget of 120. With 3 QAE evaluation qubits, the nominal resolution of the QAE output is $2^{-r} = 1/8$. Thus, these experiments use only coarse score estimation. The initial candidate pair is chosen in a nonoptimal mode whenever possible, so that the threshold-search procedure must improve the current candidate. For each random instance, we compute the exact optimal pair classically by exhaustive enumeration over all legal pairs. A run is counted as successful if the first optimal pair observed by the Q# search is one of the exact optimal pairs, allowing for ties.

For correctness, we report the success rate over random trials. For search-level behavior, we report the number of Q# circuit calls and the cumulative number of Grover iterations used before the first exact optimal pair is observed. These quantities characterize the time-to-first-optimal behavior of the Dürr–Høyer/BBHT threshold-search procedure under the configured search budget. We do not report simulator wall-clock time, since it primarily reflects classical state-vector simulation overhead rather than quantum hardware runtime.

4.3 Q# resource estimation

We estimate logical circuit resources using `qsharp.logical_counts`. The reported quantities include logical qubits, logical depth, rotation counts, and T-gate-related counts returned by the Q# toolchain. These estimates are obtained from coherent circuit entry points used for resource counting, rather than from simulator wall-clock time.

To relate these resource counts to the structure of the generated instances, we also record several quantities from the classical preprocessing. These include the observed k-mer basis size $M = |\mathcal{K}|$, the average and maximum nonzero k-mer support size, and the total sparse state-preparation schedule size $R_{\mathrm{tot}} = \sum_{i=1}^{N} R_i$, where R_i is the number of sparse state-preparation schedule entries used for sequence S_i.

4.4 Proof-of-concept circuit validation

Table 1 summarizes the proof-of-concept results for the implemented Python–Q# workflow. Across all 13 parameter settings and 65 random trials, the workflow observed an exact optimal pair before the configured Grover-iteration cap, giving an empirical success rate of 100% on the tested synthetic instances. Exact optimality was determined independently by classical exhaustive enumeration.

These results demonstrate that the implemented circuit subroutines – indexed state loading, SWAP-test-based QAE, threshold marking, and Grover-style search – can be integrated successfully within the hybrid workflow.

For descriptive purposes, the N-sweep at fixed $k = 2$ and $L = 8$ shows how the measured search-level quantities change over the small tested range. The number of legal pairs increases from $\binom{3}{2} = 3$ at $N = 3$ to $\binom{8}{2} = 28$ at $N = 8$. Over the same range, the average number of Q# circuit calls before first observing an exact optimal pair increases from 3.4 to 7.6, while the average cumulative number of Grover iterations increases from 3.6 to 13.0.

Because the experiments use only $N \leq 8$, synthetic random-DNA inputs, and five trials per setting, these measurements are intended only as circuit-level proof-of-concept validation.

4.5 Logical resource estimates

Table 2 reports the logical resource estimates obtained using `qsharp.logical_counts`. The table reports estimates for circuits with $r = 3$ QAE evaluation qubits and one Grover iteration. Each row is averaged over three random trials. All logical-count runs completed successfully.

The logical resource estimates show that circuit resources grow with the structure of the generated state-preparation schedules. In the N-sweep with $k = 2$ and $L = 8$, R_{tot} increases from 22.3 at $N = 3$ to 61.7 at $N = 8$. Over the same sweep, the mean logical depth increases from 1264.3 to 4201.3, and the reported rotation count increases from 1395.0 to 4389.0. This is consistent with the concrete implementation, which performs explicit sequence-indexed loading over the stored sequence states rather than assuming qRAM.

