Quantum k-NN and Collision Finding
Review Notes

Main theme. Swap Test gives a similarity value as a probability. QADC/amplitude estimation converts that probability into a digital value with better precision scaling. Separately, collision finding uses a quantum walk over subsets to reuse queried data and achieve an $O(N^{2/3})$ query algorithm.

1 Quantum k-NN: Cost Structure

Given a test state $|\psi\rangle$ and training states $|\phi_j\rangle$, define the fidelity similarity

$$F_j = |\langle\psi|\phi_j\rangle|^2.$$

Swap Test does not write F_j into a register directly. It produces a success probability

$$p_j = \Pr[\text{Swap Test outputs 0}] = \frac{1+F_j}{2}.$$

Thus

$$F_j = 2p_j - 1.$$

Sampling-based method

If we estimate p_j by repeatedly measuring Swap Test, then to get additive error η in F_j, we need

$$O\left(\frac{1}{\eta^2}\right)$$

measurements per training state. For M training states, the rough cost is

$$O\left(\frac{M}{\eta^2}\right).$$

Coherent QADC / amplitude-estimation method

QADC treats the Swap-Test circuit as a coherent state-preparation unitary and estimates the success probability by amplitude estimation. This improves the precision dependence from

$$O\left(\frac{1}{\eta^2}\right) \quad \text{to} \quad O\left(\frac{1}{\eta}\right).$$

If quantum k-maxima is used to find the nearest k training states, the high-level search cost is

$$O(\sqrt{kM}).$$

Because each comparison internally needs fidelity estimation through QADC, the costs multiply:

$$O(\sqrt{kM}) \cdot O\left(\frac{1}{\eta}\right) = O\left(\frac{\sqrt{kM}}{\eta}\right).$$

If k is a constant, this becomes

$$O\left(\frac{\sqrt{M}}{\eta}\right).$$

Important distinction. The $1/\eta$ factor is not added to $\sqrt{kM}$. It is multiplied because the top-k search calls a comparator oracle many times, and each comparator call requires QADC-level fidelity extraction.

2 QADC: From Probability to a Digital Fidelity

For a fixed j, let A_j be the unitary that prepares $|\psi\rangle$, prepares $|\phi_j\rangle$, and runs Swap Test up to the point before measurement. Then

$$A_j|0\rangle = \sqrt{p_j}\,|\text{good}_j\rangle + \sqrt{1-p_j}\,|\text{bad}_j\rangle,$$

where $|\text{good}_j\rangle$ is the subspace in which the Swap-Test ancilla is 0.

Write

$$p_j = \sin^2(\pi\theta_j).$$

Then

$$F_j = 2p_j - 1 = 2\sin^2(\pi\theta_j) - 1.$$

Amplitude estimation constructs a Grover-type iterate

$$G_j = -A_j S_0 A_j^\dagger S_{\text{good}},$$

whose eigenphases are related to θ_j. Quantum Phase Estimation (QPE) outputs a binary approximation

$$|\widetilde{\theta_j}\rangle.$$

Then reversible fixed-point arithmetic computes

$$|\widetilde{\theta_j}\rangle|0\rangle \mapsto |\widetilde{\theta_j}\rangle \left|\widetilde{2\sin^2(\pi\widetilde{\theta_j})} - 1\right\rangle.$$

Equivalently, since

$$2\sin^2(\pi\theta) - 1 = -\cos(2\pi\theta),$$

one may compute a finite-bit approximation to $-\cos(2\pi\theta)$.

Caveat. QADC does not exactly compute a real-valued trigonometric function. It computes a finite-bit approximation by reversible arithmetic, e.g. polynomial approximation, CORDIC-style methods, or lookup tables for small precision.

3 Collision Finding Problem

Given an input list

$$A_1, A_2, \ldots, A_N,$$

the goal is to decide whether there exist distinct indices $i \neq j$ such that

$$A_i = A_j.$$

For example,

$$[1, 3, 5, 2, 4, 3]$$

has a collision because the value 3 appears twice.

Classically:

- using a hash table, collision finding takes $O(N)$ queries/time;
- using sorting, it takes $O(N \log N)$ time.

Quantumly, Ambainis's quantum walk algorithm solves element distinctness / collision finding in

$$O(N^{2/3})$$

queries.

4 Quantum Walk Setup

Instead of searching over pairs (i, j) directly, search over subsets. Choose a parameter r and consider subsets

$$S \subseteq [N], \qquad |S| = r.$$

Each subset S is a vertex of the Johnson graph $J(N, r)$.

For a subset S, store the queried data

$$D_S = \{(i, A_i) : i \in S\}.$$

Preparing D_S initially costs

$$O(r)$$

queries.

