On the Need for (Quantum) Memory with Short Outputs

Zihan Hao[*] Zikuan Huang[†] Qipeng Liu[‡]

Abstract

In this work, we establish the first separation between computation with bounded and unbounded space, for problems with short outputs (i.e., working memory can be exponentially larger than output size), both in the classical and the quantum setting. Towards that, we introduce a problem called *nested collision finding*, and show that optimal query complexity can not be achieved, if one has much smaller memory comparing to the optimal algorithm.

Our result is based on a novel "two-oracle recording" technique, where one oracle "records" the computation's long outputs under the other oracle, effectively reducing the time-space trade-off for short-output problems to that of long-output problems. We believe this technique will be of independent interest for establishing time-space tradeoffs in other short-output settings.

[*]UC San Diego, z1hao@ucsd.edu
[†]Tsinghua University, hzk21@mails.tsinghua.edu.cn
[‡]UC San Diego, qipengliu0@gmail.com

Contents

1 Introduction

Does quantum speedup need large memory? This question has been extensively studied in different contexts, including advice-aided computation [NABT14, CGLQ20] (preprocessing model), online learning [LRZ23, CLL24] (streaming model), quantum pebbling [BHL22] (pebbling model) and space-bounded computation with massive outputs [HM23, BKW24] (general model). These works collectively demonstrate that quantum computation faces fundamental limitations when restricted to a small amount of (classical/quantum) memory, whether in the form of advice or working space.

One of the most important questions towards understands the role of memory/space in quantum computation, is the quantum collision finding problem. In this problem, one is given a function $f : [M] \to [N]$ (modeled as a random function) and the goal is to find a pair of distinct inputs $x \neq x'$ such that $f(x) = f(x')$. Classically, an algorithm needs $O(\sqrt{N})$ time to achieve the goal, and it is tight due to the Birthday Paradox. The famous Pollard-Rho algorithm [Pol78] can achieve this with $O(\sqrt{N})$ and $\widetilde{O}(1)$ space. Quantumly, the BHT algorithm [BHT98] can solve this problem with both time and space at most $O(N^{1/3})$. Unlike its classical counterpart, we do not know if $N^{1/3}$ space is needed to achieve the optimal running time. More surprisingly, we do not even know whether quantum algorithms require any nontrivial amount of memory [1] to achieve an advantage over classical algorithms. This question of understanding the time-space tradeoffs for collision finding was also listed as one of the ten open challenges by Aaronson [Aar21], yet little progress has been made.

One key difficulty in answering the above question, stems from the fact that proving time-space tradeoffs for problems with short outputs in the "generic model" is challenging even in the *classical* setting. Here, the "generic model" refers to algorithms that performs computation on bounded memory, can discard some used memory, and introduce new memory on demand. This was already pointed out by Dinur [Din20]. Since the introduction of time-space tradeoffs by Cobham [Cob66], establishing strong time-space tradeoffs for short-output problems is open; [FLVV05] demonstrated some tradeoffs for short-output problems, but both time and space are only logarithmic in the input size, making the statement not strong. Other models (preprocessing, streaming and pebbling models) do not have a similar barrier and thus most results directly apply to problems with short outputs. On the other hands, existing trade-off results in the "generic model" only apply to large-output problems, where the required memory is at least as large as the output size, like [Din20] in the classical setting and [HM23] in the quantum setting for multi-collision finding. Given these challenges, we pose the following fundamental questions in this paper:

Can we establish strong classical and quantum time-space trade-offs for problems with ***short outputs****?*

More precisely we ask:

If a problem has a short output ($\mathsf{poly}(n)$ *bits), does an optimal (quantum) algorithm need* $\exp(n)$ *space?*

Short outputs are not an edge case: decision problems and most search problems in the RO/QROM all fall into this regime. Technically, all previous method in the generic model (for long outputs) seems to require certain progress measure, which is no longer well defined for short-output problems. Even partial results — say, ruling out $T \cdot S = o(N^{\alpha})$ for some constant $\alpha > 0$ on natural short-output tasks — would represent a qualitative advance over the current logarithmic barrier and bring us materially closer to Aaronson's question.

[1]At least $\log N + \log M$ to allow classical/coherent evaluations of the function f.

1.1 Our results

In this work, we introduce the ℓ-*Nested Collision Finding* problem for a constant $\ell > 1$. Given two random oracles $H : [M] \to [N]$ and $G : [M^\ell] \to [N_0]$, the goal is to find two distinct ordered tuples $(x_1, x_2, \ldots, x_\ell) \neq (x'_1, x'_2, \ldots, x'_\ell) \in [M^\ell]$ such that:

- $\sum_{i=1}^{\ell} H(x_i) = \sum_{i=1}^{\ell} H(x'_i) = 0$, (here $\sum$ denotes summation mod N), and,
- $G(x_1, \ldots, x_\ell) = G(x'_1, \ldots, x'_\ell)$.

We will sometimes call the problem as "finding a nested collision pair". We can strengthen the problem so that instead of asking the two ℓ-sums equal to 0, we require the two ℓ-sums to be any fixed y or uniformly random y that is presented in the input. This strengthening will not change any lemma below. For the simplicity, we will just define the problem with the ℓ-sums being 0.

Note that the problem has a short output; the output is only of size poly-logarithmic in the size of the input (if we treat as input the truth table of the oracles H, G). At a high level, the goal is to find two ℓ-tuples whose images under H have the same sum 0 and whose images under G are identical. Although this formulation involves two random oracles, a single random oracle with a sufficiently large domain can serve as both H and G. However, for clarity, we will continue to work with two separate oracles throughout this paper.

Our main contribution is to show that the ℓ-Nested Collision Finding problem has time-space tradeoffs in both the classical and the quantum setting.

Quantum algorithms for short-output problems require space.

Theorem 1.1 (Main quantum theorem). *For $\ell \geq 2$, any quantum algorithm that solves the ℓ-Nested Collision Finding problem with constant probability with space $S = \Omega(\log N) = O\left(N^{\frac{1}{(\ell+1)(2\ell+1)}}\right)$ and T oracle queries must satisfy $S^{\frac{\ell+1}{2}} T = \Omega\left(N^{\frac{1}{2(\ell+1)}} N_0^{\frac{1}{4}}\right)$.*

The main theorem has both parameters ℓ and N_0 that are yet to be determined. By comparing against the best algorithm we constructed (see Theorem 6.5 for the algorithm), the ℓ-Nested Collision Finding problem establishes a separation between space-bounded and unbounded quantum computation, provided that $\ell \geq 4$. For clarity, we select the most straightforward parameter choices and state a specific instance of our result Theorem 1.1 when $\ell = 4$ and $N_0 = N$.

Theorem 1.2 (Separation between space-bounded/unbounded quantum computation)**.** *Any quantum algorithm that solves 4-Nested Collision Finding problem with oracles $H : [M] \to [N]$ and $G : [M^4] \to [N]$, with space $S = \Omega(\log N) = O(N^{1/45})$ and T oracles queries, must satisfy $S^{5/2}T = \Omega(N^{0.35})$.*

As a special case, when $S = \widetilde{\Theta}(1)$, the quantum algorithm requires at least $\widetilde{\Omega}(N^{0.35})$ quantum queries to solve the 4-Nested Collision Finding problem with a constant probability. Also, as long as $S = o(N^{1/150})$, the optimal query complexity $N^{1/3}$ can not be achieved. On the other hand, the optimal quantum algorithm only requires $O(N^{1/3})$ quantum queries (as well as $O(N^{1/3})$ memory).

Classical algorithms for short-output problems require space.

We have a similar theorem in the classical setting.

Theorem 1.3 (Main classical theorem). *For $\ell \geq 2$, any classical algorithm that solves the ℓ-Nested Collision Finding problem with constant probability with space $S = \Omega(\log N) = O\left(N^{\frac{1}{\ell^2-1}}\right)$ and T oracle queries must satisfy $S^{\frac{\ell^2-1}{2\ell}} T = \Omega\left(N^{\frac{1}{2\ell}} N_0^{\frac{1}{2}}\right)$.*

On the other hand, we construct the following classical algorithm. For any $\ell \geq 1$, there exists a classical algorithm that uses $O\left(N^{\frac{1}{\ell}} N_0^{\frac{1}{2\ell}}\right)$ queries that solve the ℓ-Nested Collision Finding Problem when $N_0 = O\left(N^{\frac{2}{\ell-1}}\right)$ with constant probability. By setting $\ell = 2$ and $N_0 = N^2$, we have the following corollary:

Theorem 1.4 (Separation between space-bounded/unbounded classical computation). *Any classical algorithm that solves 2-Nested Collision Finding problem with oracles $H : [M] \to [N]$ and $G : [M^2] \to [N^2]$, with space $S = \Omega(\log N) = O(N^{1/3})$ and T oracles queries, must satisfy $S^{3/4} T = \Omega(N^{5/4})$.*

As a special case, when $S = \widetilde{\Theta}(1)$, the classical algorithm requires at least $\widetilde{\Omega}(N^{5/4})$ classical queries to solve the same problem with a constant probability. On the other hand, the optimal classical algorithm only requires $O(N)$ classical queries.

Again, this is the first example in the classical setting, where a short-output problem requires exponential space to reach an optimal time.

1.2 Discussions and open questions

We discuss applications and directions related to our new framework.

Proof of space. By scaling down the parameter N, the classical/quantum tradeoffs yield a non-interactive proof-of-space in the random oracle model. The prover computes and returns a collision pair for the Nested Collision Finding problem as the proof; any prover lacking sufficient space incurs a longer running time to find such a collision. The main drawback is that the time gap between space-bounded and unbounded provers is only polynomial. A potential remedy could be to define a task that further nests the Nested Collision Finding problem, achieving an exponential separation between space-bounded and unbounded algorithms.

Does quantum advantage require space? Another direction is to construct a short-output problem whose classical query complexity (with sufficient space) is strictly smaller than its bounded-space quantum query complexity. This can be read as: quantum advantage requires space. If a hash function satisfies this property, then in virtually any hash-based cryptographic protocol, one could neutralize quantum advantages in attacks against the hash, as having access to exponential space is unrealistic for practical purposes. We believe that our Nested Collision Finding problem is such a candidate, but the current analysis is not tight enough for demonstrating the separation.

Distinguishing between quantum and classical space? All our results treat space as a whole, i.e. we do not distinguish between quantum and classical space. This technical limitation also appears in most prior works. Currently, there is no known general method for distinguishing quantum and classical memory in this context. The only known approaches apply to the streaming setting, such as the techniques developed in [LRZ23], or are based on unproven conjectures [LMY25].

2 Technical overview

We will only talk about lower bounds in the overview. The algorithm is relatively straightforward and is presented in Theorem 6.5.

Time-space tradeoffs in massive-output problems.

We start by reviewing the ideas behind time-space tradeoffs in massive-output problems in the general computation model and the difficulty of obtaining a tradeoff in the short-output setting. We consider the problem of finding multiple two-collisions in [HM23]. The problem is, given a random oracle $H : [M] \to [N]$, to find K pairs of disjoint inputs $(x_1, x_2), \ldots, (x_{2K-1}, x_{2K})$ such that $H(x_{2i-1}) = H(x_{2i})$ for all $i = 1, 2, \ldots, K$, which was first studied by [Din20] in the classical setting and by [HM23] in the quantum setting.

To achieve time-space tradeoffs in the general computation model, most (if not all) approaches rely on the time-segmentation method [Bor93, KŠW07]. Roughly speaking (see Figure 1), instead of enforcing a strict space bound of S at every step of an algorithm's execution, this method divides time into L segments. The algorithm is then relaxed so that the space usage is only constrained to S when transitioning between segments, while no space bound is imposed within each segment. In this model, the algorithm is allowed to output the answers on the fly; i.e., for the multi-collision problem, at the end of each segment, it can output all the pairs of collision it finds within that segment.

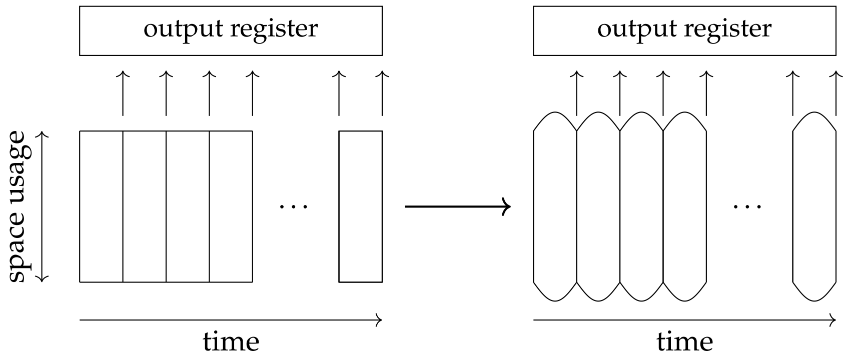


Figure 1: Conversion from a space-bounded computation to a time-segmented computation with space constraints applied only during transitions.

Therefore, if an algorithm with space constraint S in the time-segmentation framework can find more than K pairs of disjoint collisions, it must be able to find at least K/L pairs of disjoint collisions within a single segment. The key idea in time-segmentation is to track progress toward finding more than K disjoint collisions — i.e., how many disjoint collisions can be found in each segment. By appropriately choosing parameters, [HM23] proved that any quantum algorithm making T/L queries cannot find more than K/L disjoint collisions, except with probability exponentially small in S.

They further extended this argument to non-uniform algorithms with arbitrary S-qubit memory, modeling the memory passed from one segment to the next. By setting $K/L \approx \Theta(S)$, the probability of a uniform algorithm finding K/L disjoint collisions is on the order of $2^{-\Theta(S)}$. Thus, a piece of S-qubit advice can be removed by randomly guessing an advice state, introducing a multiplicative factor of 2^S. This factor is completely absorbed into $2^{-\Theta(S)}$, preserving the asymptotic order.

This is the general approach used to establish time-space tradeoffs in multi-collision finding and other time-segmentation methods. However, several inherent barriers arise when dealing with short-output problems. First, the notion of a progress measure becomes unclear. Since the algorithm is only required to produce a short output, it does not necessarily follow a structured computational pattern imposed by a predefined progress measure. Second, as the S-qubit memory is passed between segments, the only general approach to handling this memory is to guess and remove it. This introduces a multiplicative factor of 2^S in the overall success probability for each segment. To ensure that this factor does not render the bound meaningless or trivial, the success probability within each segment must be at least exponentially small in S. It appears that only massive-output problems can safely absorb this factor without changing the asymptotic order of the probability.

Nested Collision Finding problem.

From the above discussions, finding an appropriate progress measure for a short-output problem is crucial for establishing a time-space tradeoff. Our starting point is the problem of finding a *3-collision*, where the goal is to search for three distinct inputs that all map to the same output. This problem appears to be a promising candidate. Intuitively, an algorithm that finds a 3-collision must first identify multiple 2-collisions, which, as shown in the previous section, already admits a time-space tradeoff. Therefore, if we can transform an algorithm that finds a 3-collision into one that efficiently finds multiple 2-collisions, the 3-collision problem will inherently exhibit a non-trivial time-space tradeoff.

The above intuition serves as the key insight leading to our final construction and theorems. However, reducing the problem of finding multiple 2-collisions to that of finding a 3-collision has some technical difficulties, due to the following challenges:

1. **Preserving time and space:** The reduction must be both time- and space-efficient, as we aim to maintain the time-space tradeoff properties throughout the reduction.
2. **Applicability to general algorithms:** The reduction may not hold for arbitrary (non-optimal) algorithms. In particular, certain contrived algorithms may find a 3-collision without necessarily discovering many 2-collisions along the way. For instance, consider an algorithm that searches specifically for three distinct inputs mapping to zero and keeps only the preimages of zero during its execution. While this algorithm eventually finds a 3-collision with polylog space, extracting multiple 2-collisions from its execution could be challenging.

Although we are unable to apply this idea directly to the 3-collision finding problem, we implement the previous intuition through a different construction in the random oracle model. Recall the *ℓ-Nested Collision Finding* problem (see Section 6) for a constant $\ell > 1$. Given two random oracles $H : [M] \to [N]$ and $G : [M^\ell] \to [N_0]$, the goal is to find two distinct ordered tuples $(x_1, x_2, \ldots, x_\ell) \neq (x'_1, x'_2, \ldots, x'_\ell) \in [M^\ell]$ such that:

- $\sum_{i=1}^{\ell} H(x_i) = \sum_{i=1}^{\ell} H(x'_i) = 0$, and $G(x_1, \ldots, x_\ell) = G(x'_1, \ldots, x'_\ell)$.

Following the intuition from the 3-collision finding problem, we aim to establish the following two parts:

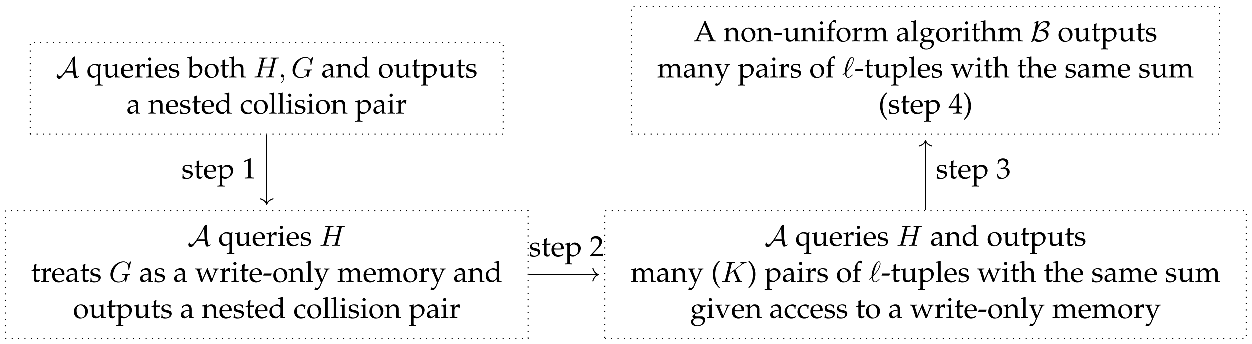
- **Part 1.** Any algorithm that finds a collision pair for the Nested Collision Finding problem can be transformed (while preserving both time and space complexity) into an algorithm that outputs multiple ℓ-tuples $(x_{1,1}, \ldots, x_{1,\ell})$, …, $(x_{K,1}, \ldots, x_{K,\ell})$ satisfying the same sum condition:

$$\sum_{j=1}^{\ell} H(x_{1,j}) = \sum_{j=1}^{\ell} H(x_{2,j}) = \cdots = \sum_{j=1}^{\ell} H(x_{K,j}) = 0.$$

We refer to this problem as "finding K ℓ-tuples with the same sum".

- **Part 2.** Establishing time-space tradeoffs for "finding K ℓ-tuples with the same sum" in both classical and quantum settings.

We elaborate on both parts in the following sections. The roadmap is given as in Figure 2.



$\mathcal{A}$ queries both H, G and outputs a nested collision pair
step 1
$\mathcal{A}$ queries H treats G as a write-only memory and outputs a nested collision pair
step 2
$\mathcal{A}$ queries H and outputs many (K) pairs of ℓ-tuples with the same sum given access to a write-only memory
step 3
A non-uniform algorithm $\mathcal{B}$ outputs many pairs of ℓ-tuples with the same sum (step 4)

Figure 2: A roadmap in establishing our main results.

"Two-oracle recording" technique that preserve time and space (step 1).

We begin by considering the classical setting. Let $\mathcal{A}$ be a classical query algorithm with oracle access to both H and G. Typically, G is viewed as a pre-sampled random function. However, here we adopt an alternative perspective by regarding G as a lazily sampled truth table. In the standard formulation, a random oracle $G : [M^{\ell}] \to [N_0]$ is chosen uniformly at random from the set of all functions. In contrast, the lazy sampling approach (often referred to as the "principle of deferred decisions") begins with G initialized to $\{\perp\}^{\otimes M^{\ell}}$, meaning that every output is initially undefined. When a query x is made, if $G(x) = \perp$ (i.e., if its value has not yet been determined), we sample a uniformly random $y \leftarrow [N_0]$ and then set $G(x) := y$. Since these two views are statistically identical, we can freely switch between them in our analysis.

Our key observation is that, in the lazy sampling view, G can be interpreted as an exponentially large memory that records the computation of $\mathcal{A}$. This allows us to view the computation of $\mathcal{A}$ using the oracles H and G in an alternative way: $\mathcal{A}$ interacts with the oracle H as usual while treating G as a memory that supports a specific form of write-only operation:

- $\mathcal{A}$ can query H as a standard random oracle.
- $\mathcal{A}$ can write to the memory G by querying on x. Initially, all entries of G are set to $\perp$, indicating that all memory cells are empty. When $\mathcal{A}$ queries $G(x)$, if $G(x) = \perp$, a random $y \leftarrow [N_0]$ is sampled, stored in the x-th memory block, and returned to $\mathcal{A}$. Otherwise, the stored value is simply returned.

Although G returns y to the algorithm, y is chosen randomly and does not depend on prior computation, so G can still be viewed as write-only memory. It is straightforward to see that this write-only memory model is equivalent to the lazy sampling view of a random oracle.

By modeling a random oracle as a write-only memory, the computation with oracle access to both H and G can now be interpreted as a computation with oracle access to H and a write-only memory G. This conversion is illustrated in Figure 3. We refer to this perspective as the "two-oracle recording" technique, which will serve as the foundation for our proofs.

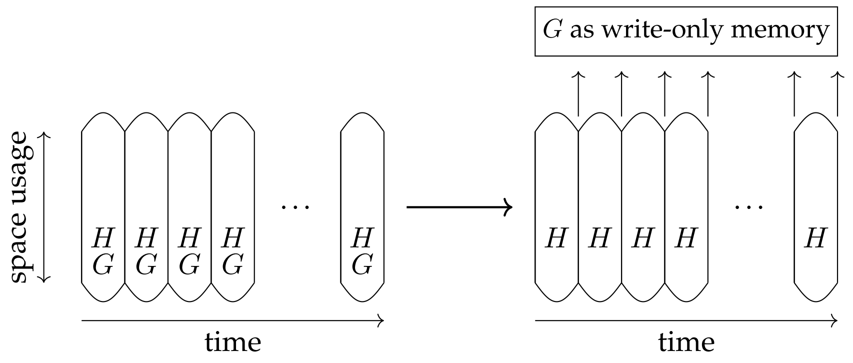


Figure 3: The LHS represents computation with oracle access to both H, G. The RHS represents computation with only oracle access to H, whereas having access to an (exponentially large) write-only memory G.

The quantum "two-oracle recording" technique (step 1, quantum).