The dependence on k and L is more instance-dependent. Increasing L at fixed $k = 2, N = 3$ increases the support size and sparse schedule size, and the logical depth grows from 668.0 at $L = 5$ to 3398.3 at $L = 16$. For the k-sweep at fixed $L = 12, N = 3$, the observed basis

Table 1 Proof-of-concept results for the Python–Q# workflow on selected synthetic random-DNA instances. Each row reports averages over five random trials. "Classical pairs" is the number of legal pairs $\binom{N}{2}$ evaluated by exhaustive classical closest-pair search. Q# calls and Grover iterations are measured until the workflow first observes an exact optimal pair, with optimality verified independently by classical exhaustive enumeration. Here $M = |\mathcal{K}|$ is the observed k-mer basis size and $R_{\text{tot}} = \sum_i R_i$ is the total sparse state-preparation schedule size over all sequences.

k	L	N	Classical pairs	Q# calls	Grover iters	Success	M	R_{tot}
2	5	3	3	1.6	1.6	5/5	8.2	11.6
2	8	3	3	3.4	3.6	5/5	11.8	21.4
2	8	4	6	5.8	7.2	5/5	14.0	29.8
2	8	5	10	4.2	4.2	5/5	15.0	38.6
2	8	6	15	5.2	5.6	5/5	15.2	45.6
2	8	7	21	5.6	6.4	5/5	15.4	54.4
2	8	8	28	7.6	13.0	5/5	15.4	63.0
2	12	3	3	2.4	2.4	5/5	13.2	26.6
2	16	3	3	2.6	2.6	5/5	15.0	32.8
3	12	3	3	6.4	7.8	5/5	24.8	38.0
3	16	3	3	4.4	5.0	5/5	30.2	52.0
4	12	3	3	3.0	3.2	5/5	25.2	37.4
4	16	3	3	5.6	6.8	5/5	34.6	52.6

size M increases with k, but the logical depth is not monotone. This reflects the fact that, on small random instances, resource counts depend not only on M, but also on the detailed sparse state-preparation schedules generated from the observed k-mer distributions.

4.6 Grover-iteration scaling

Table 3 isolates the cost of increasing the number of Grover iterations for the fixed setting $k = 2, L = 8, N = 4$ and $r = 3$ QAE evaluation qubits. As expected, the logical depth, T-count, and rotation count grow approximately linearly with the number of Grover iterations, since each Grover iteration repeats the marking and diffusion components of the coherent search circuit.

The logical depth increases from 615.0 with no Grover iterations to 5549.7 with four Grover iterations. Similarly, the rotation count increases from 663.0 to 5967.0. This approximately linear growth confirms that the Q# logical resource estimates reflect the repeated application of the Grover search iterate.

5 Conclusion

We introduced a quantum framework for alignment-free closest-pair search in biological sequence collections using distributional k-mer statistics. The central observation is that square-root amplitude encoding maps the Bhattacharyya coefficient between empirical k-mer distributions to the inner product between the corresponding quantum states. This makes the squared Bhattacharyya score accessible through a SWAP-test-based QAE procedure. The procedure is used in the finite-precision marking circuit of our proof-of-concept workflow and motivates the cost assigned to coherent comparisons in the separate oracle-level analysis.

Our main result is an oracle-level theorem. Under a consistent ε-resolved coherent-comparison-oracle assumption, Dürr–Høyer maximum finding returns, with probability at least $2/3$, a pair whose score is at least $F^\star - \varepsilon$, using $O(N)$ expected comparison-oracle applications. An optimal-score gap greater than ε implies exact recovery.

Table 2 Q# logical resource estimates obtained using `qsharp.logical_counts` for circuits with $r = 3$ QAE evaluation qubits and one Grover iteration. Each row reports the mean over three random trials. Here $M = |\mathcal{K}|$ is the observed k-mer basis size, $R_{\mathrm{tot}} = \sum_i R_i$ is the total sparse state-preparation schedule size, and the remaining columns are logical resource quantities reported by Q#.

