Element distinctness and collision problems

Separate arbitrary duplicate detection from promised collision finding and understand the role of subset walks.

Undergraduate | 21 min lesson | Dr. Charith Jayasekara

By the end, you'll be able to

- Trace a stored-subset update that finds a duplicate.
- Distinguish N^2 pair search from an N-entry input.
- Match collision promises to their query guarantees.

A duplicate is a relation between two inputs

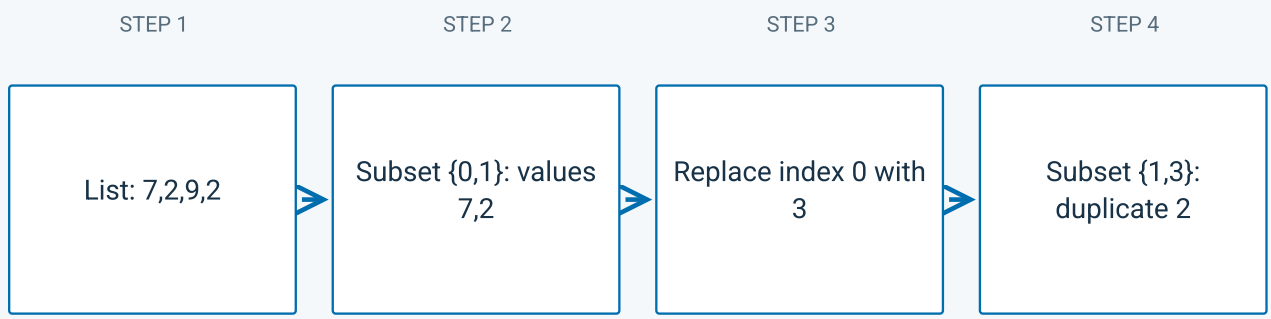
Element distinctness asks whether a list contains two equal values. Collision finding asks for different inputs with the same output of a function, often under a promise about how many inputs share each output. The problems sound similar but have different input promises and quantum query bounds. A good explanation starts by separating them rather than attaching one "collision speedup" to every duplicate-finding task.

The everyday example is a list of identifiers: 7, 2, 9, 2. The duplicate pair is at indices one and three. A classical program can detect it with sorting or hashing, but quantum query algorithms study access to the values as an oracle. Query complexity counts how often that oracle is used, while memory organisation and comparison arithmetic remain additional resources.

Review oracle access, Grover search and complexity variables. The central new idea is to walk over subsets of indices, reusing information as the subset changes.

WORKED DIAGRAM · AA16

Reuse one stored value when a subset changes



The duplicate is at indices 1 and 3. A subset walk reuses retained queried values; naive Grover search over order-N^2 pairs has order-N query scale, while the element-distinctness walk achieves order N^(2/3).

State the two setups

For element distinctness, the input is an arbitrary N-entry list and the task is to decide whether any duplicate exists, or produce a duplicate pair with bounded error. The oracle maps an index to its value coherently. No promise says that duplicates are abundant. In the hardest cases, a single equal pair may be hidden among otherwise distinct values.

For the standard promised collision problem, a function is promised to have a particular multiplicity pattern, such as being two-to-one. A collision-finding algorithm can exploit that abundance of partners. Its bound cannot automatically be applied to an arbitrary list with one duplicate. State the domain size, range and promise before comparing exponents.

Why a naive pair search is insufficient

There are roughly N squared candidate pairs. Grover search over all pairs therefore needs order N equality queries in the worst case, ignoring constants. That does not exploit the fact that nearby candidate subsets share most of their values. A more sophisticated walk stores a subset's queried values and updates only a small part of that record when moving to a neighbouring subset.

This reuse is the algorithmic insight. Quantum speedups are not always obtained by wrapping Grover around the most obvious brute-force space. Choosing a different state space and maintaining useful data can change the query trade-off. The same principle appears in classical algorithms, where choosing the right intermediate representation can be more important than accelerating one operation.

Work a subset example

For the list 7, 2, 9, 2, consider subsets of two indices. The subset containing indices zero and one stores values 7 and 2 and has no internal collision. Replacing index zero with index three gives subset one, three, whose values are both 2. Only the newly introduced value needs to be queried if the retained value is stored coherently.