The above idea extends to the quantum case with additional efforts. [Zha19] shows a formulation of lazy sampling for quantum accessible random oracle, called "compressed oracle". Since quantum queries can be made in superposition, a classical database/memory is no longer feasible to track all the information. Zhandry showed that, the notion of a "database" is still meaningful if the "database" itself is also stored in superposition. A quantum-query algorithm interacting with a quantum-accessible random oracle can be equivalent simulated (informally) as follows:

- The database register is initialized as $|\bot\rangle^{\otimes M^\ell}$: the algorithm has not yet queried anything.
- When the algorithm makes a quantum query, the database register gets updated in superposition: for a query $|x\rangle$ and a database $|D\rangle$, the simulator looks up x in D. If the x-th entry is $|\bot\rangle$, it changes this entry to an equal superposition $\sum_y |y\rangle$ (up to normalization).
  It then returns the x-th entry in the updated D also in superposition.

Any quantum computation of $\mathcal{A}$ using oracles H, G can be alternatively viewed as $\mathcal{A}$ using oracle H and has access to a memory (the random oracle G) that supports coherent write-only operation, as described above. The above description provides a high-level idea, albeit slightly inaccurate. Since a quantum algorithm can forget previously acquired information, a formal "compressed oracle" necessitates an additional step to perform these forgetting operations. We leave the details in Section 4.

Relating to massive-output problems (step 2).

Assume that an algorithm $\mathcal{A}$ finds a pair of nested collisions $(x_1, \ldots, x_\ell) \neq (x'_1, \ldots, x'_\ell)$. By the definition, we have

$$\sum_i H(x_i) = \sum_i H(x'_i) = 0$$

and

$$G(x_1, \ldots, x_\ell) = G(x'_1, \ldots, x'_\ell).$$

The classical case. Since $\mathcal{A}$ finds a collision under G, a standard lazy sampling argument implies that there exists some value z such that at least $\Omega(N_0/T)$ distinct ℓ-tuples queried to G sum to z, where T is the number of queries. This follows from the observation that if at any moment of the computation, the number of such tuples is bounded by K, then the probability of finding a collision in a single step is at most K/N_0. Given T queries, the total success probability is thus at most $(TK)/N_0$[2].

Therefore, if $\mathcal{A}$ successfully finds a nested collision with a constant probability, then by interpreting G as write-only memory (via the "two-oracle recording" technique), G must store at least $K = \Omega(N_0/T)$ pairs of ℓ-tuples of the same sum.

In the actual proof, we further strengthen this argument by showing that the claim holds even when K represents the expected number of such ℓ-tuples of the same sum stored in the lazily sampled database, where the expectation is taken uniformly at random over all possible G across all time steps. Thus, to extract K pairs of ℓ-tuples of the same sum, one can simply run $\mathcal{A}$ and halt at a uniformly random time, outputting all pairs stored in G at that moment.

The quantum case. The key difference in the quantum setting is that the random oracle G must be simulated using the compressed oracle formulation. We establish the following quantum analog:

Lemma 2.1 (Informal)**.** *If, at any moment during the computation, the compressed oracle database has a bound of at most K on the number of ℓ-tuples with the same sum, then given T queries, the total success probability is at most $(T^2K)/N_0$. By setting the probability to be a constant, we should have $K = \Omega(N_0/T^2)$.*

The proof follows from a more refined analysis using the compressed oracle technique [Zha19] and is presented in Section 5. At a high level, each query increases the amplitude (rather than the probability) of the final outcome by at most $O(\sqrt{K/N_0})$. Consequently, after T queries, the total amplitude is at most $O(T\sqrt{K/N_0})$, leading to a success probability bound of $O(T^2K/N_0)$. We further strengthen this argument by showing that the claim remains true even when K represents the expected number of such ℓ-tuples, measured at a uniformly random time step. Thus, to extract $K = \Omega(N_0/T^2)$ pairs of ℓ-tuples, one can simply run $\mathcal{A}$, halt at a uniformly random time, measure the database, and output all pairs obtained from the measurement outcomes.

Bounding the success probability in the massive-output case, with write-only memory (step 3.1).

In steps 1 and 2, we reduce an algorithm with space S and query complexity T to another algorithm that, with the same time and space complexity, outputs $K = \Omega(N_0/T^2)$ ℓ-tuples with the same sum. The next objective is to establish a time-space tradeoff for this large-output problem, which will then yield a corresponding tradeoff for the nested collision-finding problem. Similar bounds have been derived in [HM23] for multi-collision finding. However, while the setting is similar, our case presents additional challenges: in [HM23], the quantum algorithm can coherently write to a quantum output register, with each write operation stored in a new register[3]. In contrast, our setting requires each output x to be stored in a designated register (the x-th register), introducing potential interference/entanglement between the algorithm and the output register.

The approach in [HM23] (building on earlier works such as [Bor93, KŠW07, Din20]) relies on the time-segmentation method. This technique partitions an algorithm into L segments, where

[2]In the actual proof, we give an even tigher bound K^2/N_0, since $K \leq T$.

[3]More formally, each write operation coherently appends the output to the output register.

each segment is a non-uniform algorithm with S (qu)bits and at most T/L queries, with the goal of outputting K/L ℓ-tuples with the same sum. If one can show that such an algorithm, except with negligible probability, fails to output more than K/L tuples, then by a standard union bound, the original algorithm cannot produce more than K tuples (except with small probability).

This approach fails when an algorithm interacts with a memory where two distinct write-only operations can be applied to the same memory cell. Since we model the random oracle G as a write-only quantum memory using the "two-oracle recording" technique, we must consider this type of memory rather than the one analyzed in [HM23].

To explain the challenges we face, we consider the following simplified question (we call it "Sparse-Write Tail-Bound Composition Problem"):

- Consider two algorithms $\mathcal{A}, \mathcal{B}$, one working on registers AD and the other working on registers BD. All registers are initiated as all zeros; specifically, D consists of $|0^{M^\ell}\rangle$.
- $\mathcal{A}$ can apply local unitary on A as well as controlled flip operation on AD jointly: $|x\rangle|y\rangle \to |x\rangle|y \oplus e_x\rangle$ where e_x has only one 1 at its x-th location. At the end of $\mathcal{A}$, the probability that measuring D yields a string y with hamming weight more than C is at most ϵ.
- Similar for $\mathcal{B}$.
- Then the question is, can we show that if both $\mathcal{A}, \mathcal{B}$ run with the same D, the probability that measuring D yields a string y with hamming weight more than $2C$ is bounded by $\mathsf{poly}(\epsilon)$?

This question is straightforward to prove in the classical setting by incurring a union bound. However, in the quantum case, the quantum union bound [Gao15, KMW19, OV22] does not apply since potential interfence between $\mathcal{A}$ and $\mathcal{B}$'s writing behaviors. Even more, if the above question is true, it will imply another seemingly irrelevant question ("Fourier Tail-Bound Composition Problem") in the context of boolean functions:

- Let $f, g : \{-1,1\}^n \to \{-1,1\}$ be two functions. Assume $W^{>C}(f) \le \epsilon$ and $W^{>C}(g) \le \epsilon$; i.e., the Fourier weights of both functions at degrees above k are small. Then do we have $W^{>2C}(f \cdot g) \le \mathsf{poly}(\epsilon)$?

As far as we know, this simple question in boolean function analysis has not been studied yet; all standard tools seem not to work for this problem. Therefore, we do not know if the claim is true; let alone for the original 'Sparse-Write Tail-Bound Composition Problem.'

Bounding the expectation in the massive-output case, with write-only memory (step 3.2).

To address this, instead of looking at the composition of tail bounds, we look at the composition of expectations. Namely, we will examine for every $\vec{x} = (x_1, \ldots, x_\ell)$ of interest, what is the probability/expectation $w_{\vec{x}}$ that the write-only memory G has recorded $\vec{x}$. By the linearity of expectation, the final expected number is the summation of the expectation of all inputs of interests. We define the following game:

- Let H, G be two random oracles.
- Let $\vec{x}$ be a uniformly random input of interest (i.e., form a ℓ-tuple with the sum 0 under H).
- Run a (space-bounded) algorithm with oracle access to H, G except $G(x)$ is replaced with a uniform superposition $|+\rangle^{\otimes \log N_0}$; i.e., only $G(x)$ is treated as a compressed oracle[4].

[4]In the actual analysis, we will treat G as an oracle with binary outputs; i.e., split each output in $[N_0]$ into $\log N_0$ binary outputs and keep tracks the probability that each bit flipped. This is a minor difference and only introduces a multiplicative $\log N_0$ for the expectation. We do not complicate the intuition here and only provide the game without splitting outputs.

- Finally, measure the qubit in the Hadamard basis. The algorithm wins if and only if the measurement outcome is not all "$+$", indicating the x-th entry is written.

We show that we can apply the time-segmentation on the above game to remove space constraints, resulting in L segments. Within each segment, there is a T/L query algorithm with S qubits of advice and tries to maximize the expectation of the above game. Let $w_{\vec{x}}^{(i)}$ be the probability/expectation that the x-th entry is written. By a standard triangle inequality, we show that:

$$\sum_{\vec{x}} w_{\vec{x}} \leq L \cdot \sum_{i=1}^{L} \sum_{\vec{x}} w_{\vec{x}}^{(i)}.$$

Therefore, if we can bound the maximum expectation in the above game with a non-uniform algorithm of T/L queries and S qubits by W (i.e., $\sum_{\vec{x}} w_{\vec{x}}^{(i)} \leq W$ for every i), we can show that a T query and S space algorithm can have a maximum expectation $L^2 \cdot W$.

To prove an upper bound of the expectation by a non-uniform algorithm, we use the quantum presampling tool introduced in [GLLZ21] and improved in [Liu22], which is a powerful technique to remove the quantum advice by introducing a multi-instance version of the original game; for which, we leave all the details to the main body. Eventually, we are able to reduce the problem to a standard problem: how many distinct ℓ-tuples with the same sum a uniform query-bounded algorithm can find.

The time bound for finding many ℓ-tuples with the same sum (step 4).

Previous works have established similar bounds by tracking the number of tuples discovered in each step using the compressed oracle technique [Zha19]. For instance, [LZ19, HM23] analyzed the case of finding multiple 2-collisions. However, most of these works considered only cases where adding a new entry to the database introduces at most one new instance (e.g., a preimage or a collision pair), focusing solely on disjoint instances. For example, if four points x_1, x_2, x_3, x_4 map to the same value, prior work would treat them as two disjoint collisions, whereas in our setting, they correspond to six distinct collision pairs (i.e., any two distinct x_i, x_j can form a pair of collisions).

The key challenge in our proof is that in our case, adding a single new entry can introduce multiple new instances, given our target is to bound the number of ℓ-tuples. Our proof proceeds inductively based on the tuple size ℓ.

- **Base Case ($\ell = 1$).** We compute the probability that the algorithm finds a set of distinct points that all map to the same value under the random oracle using the standard compress oracle method.
- **Inductive Step ($\ell > 1$).** For larger tuples, we extend the base case by analyzing how smaller tuples can be combined to form larger ones. The key observation is that an algorithm attempting to find multiple ℓ-tuples with the same sum can proceed in one of two ways:
  - It already stores at least η distinct $(\ell - 1)$-tuples with the same sum in its database at some point during execution.
  - Alternatively, the following event occurs at least ℓ/η times: a newly added entry combines with $(\ell - 1)$ existing entries in the database to form at least one new ℓ-tuple with the target sum.

  Here, η is a threshold that we will set later. The probability of the first case can be bounded using the induction hypothesis. In the second case, by focusing on this specific event, we circumvent the challenge that arise when a single query introduces multiple new instances at once.

The details can be found in Section 7.2. Finally, by combining all the steps, we are able to show a time-space tradeoff for finding a single nested collision.

3 Preliminaries

We use $[N]$ to denote the set $\{0, 1, 2, \cdots, N-1\}$. A random oracle with domain $[M]$ and image $[N]$ is a function H sampled uniformly from all functions mapping $[M]$ to $[N]$. For two n-bit string a, b we use $\langle a, \rangle b$ to denote its inner product mod 2.

Lemma 3.1 (Chebyshev Bound)**.** *Let X be a random variable with expectation μ and variance σ, then*

$$\Pr\left[|X-\mu| \geq k\right] \leq \frac{\sigma}{k^2}.$$

Lemma 3.2 (Grover Search, [Gro96])**.** *Let $f : S \to \{0,1\}$ be a function. There exists an efficient algorithm that finds a $x \in S$ such that $f(x) = 1$ with $O\left(\sqrt{\frac{|S|}{|f|}}\right)$ calls to a Π_f gate:*

$$\Pi_f \left|x\right\rangle := (-1)^{f(x)} \left|x\right\rangle$$

if a uniform superposition over S can be efficiently generated from $|0\rangle$.

Lemma 3.3 (Jensen's Inequality, [Jen06])**.** *Let D, g be positive integers. Let $c_0, c_1, \cdots, c_{D-1}$ be a distribution over $[D]$. Let $p_0, p_1, \cdots, p_{D-1} \in \mathbb{R}^D$ satisfies that $\sum_{i\in[D]} c_i p_i > 0$. Then we have*

$$\sum_{i\in[D]} c_i p_i \leq \left(\sum_{i\in[D]} c_i p_i^g\right)^{\frac{1}{g}}.$$

4 Compress oracle and our model of computation

In this subsection, we recall the technique introduced by Zhandry [Zha19]. This part is adapted from [CGLQ20]. We will show five equivalent oracle forms: the standard/phase oracle, the compressed standard/phase oracle and the compressed Hadamard oracle.

In most cases in this paper, the computation is about a state on five registers.

- $\mathbf{X}$ is the register that stores either a oracle query or an answer waiting to be written.
- $\mathbf{U}$ is the register that stores the oracle's response or is used to store phase for the output process (will explain later).
- $\mathbf{W}$ is the register that stores as ancilla qubits in the computation.
- $\mathbf{R}$ is the output register that is used to store answers.
- $\mathbf{D}$ is the register that stores the random oracle H or its database D in the compressed oracle view.

We will explain how $\mathbf{R}$ works in Section 7.2. Intuitively, it can always be viewed as the tensor product of M' sub-registers with n'-qubits each for some M' and n'. The algorithm is only allowed to make Output gate that on state $|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}}$. This gate will add u on the sub-register corresponding to x on $\mathbf{R}$. We will write this process as:

$$\mathsf{Output}\,|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |H\rangle_{\mathbf{D}} = |x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r \oplus (x,u)\rangle_{\mathbf{R}} \otimes |H\rangle_{\mathbf{D}}.$$

Standard oracle. Now we switch to our normal oracle setting. Let H be a random oracle $[M] \to [N]$ where $N = 2^n$. We can view an algorithm that runs in the random oracle model with respect to H as the algorithm itself concatenating with a random oracle register $\mathbf{D}$ that is initialized to $\sum_H |H\rangle \langle H|_{\mathbf{D}}$ (ignoring the normalizing factor). The register $\mathbf{D}$ stores the random function $|H\rangle_{\mathbf{D}} = |H(1)\rangle |H(2)\rangle \cdots |H(N)\rangle$. The oracle unitary StO can be written as follows:

$$\mathsf{StO}\,|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |H\rangle_{\mathbf{D}} = |x\rangle_{\mathbf{X}} |u \oplus H(x)\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |H\rangle_{\mathbf{D}},$$

Note that there are some format dismatch in $\mathbf{X}$ and $\mathbf{U}$ when running StO and Output. We can assume that the algorithm itself can disambiguous on its own since it knows which type of query it is using. In general algorithm consists of interleaving StO, Output, local quantum unitaries U_i operate only on registers $\mathbf{X}\otimes\mathbf{U}\otimes\mathbf{W}$ and a final computational measurement on the output register $\mathbf{R}$. The following proposition tells that the output distribution using a standard oracle is exactly the same as using a random oracle.

Lemma 4.1 ([Zha19, Lemma 2]). *Let $\mathcal{A}$ be an (unbounded) quantum algorithm making oracle queries. The output of $\mathcal{A}$ given a random function H is exactly identical to the output of $\mathcal{A}$ given access to a standard oracle. Therefore, a random oracle with quantum query access can be perfectly simulated as a standard oracle.*

Phase oracle. Define the unitary V as $(I_{\mathbf{X}} \otimes H^{\otimes n} \otimes I_{\mathbf{W}\otimes\mathbf{R}\otimes\mathbf{D}})$ which applies $H^{\otimes n}$ on the answer register $\mathbf{U}$. Define the phase oracle operator $\mathsf{PhO} := V^\dagger \cdot \mathsf{StO} \cdot V$.

$$\begin{aligned}
&\mathsf{PhO}\,|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |H\rangle_{\mathbf{D}} \\
=&V^\dagger \cdot \mathsf{StO} \cdot \frac{1}{\sqrt{N}} \sum_{y\in[N]} (-1)^{\langle u,y\rangle} |x\rangle_{\mathbf{X}} |y\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |H\rangle_{\mathbf{D}} \\
=&V^\dagger \cdot \frac{1}{\sqrt{N}} \sum_{y\in[N]} (-1)^{\langle u,y\rangle} |x\rangle_{\mathbf{X}} |y + H(x)\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |H\rangle_{\mathbf{D}} \\
=&\frac{1}{N} \sum_{y,y'\in[N]} (-1)^{\langle u,y\rangle + \langle y+H(x),y'\rangle} |x\rangle_{\mathbf{X}} |y'\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |H\rangle_{\mathbf{D}} \\
=&\frac{1}{N}(-1)^{\langle u,H(x)\rangle} \sum_{y,y'\in[N]} (-1)^{\langle y+H(x),y'+u\rangle} |x\rangle_{\mathbf{X}} |y'\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |H\rangle_{\mathbf{D}} \\
=&|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes (-1)^{\langle u,H(x)\rangle} |H\rangle_{\mathbf{D}}.
\end{aligned}$$

The following lemma states that a phase oracle is equivalent to a standard oracle.

Lemma 4.2 ([Zha19, Lemma 3]). *Let $\mathcal{A}$ be an (unbounded) quantum algorithm making queries to a standard oracle. Let $\mathcal{B}$ be the algorithm that is identical to $\mathcal{A}$, except it performs V and $V^\dagger$ before and after each query. Then the output distributions of $\mathcal{A}$ (given access to a standard oracle) and $\mathcal{B}$ (given access to a phase oracle) are identical. Therefore, a quantum random oracle can be perfectly simulated as a phase oracle.*

Compressed standard oracle. The compressed standard oracle can be viewed a type of lazy sampling technique. Instead of initializing H at the very beginning, the compress oracle creates a database $|D\rangle_{\mathbf{D}} = |D(1)\rangle_{\mathbf{D}_1} |D(2)\rangle_{\mathbf{D}_2} \cdots |D(N)\rangle_{\mathbf{D}_N}$ where $D(x) \in \mathbb{F}_N \cup \{\perp\}$ and $|D\rangle$ is initialized to $|\emptyset\rangle_{\mathbf{D}} = |\perp, \perp, \cdots, \perp\rangle$ where $\perp$ is a symbol that indicates the lack of information of the algorithm on certain function value. Let $|D|$ denote the number of entries in D that are not $\perp$.

The database is initialized as an empty list D_0 of length N, in other words, it is initialized as the pure state $|\emptyset\rangle := |\perp, \perp, \cdots, \perp\rangle$. Let $|D|$ denote the number of entries in D that are not $\perp$. Define $D(x)$ to be the x-th entry and define $\mathbf{D}_x$ be the register for that entry.

For any D and x such that $D(x) = \perp$, we define $D \cup (x, u)$ to be the database D', such that for every $x' \neq x$, $D'(x') = D(x)$ and at the input x, $D'(x) = u$.

The compressed standard oracle is the unitary $\mathsf{CStO} := \mathsf{StdDecomp} \cdot \mathsf{CStO}' \cdot \mathsf{StdDecomp}$, where

- CStO' writes $D(x)$ to the answer register $\mathbf{U}$ by writing $u \oplus D(x)$ into it when $D(x) \neq \perp$ as usual but does nothing when $D(x) = \perp$. Or to say that we can define addition for $\perp$: $u \oplus \perp = u$, $\forall u \in [N]$. Formally,

$$\mathsf{CStO}' |x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}} = |x\rangle_{\mathbf{X}} |u \oplus D(x)\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}}.$$

- When the algorithm queries, the database calls $\mathsf{StdDecomp}$ which unfolds the database and samples a value y for positions that the algorithm does not know what the value is. More specifically, $\mathsf{StdDecomp} |x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}} := |x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes \mathsf{StdDecomp}_x |D\rangle_{\mathbf{D}}$, where $\mathsf{StdDecomp}_x$ works on $\mathbf{D}_x$.
  - If $D(x) = \perp$, $\mathsf{StdDecomp}_x$ maps $|\perp\rangle$ to $\frac{1}{\sqrt{N}} \sum_{y \in [N]} |y\rangle$.
  - If $D(x) \neq \perp$, $\mathsf{StdDecomp}_x$ works on the x-th register, and it is an identity on $\frac{1}{\sqrt{N}} \sum_{y \in [N]} (-1)^{\langle u, y\rangle} |y\rangle$ for all $u \neq 0$; it maps the uniform superposition $\frac{1}{\sqrt{N}} \sum_{y \in [N]} |y\rangle$ to $|\perp\rangle$.
    More formally, for a D' such that $D'(x) = \perp$,

$$\mathsf{StdDecomp}_x \frac{1}{\sqrt{N}} \sum_{y \in [N]} (-1)^{\langle u, y\rangle} |D' \cup (x, y)\rangle_{\mathbf{D}} = \frac{1}{\sqrt{N}} \sum_{y \in [N]} (-1)^{\langle u, y\rangle} |D' \cup (x, y)\rangle_{\mathbf{D}} \text{ for any } u \neq 0,$$

    and,

$$\mathsf{StdDecomp}_x \frac{1}{\sqrt{N}} \sum_{y \in [N]} |D' \cup (x, y)\rangle_{\mathbf{D}} = |D'\rangle_{\mathbf{D}}.$$

  Intuitively, it swaps a uniform superposition $\frac{1}{\sqrt{N}} \sum_{y \in [N]} |y\rangle$ with $|\perp\rangle$ on $\mathbf{D}_x$ and does nothing on other orthogonal basis. So it is a well defined unitary.

Zhandry proves that, StO and CStO are perfectly indistinguishable by any *unbounded* quantum algorithm.

Lemma 4.3 ([Zha19, Lemma 4])**.** *Let $\mathcal{A}$ be an (unbounded) quantum algorithm making oracle queries. The output of $\mathcal{A}$ given access to the standard oracle is exactly identical to the output of $\mathcal{A}$ given access to a compressed standard oracle.*

In this work, we only consider query complexity, and thus simulation efficiency is irrelevant to us.