k	L	N	M	R_{tot}	Qubits	Depth	T-count	Rotations
2	5	3	8.7	12.7	29.7	668.0	978.0	735.0
2	8	3	12.7	22.3	31.0	1264.3	1518.0	1395.0
2	8	4	14.7	31.7	31.0	1848.7	1926.0	1989.0
2	8	5	14.3	38.0	35.0	2323.0	1950.0	2415.0
2	8	6	14.7	47.3	35.0	3175.3	2322.0	3315.0
2	8	7	15.7	54.0	35.0	3483.7	2682.0	3627.0
2	8	8	15.3	61.7	35.0	4201.3	2766.0	4389.0
2	12	3	14.7	31.3	31.0	2322.3	1698.0	2475.0
2	16	3	14.7	33.7	31.0	3398.3	1758.0	3555.0
3	12	3	23.3	39.3	35.0	1931.7	2778.0	2055.0
3	16	3	31.0	54.3	36.3	3292.7	3078.0	3495.0
4	12	3	26.3	38.0	35.0	1609.7	2778.0	1695.0
4	16	3	38.7	59.0	39.0	2864.7	3498.0	3135.0

Table 3 Effect of the number of Grover iterations on Q# logical resource estimates for the setting $k = 2$, $L = 8$, $N = 4$, and $r = 3$ QAE evaluation qubits. The logical depth, T-count, and rotation count grow approximately linearly with the number of Grover iterations.

Grover iters	Logical qubits	Logical depth	T-count	Rotations
0	25.0	615.0	642.0	663.0
1	31.0	1848.7	1926.0	1989.0
2	31.0	3082.3	3210.0	3315.0
4	31.0	5549.7	5778.0	5967.0

The resulting gate bounds are conditional on both the efficient consistent-order implementation assumed in Assumption 3 and the chosen data-access model. Combining this comparison-order assumption with idealized qRAM-style indexed loading gives a sequential gate complexity of $\widetilde{O}\left(\frac{NL}{\varepsilon}\right)$. Combining the same comparison-order assumption with explicit multiplexed indexed loading instead gives $\widetilde{O}\left(\frac{N^2 L}{\varepsilon}\right)$.

The current Q# implementation validates the finite-precision loading, QAE, marking, and Grover-search components, but does not establish the execution-wide consistency assumption used by the main theorem. It should therefore be viewed as a circuit-level proof of concept that makes the coherent loading cost explicit.

Several directions remain for future work. One direction is to give a complete circuit-level realization of the consistent comparison-oracle assumption and to analyze comparison consistency near the additive-resolution boundary. A second direction is to develop more efficient coherent indexed-loading mechanisms, including qRAM-style or hybrid data structures that reduce the cost of loading sparse k-mer distribution states while accounting for routing, coherence, and fault-tolerant overheads [11, 3]. Another direction is to move beyond the black-box maximum-finding formulation by exploiting biological or statistical structure in k-mer distributions, sequence databases, or induced feature spaces. Finally, compact classical

sketches such as MinHash reduce memory and runtime by approximately preserving similarity, and future quantum or hybrid methods could study whether sketch-derived features can be represented as quantum states and compared using overlap-estimation primitives.

Together, these directions suggest a path toward quantum and hybrid algorithms for alignment-free omics search that combine coherent similarity estimation, maximum finding, and practical compressed sequence representations.

References

1 Scott Aaronson, Nai-Hui Chia, Han-Hsuan Lin, Chunhao Wang, and Ruizhe Zhang. On the quantum complexity of closest pair and related problems. In *35th Computational Complexity Conference (CCC 2020)*, volume 169 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:43. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CCC.2020.16.

2 Farid Ablayev, Kamil Khadiev, Alexander Vasiliev, and Mansur Ziiatdinov. Theory and applications of quantum hashing. *Quantum Reports*, 7(2):24, 2025. doi:10.3390/quantum7020024.

3 Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17:123010, 2015. doi:10.1088/1367-2630/17/12/123010.

4 Afrad Basheer, A. Afham, and Sandeep K. Goyal. Quantum k-nearest neighbors algorithm. *arXiv preprint*, 2020. arXiv:2003.09187.

5 Anil Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distribution. *Bulletin of the Calcutta Mathematical Society*, 35:99–110, 1943.