A subset is marked if it contains a collision:

$$S \text{ is marked} \iff \exists i \neq j \in S \text{ such that } A_i = A_j.$$

If there is one hidden collision pair (a, b), then a random subset S of size r is marked exactly when it contains both a and b. Hence the marked fraction is approximately

$$\varepsilon \approx \frac{r}{N} \cdot \frac{r-1}{N-1} \approx \frac{r^2}{N^2}.$$

Here ε means marked fraction, not error tolerance.

5 Johnson Graph Walk: One Out, One In

The Johnson graph step is

$$S \longrightarrow S - \{a\} + \{b\},$$

where $a \in S$ and $b \notin S$.

The key data-reuse idea is:

$$D_{S-\{a\}+\{b\}} = D_S - \{(a, A_a)\} + \{(b, A_b)\}.$$

The values of the other $r - 1$ elements are reused. Only the new value A_b must be queried. With reversible cleanup, this may take one or two calls to the input oracle, but in query complexity it is still

$$O(1)$$

per walk step.

Important. The collision check is not a magical oracle for the whole input. Once D_S is already stored, checking whether S contains a collision is ordinary reversible computation on the cached table. Naively this costs $O(r^2)$ comparisons, but it costs no new input queries.

6 Why the Spectral Gap Appears

A walk on the Johnson graph changes only one element of the subset at a time. Therefore the new subset is highly correlated with the old subset.

A particular element currently in S is removed with probability about $1/r$ per step. Thus it takes about r steps for the subset to forget much of its current content. This corresponds to a spectral gap

$$\delta \approx \frac{1}{r}.$$

Intuitively:

$$\frac{1}{\delta}$$

is the time scale for the walk to produce a substantially new subset.

7 Quantum Walk Search Complexity

For graph search, two quantities matter:

- ε: fraction of marked vertices;
- δ: spectral gap of the underlying random walk.

Classical local random walk search scales roughly like

$$O\left(\frac{1}{\varepsilon\delta}\right).$$

Quantum walk search gives the square-root improvement:

$$O\left(\frac{1}{\sqrt{\varepsilon\delta}}\right).$$

For element distinctness,

$$\varepsilon \approx \frac{r^2}{N^2}, \qquad \delta \approx \frac{1}{r}.$$

Hence

$$\varepsilon\delta \approx \frac{r^2}{N^2} \cdot \frac{1}{r} = \frac{r}{N^2}.$$

So the number of quantum walk iterations is

$$\frac{1}{\sqrt{\varepsilon\delta}} = \frac{1}{\sqrt{r/N^2}} = \frac{N}{\sqrt{r}}.$$

8 Optimizing the Total Cost

The total query cost is

$$O(r) + O\left(\frac{N}{\sqrt{r}}\right).$$

The first term is the setup cost for the initial subset table. The second term is the quantum walk search cost.

Balance the two terms:

$$r = \frac{N}{\sqrt{r}}.$$

Then

$$r^{3/2} = N,$$

so

$$r = N^{2/3}.$$

At this choice,

$$r + \frac{N}{\sqrt{r}} = N^{2/3} + \frac{N}{N^{1/3}} = 2N^{2/3}.$$

Thus the query complexity is

$$O(N^{2/3}).$$

9 Classical Random Walk Comparison

Classical local random walk search takes roughly

$$O\left(\frac{1}{\varepsilon\delta}\right)$$

steps. Substituting the same values gives

$$\frac{1}{\varepsilon\delta} = \frac{1}{r/N^2} = \frac{N^2}{r}.$$

Including setup cost,

$$O(r) + O\left(\frac{N^2}{r}\right).$$

Optimizing this gives $r = N$, and the total cost is

$$O(N).$$

Thus the quantum walk gives a genuine improvement:

$$O(N) \quad \longrightarrow \quad O(N^{2/3}).$$

10 Key Intuition

- A vertex is a subset S of size r.
- A marked vertex is a subset containing the hidden collision pair.
- The probability that a random subset is marked is $\varepsilon \approx r^2/N^2$.
- The Johnson graph changes only one element at a time, so its spectral gap is $\delta \approx 1/r$.
- Classical local search costs about $1/(\varepsilon\delta)$.
- Quantum walk search costs about $1/\sqrt{\varepsilon\delta}$.
- The total quantum query cost is $r + N/\sqrt{r}$.
- The optimal subset size is $r = N^{2/3}$.

One-sentence summary. Collision finding is sped up by walking over cached subsets: the algorithm pays r queries once to prepare a subset, updates the subset with $O(1)$ new queries per step, and uses quantum walk search to find a marked subset in $N/\sqrt{r}$ steps. Optimizing gives $r = N^{2/3}$ and total query complexity $O(N^{2/3})$.