Compressed phase oracle The compressed phase oracle is the unitary $\mathsf{CPhO} := \mathsf{StdDecomp} \cdot \mathsf{CPhO}' \cdot \mathsf{StdDecomp}$, where

$$\begin{aligned} &\mathsf{CPhO}' \left|x\right\rangle_{\mathbf{X}} \left|u\right\rangle_{\mathbf{U}} \left|w\right\rangle_{\mathbf{W}} \left|r\right\rangle_{\mathbf{R}} \otimes \left|D\right\rangle_{\mathbf{D}} \\ =& \left|x\right\rangle_{\mathbf{X}} (-1)^{\langle u, D(x)\rangle} \left|u\right\rangle_{\mathbf{U}} \left|w\right\rangle_{\mathbf{W}} \left|r\right\rangle_{\mathbf{R}} \otimes \left|D\right\rangle_{\mathbf{D}} \\ =& \left|x\right\rangle_{\mathbf{X}} \left|u\right\rangle_{\mathbf{U}} \left|w\right\rangle_{\mathbf{W}} \left|r\right\rangle_{\mathbf{R}} \otimes (-1)^{\langle u, D(x)\rangle} \left|D\right\rangle_{\mathbf{D}} . \end{aligned}$$

Redefine the unitary V as $(I_{\mathbf{X}} \otimes H^{\otimes n} \otimes I_{\mathbf{W}\otimes\mathbf{R}\otimes\mathbf{D}})$. Note that $\mathsf{CPhO}' = V^\dagger \cdot \mathsf{CStO}' \cdot V$ and V commutes with $\mathsf{StdDecomp}$. Thus

$$\begin{aligned} \mathsf{CPhO} =& \mathsf{StdDecomp} \cdot \mathsf{CPhO}' \cdot \mathsf{StdDecomp} \\ =& \mathsf{StdDecomp} \cdot V^\dagger \cdot \mathsf{CStO}' \cdot V \cdot \mathsf{StdDecomp} \\ =& V^\dagger \cdot \mathsf{StdDecomp} \cdot \mathsf{CStO}' \cdot \mathsf{StdDecomp} \cdot V \\ =& V^\dagger \cdot \mathsf{CStO} \cdot V. \end{aligned}$$

By this equivalence, we obtain the following corollary:

Corollary 4.4. *Any quantum algorithm $\mathcal{A}$ equipped with a quantum random oracle can be perfectly simulated by another algorithm $\mathcal{B}$ equipped with a compressed phase oracle.*

The following lemma states that the size of the database is at most the number of queries.

Lemma 4.5 ([CGLQ20], Lemma 2.6)**.** *Let $\mathcal{A}$ be a quantum algorithm making at most T queries to a compressed phase oracle. The overall state of $\mathcal{A}$ and the oracle database can be written as*

$$\sum_{x,u,w,r,D:|D|\le T} \alpha_{x,u,w,r,D} \left|x, u, w, r\right\rangle \otimes \left|D\right\rangle .$$

Moreover, it is true even if the state is conditioned on arbitrary outcomes (with non-zero probability) of $\mathcal{A}$'s intermediate measurements.

Here we further formalize the way an algorithm interacts with a compressed phase oracle. For a quantum algorithm $\mathcal{A}$ interacting with a compressed phase oracle CPhO, suppose it makes T queries in total. Then the algorithm is equivalent to applying a sequence of unitaries on the joint system $\mathbf{X} \otimes \mathbf{U} \otimes \mathbf{W} \otimes \mathbf{R} \otimes \mathbf{D}$ to a certain initialized state and then measuring the output register $\mathbf{R}$ to obtain the output. Without loss of generality, we initialize the system to be

$$\left|\Phi_{\mathsf{init}}\right\rangle = \left|0\right\rangle_{\mathbf{X}} \left|0\right\rangle_{\mathbf{U}} \left|0\right\rangle_{\mathbf{W}} \left|0\right\rangle_{\mathbf{R}} \otimes \left|\emptyset\right\rangle_{\mathbf{D}} .$$

The algorithm $\mathcal{A}$ could be characterized as a sequence of local unitaries and compressed phase oracle queries,

$$(U_T \otimes I_{\mathbf{D}}) \cdot \mathsf{CPhO} \cdot (U_{T-1} \otimes I_{\mathbf{D}}) \cdot \mathsf{CPhO} \cdots \mathsf{CPhO} \cdot (U_0 \otimes I_{\mathbf{D}}),$$

where $U_0, \cdots, U_T$ are unitaries on the algorithm space $\mathbf{X} \otimes \mathbf{U} \otimes \mathbf{W} \otimes \mathbf{R}$. Then,we denote $\left|\Phi_i\right\rangle$ to be the intermediate system state after applying i oracle queries and some local unitaries,

$$\left|\Phi_i\right\rangle = (U_i \otimes I_{\mathbf{D}}) \cdot \mathsf{CPhO} \cdot (U_{i-1} \otimes I_{\mathbf{D}}) \cdot \mathsf{CPhO} \cdots \mathsf{CPhO} \cdot (U_0 \otimes I_{\mathbf{D}}) \left|\Phi\right\rangle_0 ,$$

where $i \in [T]$.

We now relate the winning probability of the algorithm to the final state of the system, $|\Phi_T\rangle$. For the scenario of an algorithm interacting with a compressed oracle, we determine whether it wins or not by measuring the database $\mathbf{D}$ and output register $\mathbf{R}$ first, and then check whether the outcoming database D and output r satisfy the goal of the algorithm. More formally, suppose that we form the goal of the algorithm as a relation $\mathcal{R}$ over all pairs of possible output and database, and the algorithm wins if and only if it outputs a pair of (D, r) such that $(D, r) \in \mathcal{R}$. For example, in the case of the collision finding problem, $\mathcal{R} = \{(r, D) | r = (x, x'), x \neq x', D(x) = D(x') \neq \perp\}$. We then define Π_{win} as to be a projector onto all basis states $|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}}$ such that $(r, D) \in \mathcal{R}$, in other words projecting on all "winning" output. Therefore, the winning probability of $\mathcal{A}$ equals to $\|\Pi_{\mathsf{win}}|\Phi_T\rangle\|^2$.

The following lemma establishes a connection between the winning probability when interacting with a random oracle and when interacting with a compressed oracle.

Lemma 4.6 ([Zha19, Lemma 5]). *Consider a quantum algorithm $\mathcal{A}$ making phase queries to a random oracle H and outputting tuples $(x_1, \cdots, x_k, y_1, \cdots, y_k, z)$. Let $\mathcal{R}$ be a collection of such tuples. Let p be the probability that, $\mathcal{A}$ outputs a tuple such that (1) the tuple is in $\mathcal{R}$, and (2) $H(x_i) = y_i$ for all i. Now consider running $\mathcal{A}$ with the oracle* CPhO*, and suppose the database register $\mathbf{D}$ is measured after $\mathcal{A}$ produces its output. Let p' be the probability that, $\mathcal{A}$ outputs a tuple such that (1) the tuple is in $\mathcal{R}$, and (2) $D(x_i) = y_i \neq \perp$ for all i. Then we have $\sqrt{p} \leq \sqrt{p'} + \sqrt{k/N}$, where N is the range size.*

Remark 4.7. *The Lemma 4.6 is a little different from the original statement of the lemma 5 in Zhandry's work [Zha19], but it is not hard to see that they are equivalent. In the original lemma[Zha19], the algorithm $\mathcal{A}$ is making standard queries in the first scenario, and making queries to oracle* CStO *in the second scenario. Here in the statement the algorithm use phase queries and* CPhO *oracle instead. However, since we show previously that* $\mathsf{PhO} := V^\dagger \cdot \mathsf{StO} \cdot V$*, and that* $\mathsf{CPhO} = V^\dagger \cdot \mathsf{CStO} \cdot V$*, the analysis could similarly apply to the phase oracle setting in Lemma 4.6.*

The following lemma shows how CPhO affects the database under the standard basis:

Lemma 4.8 ([HM23], Lemma 4.1). *If the operator* CPhO *is applied to a basis state $|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}}$ where $u \neq 0$ then the register $|D(x)\rangle_{\mathbf{D}_x}$ is mapped to*

- $\displaystyle\sum_{y \in [N]} \frac{(-1)^{\langle u,y\rangle}}{\sqrt{N}} |y\rangle$ *if $D(x) = \perp$*
- $\frac{(-1)^{\langle u,D(x)\rangle}}{\sqrt{N}} |\perp\rangle + \frac{1+(-1)^{\langle u,D(x)\rangle}(N-2)}{N} |D(x)\rangle + \sum_{y \in [N]\setminus\{D(x)\}} \frac{1-(-1)^{\langle u,y\rangle}-(-1)^{\langle u,D(x)\rangle}}{N} |y\rangle$ *if $D(x) \in [N]$*

and other registers are unchanged. If $u = 0$ then none of the registers are changed.

Hadamard Oracle Define $|\widehat{v}\rangle_{\mathbf{D}_x} := \sum_{y \in [N]} (-1)^{\langle x,y\rangle} |y\rangle_{\mathbf{D}_x}$ for $v \neq 0$ and $|\widehat{0}\rangle_{\mathbf{D}_x} = |\perp\rangle_{\mathbf{D}_x}$. Define $\mathbf{D}_{-x} := \otimes_{x' \in [M]-\{x\}} \mathbf{D}_{x'}$. Then we have

$$\begin{aligned}
&\mathsf{CPhO}\, |x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |\widehat{d}\rangle_{\mathbf{D}_x} \otimes |D\rangle_{\mathbf{D}_{-x}} \\
=& \mathsf{StdDecomp} \cdot \mathsf{CPhO}' \cdot \mathsf{StdDecomp}\, |x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |\widehat{d}\rangle_{\mathbf{D}_x} \otimes |D\rangle_{\mathbf{D}_{-x}} \\
=& \mathsf{StdDecomp} \cdot \mathsf{CPhO}'\, |x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes \left(\sum_{y \in [N]} \frac{(-1)^{\langle d,y\rangle}}{\sqrt{N}} |y\rangle_{\mathbf{D}_x} \right) \otimes |D\rangle_{\mathbf{D}_{-x}}
\end{aligned}$$

$$=\mathsf{StdDecomp}\,|x\rangle_{\mathbf{X}}\,|u\rangle_{\mathbf{U}}\,|w\rangle_{\mathbf{W}}\,|r\rangle_{\mathbf{R}}\otimes\left(\sum_{y\in[N]}\frac{(-1)^{\langle d+u,y\rangle}}{\sqrt{N}}\,|y\rangle_{\mathbf{D}_x}\right)\otimes|D\rangle_{\mathbf{D}_{-x}}$$
$$=|x\rangle_{\mathbf{X}}\,|u\rangle_{\mathbf{U}}\,|w\rangle_{\mathbf{W}}\,|r\rangle_{\mathbf{R}}\otimes|\widehat{d\oplus u}\rangle_{\mathbf{D}_x}\otimes|D\rangle_{\mathbf{D}_{-x}}.$$

The compressed Hadamard oracle HaO uses the database register under a different basis. Define $\mathsf{HaO}\,|x\rangle_{\mathbf{X}}\,|u\rangle_{\mathbf{U}}\,|w\rangle_{\mathbf{W}}\,|r\rangle_{\mathbf{R}}\otimes|D\rangle_{\mathbf{D}}:=|x\rangle_{\mathbf{X}}\,|u\rangle_{\mathbf{U}}\,|w\rangle_{\mathbf{W}}\,|r\rangle_{\mathbf{R}}\otimes|D\oplus(x,u)\rangle_{\mathbf{D}}$ where $D\oplus(x,u)$ is the database resulted by applying $+u$ under $\mathbb{F}_2^n$ to the value on $\mathbf{D}_x$ register on D. Since HaO and CPhO are two identical operator defined under two different basis and the basis choice of $\mathbf{D}$ does not affect the algorithm, we have the following lemma.

Lemma 4.9. *Let $\mathcal{A}$ be an (unbounded) quantum algorithm making oracle queries. The output of $\mathcal{A}$ given access to the compressed phase oracle is exactly identical to the output of $\mathcal{A}$ given access to a compressed Hadamard oracle.*

Proof. A Hadamard oracle is essentially a phase oracle under Hadamard basis. □

5 Finding collision implies many entries in the compressed oracle

Below we present a new lemma showing intuitively that, if an algorithm can find a pair of collision with high probability, under the view of compressed oracle, the expected number of non-$\perp$ entries in the compressed oracle can not be very small. Our main proof idea is to first bound the increment in winning probability after each of the T queries, then show that the overall winning probability is bounded by a combination of each increment. In Proposition 5.5, we carefully bound the weight increase on the basis where the database contains collision, by analyzing the evolution of the database during oracle query separately according to their original non-$\perp$ entries numbers. Then in Theorem 5.6, we combine the winning probability increment for each query together, and connect it to the final winning probability.

We clarify that the lemma works for a slightly generalized version of the collision finding problem, namely the *Labeled Collision Finding* problem, defined as following:

Definition 5.1 (Labeled Collision Finding)**.** *Given a random oracle $G:[M]\to[N_0]$, as well as a label function $\mathsf{Lbl}:[M]\to[N]$, for some fixed y^* the goal is to find a pair $(x,x')\in[M]^2$ such that:*

- $x\neq x'$.
- $\mathsf{Lbl}(x)=\mathsf{Lbl}(x')=y^*$.
- $G(x)=G(x')$.

We call the satisfying pair (x,x') a pair of label-y^* collisions.

Notice that when the label function maps all input $x\in[M]$ to the same label, this problem becomes the standard collision finding problem, requiring only $G(x)=G(x'),x\neq x'$. For nontrivial label functions, the problem is essentially finding a collision pair of random oracle G that are also of the same label. We start by introducing a set of projectors on the joint system of algorithm register and database, we may use them to categorize which condition the database is on after or before each oracle query.

Definition 5.2. *Given a label function $\mathsf{Lbl}:[M]\to[N]$, we define the following projectors by giving the basis states on which they project:*

- Π_v: *all basis states* $|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}}$ *such that there is* v *non-*$\perp$ *entries of the label* y^**, i.e.* $\sum_{x\in[M]} \mathbb{1}\{D(x) \neq \perp, \mathsf{Lbl}(x) = y^*\} = v$*. Let* $|D|_{y^*}$ *denote the number of non-*$\perp$ *entries in* D *with label* y^*
- P*: all basis states* $|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}}$ *such that* $|D\rangle_{\mathbf{D}}$ *contains a pair of labeled-collision, i.e.,* $\exists x \neq x'$, $D(x) = D(x') \neq \perp, \mathsf{Lbl}(x) = \mathsf{Lbl}(x') = y^*$.
- O*: all basis states* $|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}}$ *such that:*
  (1) there are no label-y^* *collisions in* D*; (2)*$\mathsf{Lbl}(x) \neq y^*$.
- Q*: all basis states* $|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}}$ *such that:*
  (1) there are no label-y^* *collisions in* D*; (2)* $D(x) = \perp$*; (3)* $\mathsf{Lbl}(x) = y^*$*; (4)* $u \neq 0$.
- R*: all basis states* $|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}}$ *such that:*
  (1) there are no label-y^* *collisions in* D*; (2)* $D(x) \neq \perp$*; (3)* $\mathsf{Lbl}(x) = y^*$*; (4)* $u \neq 0$.
- S*: all basis states* $|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}}$ *such that:*
  (1) there are no label-y^* *collisions in* D*; (2)* $u = 0$*.(3)* $\mathsf{Lbl}(x) = y^*$.
- *Also define* $P_v = \Pi_v P$, $O_v = \Pi_v O$, $Q_v = \Pi_v Q$, $R_v = \Pi_v R$, $S_v = \Pi_v S$.

Then we have the following properties:

- $\sum_{v=0}^{M} \Pi_v = I$.
- $P + O + Q + R + S = I$, and that P, O, Q, R, S are orthogonal.
- $P_v + O_v + Q_v + R_v + S_v = \Pi_v$, $\forall v \in [0, M]$, and that P_v, O_v, Q_v, R_v, S_v are orthogonal.
- $\forall v \neq v'$, their corresponding projectors are orthogonal, it also holds for P_v, O_v, Q_v, R_v, and S_v.
- Notice that the whether a state lies in the projected space of P and Π_v only depends on the $\mathbf{D}$ register information of the state, so these projectors commute with local unitaries on algorithm space, $\Pi_v(U \otimes I_{\mathbf{D}}) = (U \otimes I_{\mathbf{D}})\Pi_v$, and it is the same for P, O, Q, R, S.

Based on the projectors, we could define the average non-empty size of the database, throughout the process of an algorithm interacting with compressed oracle.

Definition 5.3. *Suppose we have an algorithm* $\mathcal{A}$ *interacting with a compressed phase oracle* CPhO*, with a total of* T *queries,* CPhO *corresponds to a database with* M *entries and each entry has an element in* $\mathbb{F}_{N_0} \cup \{\perp\}$*. Consider the case when we measure the database register after* i*-th query, we define*

$$p_{i,v} = \|\Pi_v |\Phi_i\rangle\|^2$$

to be the probability of collapsing into a database with v *non-*$\perp$ *entries with label* y^**,* $\forall i \in [T], v \in [0, M]$*. Then, we define*

$$V_i = \sum_{v=0}^{M} v \cdot p_{i,v}$$

to be the expectation of the number of non-$\perp$ *entries with label* y^* *after* i *queries,* $\forall i \in [T]$*. Furthermore, we define* V *for the case when we randomly pick* $i \leftarrow [T]$ *and measure the database register* $\mathbf{D}$ *after* i *oracle queries,*

$$V = \frac{1}{T}\sum_{i=1}^{T} V_i = \frac{1}{T}\sum_{i=1}^{T}\sum_{v=0}^{M} v \|\Pi_v |\Phi_i\rangle\|^2 .$$

Remark 5.4. *Notice that after* i *queries to the oracle, we have* $V_i = \sum_{v=0}^{M} v \cdot p_{i,v}$*. It seems that we are adding up* M *terms, but from Lemma 4.5 we know that at the moment the database register only has support over states that have at most* i *non-*$\perp$ *entries, and thus the maximal single-labeled non-*$\perp$ *capacity is also at most* i*. Therefore, for* $v > i$*, we have* $p_{i,v} = 0$*, and* $V_i = \sum_{v=0}^{i} v \cdot p_{i,v}$.

Now we move on to prove a proposition, intuitively bounding the increase in success probability after each oracle query.

Proposition 5.5. *For an arbitrary state $|\psi\rangle$ in joint space* $\mathbf{XUWRD}$*, and arbitrary local unitary U on algorithm space* $\mathbf{XUWR}$*, we have*

$$\|P_v \cdot \mathsf{CPhO} \cdot (U \otimes I_{\mathbf{D}}) \cdot (I-P)\,|\psi\rangle\| \leq 3\sqrt{\frac{(v-1)}{N_0}}\,\|\Pi_v\,|\psi\rangle\| + \sqrt{\frac{v-1}{N_0}}\,\|\Pi_{v-1}\,|\psi\rangle\|\,,$$

where N_0 is the range size of the compressed oracle, $v > 0$ and P, Π_v, P_v are projectors defined in Definition 5.2.

Proof. Define $|\phi\rangle = (U \otimes I_{\mathbf{D}})\,|\psi\rangle$. Recall that $U \otimes I_{\mathbf{D}}$ commute with $I-P$, then

$$P_v \cdot \mathsf{CPhO} \cdot (U \otimes I_{\mathbf{D}}) \cdot (I-P)\,|\psi\rangle = P_v \cdot \mathsf{CPhO} \cdot (I-P)\,|\phi\rangle\,. \tag{1}$$

We now analyze the resulting state after applying CPhO to each support of $(I-P)\,|\phi\rangle$, considering that $(I-P)\,|\phi\rangle = O\,|\phi\rangle + Q\,|\phi\rangle + R\,|\phi\rangle + S\,|\phi\rangle$.