A quantum walk explores a superposition of such subsets, with marked subsets containing a duplicate. Its checking operation uses the stored values. The walk is not a sequence of classical random subset samples, because amplitudes from different paths can interfere. The data register and subset register must be updated reversibly so the walk remains a valid unitary process.

For a larger list with one duplicate pair and a random subset of size r, the exact chance that both duplicate indices are included is $r(r-1)/(N(N-1))$, approximately $(r/N)^2$ when the sizes permit that approximation. For N equal to 100 and r equal to 10, the exact probability is 90/9900, about 0.00909. Larger subsets cost more to set up but make marked subsets less rare. The optimal choice balances setup, update and detection costs.

The probability that a subset contains a fixed duplicate pair follows from counting subsets.

$$P_{\text{marked}} = \frac{\binom{N-2}{r-2}}{\binom{N}{r}} = \frac{r(r-1)}{N(N-1)}, \qquad N=100,\ r=10: \quad P_{\text{marked}} = \frac{90}{9900} \approx 0.00909.$$

N is the list length and r the uniformly chosen subset size. The numerator fixes the two duplicate indices and chooses the remaining $r-2$ from the other $N-2$ indices. Larger subsets increase marked weight but also increase setup cost.

The element-distinctness result

Ambainis's quantum-walk algorithm achieves order N to the two-thirds query complexity for element distinctness, improving on naive pair search. The theorem concerns the oracle problem and a carefully chosen walk, not a claim that every practical deduplication service runs faster on a quantum processor. Read the element-distinctness paper.

The characteristic two-thirds exponent arises from balancing the subset size against the walk search cost. Memory and reversible data handling are important implementation parameters. A query theorem can be optimal while a particular hardware implementation remains expensive. Preserve the distinction between a mathematical access lower bound and an engineering benchmark.

A collision-table intuition

For a promised many-collision function, another strategy stores a table of sampled input–output pairs, then uses quantum search for an outside input whose output matches a stored one. Increasing the table makes a partner easier to find but increases setup and memory. Balancing those costs produces a cube-root query scale in standard promised settings.

Brassard, Høyer and Tapp introduced a collision-finding approach combining a stored table with Grover search and analysed its query and space trade-offs. Its multiplicity assumptions are part of the result. Read the collision-finding paper. Do not replace those assumptions with the statement that any duplicate can always be found with a cube-root number of queries.

Follow a practical analysis

Identify whether the input is arbitrary or promised and whether the output is a decision or an explicit pair. Specify coherent value access, equality precision and memory. Choose the subset-walk or table-search construction appropriate to that problem. Count setup, updates, checks, repetitions and verification. Then compare with classical hashing, sorting or external-memory methods under the same data-access constraints.

For cryptographic collision examples, distinguish an ideal random-function model from a particular hash implementation and include reversible evaluation cost. A query exponent does not directly specify the number of physical qubits or the runtime of an attack. Parallelism, memory restrictions and fault-tolerant gates can change the useful resource trade-off.

Practice with answers

Exercise 1. Why does Grover search over all index pairs give an order-N query scale rather than order square-root-N?

> Check the worked answer
>
> The candidate space has order N squared pairs. Taking its square root gives order N. Query complexity depends on the size of the searched space, not merely the length of the original list.

Exercise 2. Can a cube-root collision theorem for a two-to-one function be applied to a list with exactly one equal pair?

> Check the worked answer
>
> Not directly. The two-to-one promise provides many collision partners and changes the marked fraction. The arbitrary element-distinctness problem has a different guarantee and algorithmic trade-off. Match the promise before quoting a bound.

Check your understanding

PAUSE & THINK

Why does Grover over all index pairs have order-N query scale?

- [ ] There are order N^2 candidate pairs
- [ ] Grover cannot search pairs
- [ ] Every list is two-to-one

Answer: There are order N^2 candidate pairs. Taking the square root of an order-N^2 search space gives order N. A different walk state space is needed for the stronger element-distinctness result.

Sources

- Read the element-distinctness paper
- Read the collision-finding paper