6 Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4–5):493–505, 1998. arXiv:quant-ph/9605034.

7 Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *arXiv preprint quant-ph/0005055*, 2000.

8 Harry Buhrman, Richard Cleve, John Watrous, and Ronald De Wolf. Quantum fingerprinting. *Physical Review Letters*, 87(16):167902, 2001.

9 Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.

10 Robert C. Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461, 2010. doi:10.1093/BIOINFORMATICS/BTQ461.

11 Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Architectures for a quantum random access memory. *Physical Review A*, 78(5):052310, 2008. doi:10.1103/PhysRevA.78.052310.

12 Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16):160501, 2008. doi:10.1103/PhysRevLett.100.160501.

13 Niels Gleinig and Torsten Hoefler. An efficient algorithm for sparse quantum state preparation. In *Proceedings of the 58th ACM/IEEE Design Automation Conference*, pages 433–438, 2021. doi:10.1109/DAC18074.2021.9586240.

14 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC)*, pages 212–219, 1996. doi:10.1145/237814.237866.

15 Lov K. Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint quant-ph/0208112*, 2002.

16 Peter Høyer, Michele Mosca, and Ronald de Wolf. Quantum search on bounded-error inputs. In *Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 291–299. Springer, 2003. doi:10.1007/3-540-45061-0_25.

17 Weizhong Li, Lukasz Jaroszewski, and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006. doi:10.1093/BIOINFORMATICS/BTL158.

18 Guillaume Marçais, Dan DeBlasio, and Carl Kingsford. Sketching methods with small window guarantee using minimum decycling sets. *Journal of Computational Biology*, 31(7):597–615, 2024. doi:10.1089/CMB.2024.0544.

19 Brian D. Ondov, Gregory J. Starrett, Nicholas H. Sappington, et al. Mash Screen: high-throughput sequence containment estimation for genome discovery. *Genome Biology*, 20:232, 2019.

20 Brian D. Ondov, Todd J. Treangen, Péter Melsted, et al. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biology*, 17:132, 2016.

21 Yihui Quek, Clément L. Canonne, and Patrick Rebentrost. Robust quantum minimum finding with an application to hypothesis selection. *arXiv preprint*, 2020. arXiv:2003.11777.

22 Debora Ramacciotti, Andreea-Iulia Lefterovici, and Antonio F. Rotundo. A simple quantum algorithm to efficiently prepare sparse states. *Physical Review A*, 110:032609, 2024.

23 Torbjørn Rognes, Tomáš Flouri, Ben Nichols, Christopher Quince, and Frédéric Mahé. VSEARCH: a versatile open source tool for metagenomics. *PeerJ*, 4:e2584, 2016.

24 Karthik C. S. and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics*, pages 17:1–17:16, 2019. doi:10.4230/LIPIcs.ITCS.2019.17.

25 Michael Ian Shamos and Dan Hoey. Closest-point problems. In *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, pages 151–162. IEEE, 1975. doi:10.1109/SFCS.1975.8.

26 Jim Shaw and Yun William Yu. Fast and robust metagenomic sequence comparison through sparse chaining with skani. *Nature Methods*, 20:1661–1665, 2023.

27 Krysta M. Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. Q#: enabling scalable quantum computing and development with a high-level domain-specific language. *arXiv preprint*, 2018. arXiv:1803.00652.

28 Nathan Wiebe, Ashish Kapoor, and Krysta M. Svore. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Information and Computation*, 15(3–4):316–356, 2015. doi:10.26421/QIC15.3-4-7.

29 Shengli Zhang and Tianming Wang. A novel alignment-free method for phylogenetic analysis of protein sequences. In *Proceedings of the 10th WSEAS International Conference on Applied Computer Science*, pages 67–71, 2010.

30 Hongyu Zheng, Guillaume Marçais, and Carl Kingsford. Creating and using minimizer sketches in computational genomics. *Journal of Computational Biology*, 30(12):1251–1276, 2023. doi:10.1089/CMB.2023.0094.