We first consider $\mathsf{CPhO} \cdot O\,|\phi\rangle$. Since $\mathsf{Lbl}(x) \neq y^*$, adding this point to the database does not change the structure of the database on label-y^* entries. We have

$$P_v \cdot \mathsf{CPhO} \cdot O\,|\phi\rangle = 0$$

because $P_v\,|\phi\rangle = 0$.
We than consider $\mathsf{CPhO} \cdot Q\,|\phi\rangle$, which stands for the case of filling a new entry with label y^* into the database. Since $Q\,|\phi\rangle = \sum_{b=0}^{M} Q_b\,|\phi\rangle$, we analyze them separately. Assume $Q_b\,|\phi\rangle = \sum_{x,u,w,r,D'} \alpha^{(b)}_{x,u,w,r,D'}\,|x,u,w,r\rangle\,|D'\rangle$, where x,u,w,r takes arbitrary value from the register space of $\mathbf{X},\mathbf{U},\mathbf{W},\mathbf{R}$, and D' takes the value of all database that $D'(x) = \bot$ and $|D|_{y^*} = b$. Then by Lemma 4.8, we have

$$\begin{aligned}
P_v \cdot \mathsf{CPhO} \cdot Q\,|\phi\rangle &= \sum_{b=0}^{M} P_v \cdot \mathsf{CPhO} \cdot Q_b\,|\phi\rangle \\
&= \sum_{b=0}^{M} P_v \cdot \mathsf{CPhO}\left(\sum_{\substack{x,u,w,r,D' \\ |D|_{y^*}=b}} \alpha^{(b)}_{x,u,w,r,D'}\,|x,u,w,r\rangle\,|D'\rangle\right) \\
&= \sum_{b=0}^{M} \sum_{\substack{x,u,w,r,D' \\ |D|_{y^*}=b}} \alpha^{(b)}_{x,u,w,r,D'} P \cdot \Pi_v\left(|x,u,w,r\rangle \otimes \sum_{y\in[N_0]} \frac{(-1)^{\langle u,y\rangle}}{\sqrt{N_0}}\,|D' \cup (x,y)\rangle\right) \\
&= \sum_{\substack{x,u,w,r,D' \\ |D|_{y^*}=v-1}} \alpha^{(v-1)}_{x,u,w,r,D'} P\left(|x,u,w,r\rangle \otimes \sum_{y\in[N_0]} \frac{(-1)^{\langle u,y\rangle}}{\sqrt{N_0}}\,|D' \cup (x,y)\rangle\right) \\
&= \sum_{\substack{x,u,w,r,D' \\ |D|_{y^*}=v-1}} \sum_{\substack{y \\ \exists x', D'(x')=y \text{ and } \mathsf{Lbl}(x)=y^*}} \alpha^{(v-1)}_{x,u,w,r,D'} \frac{(-1)^{\langle u,y\rangle}}{\sqrt{N_0}}\,|x,u,w,r\rangle\,|D' \cup (x,y)\rangle\,.
\end{aligned}$$

The fourth equality holds as we want $|x, u, w, r\rangle |D' \cup (x, y)\rangle$ to be inside the projected subspace of Π_v, so $|D' \cup (x, y)| = b + 1 = v$. The fifth equality holds as we want $|x, u, w, r\rangle |D' \cup (x, y)\rangle$ to be inside the projected subspace of P, so y needs to collide with one of the $v - 1$ existing different value inside D'. Then, we have

$$\begin{aligned}
\|P_v \cdot \mathsf{CPhO} \cdot Q |\phi\rangle\|^2 =& \sum_{\substack{x,u,w,r,D' \\ |D|_{y^*}=v-1}} \sum_{\substack{y \\ \exists x', D'(x')=y \text{ and } \mathsf{Lbl}(x)=y^*}} \left| \alpha^{(v-1)}_{x,u,w,r,D'} \frac{(-1)^{\langle u,y\rangle}}{\sqrt{N_0}} \right|^2 \\
\leq& \frac{1}{\sqrt{N_0}} \sum_{\substack{x,u,w,r,D' \\ |D|_{y^*}=v-1}} |D|_{y^*} \cdot \left| \alpha^{(v-1)}_{x,u,w,r,D'} \right|^2 \qquad (2) \\
=& \frac{v-1}{N_0} \|Q_{v-1} |\phi\rangle\|^2 .
\end{aligned}$$

We now consider $\mathsf{CPhO} \cdot R |\phi\rangle$, which stands for rerandomizing an existing entry in the database. Similarly, we assume $R_b |\phi\rangle = \sum_{x,u,w,r,D',y} \beta^{(b)}_{x,u,w,r,D',y} |x, u, w, r\rangle |D' \cup (x, y)\rangle$. Here x, u, w, r takes arbitrary value; D' still takes the value of all database that $D'(x) = \perp$ and $|D'|_{y^*} = b - 1$, so that $|D' \cup (x, y)| = b$; and y takes all value in $\{1, 2, \cdots, N_0 - 1\}$ such that $D' \cup (x, y)$ has no label-y^* collision, therefore the state lies in support of R_b. Then by Lemma 4.8, we have

$$\begin{aligned}
& P_v \cdot \mathsf{CPhO} \cdot R |\phi\rangle \\
=& \sum_{b=0}^{M} P_v \cdot \mathsf{CPhO} \cdot R_b |\phi\rangle \\
=& \sum_{b=0}^{M} P_v \cdot \mathsf{CPhO} \Bigg(\sum_{\substack{x,u,w,r,D',y \\ |D|_{y^*}=b-1, y\neq 0}} \beta^{(b)}_{x,u,w,r,D',y} |x, u, w, r\rangle |D' \cup (x, y)\rangle \Bigg) \\
=& \sum_{b=0}^{M} \sum_{\substack{x,u,w,r,D',y \\ |D|_{y^*}=b-1, y\neq 0}} \beta^{(b)}_{x,u,w,r,D',y} P \cdot \Pi_v \Bigg[|x, u, w, r\rangle \otimes \Bigg(\frac{(-1)^{\langle u,y\rangle}}{\sqrt{N_0}} |D'\rangle + \frac{1 + (-1)^{\langle u,y\rangle}(N_0 - 2)}{N_0} |D' \cup (x, y)\rangle \\
& + \sum_{y'\neq y} \frac{1 - (-1)^{\langle u,y'\rangle} - (-1)^{\langle u,y\rangle}}{N_0} |D' \cup (x, y')\rangle \Bigg) \Bigg] \\
=& \sum_{\substack{x,u,w,r,D',y \\ |D|_{y^*}=v, y\neq 0}} \beta^{(v+1)}_{x,u,w,r,D',y} P \Bigg(|x, u, w, r\rangle \otimes \frac{(-1)^{\langle u,y\rangle}}{\sqrt{N_0}} |D'\rangle \Bigg) + \sum_{\substack{x,u,w,r,D',y \\ |D|_{y^*}=v-1, y\neq 0}} \beta^{(v)}_{x,u,w,r,D',y} P \\
& \cdot \Bigg[|x, u, w, r\rangle \otimes \Bigg(\frac{1 + (-1)^{\langle u,y\rangle}(N_0 - 2)}{N_0} |D' \cup (x, y)\rangle + \sum_{y'\neq y} \frac{1 - (-1)^{\langle u,y'\rangle} - (-1)^{\langle u,y\rangle}}{N_0} |D' \cup (x, y')\rangle \Bigg) \Bigg] \\
=& \sum_{\substack{x,u,w,r,D',y \\ |D|_{y^*}=v-1, y\neq 0}} \sum_{\substack{y' \ s.t. \\ \exists x', D'(x')=y' \text{ and } \mathsf{Lbl}(x)=y^*}} \beta^{(v)}_{x,u,w,r,D',y} \frac{1 - (-1)^{\langle u,y'\rangle} - (-1)^{\langle u,y\rangle}}{N_0} |x, u, w, r\rangle |D' \cup (x, y')\rangle .
\end{aligned}$$

Similarly, the fourth equality holds as we want the database corresponds to each state to have v non-$\perp$ entries. The fifth equality holds as given that there is no collisions in $D \cup (x, y)$, the only term that could fall in the projected subspace of P is $|x,u,w,r\rangle\,|D' \cup (x,y')\rangle$, conditioning on that y' needs to collide with one of the $v-1$ existing different value inside D'. Then, we have

$$\begin{aligned}
\|P_v \cdot \mathsf{CPhO} \cdot R\,|\phi\rangle\|^2 = &\sum_{\substack{x,u,w,r,D',y \\ |D|_{y^*}=v-1, y\neq 0}} \sum_{\substack{y' \ s.t. \\ \exists x', D'(x')=y' \text{ and } \mathsf{Lbl}(x)=y^*}} \left| \beta^{(v)}_{x,u,w,r,D',y} \frac{1-(-1)^{\langle u,y'\rangle}-(-1)^{\langle u,y\rangle}}{N_0} \right|^2 \\
\leq &\frac{1}{N_0} \sum_{\substack{x,u,w,r,D',y \\ |D|_{y^*}=v-1, y\neq 0}} |D|_{y^*} \cdot \left|\beta^{(v)}_{x,u,w,r,D',y}\right| \cdot \left|1-(-1)^{\langle u,y'\rangle}-(-1)^{\langle u,y\rangle}\right|^2 \\
\leq &\frac{9(v-1)}{N_0} \sum_{\substack{x,u,w,r,D',y \\ |D|_{y^*}=v-1, y\neq 0}} \left|\beta^{(v)}_{x,u,w,r,D',y}\right|^2 \\
= &\frac{9(v-1)}{N_0} \|R_v\,|\phi\rangle\|^2 .
\end{aligned} \tag{3}$$

We finally consider $\mathsf{CPhO} \cdot S\,|\phi\rangle$, in which case the oracle query does not change the database. $\mathsf{CPhO}\,|x,0,w,r\rangle\,|D\rangle = (-1)^{\langle 0, D(x)\rangle}\,|x,0,w,r\rangle\,|D\rangle = |x,0,w,r\rangle\,|D\rangle$ for all x,w,r,D , so that

$$\mathsf{CPhO} \cdot S\,|\phi\rangle = S\,|\phi\rangle\,,\ \|P_v \cdot \mathsf{CPhO} \cdot S\,|\phi\rangle\| = 0. \tag{4}$$

Combining Equations (1) to (4), we have

$$\begin{aligned}
&\|P_v \cdot \mathsf{CPhO} \cdot (U \otimes I_{\mathbf{D}}) \cdot (I-P)\,|\psi\rangle\| \\
=&\|P_v \cdot \mathsf{CPhO} \cdot (I-P)\,|\phi\rangle\| \\
\leq&\|P_v \cdot \mathsf{CPhO} \cdot O\,|\phi\rangle\| + \|P_v \cdot \mathsf{CPhO} \cdot Q\,|\phi\rangle\| + \|P_v \cdot \mathsf{CPhO} \cdot R\,|\phi\rangle\| + \|P_v \cdot \mathsf{CPhO} \cdot S\,|\phi\rangle\| \\
\leq&\sqrt{\frac{v-1}{N_0}}\,\|Q_{v-1}\,|\phi\rangle\| + 3\sqrt{\frac{(v-1)}{N_0}}\,\|R_v\,|\phi\rangle\| .
\end{aligned} \tag{5}$$

Substituting $|\phi\rangle = (U \otimes I_{\mathbf{D}})\,|\psi\rangle$, since $(U \otimes I_{\mathbf{D}})$ commutes with Q_{v-1}, R_v, we have $\|Q_{v-1}\,|\phi\rangle\| = \|(U \otimes I_{\mathbf{D}})Q_{v-1}\,|\psi\rangle\| = \|Q_{v-1}\,|\psi\rangle\|$. Similarly, $\|R_v\,|\phi\rangle\| = \|R_v\,|\psi\rangle\|$. Finally, substituting these into Equation (5), we have

$$\begin{aligned}
\|P_v \cdot \mathsf{CPhO} \cdot (U \otimes I_{\mathbf{D}}) \cdot (I-P)\,|\psi\rangle\| \leq&\sqrt{\frac{v-1}{N_0}}\,\|Q_{v-1}\,|\psi\rangle\| + 3\sqrt{\frac{(v-1)}{N_0}}\,\|R_v\,|\psi\rangle\| \\
\leq&\sqrt{\frac{v-1}{N_0}}\,\|\Pi_{v-1}\,|\psi\rangle\| + 3\sqrt{\frac{(v-1)}{N_0}}\,\|\Pi_v\,|\psi\rangle\| .
\end{aligned}$$

□

We now proceed to the main helper lemma of this section.

Theorem 5.6 (Helper lemma on bounded capacity collision finding)**.** *For any fixed label function* $\mathsf{Lbl} : [M] \to [N]$*, let $\mathcal{A}$ be a quantum algorithm making T queries to a compressed phase oracle, its goal is to find a pair of label-y^* collision. Define V according to Definition 5.3. Then its success probability is at most $O(T^2V/N_0)$, where N_0 is the range size. Also, the constant in O does not depend on* Lbl.

The lemma basically shows that if we want a quantum algorithm to find collisions efficiently, the average non-$\perp$ entry number has to be sufficiently large. If an algorithm achieves the optimal $T = O\left(N_0^{1/3}\right)$ running time, then during its interaction with the compressed oracle, the database has an average of $O(T)$ non-$\perp$ entry. In other words, it is impossible to find an algorithm that can achieve optimal time efficiency while only touching few entries in the database.

Proof. Define Π_{col} as a projector onto all basis states $|x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} \otimes |D\rangle_{\mathbf{D}}$ such that $r = (r_1, r_2)$ satisfies $D(r_1) = D(r_2) \neq \perp, \mathsf{Lbl}(r_1) = \mathsf{Lbl}(r_2) = y^*, r_1 \neq r_2$. The projector describes the case when the algorithm $\mathcal{A}$ outputs a valid pair of collision that could be verified by checking the database, and the probability of such case equals to $\|\Pi_{\mathsf{col}} |\Phi_T\rangle\|^2$, while recall that $|\Phi_T\rangle$ is the final state after T-th query. By Lemma 4.6, we have

$$\sqrt{\Pr[\mathcal{A}\ wins]} \leq \|\Pi_{\mathsf{col}} |\Phi_T\rangle\| + \sqrt{2/N_0}. \tag{6}$$

Notice that the subspace that Π_{col} projects onto strictly contains the subspace that P projects onto, as P only requires the existence of a collision inside D, while Π_{col} also requires the output r to be that collision.In other words, we have $\Pi_{\mathsf{col}} P = P \Pi_{\mathsf{col}} = \Pi_{\mathsf{col}}$. Since projectors do not increase the norm of a vector, we have

$$\|\Pi_{\mathsf{col}} |\Phi_T\rangle\| = \|\Pi_{\mathsf{col}} P |\Phi_T\rangle\| \leq \|P |\Phi_T\rangle\|. \tag{7}$$

We then focus on the projected state $P|\Phi_T\rangle$,

$$\begin{aligned} P|\Phi_T\rangle =& P \cdot \mathsf{CPhO} \cdot (U_{T-1} \otimes I_{\mathbf{D}}) \cdot \mathsf{CPhO} \cdots \mathsf{CPhO} \cdot (U_0 \otimes I_{\mathbf{D}}) |\Phi_0\rangle \\ =& P \cdot \mathsf{CPhO} \cdot (U_{T-1} \otimes I_{\mathbf{D}}) \cdot (P + I - P) \cdot \mathsf{CPhO} \cdots (P + I - P) \cdot \mathsf{CPhO} \cdot (U_0 \otimes I_{\mathbf{D}}) \cdot (P + I - P) |\Phi_0\rangle \\ =& \sum_{i=1}^{T} P \cdot \mathsf{CPhO} \cdot (U_{T-1} \otimes I_{\mathbf{D}}) \cdot P \cdot \mathsf{CPhO} \cdot (U_{T-2} \otimes I_{\mathbf{D}}) \cdots P \cdot \mathsf{CPhO} \cdot (U_{i-1} \otimes I_{\mathbf{D}}) \cdot (I - P) \\ & \cdot \mathsf{CPhO} \cdot (U_{i-2} \otimes I_{\mathbf{D}}) \cdot \mathsf{CPhO} \cdots \mathsf{CPhO} \cdot (U_0 \otimes I_{\mathbf{D}}) |\Phi_0\rangle. \end{aligned}$$

Strictly speaking, there should also be a term $P|\Phi_0\rangle$ added to the right hand side, but as $|\Phi_0\rangle$ stands for an empty database, we have $|\Phi_0\rangle = 0$, so we omit this term. For $i \in [T]$, we define $|\phi_i\rangle$ to be the state projected onto the case that: (1) before the i-th query there are no collisions in the database; (2) the i-th query forms a collision. Specifically,

$$|\phi_i\rangle = P \cdot \mathsf{CPhO} \cdot (U_{i-1} \otimes I_{\mathbf{D}}) \cdot (I - P) \cdot \mathsf{CPhO} \cdot (U_{i-2} \otimes I_{\mathbf{D}}) \cdot \mathsf{CPhO} \cdots \mathsf{CPhO} \cdot (U_0 \otimes I_{\mathbf{D}}) |\Phi_0\rangle.$$

Then, we have

$$P|\Phi_T\rangle = \sum_{i=1}^{T} P \cdot \mathsf{CPhO} \cdot (U_{T-1} \otimes I_{\mathbf{D}}) \cdots P \cdot \mathsf{CPhO} \cdot (U_i \otimes I_{\mathbf{D}}) |\phi_i\rangle.$$

Take the norm of the state, and apply the Cauchy-Schwartz inequality, we have

$$\begin{aligned} \|P|\Phi_T\rangle\|^2 \leq & T \sum_{i=1}^{T} \|P \cdot \mathsf{CPhO} \cdot (U_{T-1} \otimes I_{\mathbf{D}}) \cdots P \cdot \mathsf{CPhO} \cdot (U_i \otimes I_{\mathbf{D}}) |\phi_i\rangle\|^2 \\ \leq & T \sum_{i=1}^{T} \||\phi_i\rangle\|^2 \end{aligned}$$

$$=T\sum_{i=1}^{T}\left\|\sum_{v=0}^{M}\Pi_v\left|\phi_i\right\rangle\right\|^2.$$

For $i \in [T]$, $v \in [0, M]$ we define $|\phi_{i,v}\rangle = \Pi_v |\phi_i\rangle$ to represent the case when there are v non-$\perp$ entries in the database, by the orthogonality of Π_v, we have

$$\|P\left|\Phi_T\right\rangle\|^2 \le T\sum_{i=1}^{T}\left\|\sum_{v=0}^{M}\left|\phi_{i,v}\right\rangle\right\|^2 = T\sum_{i=1}^{T}\sum_{v=0}^{M}\||\phi_{i,v}\rangle\|^2. \tag{8}$$

Now we focus on how to bound $\||\phi_{i,v}\rangle\|$. For arbitrary $i \in [T]$, $v \in [0, M]$, we have

$$\begin{aligned}|\phi_{i,v}\rangle =&P_v \cdot \mathsf{CPhO}\cdot(U_{i-1}\otimes I_{\mathbf{D}})\cdot(I-P)\cdot\mathsf{CPhO}\cdot(U_{i-2}\otimes I_{\mathbf{D}})\cdot\mathsf{CPhO}\cdots\mathsf{CPhO}\cdot(U_0\otimes I_{\mathbf{D}})\left|\Phi_0\right\rangle\\ =&P_v\cdot\mathsf{CPhO}\cdot(U_{i-1}\otimes I_{\mathbf{D}})\cdot(I-P)\left|\Phi_{i-1}\right\rangle.\end{aligned}$$

Also notice that $P_v = 0$ when $v = 0$, as a database without non-$\perp$ entry cannot contain collisions. Then, $|\phi_{i,0}\rangle = 0$. For other terms when $v > 0$, by applying Proposition 5.5, we have

$$\begin{aligned}\||\phi_{i,v}\rangle\| =&\|P_v\cdot\mathsf{CPhO}\cdot(U_{i-1}\otimes I_{\mathbf{D}})\cdot(I-P)\left|\Phi_{i-1}\right\rangle\|\\ \le&3\sqrt{\frac{v-1}{N_0}}\,\|\Pi_v\left|\Phi_{i-1}\right\rangle\| + \sqrt{\frac{v-1}{N_0}}\,\|\Pi_{v-1}\left|\Phi_{i-1}\right\rangle\|\\ =&3\sqrt{\frac{v-1}{N_0}}\sqrt{p_{i-1,v}} + \sqrt{\frac{v-1}{N_0}}\sqrt{p_{i-1,v-1}},\end{aligned}$$

where $p_{i,v}$ is defined in Definition 5.3. Applying another Cauchy-Schwartz inequality, we have

$$\||\phi_{i,v}\rangle\|^2 \le 2\left(\frac{9(v-1)}{N_0}p_{i-1,v} + \frac{v-1}{N_0}p_{i-1,v-1}\right).$$

Substituting that into Equation (8), we have

$$\begin{aligned}\|P\left|\Phi_T\right\rangle\|^2 \le&T\sum_{i=1}^{T}\sum_{v=0}^{M}\||\phi_{i,v}\rangle\|^2 = T\sum_{i=1}^{T}\sum_{v=1}^{M}\||\phi_{i,v}\rangle\|^2\\ \le&T\sum_{i=1}^{T}\sum_{v=1}^{M}\frac{18(v-1)}{N_0}p_{i-1,v} + T\sum_{i=1}^{T}\sum_{v=1}^{M}\frac{2(v-1)}{N_0}p_{i-1,v-1}\\ \le&\frac{18T}{N_0}\sum_{i=1}^{T}\sum_{v=1}^{M}v\cdot p_{i-1,v} + \frac{2T}{N_0}\sum_{i=1}^{T}\sum_{v=0}^{M-1}v\cdot p_{i-1,v}\\ \le&\frac{20T}{N_0}\sum_{i=1}^{T}\sum_{v=0}^{M}v\cdot p_{i-1,v}\\ =&\frac{20T}{N_0}(TV - V_T + V_0).\end{aligned}$$

Notice that according to Remark 5.4, $V_i = \sum_{v=0}^{i} v\cdot p_{i,v}$, then $V_0 = 0$. Therefore, we have

$$\|P\left|\Phi_T\right\rangle\|^2 \le \frac{20T}{N_0}(TV - V_T) \le 20\frac{T^2V}{N_0}, \tag{9}$$

as V_T being a sum of probabilities has to be non-negative. Combining Equations (6), (7) and (9), we have

$$\Pr[\mathcal{A}\ wins] \le \left(\|P\,|\Phi_T\rangle\| + \sqrt{\frac{2}{N_0}}\right)^2 \le \left(\sqrt{\frac{20T^2V}{N_0}} + \sqrt{\frac{2}{N_0}}\right)^2 = O\left(\frac{T^2V}{N_0}\right).$$

□

6 ℓ-Nested collision finding and upper bounds

Definition 6.1 (ℓ-Nested Collision Finding). *Let N be the range size, let N_0 and M be polynomial of N. For any constant integer $\ell > 0$ and any target distribution $\mathcal{D}$, the following problem is called an ℓ-Nested Collision Finding Problem:*

- *Input: two random oracles $H : [M] \to [N], G : [M^\ell] \to [N_0]$ and the target $y \leftarrow \mathcal{D}$.*
- *The goal is to find two different ℓ-tuples $(x_1, x_2, \cdots, x_\ell), (x'_1, x'_2, \cdots, x'_\ell)$ such that*
  - *$0 \le x_1 < x_2 < \cdots < x_\ell < M$, and $0 \le x'_1 < x'_2 < \cdots < x'_\ell < M$. We will call such tuple valid tuples.*
  - *$\sum_{i=1}^{\ell} H(x_i) \equiv \sum_{i=1}^{\ell} H(x'_i) \equiv y \mod N$.*
  - *$G(x_1, x_2, \cdots, x_\ell) = G(x_1, x_2, \cdots, x_\ell)$.*

Remark 6.2. *Two special cases of the problem would be:*

- *$\mathcal{D}$ is the uniform distribution. Meaning that the input is two random function H, G and a random target y.*
- *$\mathcal{D}$ is the distribution with unique support y. For example when $y = 0$ the problem becomes:*
  - *Input: two random oracles $H : [M] \to [N], G : [M^\ell] \to [N_0]$.*
  - *The goal is to find two different valid ℓ-tuples $(x_1, x_2, \cdots, x_\ell), (x'_1, x'_2, \cdots, x'_\ell)$ with sum 0 on H such that $G(x_1, x_2, \cdots, x_\ell) = G(x_1, x_2, \cdots, x_\ell)$.*

Now we present a classical algorithm and a quantum algorithm that solves this problem. In later sections, we show that without sufficient classical/quantum memory, no algorithm can perform as good as them, in terms of query complexity. Before that, we need the following lemma:

Lemma 6.3. *For any y and a random function $H : [M] \to [N]$. If we query T points, the probability that we get $\frac{\binom{T}{\ell}}{2N}$ ℓ-tuples (formed by queried points) with sum y is constant.*

Proof. Let S be the set of queried points. For $|I| = \ell$ being a subset of S define K_I be event that the sum of the ℓ-tuple on H formed by points in I is y. We have $\mathbb{E}[K_I] = \frac{1}{N}$ and $\mathrm{Var}(K_I) = \frac{N-1}{N^2}$. Notice that $\{K_I\}_I$ is pairwise independent. Thus

$$\mathbb{E}[K] := \mathbb{E}\left(\sum_{I\subseteq S, |I|=\ell} K_I\right) = \sum_{I\subseteq S, |I|=\ell} \mathbb{E}(K_I) = \binom{N}{\ell}\frac{1}{N}$$

$$\mathrm{Var}(K) := \mathrm{Var}\left(\sum_{I\subseteq S, |I|=\ell} K_I\right) = \sum_{I\subseteq S, |I|=\ell} \mathrm{Var}(K_I) = \binom{N}{\ell}\frac{N-1}{N^2}.$$

By Chebyshev bound Lemma 3.1, we have

$$\Pr\left[K \le \frac{1}{2} \cdot \frac{\binom{T}{\ell}}{N}\right] \le \frac{\mathrm{Var}(K)}{\frac{1}{4} \cdot (\mathbb{E}[K])^2} < \frac{1}{N}.$$

□

Theorem 6.4. *For any $\ell \ge 1$, there exists a classical algorithm that uses $\Theta\left(N^{\frac{1}{\ell}} N_0^{\frac{1}{2\ell}}\right)$ queries that solve the ℓ-Nested Collision Finding Problem with constant probability when $N_0 = O\left(N^{\frac{2}{\ell-1}}\right)$.*

Proof. The algorithm is as follows:

1. In the first step, the algorithm produces $K = \Theta\left(N_0^{\frac{1}{2}}\right)$ ℓ-tuples with the sum y by random querying $T_1 = \Theta\left(K^{\frac{1}{\ell}} N^{\frac{1}{\ell}}\right)$ points on H. This is by setting appropriate constant and applying Lemma 6.3.
2. Query all the G values of these ℓ -tuples, this step costs $T_2 = K$ queries. Output a collision if there is one.

The success probability of this algorithm is constant since $K = \Theta\left(N_0^{\frac{1}{2}}\right)$ points of G is sufficient for finding a collision with constant probability. When $N_0 = O\left(N^{\frac{2}{\ell-1}}\right)$, the query complexity of the first step is larger than the one in the second step thus the overall query complexity when $N_0 = O\left(N^{\frac{2}{\ell-1}}\right)$ is $\Theta\left(N^{\frac{1}{\ell}} N_0^{\frac{1}{2\ell}}\right)$. □

Theorem 6.5. *For any $\ell \ge 2$, there exists a quantum algorithm that uses $\Theta\left(N^{\frac{2}{2\ell+1}} N_0^{\frac{1}{2\ell+1}}\right)$ queries that solves the ℓ-Nested Collision Finding problem with constant probability when $N_0 = O\left(N^{\frac{3}{\ell-1}}\right) = \Omega\left(N^{\frac{1}{\ell}}\right)$.*

Proof. The algorithm is as follows:

1. In the first step, the algorithm produces K ℓ-tuples with the same sum by random querying $T_1 = \Theta\left(K^{\frac{1}{\ell}} N^{\frac{1}{\ell}}\right)$ points on H. This is by setting appropriate constant and applying Lemma 6.3.
2. Query all G value of these ℓ-tuples, this step costs $T_2 = K$ queries.
3. Query T_3 individual H values different from points queried in the first step. This will produce $\Theta\left(T_3^{\ell-1}\right)$ $\ell - 1$ tuples.
4. Now we run Grover search to find a ℓ-tuple with:
   - Sum y on H.
   - Its $\ell - 1$ sub-tuple (the $\ell - 1$-tuple formed by the first $\ell - 1$ elements) being one of the $\ell - 1$-tuple founded in step 3. Which means that we only need to run the Grover search on the last element.
   - Its G value collide with one of the ℓ-tuple in step 2.

The query complexity is $\Theta\left(\sqrt{\frac{N_0}{K}} \cdot \sqrt{\frac{N}{T_3^{\ell-1}}}\right)$ when $K = O(N_0)$ and $T_3^{\ell-1} = O(N)$ by Lemma 3.2. This is because we can run a grover search on the last element of the ℓ-tuple. For each such element, the probability that there exists a $\ell - 1$-tuple in step 3, with this element, adds up to sum y on H, is $\frac{T_3^{\ell-1}}{N}$. And the probability that this ℓ-tuple collides with a ℓ-tuple in step 2 on G is $\frac{K}{N_0}$.

Set $T_3 = O\left(\left(\frac{N_0 N}{K}\right)^{\frac{1}{\ell+1}}\right)$ and $K = \Theta\left(\frac{N_0^{\frac{\ell}{2\ell+1}}}{N^{\frac{1}{2\ell+1}}}\right) = \Omega(1)$ because $N_0 = \Omega\left(N^{\frac{1}{\ell}}\right)$. Now we have

1. Step 1 costs $\Theta\left(N^{\frac{2}{2\ell+1}} N_0^{\frac{1}{2\ell+1}}\right)$ queries.
2. Step 2 costs $\Theta\left(\frac{N_0^{\frac{\ell}{2\ell+1}}}{N^{\frac{1}{2\ell+1}}}\right)$ queries. To make this step not a bottleneck we need $N_0 = O\left(N^{\frac{3}{\ell-1}}\right)$.
3. Step 3 costs $\Theta\left(N^{\frac{2}{2\ell+1}} N_0^{\frac{1}{2\ell+1}}\right)$. We also need $T_3^{\ell-1} = O(N_0)$. Adding constraint $N_0 = O\left(N^{\frac{3}{\ell-1}}\right)$ again.
4. Step 4 costs $\Theta\left(N^{\frac{2}{2\ell+1}} N_0^{\frac{1}{2\ell+1}}\right)$ queries.

□

7 The time bound for finding K ℓ-tuples with the same sum

In this section we first prove bounds on the probability that an algorithm on T oracle queries, produces K ℓ-tuples with the same sum. Here, the sum of a ℓ-tuple $(x_1, x_2, \cdots, x_\ell)$ is defined by $\sum_{i=1}^{\ell} H(x_i) \mod N$ where H is the random oracle which will be clear from the content. We first prove the case where the algorithm is classical to demonstrate our idea. Then we shift to the quantum case and use the compress oracle technique to derive a similar bound.

7.1 The classical bound

Theorem 7.1. *Let $H : [M] \to [N]$ be a random oracle with sufficient large M polynomial in N and let $K = \Omega(\log N)^\ell$. For any classical algorithm with $T = \Omega(K)$ oracle queries, the probability that there are K distinct ℓ-tuples consist only queried points with the same sum is at most $O\left(\frac{T^\ell}{K^{\frac{1}{\ell}} N}\right)^{K^{\frac{1}{\ell}}}$.*

Remark 7.2. *At the first glance, the bound may seem strange due to the superscript $\frac{1}{\ell}$ on K. This is in fact natural, because if an algorithm queries $O\left(K^{\frac{1}{\ell}}\right)$ points, with probability $\left(\frac{1}{N}\right)^{O\left(K^{\frac{1}{\ell}}\right)}$ it gets value 0 on all queried points and the task is done automatically. So one would expect the term $K^{\frac{1}{\ell}}$ to exists in reasonable bounds.*

Proof. In the classical case, query strategy is meaningless since each point of the random oracle is independent. Thus we assume that the algorithm just simply query points one-by-one. We prove by induction on ℓ. For $\ell = 1$, let $y \in [N]$ and $\Delta_{t,k}^y$ be the probability that after t queries one can find out at least k distinct queried points $x_1, x_2, \cdots, x_k$ such that $H(x_1) = H(x_2) = \cdots = H(x_k) = y$. We have

$$\Delta^y_{t,k} \le \Delta^y_{t-1,k} + \frac{1}{N} \cdot \Delta^y_{t-1,k-1}.$$

Thus

$$\Delta^y_{t,k} \le \binom{t}{k} \left(\frac{1}{N}\right)^k.$$

The probability that the there are K distinct points $x_1, x_2, \cdots, x_k$ of the same value is at most

$$\begin{aligned} &\sum_{y\in[N]} \Delta^y_{T,K} \\ \le & N\binom{T}{K}\left(\frac{1}{N}\right)^K \\ = & O\left(\frac{T}{KN}\right)^K. \end{aligned}$$

Suppose that the statement holds for $\ell = \ell^* - 1$, now we prove that the statement holds for $\ell = \ell^*$. After T queries there are two cases: for the first case, one can find out at least $K^{\frac{\ell-1}{\ell}}$ distinct $\ell - 1$ tuples consist of only queried points with the same sum. The probability of this case, by induction, is $O\left(\frac{t^{\ell-1}}{K^{\frac{1}{\ell}}N}\right)^{K^{\frac{1}{\ell}}}$. The rest form the second case. Let $y \in [N]$. Now we want to calculate how many fraction of the second case satisfies that at some point one can find out at least K distinct ℓ tuples with sum y. Note that number of distinct $\ell - 1$ tuples with the same sum is at most $K^{\frac{\ell-1}{\ell}}$ since we are in the second case. Thus we can relax the requirement by assuming that we can find $K^{\frac{\ell-1}{\ell}}$ ℓ-tuples of sum y whenever a new point is added for which there exists a $\ell-1$ tuple in the database such that its sum plus the value of the new entry is y. Let $\Delta^y_{t,k}$ denote the probability that after t queries there exists k queries out of these t queries such that that queried point with all the previous queried points can form a ℓ-tuple of sum y that contains that queried point. Since there are at most $(t + k_0)^{\ell-1}$ possible sum for $\ell - 1$ tuples consists of only queried points we have,

$$\Delta^y_{t,k} \le \Delta^y_{t-1,k} + \frac{t^{\ell-1}}{N} \cdot \Delta^y_{t-1,k-1}.$$

Thus

$$\Delta^y_{t,k} \le \binom{t}{k}\left(\frac{t^{\ell-1}}{N}\right)^k.$$

By union bound, the probability that an algorithm outputs K distinct ℓ-tuples with the same sum is bounded by the sum of probability that this algorithm outputs K distinct ℓ-tuples with sum y over all $y \in [N]$. If an algorithm outputs K distinct ℓ-tuples with sum y, by above analysis, either one can find out $K^{\frac{\ell-1}{\ell}}$ distinct $\ell - 1$ tuples with the same sum in its queried points or there exists $K^{\frac{1}{\ell}}$ queries out of these t queries such that the queried point with all the previous queried points can form a ℓ-tuple of sum y that contains the queried point. Thus, the overall success probability of any algorithm is bounded by

$$\sum_{y\in[N]} \Delta^y_{T,K^{\frac{1}{\ell}}} + O\left(\frac{T^{\ell-1}}{K^{\frac{1}{\ell}}N}\right)^{K^{\frac{1}{\ell}}}$$

$$
\begin{aligned}
&\leq N\left(\frac{T}{K^{\frac{1}{\ell}}}\right)\left(\frac{T^{\ell-1}}{N}\right)^{K^{\frac{1}{\ell}}}+O\left(\frac{T^{\ell-1}}{K^{\frac{1}{\ell}}N}\right)^{K^{\frac{1}{\ell}}}\\
&=O\left(\frac{T^{\ell}}{K^{\frac{1}{\ell}}N}\right)^{K^{\frac{1}{\ell}}}+O\left(\frac{T^{\ell-1}}{K^{\frac{1}{\ell}}N}\right)^{K^{\frac{1}{\ell}}}\\
&=O\left(\frac{T^{\ell}}{K^{\frac{1}{\ell}}N}\right)^{K^{\frac{1}{\ell}}}.
\end{aligned}
$$

Note that since $\ell > 0$ is a constant, doing induction on it preserves the $O(\cdot)$ notation. □

Theorem 7.3. *Let $H : [M] \to [N]$ be a random oracle with sufficient large M polynomial in N and let $K = \Omega(\log N)^{\ell}$. For any classical algorithm with $T = \Omega(K)$ oracle queries, the probability that it outputs K distinct ℓ-tuples with the same sum is at most $O\left(\frac{T^{\ell}}{K^{\frac{1}{\ell}}N}\right)^{K^{\frac{1}{\ell}}}$.*

Proof. For any algorithm $\mathcal{A}$ with $T = \Omega(K)$ oracle queries, we can construct an algorithm $\mathcal{A}'$ such that all outputted points are queried at some point by running $\mathcal{A}$ first and querying any index that does not exist in its database at the end of the $\mathcal{A}$. By theorem 7.1, we prove the statement. □

7.2 The quantum bound

In this section we are going to prove a similar result in the quantum setting. Suppose that we start with an **arbitrary** state:

$$
|\psi_{\mathsf{st}}\rangle = \sum_{x,u,w,r,D} \alpha_{x,u,w,r,D} |x\rangle_{\mathbf{X}} |u\rangle_{\mathbf{U}} |w\rangle_{\mathbf{W}} |r\rangle_{\mathbf{R}} |D\rangle_{\mathbf{D}}.
$$

Define the following operators:

- We extend the phase oracle unitary CPhO and standard decompose unitary $\mathsf{StdDecomp}_x$ to the case where $x = \bot$. In this case these unitaries acts as an identity over all registers. As a special reminder, not like the case in the compressed oracle setting, here $|\bot\rangle_{\mathbf{X}}$ is orthogonal to any $|x\rangle_{\mathbf{X}}$ for $x \in [M]$.
- Define $\Pi_{\geq k}$ as the projector that projects to the subspace spanned by all state $|x,u,w,r\rangle |D\rangle$ such that D contains at least k non-bot entries. Define $\Pi_{=k}, \Pi_{<k}$ in the similar manner.
- Let $y \in [N]$, define $\Pi^{\ell,y}_{\geq k}$ as the projector that projects to the subspace spanned by all state $|x,u,w,r\rangle |D\rangle$ such that D contains at least k distinct ℓ-tuples with sum y. Define $\Pi^{\ell,y}_{=k}, \Pi^{\ell,y}_{<k}$ in the similar manner. Also if $y = *$ it means any y, for example, $\Pi^{\ell,*}_{\geq k}$ projects to the subspace spanned by all state $|x,u,w,r\rangle |D\rangle$ such that D contains at least k distinct ℓ-tuples with the same sum.
- Let $y \in [N]$ and $\mathcal{O}$ be either the compressed phase oracle CPhO or $\mathsf{StdDecomp}$, define $\Pi^{y,\mathcal{O}}_{\mathsf{inc}}$ as the operator that 'projects' to the state that after calling $\mathcal{O}$ the number of distinct ℓ-tuples with sum y increases. That is, $\Pi^{y,\mathcal{O}}_{\mathsf{inc}} = \sum_k \left(\mathcal{O}^{-1}\Pi^{\ell,y}_{>k}\mathcal{O}\Pi^{\ell,y}_{=k}\right)$. **Note that although we use Π, this may not be a projection**.

Theorem 7.4. *For $\ell > 0$ be an integer. Let $k_0, k_1, k_2, \cdots, k_\ell \geq 0$ be integers. Assume the following:*

- *The number of non-bot entries in the database register is at most k_0. In other words,*

$$
\Pi_{>k_0} |\psi_{\mathsf{st}}\rangle = 0.
$$

- *For $i = 1, 2, \cdots, \ell$, the number of i-tuples with the same sum is at most k_i. In other words, for $i = 1, 2, \cdots, \ell - 1$*
$$\Pi^{i,*}_{>k_i} |\psi_{\mathsf{st}}\rangle = 0.$$

Let M be a sufficient large and let $K > \max_{i=1}^{\ell} k_i$. Let K_{sol} be the only real non-negative root of the following function
$$f_{K,k_1,k_2,\cdots,k_\ell}(x) = x^\ell + \sum_{i=0}^{\ell-1} k_{\ell-i} x^i - K.$$
Note that when $x = 0$, $f_{K,k_1,k_2,\cdots,k_\ell}(x) = k_\ell - K \leq 0$ and when $x \to \infty$, $f(x) \to \infty$. Thus there must be a real non-negative root. Also we know that this root is unique since all coefficients except the constant term are positive, meaning that $f_{K,k_1,k_2,\cdots,k_\ell}(x)$ is monotonically increasing when $x \geq 0$.
If $K_{\mathsf{sol}} = \Omega(\log N)$ then for any quantum algorithm with $T = \Omega(K)$ compressed phase oracle queries, and T additional StdDecomp *after that, the probability that* ***at some point during the algorithm*** *there are K distinct ℓ-tuples with the same sum in the database register under the view of compress oracle is at most $O\left(\frac{T^2(T+k_0)^{\ell-1}}{K_{\mathsf{sol}}^2 N}\right)^{K_{\mathsf{sol}}}$. More specifically,*
$$\left|\left(\sum_{i=0}^{''-1} \prod_{i+1}^{j=2T-1} (\mathcal{O}_j U_j)\, \Pi^{\ell,*}_{\geq K} \mathcal{O}_i U_i \prod_{0}^{j=i-1} \left(\Pi^{\ell,*}_{<K} \mathcal{O}_j U_j\right)\right) |\psi_{\mathsf{st}}\rangle\right|^2 = O\left(\frac{T^{\ell+1}}{K_{\mathsf{sol}}^2 N}\right)^{K_{\mathsf{sol}}}.$$
where we additionally define $\mathcal{O}_i = \mathsf{CPhO}$ when $i < T$ and $\mathcal{O}_i = \mathsf{StdDecomp}$ otherwise.

Proof. We prove by induction on ℓ. For $\ell = 1$, define $\Delta^y_{t,k}$ as the amplitude on states after t queries (from now on, if $t > T$ it means after the first $t - T$ StdDecomp) and at some point one can find out at least k distinct points with value y in the database,
$$\Delta^y_{t,k} = \left|\left(\sum_{i=0}^{t-1} \prod_{i+1}^{j=t-1} (\mathcal{O}_j U_j)\, \Pi^{1,y}_{\geq k} \mathcal{O}_i U_i \prod_{0}^{j=i-1} \left(\Pi^{1,y}_{<k} \mathcal{O}_j U_j\right)\right) |\psi_{\mathsf{st}}\rangle\right|.$$
Since we assume that the number of distinct points with every value $y \in [N]$ in the database of $|\psi_{\mathsf{st}}\rangle$ are at most k_1 we have:
$$\Delta^y_{0,0} = \Delta^y_{0,1} = \cdots = \Delta^y_{0,k_1} = 1. \tag{10}$$
And
$$0 = \Delta^y_{0,k_1+1} = \Delta^y_{0,k_1+2} = \cdots. \tag{11}$$
Now we try to calculate the recursion formula for $\Delta^y_{t,k}$:
$$\begin{aligned}
\Delta^y_{t,k} &= \left|\left(\sum_{i=0}^{t-1} \prod_{i+1}^{j=t-1} (\mathcal{O}_j U_j)\, \Pi^{1,y}_{\geq k} \mathcal{O}_i U_i \prod_{0}^{j=i-1} \left(\Pi^{1,y}_{<k} \mathcal{O}_j U_j\right)\right) |\psi_{\mathsf{st}}\rangle\right| \\
&\leq \left|\left(\sum_{i=0}^{t-2} \prod_{i+1}^{j=t-1} (\mathcal{O}_j U_j)\, \Pi^{1,y}_{\geq k} \mathcal{O}_i U_i \prod_{0}^{j=i-1} \left(\Pi^{1,y}_{<k} \mathcal{O}_j U_j\right)\right) |\psi_{\mathsf{st}}\rangle\right| + \left|\Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} U_{t-1} \prod_{0}^{j=t-2} \left(\Pi^{1,y}_{<k} \mathcal{O}_j U_j\right) |\psi_{\mathsf{st}}\rangle\right| \\
&= \left|\left(\sum_{i=0}^{t-2} \prod_{i+1}^{j=t-2} (\mathcal{O}_j U_j)\, \Pi^{1,y}_{\geq k} \mathcal{O}_i U_i \prod_{0}^{j=i-1} \left(\Pi^{1,y}_{<k} \mathcal{O}_j U_j\right)\right) |\psi_{\mathsf{st}}\rangle\right| + \left|\Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} U_{t-1} \prod_{0}^{j=t-2} \left(\Pi^{1,y}_{<k} \mathcal{O}_j U_j\right) |\psi_{\mathsf{st}}\rangle\right|
\end{aligned}$$

$$
\begin{aligned}
&=\Delta^y_{t-1,k} + \left| \Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} \Pi^{1,y}_{<k} U_{t-1} \mathcal{O}_{t-2} U_{t-2} \prod_{0}^{j=t-3} \left(\Pi^{1,y}_{<k} \mathcal{O}_j U_j \right) |\psi_{\mathsf{st}}\rangle \right| \\
&\leq \Delta^y_{t-1,k} + \left| \Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} \Pi^{1,y}_{<k} |\psi_t\rangle \right| \left| \Pi^{1,y}_{<k} \mathcal{O}_{t-2} U_{t-2} \prod_{0}^{j=t-3} \left(\Pi^{1,y}_{<k} \mathcal{O}_j U_j \right) |\psi_{\mathsf{st}}\rangle \right|
\end{aligned}
$$

where $|\psi_t\rangle \propto \Pi^{1,y}_{<k} U_{t-1} \mathcal{O}_{t-2} U_{t-2} \prod_0^{j=t-3} \left(\Pi^{1,y}_{<k} \mathcal{O}_j U_j \right) |\psi_{\mathsf{st}}\rangle$ is the normalized state before the t-th oracle query. Define the following projectors:

- $Q^{\ell,y}_{=k}$ projects to all state spanned by $|x,u,w,r\rangle |D\rangle$ such that there are k distinct ℓ-tuples of value y, $D(x) = \perp$ and $u \neq 0$.
- $R^{\ell,y}_{=k}$ projects to all state spanned by $|x,u,w,r\rangle |D\rangle$ such that there are k distinct ℓ-tuples of value y, $D(x) \neq \perp$ and $u \neq 0$.
- $S^{\ell,y}_{=k}$ projects to all state spanned by $|x,u,w,r\rangle |D\rangle$ such that there are k distinct ℓ-tuples of value y and $u = 0$.

Because each oracle query add at most one entry in the database we have that

$$
\begin{aligned}
\Delta^y_{t,k} &\leq \Delta^y_{t-1,k} + \left| \Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} \Pi^{1,y}_{<k} |\psi_t\rangle \right| \left| \Pi^{1,y}_{<k} \mathcal{O}_{t-2} U_{t-2} \prod_{0}^{j=t-3} \left(\Pi^{1,y}_{<k} \mathcal{O}_j U_j \right) |\psi_{\mathsf{st}}\rangle \right| \\
&\leq \Delta^y_{t-1,k} + \left| \Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} \left(Q^{1,y}_{=k-1} + R^{1,y}_{=k-1} + S^{1,y}_{=k-1} \right) |\psi_t\rangle \right| \cdot \Delta^y_{t-1,k-1}.
\end{aligned}
$$

To bound $\left| \Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} \left(Q^{1,y}_{=k-1} + R^{1,y}_{=k-1} + S^{1,y}_{=k-1} \right) |\psi_t\rangle \right|$, we need to consider two cases:

- When $t \leq T$, $\mathcal{O}_{t-1} = \mathsf{CPhO}$ so we invoke lemma 4.8. For $|x,u,w,r\rangle |D\rangle$ in $S^{1,y}_{=k-1}$ the state is unchanged after $\mathcal{O}_{t-1}$. For $|x,u,w,r\rangle |D\rangle$ in $Q^{1,y}_{=k-1}$, we have $|D(x)\rangle_{\mathbf{D}_x} = \sum_{y' \in [N]} \frac{(-1)^{\langle u,y' \rangle}}{\sqrt{N}} |y'\rangle$ after $\mathcal{O}_{t-1}$. Since only one uniform random value y' is added, we have $\left| \Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} Q^{1,y}_{=k-1} |\psi_t\rangle \right| \leq \sqrt{\frac{1}{N}}$. For $|x,u,w,r\rangle |D\rangle$ in $R^{1,y}_{=k-1}$, we have $|D(x)\rangle_{\mathbf{D}_x} = \frac{(-1)^{\langle u,D(x) \rangle}}{\sqrt{N}} |\perp\rangle + \frac{1+(-1)^{\langle u,D(x) \rangle}(N-2)}{N} |D(x)\rangle + \sum_{y' \in [N] \setminus \{D(x)\}} \frac{1-(-1)^{\langle u,y' \rangle} - (-1)^{\langle u,D(x) \rangle}}{N} |y'\rangle$ after $\mathcal{O}_{t-1}$. Since the only case where a new value y' is added has amplitude $\frac{1-(-1)^{\langle u,y' \rangle} - (-1)^{\langle u,D(x) \rangle}}{N}$, we have $\left| \Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} R^{1,y}_{=k-1} |\psi_t\rangle \right| \leq \frac{3}{N}$.
- When $t > T$, $\mathcal{O}_{t-1} = \mathsf{StdDecomp}$. For $|x,u,w,r\rangle |D\rangle$ in $S^{1,y}_{=k-1}$ the state is unchanged after $\mathcal{O}_{t-1}$. For $|x,u,w,r\rangle |D\rangle$ in $Q^{1,y}_{=k-1}$, we have $|D(x)\rangle_{\mathbf{D}_x} = \frac{1}{\sqrt{N}} \sum_{y' \in [N]} |y'\rangle$. Since only one uniform random value y' is added, we have $\left| \Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} Q^{1,y}_{=k-1} |\psi_t\rangle \right| \leq \sqrt{\frac{1}{N}}$. For $|x,u,w,r\rangle |D\rangle$ in $R^{1,y}_{=k-1}$, we have $|D(x)\rangle_{\mathbf{D}_x} = \frac{N-1}{N} |D(x)\rangle + \frac{1}{\sqrt{N}} |\perp\rangle - \frac{1}{N} \sum_{y' \in [N] \setminus \{D(x)\}} |y'\rangle$ after $\mathcal{O}_{t-1}$. Since the only case where a new value y' is added has amplitude $\frac{1}{N}$, we have $\left| \Pi^{1,y}_{\geq k} \mathcal{O}_{t-1} R^{1,y}_{=k-1} |\psi_t\rangle \right| \leq \frac{1}{N}$.

Combining all cases above, we obtain the following recursion formula for $\Delta^y_{t,k}$:

$$
\Delta^y_{t,k} \leq \Delta^y_{t-1,k} + 4\sqrt{\frac{1}{N}} \cdot \Delta^y_{t-1,k-1}
$$

combine with eq. (10) and eq. (11) we get

$$\Delta^y_{t,k} \le \binom{t}{k-k_1}\left(4\sqrt{\frac{1}{N}}\right)^{k-k_1}$$

for $k > k_1$ and $y \in [N]$. The probability that at some point during the algorithm there are K distinct points $x_1, x_2, \cdots, x_K$ of the same value in the database is at most

$$\begin{aligned}
&\left|\left(\sum_{i=0}^{2T-1}\prod_{i+1}^{j=2T-1}(\mathcal{O}_jU_j)\,\Pi^{1,*}_{\ge K}\mathcal{O}_iU_i\prod_{0}^{j=i-1}\left(\Pi^{1,*}_{<K}\mathcal{O}_jU_j\right)\right)|\psi_{\mathsf{st}}\rangle\right|^2\\
\le&\left|\left(\sum_{y\in[N]}\sum_{i=0}^{2T-1}\prod_{i+1}^{j=2T-1}(\mathcal{O}_jU_j)\,\Pi^{1,y}_{\ge K}\mathcal{O}_iU_i\prod_{0}^{j=i-1}\left(\Pi^{1,y}_{<K}\mathcal{O}_jU_j\right)\right)|\psi_{\mathsf{st}}\rangle\right|^2\\
\le& N\sum_{y\in[N]}\left|\left(\sum_{i=0}^{2T-1}\prod_{i+1}^{j=2T-1}(\mathcal{O}_jU_j)\,\Pi^{1,y}_{\ge K}\mathcal{O}_iU_i\prod_{0}^{j=i-1}\left(\Pi^{1,y}_{<K}\mathcal{O}_jU_j\right)\right)|\psi_{\mathsf{st}}\rangle\right|^2\\
\le& N\sum_{y\in[N]}\left(\Delta^y_{2T,K}\right)^2\\
\le& N^2\binom{2T}{K-k_1}^2\left(\frac{16}{N}\right)^{K-k_1}\\
=& O\left(\frac{T^2}{(K-k_1)^2N}\right)^{K-k_1}.
\end{aligned}$$

N is absorbed into the $O(\cdot)$ notation because $K - k_1 = \Omega(\log N)$.
Suppose that the statement holds for $\ell = \ell^* - 1$, now we prove that the statement holds for $\ell = \ell^*$. Imagine an algorithm that at some point one can find out at least K distinct ℓ tuples with the same sum y in its database. This algorithm must satisfy:

- Either at some point of this algorithm, one can find out at least K_{thd} distinct $\ell - 1$ tuples with the same sum.
- Or there exists at least $\frac{K-k_\ell}{K_{\mathsf{thd}}}$ 'effective' queries, meaning that after that query the queried point, with known points in the database forms at least one new ℓ tuple of sum y.

Intuitively, this is because if an algorithm never stores K_{thd} distinct $\ell-1$-tuples with the same sum in its database, each 'effective' query can at most generate $K_{\mathsf{thd}} - 1$ ℓ-tuples with sum y thus one need at least $\frac{K-k_\ell}{K_{\mathsf{thd}}}$ of them.
Now let us formalize our proof. Define $\Gamma^y_{t,k}$ as the amplitude on states after t queries and at some point there one can find at least K ℓ-tuples with sum y in the database,

$$\begin{aligned}
\Gamma^y_{t,k} &= \left|\left(\sum_{i=0}^{t-1}\prod_{i+1}^{j=t-1}(\mathcal{O}_jU_j)\,\Pi^{\ell,y}_{\ge k}\mathcal{O}_iU_i\prod_{0}^{j=i-1}\left(\Pi^{\ell,y}_{<k}\mathcal{O}_jU_j\right)\right)|\psi_{\mathsf{st}}\rangle\right|\\
&= \left|\left(\sum_{\substack{c_0,c_1,\cdots,c_t\\ c_0\le k_\ell \text{ and } \exists i \text{ s.t. } c_i\ge k}}\prod_{0}^{j=t-1}\left(\Pi^{\ell,y}_{=c_{j+1}}\mathcal{O}_jU_j\right)\Pi^{\ell,y}_{=c_0}\right)|\psi_{\mathsf{st}}\rangle\right|
\end{aligned}$$

$$\leq \left| \left(\sum_{\substack{c_0, c_1, \cdots, c_t \\ \exists i \text{ s.t. } c_i \geq c_{i-1} + K_{\mathsf{thd}}}} \prod_{0}^{j=t-1} \left(\Pi^{\ell, y}_{=c_{j+1}} \mathcal{O}_j U_j \right) \Pi^{\ell, y}_{=c_0} \right) |\psi_{\mathsf{st}}\rangle \right|$$
$$+ \left| \left(\sum_{\substack{c_0, c_1, \cdots, c_t \\ \forall i, c_i < c_{i-1} + K_{\mathsf{thd}} \\ \exists i \text{ s.t. } c_i \geq k}} \prod_{0}^{j=t-1} \left(\Pi^{\ell, y}_{=c_{j+1}} \mathcal{O}_j U_j \right) \Pi^{\ell, y}_{=c_0} \right) |\psi_{\mathsf{st}}\rangle \right|$$
$$\leq \left| \left(\sum_{i=0}^{t-1} \prod_{i+1}^{j=t-1} (\mathcal{O}_j U_j) \Pi^{\ell-1,*}_{\geq K_{\mathsf{thd}}} \mathcal{O}_i U_i \prod_{0}^{j=i-1} \left(\Pi^{\ell-1,*}_{<K_{\mathsf{thd}}} \mathcal{O}_j U_j \right) \right) |\psi_{\mathsf{st}}\rangle \right|$$
$$+ \left| \left(\sum_{\substack{c_0, c_1, \cdots, c_t \\ \sum_{i=1}^{t} \mathbb{I}[c_i > c_{i-1}] \geq \frac{k - k_\ell}{K_{\mathsf{thd}}}}} \prod_{0}^{j=t-1} \left(\Pi^{\ell, y}_{=c_{j+1}} \mathcal{O}_j U_j \right) \Pi^{\ell, y}_{=c_0} \right) |\psi_{\mathsf{st}}\rangle \right|$$

The first term is bounded by induction (we will show it later). Now we bound the second term. We redefine $\Delta^{y}_{t,k}$ as the amplitude on states after t queries and there exists k queries such that the number of distinct ℓ-tuples with sum y increases after it,

$$\Delta^{y}_{t,k} = \left| \left(\sum_{\substack{0 = p_0 < p_1 < \\ \cdots < p_k \leq t}} \prod_{p_k}^{j=t-1} (\mathcal{O}_j U_j) \prod_{1}^{i<k} \left(\mathsf{Inc}^{y}_{p_{i+1}-1} \prod_{p_i}^{j=p_{i+1}-2} \mathsf{NonInc}^{y}_{j} \right) \right) |\psi_{\mathsf{st}}\rangle \right|$$

where $\mathsf{Inc}^{y}_{i} = \mathcal{O}_i \Pi^{y, \mathcal{O}_i}_{\mathsf{Inc}} U_i$, $\mathsf{NonInc}^{y}_{i} = \mathcal{O}_i \left(I - \Pi^{y, \mathcal{O}_i}_{\mathsf{Inc}} \right) U_i$. And we have,

$$\Delta^{y}_{t,k} = \left| \left(\sum_{\substack{0 = p_0 < p_1 < \\ \cdots < p_k \leq t}} \prod_{p_k}^{j=t-1} (\mathcal{O}_j U_j) \prod_{1}^{i<k} \left(\mathsf{Inc}^{y}_{p_{i+1}-1} \prod_{p_i}^{j=p_{i+1}-2} \mathsf{NonInc}^{y}_{j} \right) \right) |\psi_{\mathsf{st}}\rangle \right|$$
$$\leq \left| \left(\sum_{\substack{0 = p_0 < p_1 < \\ \cdots < p_k < t}} \prod_{p_k}^{j=t-1} (\mathcal{O}_j U_j) \prod_{1}^{i<k} \left(\mathsf{Inc}^{y}_{p_{i+1}-1} \prod_{p_i}^{j=p_{i+1}-2} \mathsf{NonInc}^{y}_{j} \right) \right) |\psi_{\mathsf{st}}\rangle \right|$$
$$+ \left| \left(\sum_{\substack{0 = p_0 < p_1 < \\ \cdots < p_k = t}} \prod_{1}^{i<k} \left(\mathsf{Inc}^{y}_{p_{i+1}-1} \prod_{p_i}^{j=p_{i+1}-2} \mathsf{NonInc}^{y}_{j} \right) \right) |\psi_{\mathsf{st}}\rangle \right|$$
$$= \left| \left(\sum_{\substack{0 = p_0 < p_1 < \\ \cdots < p_k \leq t-1}} \prod_{p_k}^{j=t-2} (\mathcal{O}_j U_j) \prod_{1}^{i<k} \left(\mathsf{Inc}^{y}_{p_{i+1}-1} \prod_{p_i}^{j=p_{i+1}-2} \mathsf{NonInc}^{y}_{j} \right) \right) |\psi_{\mathsf{st}}\rangle \right|$$

$$+\left|\left(\sum_{\substack{0=p_0<p_1<\\ \cdots<p_{k-1}<t}} \mathcal{O}_{t-1}\Pi_{\mathsf{inc}}^{y,\mathcal{O}_{t-1}} U_{t-1} \prod_{p_{k-1}}^{j=t-2} \mathsf{NonInc}_j^y \prod_{1}^{i=k-2}\left(\mathsf{Inc}_{p_{i+1}-1}^y \prod_{p_i}^{j=p_{i+1}-2} \mathsf{NonInc}_j^y\right)\right)|\psi_{\mathsf{st}}\rangle\right|$$

$$\leq \Delta_{t-1,k}^y + \left|\mathcal{O}_{t-1}\Pi_{\mathsf{inc}}^{y,\mathcal{O}_{t-1}}|\psi_t\rangle\right|$$

$$\cdot\left|\sum_{\substack{0=p_0<p_1<\\ \cdots<p_{k-1}<t}} \prod_{p_{k-1}}^{j=t-2}(\mathcal{O}_jU_j)\prod_{1}^{i=k-2}\left(\mathsf{Inc}_{p_{i+1}-1}^y \prod_{p_i}^{j=p_{i+1}-2}\mathsf{NonInc}_j^y\right)|\psi_{\mathsf{st}}\rangle\right|$$

where $|\psi_t\rangle \propto \sum_{\substack{0=p_0<p_1<\\ \cdots<p_{k-1}<t}}\left(U_{t-1}\prod_{p_{k-1}}^{j=t-2}(\mathcal{O}_jU_j)\prod_1^{i=k-2}\left(\mathsf{Inc}_{p_{i+1}-1}^y\prod_{p_i}^{j=p_{i+1}-2}\mathsf{NonInc}_j^y\right)\right)|\psi_{\mathsf{st}}\rangle$ is the state (normalized) before the t-th oracle query. Define the following projectors:

- Q projects to all state spanned by $|x,u,w,r\rangle|D\rangle$ such that $D(x)=\perp$ and $u\neq 0$.
- R projects to all state spanned by $|x,u,w,r\rangle|D\rangle$ such that $D(x)\neq\perp$ and $u\neq 0$.
- S projects to all state spanned by $|x,u,w,r\rangle|D\rangle$ such that $u=0$.

Use these projectors we have

$$\begin{aligned}\Delta_{t,k}^y \leq&\Delta_{t-1,k}^y + \left|\mathcal{O}_{t-1}\Pi_{\mathsf{inc}}^{y,\mathcal{O}_{t-1}}|\psi_t\rangle\right|\\ &\cdot\left|\sum_{\substack{0=p_0<p_1<\\ \cdots<p_{k-1}<t}} \prod_{p_{k-1}}^{j=t-2}(\mathcal{O}_jU_j)\prod_{1}^{i=k-2}\left(\mathsf{Inc}_{p_{i+1}-1}^y \prod_{p_i}^{j=p_{i+1}-2}\mathsf{NonInc}_j^y\right)|\psi_{\mathsf{st}}\rangle\right|\\ \leq&\Delta_{t-1,k}^y + \left|\mathcal{O}_{t-1}\Pi_{\mathsf{inc}}^{y,\mathcal{O}_{t-1}}(Q+R+S)|\psi_t\rangle\right|\cdot\Delta_{t-1,k-1}^y.\end{aligned}$$

To bound $\left|\mathcal{O}_{t-1}\Pi_{\mathsf{inc}}^{y,\mathcal{O}_{t-1}}(Q+R+S)|\psi_t\rangle\right|$, we need to consider two cases:

- When $t\leq T$, $\mathcal{O}_{t-1}=\mathsf{CPhO}$ so we invoke lemma 4.8. For $|x,u,w,r\rangle|D\rangle$ in S the state is unchanged after $\mathcal{O}_{t-1}$. For $|x,u,w,r\rangle|D\rangle$ in R, we have $|D(x)\rangle_{\mathbf{D}_x}=\sum_{y'\in[N]}\frac{(-1)^{\langle u,y'\rangle}}{\sqrt{N}}|y'\rangle$ after $\mathcal{O}_{t-1}$. Since only one uniform random value y' is added, there are at most $(t+k_0)^{\ell-1}$ possible value of y' such that it can form a ℓ-tuple with sum y with $\ell-1$ elements already in the database. Thus we have $\left|\mathcal{O}_{t-1}\Pi_{\mathsf{inc}}^{y,\mathcal{O}_{t-1}}Q|\psi_t\rangle\right|\leq\sqrt{\frac{(t+k_0)^{\ell-1}}{N}}$. For $|x,u,w,r\rangle|D\rangle$ in R, we have $|D(x)\rangle_{\mathbf{D}_x}=\frac{(-1)^{\langle u,D(x)\rangle}}{\sqrt{N}}|\perp\rangle+\frac{1+(-1)^{\langle u,D(x)\rangle}(N-2)}{N}|D(x)\rangle+\sum_{y'\in[N]\setminus\{D(x)\}}\frac{1-(-1)^{\langle u,y'\rangle}-(-1)^{\langle u,D(x)\rangle}}{N}|y'\rangle$ after $\mathcal{O}_{t-1}$. Since the only case where a new value y' is added has amplitude $\frac{1-(-1)^{\langle u,y'\rangle}-(-1)^{\langle u,D(x)\rangle}}{N}$ and there are at most $(t+k_0)^{\ell-1}$ possible value of y' such that it can form a ℓ-tuple with sum y with $\ell-1$ elements already in the database, we have $\left|\mathcal{O}_{t-1}\Pi_{\mathsf{inc}}^{y,\mathcal{O}_{t-1}}R|\psi_t\rangle\right|\leq\frac{3(t+k_0)^{\ell-1}}{N}$.
- When $t>T$, $\mathcal{O}_{t-1}=\mathsf{StdDecomp}$. For $|x,u,w,r\rangle|D\rangle$ in S the state is unchanged after $\mathcal{O}_{t-1}$. For $|x,u,w,r\rangle|D\rangle$ in Q, we have $|D(x)\rangle_{\mathbf{D}_x}=\frac{1}{\sqrt{N}}\sum_{y'\in[N]}|y'\rangle$ after $\mathcal{O}_{t-1}$. Since only one uniform random value y' is added, there are at most $(t+k_0)^{\ell-1}$ possible value of y' such that it can form a ℓ-tuple with sum y with $\ell-1$ elements already in the database. Thus we have $\left|\mathcal{O}_{t-1}\Pi_{\mathsf{inc}}^{y,\mathcal{O}_{t-1}}Q|\psi_t\rangle\right|\leq\sqrt{\frac{(t+k_0)^{\ell-1}}{N}}$. For $|x,u,w,r\rangle|D\rangle$ in R, we have $|D(x)\rangle_{\mathbf{D}_x}=\frac{N-1}{N}|D(x)\rangle+\frac{1}{\sqrt{N}}|\perp\rangle-\frac{1}{N}\sum_{y'\in[N]\setminus\{D(x)\}}|y'\rangle$ after $\mathcal{O}_{t-1}$. Since the only case where a new

value y' is added has amplitude $\frac{1}{N}$ and there are at most $(t+k_0)^{\ell-1}$ possible value of y' such that it can form a ℓ-tuple with sum y with $\ell-1$ elements already in the database, we have $\left|\mathcal{O}_{t-1}\Pi_{\mathsf{inc}}^{y,\mathcal{O}_{t-1}} R\left|\psi_t\right\rangle\right| \le \frac{(t+k_0)^{\ell-1}}{N}$.

Combining all cases above, we obtain the following recursion formula for $\Delta_{t,k}^y$:

$$\Delta_{t,k}^y \le \Delta_{t-1,k}^y + \Delta_{t-1,k-1}^y \cdot \left(4\sqrt{\frac{(t+k_0)^{\ell-1}}{N}}\right).$$

Thus

$$\Delta_{t,k}^y \le \binom{t}{k-k_\ell}\left(\sqrt{4\frac{(t+k_0)^{\ell-1}}{N}}\right)^{k-k_\ell}.$$

The probability that at some point during the algorithm there are K distinct ℓ-tuples of sum $y \in [N]$ in the database is at most

$$\begin{aligned}
&\left|\left(\sum_{i=0}^{2T-1}\prod_{i+1}^{j=2T-1}\left(\mathcal{O}_jU_j\right)\Pi_{\ge K}^{\ell,*}\mathcal{O}_iU_i\prod_{0}^{j=i-1}\left(\Pi_{<K}^{\ell,*}\mathcal{O}_jU_j\right)\right)\left|\psi_{\mathsf{st}}\right\rangle\right|^2\\
\le&\left|\left(\sum_{y\in[N]}\sum_{i=0}^{2T-1}\prod_{i+1}^{j=2T-1}\left(\mathcal{O}_jU_j\right)\Pi_{\ge K}^{\ell,y}\mathcal{O}_iU_i\prod_{0}^{j=i-1}\left(\Pi_{<K}^{\ell,y}\mathcal{O}_jU_j\right)\right)\left|\psi_{\mathsf{st}}\right\rangle\right|^2\\
\le& N\sum_{y\in[N]}\left|\left(\sum_{i=0}^{2T-1}\prod_{i+1}^{j=2T-1}\left(\mathcal{O}_jU_j\right)\Pi_{\ge K}^{\ell,y}\mathcal{O}_iU_i\prod_{0}^{j=i-1}\left(\Pi_{<K}^{\ell,y}\mathcal{O}_jU_j\right)\right)\left|\psi_{\mathsf{st}}\right\rangle\right|^2\\
\le& N\sum_{y\in[N]}\left(\Gamma_{2T,K}^y\right)^2\\
\le& N\left|\left(\sum_{i=0}^{2T-1}\prod_{i+1}^{j=2T-1}\left(\mathcal{O}_jU_j\right)\Pi_{\ge K_{\mathsf{thd}}}^{\ell-1,*}\mathcal{O}_iU_i\prod_{0}^{j=i-1}\left(\Pi_{<K_{\mathsf{thd}}}^{\ell-1,*}\mathcal{O}_jU_j\right)\right)\left|\psi_{\mathsf{st}}\right\rangle\right|^2\\
&+N\left|\left(\sum_{\substack{c_0,c_1,\cdots,c_t\\ \sum_{i=1}^T\mathbb{I}[c_i>c_{i-1}]\ge\frac{K-k_\ell}{K_{\mathsf{thd}}}}}\prod_{0}^{j=2T-1}\left(\Pi_{=c_{j+1}}^{\ell,y}\mathcal{O}_jU_j\right)\Pi_{=c_0}^{\ell,y}\right)\left|\psi_{\mathsf{st}}\right\rangle\right|^2\\
\le& NO\left(\frac{T^2(T+k_0)^{\ell-2}}{\left(K'_{\mathsf{sol}}\right)^2N}\right)^{K'_{\mathsf{sol}}}+N\left(\Delta_{2T,\frac{K-k_\ell}{K_{\mathsf{thd}}}}^y\right)^2\\
\le& NO\left(\frac{T^2(T+k_0)^{\ell-2}}{\left(K'_{\mathsf{sol}}\right)^2N}\right)^{K'_{\mathsf{sol}}}+N\binom{2T}{\frac{K-k_\ell}{K_{\mathsf{thd}}}}^2\left(\frac{16(2T+k_0)^{\ell-1}}{N}\right)^{\frac{K-k_\ell}{K_{\mathsf{thd}}}}\\
\le& NO\left(\frac{T^2(T+k_0)^{\ell-2}}{\left(K'_{\mathsf{sol}}\right)^2N}\right)^{K'_{\mathsf{sol}}}+NO\left(\frac{T^2(T+k_0)^{\ell-1}}{\left(\frac{K-k_\ell}{K_{\mathsf{thd}}}\right)^2N}\right)^{\frac{K-k_\ell}{K_{\mathsf{thd}}}}.
\end{aligned}$$

We set K_{thd} in a way that $K_{\mathsf{sol}} := \frac{K-k_\ell}{K_{\mathsf{thd}}}$ is the only real non-negative root of the function

$$f_{K,k_1,k_2,\cdots,k_\ell}(x) = x^\ell + \sum_{i=0}^{\ell-1} k_{\ell-i}x^i - K.$$

Since we know that K'_{sol} is the only real non-negative root of the function

$$f_{K_{\mathsf{thd}},k_1,k_2,\cdots,k_{\ell-1}}(x) = x^{\ell-1} + \sum_{i=0}^{\ell-2} k_{\ell-i-1}x^i - K_{\mathsf{thd}}.$$

We have that $K'_{\mathsf{sol}} = \frac{K-k_\ell}{K_{\mathsf{thd}}}$. In other words, $\frac{K-k_\ell}{K_{\mathsf{thd}}}$ is also the only real non-negative root of the function $f_{K_{\mathsf{thd}},k_1,k_2,\cdots,k_{\ell-1}}(x)$. Proved by

$$\begin{aligned}
&f_{K_{\mathsf{thd}},k_1,k_2,\cdots,k_{\ell-1}}\left(\frac{K-k_\ell}{K_{\mathsf{thd}}}\right)\\
=&\left(\frac{K-k_\ell}{K_{\mathsf{thd}}}\right)^{\ell-1} + \sum_{i=0}^{\ell-2} k_{\ell-i-1}\left(\frac{K-k_\ell}{K_{\mathsf{thd}}}\right)^i - K_{\mathsf{thd}}\\
=&\left(\frac{K_{\mathsf{thd}}}{K-k_\ell}\right)\left(\left(\frac{K-k_\ell}{K_{\mathsf{thd}}}\right)^{\ell} + \sum_{i=1}^{\ell-1} k_{\ell-i}\left(\frac{K-k_\ell}{K_{\mathsf{thd}}}\right)^i - K + k_\ell\right)\\
=&\left(\frac{K_{\mathsf{thd}}}{K-k_\ell}\right)\left(\left(\frac{K-k_\ell}{K_{\mathsf{thd}}}\right)^{\ell} + \sum_{i=0}^{\ell-1} k_{\ell-i}\left(\frac{K-k_\ell}{K_{\mathsf{thd}}}\right)^i - K\right)\\
=&\left(\frac{K_{\mathsf{thd}}}{K-k_\ell}\right) f_{K,k_1,k_2,\cdots,k_\ell}\left(\frac{K-k_\ell}{K_{\mathsf{thd}}}\right) = 0.
\end{aligned}$$

Finishing the computation, the probability that at some point during the algorithm there are K distinct ℓ-tuples with the same sum in the database is at most

$$\begin{aligned}
&NO\left(\frac{T^2(T+k_0)^{\ell-2}}{\left(K'_{\mathsf{sol}}\right)^2 N}\right)^{K'_{\mathsf{sol}}} + NO\left(\frac{T^2(T+k_0)^{\ell-1}}{\left(\frac{K-k_\ell}{K_{\mathsf{thd}}}\right)^2 N}\right)^{\frac{K-k_\ell}{K_{\mathsf{thd}}}}\\
\le& NO\left(\frac{T^2(T+k_0)^{\ell-2}}{K_{\mathsf{sol}}^2 N}\right)^{K_{\mathsf{sol}}} + NO\left(\frac{T^2(T+k_0)^{\ell-1}}{K_{\mathsf{sol}}^2 N}\right)^{K_{\mathsf{sol}}}\\
\le& O\left(\frac{T^2(T+k_0)^{\ell-1}}{K_{\mathsf{sol}}^2 N}\right)^{K_{\mathsf{sol}}}
\end{aligned}$$

The N factor can be absorbed into the big-O notation because $K_{\mathsf{sol}} = \Omega(\log N)$. Note that since $\ell > 0$ is a constant, doing induction on it preserves the $O(\cdot)$ notation. □

8 The time bound for finding K ℓ-tuples with the same sum with advice

Now we consider an algorithm starting with S-qubit advice state

$$|\psi_{\mathsf{st}}\rangle = \frac{1}{\sqrt{N^M 2^{M^\ell}}} \sum_{H,\hat{r}} |\psi_{H,\hat{r}}\rangle_{\mathbf{XUW}} |\hat{r}\rangle_{\mathbf{R}} |H\rangle_{\mathbf{D}},$$

where $|\psi_{H,\hat{r}}\rangle_{\mathbf{XUW}}$ is of size S. $\hat{r}$ is in the Hadamard basis representation and H is a standard oracle representation. $\mathbf{R}$ contains M^ℓ qubits as the first part which is labeled by ℓ-tuples and an arbitrary size second part (polynomial in N). Let $\hat{r}_{(x_1,x_2,\cdots,x_\ell)}/r_{(x_1,x_2,\cdots,x_\ell)}$ be the value on the qubit in the first part corresponding to the tuple $(x_1, x_2, \cdots, x_\ell)$. The structure $|\psi_{\mathsf{st}}\rangle$ means that $\mathbf{R}$ under Hadamard basis is a classical control register, we will use this fact in the next section. Another random challenge register $\mathbf{S}$ being initialized to $|0\rangle_{\mathbf{S}}$ and a memory qubit $\mathbf{B}$ initialized to $|0\rangle_{\mathbf{B}}$. For a sequence of target values $\{y_H\}_{H\in[N]^M}$ (we will just use $\{y_H\}$ later) and an algorithm $\mathcal{A}$:

- $\mathcal{A}$ is an isometry that receives S qubit advice $|\psi_{H,\hat{r}}\rangle_{\mathbf{XUW}}$. Its purification $\mathcal{A}'$ is a unitary (with access to the oracle and the output register) on $\mathbf{XUWARD}$ where $\mathbf{A}$ is the purification register of arbitrary size initialized to $|0\rangle_{\mathbf{A}}$.
- $\mathcal{A}'$ only queries the oracle register by PhO at most T times.
- $\mathcal{A}'$ only interact with the output register by $\mathtt{Output}$ at most T times. It can interact with both parts of the output register.

Remark 8.1. *In general, the advice may not be a pure state. But later on we will see that all experiments are linear which means we can take an average when the state is mixed. This will be used in later sections.*

We may also consider other inputs for $\mathcal{A}'$ other than $|\psi_{\mathsf{st}}\rangle$ later in this section. Define

$$\rho_{\mathsf{mixed}} = \left(\frac{I}{2^S}\right)_{\mathbf{XUW}},$$

$$\begin{aligned}\rho_{\mathsf{standard}} =& \rho_{\mathsf{mixed}} \otimes |0\rangle_{\mathbf{A}}\langle 0| \otimes |0\rangle_{\mathbf{R}}\langle 0| \otimes \frac{1}{N^M}\left(\sum_H |H\rangle_{\mathbf{D}}\right)\left(\sum_H \langle H|_{\mathbf{D}}\right) \otimes |0\rangle_{\mathbf{S}}\langle 0| \otimes |0\rangle_{\mathbf{B}}\langle 0| \\ =& \rho_{\mathsf{mixed}} \otimes |0\rangle_{\mathbf{A}}\langle 0| \otimes \frac{1}{N^M 2^{M^\ell}}\left(\sum_{H,\hat{r}} |\hat{r}\rangle_{\mathbf{R}} |H\rangle_{\mathbf{D}}\right)\left(\sum_{H,\hat{r}} \langle \hat{r}|_{\mathbf{R}} \langle H|_{\mathbf{D}}\right) \otimes |0\rangle_{\mathbf{S}}\langle 0| \otimes |0\rangle_{\mathbf{B}}\langle 0|\end{aligned}$$

and

$$\rho_{\mathsf{compressed}} = \rho_{\mathsf{mixed}} \otimes |0\rangle_{\mathbf{A}}\langle 0| \otimes |0\rangle_{\mathbf{R}}\langle 0| \otimes |\bot,\bot,\cdots,\bot\rangle_{\mathbf{D}}\langle \bot,\bot,\cdots,\bot| \otimes |0\rangle_{\mathbf{S}}\langle 0| \otimes |0\rangle_{\mathbf{B}}\langle 0|$$

which are inputs that the advice is removed.

Step 1: We are interested in the probability $P^{(1)}_{\mathsf{FlipTest}} := \left|\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}} \left(|\psi_{\mathsf{st}}\rangle |0\rangle_{\mathbf{A}} |0\rangle_{\mathbf{S}} |0\rangle_{\mathbf{B}}\right)\right|^2$ where $\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}$ is shown in Figure 4.

The idea of this experiment is that we want to know the probability of the algorithm $\mathcal{A}'$ recognizing a random ℓ-tuple. A random ℓ-tuple is stored in the register $\mathbf{S}$ and the projector projects to the subspace where that the bit corresponding to that ℓ-tuple is flipped after running $\mathcal{A}'$ and that ℓ-tuples has sum y_H. Ideally, we want to prove that the probability that any algorithm $\mathcal{A}'$ on any

$\Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}}$:

Domain: XUWRDSB.

1. Apply the unitary Gen that maps $|0\rangle_{\mathbf{S}}$ to $\sum_{(x_1,x_2,\cdots,x_\ell)\in[M]^\ell} |(x_1,x_2,\cdots,x_\ell)\rangle$.
2. Apply unitary Store:
$$|r\rangle_{\mathbf{R}} |(x_1,x_2,\cdots,x_\ell)\rangle_{\mathbf{S}} |b\rangle_{\mathbf{B}} \to |r\rangle_{\mathbf{R}} |(x_1,x_2,\cdots,x_\ell)\rangle_{\mathbf{S}} |b+r_{(x_1,x_2,\cdots,x_\ell)}\rangle_{\mathbf{B}}.$$
Which is a unitary that copies the output bit (in computational basis) on tuple $(x_1,x_2,\cdots,x_\ell)$ into the memory register.
3. Run $\mathcal{A}'$.
4. Apply unitary Store†.
5. Project the state onto the state where
   (a) The value on $\mathbf{B}$ is 1 under computational basis.
   (b) The ℓ-tuple on $\mathbf{S}$ is valid and has sum y_H where H is the oracle on $\mathbf{D}$.
6. Apply unitary Store again.
7. Run $\mathcal{A}'^\dagger$ (run $\mathcal{A}'$ in the reverse order and change every unitary into its conjugate)
8. Apply Gen†Store†.

Figure 4: Challenge an algorithm for finding a random ℓ-tuple.

starting state $|\psi_{\mathsf{st}}\rangle$ recognizing a random ℓ-tuple of sum y_H is extremely small when T is small. Because from last section we learn that for any uniform algorithm with T queries, the chance of finding too many ℓ-tuples of the same sum is very small. Even if we have a S-qubit advice, it should only increase the success probability by 2^S multiplicatively, which is not enough. The above reasoning seems to work out; however, in fact it takes many steps to implement it. Here is a road map with links on steps and connections between steps Figure 5 that provides an intuition of our proof.

Step 2: We first bound the probability $P_{\mathsf{FlipTest}}^{(1)}$ by the success probability of an alternating experiment. We introduce a [MW05]-form alternating experiment and prove the following result. Lemma 8.2, Lemma 8.3 and Lemma 8.4 are lemmas from [Liu22], we include the proof in Appendix A for completeness.

Lemma 8.2 ([Liu22], Lemma 6.5)**.** *Let* $P_{\mathsf{FlipTest}}^{(k)} := \left|\left(\Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}}\Pi_{\mathsf{reset}}\right)^k (|\psi_{\mathsf{st}}\rangle\,|0\rangle_{\mathbf{A}}\,|0\rangle_{\mathbf{S}}\,|0\rangle_{\mathbf{B}})\right|^2$ *where* $\Pi_{\mathsf{reset}} := |0\rangle_{\mathbf{S}}\langle 0| \otimes |0\rangle_{\mathbf{B}}\langle 0|$. *We have*

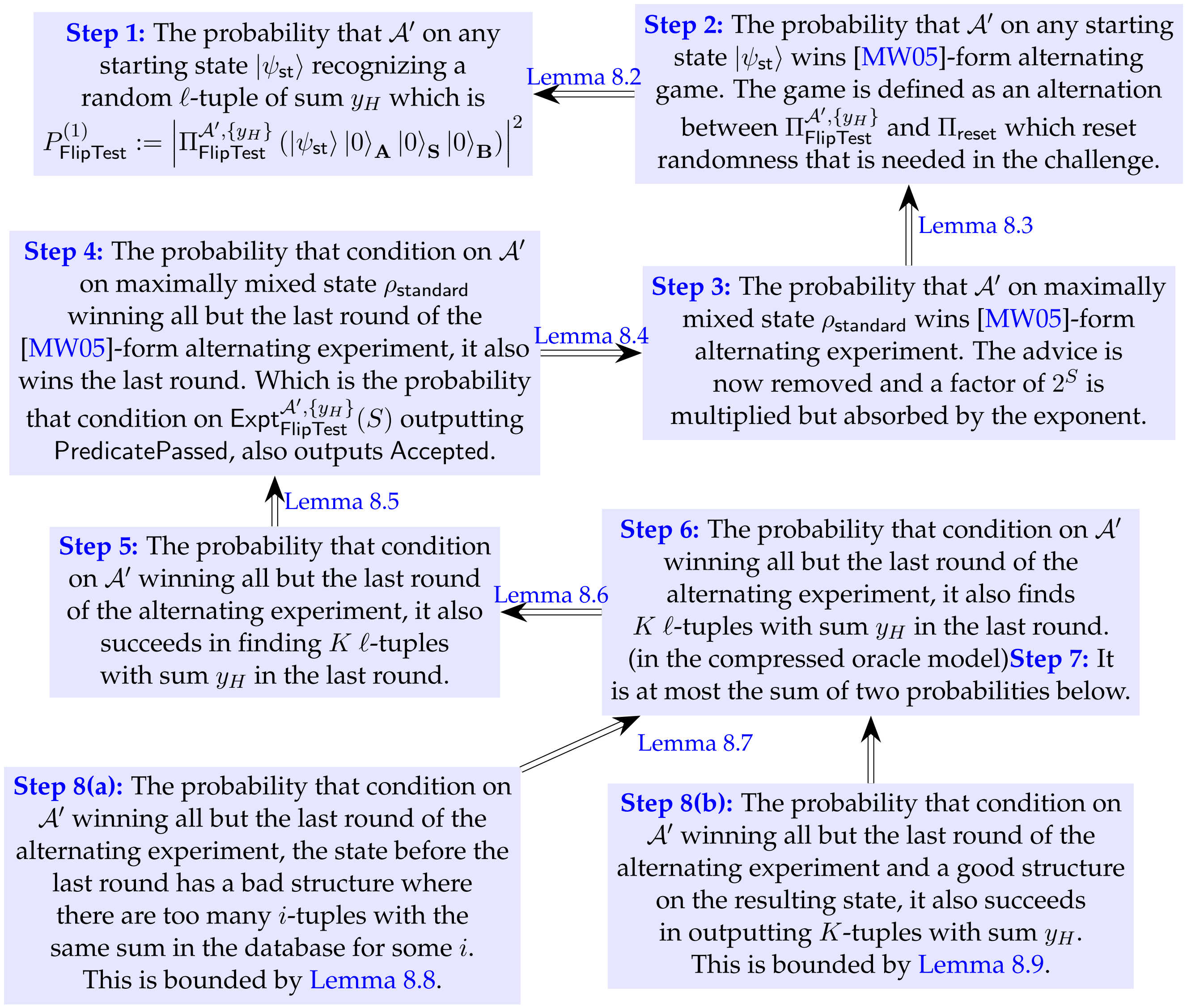
$$P_{\mathsf{FlipTest}}^{(1)} \le \left(P_{\mathsf{FlipTest}}^{(k)}\right)^{\frac{1}{2k-1}}.$$

Step 3: Now we try to substitute the advice state with a S-qubit maximally mixed state and absorb the 2^S factor by the exponent.

Lemma 8.3. *Let* $P_{\mathsf{FlipTest}}^{\mathsf{uniform},(k)} = \mathsf{Tr}\left(\left(\Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}}\Pi_{\mathsf{reset}}\right)^k \rho_{\mathsf{standard}} \left(\Pi_{\mathsf{reset}}\Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}}\right)^k\right)$. *We have*

$$P_{\mathsf{FlipTest}}^{(1)} \le 2\left(P_{\mathsf{FlipTest}}^{\mathsf{uniform},(S)}\right)^{\frac{1}{2S-1}}.$$

Step 4: Now we try to bound $P_{\mathsf{FlipTest}}^{\mathsf{uniform},(S)}$ and link it to a new experiment. Before that we first show a 'monotone' theorem on each alternation.


Step 1: The probability that $\mathcal{A}'$ on any starting state $|\psi_{\mathsf{st}}\rangle$ recognizing a random ℓ-tuple of sum y_H which is $P^{(1)}_{\mathsf{FlipTest}} := \left|\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}} \left(|\psi_{\mathsf{st}}\rangle |0\rangle_{\mathbf{A}} |0\rangle_{\mathbf{S}} |0\rangle_{\mathbf{B}}\right)\right|^2$
Lemma 8.2
Step 2: The probability that $\mathcal{A}'$ on any starting state $|\psi_{\mathsf{st}}\rangle$ wins [MW05]-form alternating game. The game is defined as an alternation between $\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}$ and Π_{reset} which reset randomness that is needed in the challenge.
Lemma 8.3
Step 4: The probability that condition on $\mathcal{A}'$ on maximally mixed state ρ_{standard} winning all but the last round of the [MW05]-form alternating experiment, it also wins the last round. Which is the probability that condition on $\mathsf{Expt}^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}(S)$ outputting PredicatePassed, also outputs Accepted.
Lemma 8.4
Step 3: The probability that $\mathcal{A}'$ on maximally mixed state ρ_{standard} wins [MW05]-form alternating experiment. The advice is now removed and a factor of 2^S is multiplied but absorbed by the exponent.
Lemma 8.5
Step 5: The probability that condition on $\mathcal{A}'$ winning all but the last round of the alternating experiment, it also succeeds in finding K ℓ-tuples with sum y_H in the last round.
Lemma 8.6
Step 6: The probability that condition on $\mathcal{A}'$ winning all but the last round of the alternating experiment, it also finds K ℓ-tuples with sum y_H in the last round. (in the compressed oracle model)**Step 7:** It is at most the sum of two probabilities below.
Lemma 8.7
Step 8(a): The probability that condition on $\mathcal{A}'$ winning all but the last round of the alternating experiment, the state before the last round has a bad structure where there are too many i-tuples with the same sum in the database for some i. This is bounded by Lemma 8.8.
Step 8(b): The probability that condition on $\mathcal{A}'$ winning all but the last round of the alternating experiment and a good structure on the resulting state, it also succeeds in outputting K-tuples with sum y_H. This is bounded by Lemma 8.9.
Figure 5: An overview of this section.

Lemma 8.4 ([Liu22], Corollary 6.10)**.** *For $k \geq 2$ we have*

$$\frac{P_{\mathsf{FlipTest}}^{\mathsf{uniform},(k)}}{P_{\mathsf{FlipTest}}^{\mathsf{uniform},(k-1)}} \leq \frac{P_{\mathsf{FlipTest}}^{\mathsf{uniform},(k+1)}}{P_{\mathsf{FlipTest}}^{\mathsf{uniform},(k)}}.$$

From this theorem, we can first bound $P_{\mathsf{FlipTest}}^{\mathsf{uniform},(S)}$ in the following way:

$$P_{\mathsf{FlipTest}}^{\mathsf{uniform},(S)} = \prod_{k=1}^{S} \left(\frac{P_{\mathsf{FlipTest}}^{\mathsf{uniform},(k)}}{P_{\mathsf{FlipTest}}^{\mathsf{uniform},(k-1)}} \right) \leq \left(\frac{P_{\mathsf{FlipTest}}^{\mathsf{uniform},(S)}}{P_{\mathsf{FlipTest}}^{\mathsf{uniform},(S-1)}} \right)^{S}.$$

Define $\Pi_{\mathsf{alter}}^{(k)} = \left(\Pi_{\mathsf{reset}} \Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}} \right)^{k}$. Define the following experiment Figure 6. We can notice that

$$\begin{aligned}
\frac{P_{\mathsf{FlipTest}}^{\mathsf{uniform},(S)}}{P_{\mathsf{FlipTest}}^{\mathsf{uniform},(S-1)}} &= \frac{\mathsf{Tr}\left(\left(\Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}} \Pi_{\mathsf{reset}} \right)^{S} \rho_{\mathsf{standard}} \left(\Pi_{\mathsf{reset}} \Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}} \right)^{S} \right)}{\mathsf{Tr}\left(\left(\Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}} \Pi_{\mathsf{reset}} \right)^{S-1} \rho_{\mathsf{standard}} \left(\Pi_{\mathsf{reset}} \Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}} \right)^{S-1} \right)} \\
&\leq \frac{\mathsf{Tr}\left(\Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}} \Pi_{\mathsf{alter}}^{(S-1)} \rho_{\mathsf{standard}} \Pi_{\mathsf{alter}}^{(S-1)} \Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}} \right)}{\mathsf{Tr}\left(\Pi_{\mathsf{alter}}^{(S-1)} \rho_{\mathsf{standard}} \Pi_{\mathsf{alter}}^{(S-1)} \right)} \\
&= \frac{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accepted}\} \subseteq \mathsf{Expt}_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}}(S) \right]}{\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}}(S) \right]}.
\end{aligned}$$

Step 5: Now we turn to another new experiment Figure 7. Instead of examine whether a single

$\mathsf{Expt}_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}}$:

Parameters: A round number k.

Setup: The algorithm starts with the empty state ρ_{standard}.

1. Apply a 2-outcome measurement $(\Pi_{\mathsf{alter}}^{(k-1)}, I - \Pi_{\mathsf{alter}}^{(k-1)})$ to the whole state.
   - If the outcome state is in $\Pi_{\mathsf{alter}}^{(k-1)}$, let $R = \{\mathsf{PredicatePassed}\}$.
   - If the outcome state is in $I - \Pi_{\mathsf{alter}}^{(k-1)}$, let $R = \{\mathsf{PredicateFailed}\}$.
2. Apply a 2-outcome measurement $(\Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}}, I - \Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}})$.
   - If the outcome state is in $\Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}}$, output $R \cup \{\mathsf{Accepted}\}$.
   - If the outcome state is in $I - \Pi_{\mathsf{FlipTest}}^{\mathcal{A}',\{y_H\}}$, output $R \cup \{\mathsf{Rejected}\}$.

Figure 6: Challenge an algorithm for finding a random ℓ-tuple with predicate.

ℓ-tuple is recognized, this experiment now cares about the probability of $\mathcal{A}'$ outputting at least K ℓ-tuples of the same sum. Here we say that $\mathcal{A}'$ outputs a ℓ-tuple $(x_1, x_2, \cdots, x_\ell)$ if $r_{(x_1,x_2,\cdots,x_\ell)}$ is non-zero. Intuitively, PredicatePassed/PredicateFailed in the output of $\mathsf{Expt}_{\mathsf{Standard}}^{\mathcal{A}'}$ means that the algorithm passes/fails in the predicate test and Accepted/Rejected in the output of $\mathsf{Expt}_{\mathsf{Standard}}^{\mathcal{A}'}$ means the algorithm is/isn't able to find K ℓ-tuples of the same sum.

$\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}$:

Parameters: A projector $\Pi_{\mathsf{predicate}}$ and a threshold K.
Setup: The algorithm starts with the state ρ_{standard}.

1. Apply a 2-outcome measurement $(\Pi_{\mathsf{predicate}}, I - \Pi_{\mathsf{predicate}})$ to the whole state.
   - If the outcome state is in $\Pi_{\mathsf{predicate}}$, let $R = \{\mathsf{PredicatePassed}\}$.
   - If the outcome state is in $I - \Pi_{\mathsf{predicate}}$, let $R = \{\mathsf{PredicateFailed}\}$.
2. For every $y \in [N]$, count how many distinct ℓ-tuples outputted by the prover has sum y for every y. If there exists $y \in [N]$ such that this number is at least K then let $R \leftarrow R \cup \{\mathsf{Accept0}\}$. Otherwise let $R \leftarrow R \cup \{\mathsf{Rejected\_0}\}$.
3. Run the second stage of the algorithm $\mathcal{A}'$. All oracle queries use PhO, interacting with the oracle register H.
4. For every $y \in [N]$, count how many distinct ℓ-tuples outputted by the prover has sum y for every y. If there exists $y \in [N]$ such that this number is at least K then output $R \cup \{\mathsf{Accept1}\}$. Otherwise output $R \cup \{\mathsf{Rejected\_1}\}$.

Figure 7: Verifying whether an algorithm succeeds in finding K distinct ℓ-tuples the same sum with a predicate.

Lemma 8.5. *For any $k \geq 1$ and K we have*

$$\begin{aligned}&\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}(k)\right]\\ =&\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(k-1)}_{\mathsf{alter}}, K\right)\right]\end{aligned}$$

and

$$\begin{aligned}&\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accepted}\} \subseteq \mathsf{Expt}^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}(k)\right]\\ \leq&\frac{8K}{M^\ell}\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(k-1)}_{\mathsf{alter}}, K\right)\right]\\ &+4\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accept0}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(k-1)}_{\mathsf{alter}}, K\right)\right]\\ &+4\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accept1}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(k-1)}_{\mathsf{alter}}, K\right)\right].\end{aligned}$$

Proof. Let $\rho_{\mathsf{pass}} = \Pi^{(k-1)}_{\mathsf{alter}}\rho_{\mathsf{standard}}\Pi^{(k-1)}_{\mathsf{alter}}$ (not normalized). This is the state after step 1 in $\mathsf{Expt}^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}(k)$ and it is also the state after step 1 in $\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}$. Thus the two probabilities in the first equation are exactly the same:

$$\begin{aligned}&\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}(k)\right]\\ =&\mathsf{Tr}\left(\rho_{\mathsf{pass}}\right)\\ =&\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(k-1)}_{\mathsf{alter}}, K\right)\right].\end{aligned}$$

Now we prove the second inequality. Define

$$\Pi_b = \sum_{\substack{H,(x_1,x_2,\cdots,x_\ell)\text{ is a}\\ \text{valid }\ell\text{-tuple of sum }y_H}} |H\rangle_{\mathbf{D}}\langle H| \otimes |(x_1,x_2,\cdots,x_\ell)\rangle_{\mathbf{S}}\langle(x_1,x_2,\cdots,x_\ell)| \otimes |b\rangle_{\mathbf{R}_{(x_1,x_2,\cdots,x_\ell)}}\langle b|$$

and

$$\Pi_{\mathsf{Accepted}} = \sum_{\substack{H,r \text{ such that there are } K \\ \text{non-zero entries on } r \text{ that correspond} \\ \text{to a valid } \ell\text{-tuple of sum } y_H}} |H\rangle_{\mathbf{D}} \langle H| \otimes |r\rangle_{\mathbf{R}} \langle r| .$$

We have

$$\begin{aligned}
&\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accepted}\} \subseteq \mathsf{Expt}^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}(k)\right] \\
=&2\mathsf{Tr}\left(\Pi_0\mathcal{A}'\Pi_1\mathsf{Gen}\cdot\rho_{\mathsf{pass}}\cdot\mathsf{Gen}^{\dagger}\Pi_1\mathcal{A}'^{\dagger}\Pi_0\right) + \mathsf{Tr}\left(\Pi_1\mathcal{A}'\Pi_0\mathsf{Gen}\cdot\rho_{\mathsf{pass}}\cdot\mathsf{Gen}^{\dagger}\Pi_0\mathcal{A}'^{\dagger}\Pi_1\right) \\
\leq&2\mathsf{Tr}\left(\Pi_1\mathsf{Gen}\cdot\rho_{\mathsf{pass}}\cdot\mathsf{Gen}^{\dagger}\Pi_1\right) + 2\mathsf{Tr}\left(\Pi_1\mathcal{A}'\cdot\mathsf{Gen}\cdot\rho_{\mathsf{pass}}\cdot\mathsf{Gen}^{\dagger}\mathcal{A}'^{\dagger}\Pi_1\right) \\
\leq&4\mathsf{Tr}\left(\Pi_1\mathsf{Gen}(I-\Pi_{\mathsf{Accepted}})\rho_{\mathsf{pass}}(I-\Pi_{\mathsf{Accepted}})\mathsf{Gen}^{\dagger}\Pi_1\right) \\
&+4\mathsf{Tr}\left(\Pi_1(I-\Pi_{\mathsf{Accepted}})\mathcal{A}'\cdot\mathsf{Gen}\cdot\rho_{\mathsf{pass}}\cdot\mathsf{Gen}^{\dagger}\mathcal{A}'^{\dagger}(I-\Pi_{\mathsf{Accepted}})\Pi_1\right) \\
&+4\mathsf{Tr}\left(\Pi_1\mathsf{Gen}\Pi_{\mathsf{Accepted}}\cdot\rho_{\mathsf{pass}}\Pi_{\mathsf{Accepted}}\mathsf{Gen}^{\dagger}\Pi_1\right) \\
&+4\mathsf{Tr}\left(\Pi_1\Pi_{\mathsf{Accepted}}\mathcal{A}'\cdot\mathsf{Gen}\cdot\rho_{\mathsf{pass}}\cdot\mathsf{Gen}^{\dagger}\mathcal{A}'^{\dagger}\Pi_{\mathsf{Accepted}}\Pi_1\right) \\
\leq&\frac{8K}{M^{\ell}}\mathsf{Tr}\left(\rho_{\mathsf{pass}}\right) + 4\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accept0}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(k-1)}_{\mathsf{alter}}, K\right)\right] \\
&+4\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accept1}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(k-1)}_{\mathsf{alter}}, K\right)\right] \\
\leq&\frac{8K}{M^{\ell}}\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(k-1)}_{\mathsf{alter}}, K\right)\right] \\
&+4\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accept0}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(k-1)}_{\mathsf{alter}}, K\right)\right] \\
&+4\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accept1}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(k-1)}_{\mathsf{alter}}, K\right)\right] .
\end{aligned}$$

The reason on why

$$\mathsf{Tr}\left(\Pi_1\mathsf{Gen}(I-\Pi_{\mathsf{Accepted}})\rho_{\mathsf{pass}}(I-\Pi_{\mathsf{Accepted}})\mathsf{Gen}^{\dagger}\Pi_1\right) \leq \frac{K}{M^{\ell}}\mathsf{Tr}\left(\rho_{\mathsf{pass}}\right)$$

and

$$\mathsf{Tr}\left(\Pi_1(I-\Pi_{\mathsf{Accepted}})\mathcal{A}'\cdot\mathsf{Gen}\cdot\rho_{\mathsf{pass}}\cdot\mathsf{Gen}^{\dagger}\mathcal{A}'^{\dagger}(I-\Pi_{\mathsf{Accepted}})\Pi_1\right) \leq \frac{K}{M^{\ell}}\mathsf{Tr}\left(\rho_{\mathsf{pass}}\right)$$

is that once the state is in the span of $(I - \Pi_{\mathsf{Accepted}})$ then the number of 1 that corresponds to a valid ℓ-tuple of sum y_H is less or equal to K. For a random challenge on **S** the probability that the challenge hits one of the valid ℓ-tuple of sum y_H is at most $\frac{K}{M^{\ell}}$. □

Step 6: Now we switch the standard oracle into a compressed oracle.

Lemma 8.6. *For any algorithm $\mathcal{A}'$ the probability of any event happening in the experiment* $\mathsf{Expt}_{\mathsf{Standard}}$ *is the same as that in the experiment* $\mathsf{Expt}_{\mathsf{Compressed}}$. *That is, for any K and any subset of results*

$$R \subseteq \{\mathsf{PredicatePassed}, \mathsf{PredicateFailed}, \mathsf{Accept0}, \mathsf{Accept1}, \mathsf{Rejected\_0}, \mathsf{Rejected\_1}\},$$

we have

$$\Pr\left[R \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}(\Pi_{\mathsf{predicate}}, K)\right]$$

$\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Compressed}}$:

Parameters: A projector $\Pi_{\mathsf{predicate}}$ and a threshold K.
Setup: The algorithm starts with the state $\rho_{\mathsf{compressed}}$.

1. Define $\Pi'_{\mathsf{predicate}} = \mathsf{DecompressAll}^{\dagger}\Pi_{\mathsf{predicate}}\mathsf{DecompressAll}$ where $\mathsf{DecompressAll}$ is the unitary that applies $\mathsf{StdDecomp}_x$ for every $x \in [M]$. Apply a 2-outcome measurement $(\Pi'_{\mathsf{predicate}}, I - \Pi'_{\mathsf{predicate}})$ to the whole state.
    - If the outcome state is in $\Pi'_{\mathsf{predicate}}$, let $R = \{\mathsf{PredicatePassed}\}$.
    - If the outcome state is in $I - \Pi'_{\mathsf{predicate}}$, let $R = \{\mathsf{PredicateFailed}\}$.
2. For every $y \in [N]$, count how many distinct ℓ-tuples outputted by the prover has sum y for every y. To do this, first run $\mathsf{DecompressAll}$ to convert the compress oracle into a standard oracle, count it and run $\mathsf{DecompressAll}^{\dagger}$ to recover. If there exists $y \in [N]$ such that this number is at least K then let $R \leftarrow R \cup \{\mathsf{Accept0}\}$. Otherwise let $R \leftarrow R \cup \{\mathsf{Rejected\_0}\}$.
3. Run the second stage of the algorithm $\mathcal{A}'$. All oracle queries use CPhO, interacting with the database $\mathbf{D}$.
4. For every $y \in [N]$, count how many distinct ℓ-tuples outputted by the prover has sum y for every y. To do this, first run $\mathsf{DecompressAll}$ to convert the compress oracle into a standard oracle, count it and run $\mathsf{DecompressAll}^{\dagger}$ to recover. If there exists $y \in [N]$ such that this number is at least K then output $R \cup \{\mathsf{Accept1}\}$. Otherwise output $R \cup \{\mathsf{Rejected\_1}\}$.

Figure 8: Verifying whether an algorithm succeeds in finding K distinct ℓ-tuples the same sum with a predicate.

$$= \Pr\left[R \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Compressed}}(\Pi_{\mathsf{predicate}}, K)\right].$$

Proof. We prove that at any time the state on $\mathbf{D}_x$ in $\mathsf{Expt}_{\mathsf{Compressed}}$ and in $\mathsf{Expt}_{\mathsf{Standard}}$ differs by exactly $\mathsf{StdDecomp}_x$. At some point, the state on $\mathbf{D}_x$ in $\mathsf{Expt}_{\mathsf{Compressed}}$ is

$$\begin{aligned}
&\mathsf{StdDecomp}_x \cdot \prod_j \mathsf{CPhO}_{u_j} |\bot\rangle \\
=&\mathsf{StdDecomp}_x \cdot \prod_j \left(\mathsf{StdDecomp}_x \cdot \mathsf{CPhO}'_{u_j} \cdot \mathsf{StdDecomp}_x\right) |\bot\rangle \\
=&\prod_j \left(\mathsf{CPhO}'_{u_j}\right) \mathsf{StdDecomp}_x |\bot\rangle \\
=&\prod_j \mathsf{CPhO}'_{u_j} \left(\sum_{y\in[N]} \frac{1}{\sqrt{N}} |y\rangle\right)
\end{aligned}$$

where CPhO_{u_j} are all gates that acts on this register.
One can see that using compress oracle queries in the middle and a StdDecomp at the end is equivalent to initializing that database with a uniform random value and then using CPhO' for each oracle query. If you interpret the initialization $|\bot\rangle \to \left(\sum_{y\in[N]} \frac{1}{\sqrt{N}} |y\rangle\right)$ as the initialization of H you will find that this two approaches are exactly the same. Thus this two experiments must have the same result. □

Step 7: Now we consider the another experiment Figure 9 which is similar to Figure 8 but another structure projector is added in between step 1 and step 2. The intuition on this experiment is that it split the state into two types. In the case where the state has a good structure, which means that the number of i-tuples in the database is small for all i, the success probability in later stages will be bounded due to a lack of information on tuples. And we argue that it is unlikely for the state to have a bad structure, which means that there are too many i-tuples of the same sum for some i that provides to much information in later stages.

Intuitively, $\mathsf{GoodStructure}/\mathsf{BadStructure}$ in the output of $\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi_{\mathsf{predicate}}, K, k_0, k_1, \cdots, k_\ell)$

$\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}$:

Parameters: A projector $\Pi_{\mathsf{predicate}}$, a threshold K and integers $k_0, k_1, \cdots, k_\ell$.

Setup: The algorithm starts with the state $\rho_{\mathsf{compressed}}$.

1. Define $\Pi'_{\mathsf{predicate}} = \mathsf{DecompressAll}^\dagger \Pi_{\mathsf{predicate}} \mathsf{DecompressAll}$ where $\mathsf{DecompressAll}$ is the unitary that applies $\mathsf{StdDecomp}_x$ for every $x \in [M]$. Apply a 2-outcome measurement $(\Pi'_{\mathsf{predicate}}, I - \Pi'_{\mathsf{predicate}})$ to the whole state.
   - If the outcome state is in $\Pi'_{\mathsf{predicate}}$, let $R = \{\mathsf{PredicatePassed}\}$.
   - If the outcome state is in $I - \Pi'_{\mathsf{predicate}}$, let $R = \{\mathsf{PredicateFailed}\}$.
2. Define $\Pi_{\mathsf{structure}} = \Pi^{>k_0} \prod_{i=1}^{\ell} \Pi^{i,*}_{>k_i}$. Apply a 2-outcome measurement $(\Pi_{\mathsf{structure}}, I - \Pi_{\mathsf{structure}})$ to the whole state.
   - If the outcome state is in $\Pi_{\mathsf{structure}}$, let $R \leftarrow R \cup \{\mathsf{GoodStructure}\}$.
   - If the outcome state is in $I - \Pi_{\mathsf{structure}}$, let $R \leftarrow R \cup \{\mathsf{BadStructure}\}$.
3. For every $y \in [N]$, count how many distinct ℓ-tuples outputted by the prover has sum y for every y. To do this, first run $\mathsf{DecompressAll}$ to convert the compress oracle into a standard oracle, count it and run $\mathsf{DecompressAll}^\dagger$ to recover. If there exists $y \in [N]$ such that this number is at least K then let $R \leftarrow R \cup \{\mathsf{Accept0}\}$. Otherwise let $R \leftarrow R \cup \{\mathsf{Rejected\_0}\}$.
4. Run the second stage of the algorithm $\mathcal{A}'$. All oracle queries use CPhO, interacting with the database $\mathbf{D}$.
5. For every $y \in [N]$, count how many distinct ℓ-tuples outputted by the prover has sum y for every y. To do this, first run $\mathsf{DecompressAll}$ to convert the compress oracle into a standard oracle, count it and run $\mathsf{DecompressAll}^\dagger$ to recover. If there exists $y \in [N]$ such that this number is at least K then output $R \cup \{\mathsf{Accept1}\}$. Otherwise output $R \cup \{\mathsf{Rejected\_1}\}$.

Figure 9: Verifying whether an algorithm succeeds in finding K distinct ℓ-tuples the same sum with a predicate and a structural test.

means that there doesn't/does exist $i \in \{0, 1, \cdots, \ell\}$ and $y \in [N]$ such that there are more than k_i i-tuples of sum y in the database after step 1.

Lemma 8.7. *For any $k_1, k_2, \cdots, k_\ell < K$. We can bound the success probability of the algorithm in the standard experiment condition on the predicate test passed by, intuitively, the probability of acceptance condition on a good structure on the database plus the probability of having a bad structure condition on predicate test passed:*

$$\frac{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accept0}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}(\Pi^{(k)}_{\mathsf{alter}}, K)\right]}{\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}(\Pi^{(k)}_{\mathsf{alter}}, K)\right]} + \frac{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{Accept1}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}(\Pi^{(k)}_{\mathsf{alter}}, K)\right]}{\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}(\Pi^{(k)}_{\mathsf{alter}}, K)\right]}$$
$$\leq \frac{2\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{GoodStructure}, \mathsf{Accept0}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell)\right]}{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{GoodStructure}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell)\right]}$$

$$+\frac{2\Pr\left[\{\mathsf{PredicatePassed},\mathsf{GoodStructure},\mathsf{Accept1}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}{\Pr\left[\{\mathsf{PredicatePassed},\mathsf{GoodStructure}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}$$
$$+\frac{4\Pr\left[\{\mathsf{PredicatePassed},\mathsf{BadStructure}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}{\Pr\left[\mathsf{PredicatePassed}\in\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}.$$

Proof.

$$\begin{aligned}
&\frac{\Pr\left[\{\mathsf{PredicatePassed},\mathsf{Accept0}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}(\Pi^{(k)}_{\mathsf{alter}},K)\right]}{\Pr\left[\mathsf{PredicatePassed}\in\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}(\Pi^{(k)}_{\mathsf{alter}},K)\right]}
+\frac{\Pr\left[\{\mathsf{PredicatePassed},\mathsf{Accept1}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}(\Pi^{(k)}_{\mathsf{alter}},K)\right]}{\Pr\left[\mathsf{PredicatePassed}\in\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}(\Pi^{(k)}_{\mathsf{alter}},K)\right]}\\
=&\frac{\Pr\left[\{\mathsf{PredicatePassed},\mathsf{Accept0}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Compressed}}(\Pi^{(k)}_{\mathsf{alter}},K)\right]}{\Pr\left[\mathsf{PredicatePassed}\in\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Compressed}}(\Pi^{(k)}_{\mathsf{alter}},K)\right]}
+\frac{\Pr\left[\{\mathsf{PredicatePassed},\mathsf{Accept1}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Compressed}}(\Pi^{(k)}_{\mathsf{alter}},K)\right]}{\Pr\left[\mathsf{PredicatePassed}\in\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Compressed}}(\Pi^{(k)}_{\mathsf{alter}},K)\right]}\\
\leq&\frac{2\Pr\left[\{\mathsf{PredicatePassed},\mathsf{Accept0}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}{\Pr\left[\mathsf{PredicatePassed}\in\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}
+\frac{2\Pr\left[\{\mathsf{PredicatePassed},\mathsf{Accept1}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}{\Pr\left[\mathsf{PredicatePassed}\in\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}\\
\leq&\frac{2\Pr\left[\{\mathsf{PredicatePassed},\mathsf{GoodStructure},\mathsf{Accept0}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}{\Pr\left[\{\mathsf{PredicatePassed},\mathsf{GoodStructure}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}\\
&+\frac{2\Pr\left[\{\mathsf{PredicatePassed},\mathsf{GoodStructure},\mathsf{Accept1}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}{\Pr\left[\{\mathsf{PredicatePassed},\mathsf{GoodStructure}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}\\
&+\frac{4\Pr\left[\{\mathsf{PredicatePassed},\mathsf{BadStructure}\}\subseteq\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}{\Pr\left[\mathsf{PredicatePassed}\in\mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}.
\end{aligned}$$

The constant 2 in the second equation is because of the additional 2-outcome measurement on the structure. □

Step 8(a): Now we prove that it is unlikely for the state to have a bad structure.

Lemma 8.8. *For $k_0 = 2kT$ and $k_1, \cdots, k_\ell$, if $T = \Omega(k_i)$ for every k_i then*

$$\Pr\left[\mathsf{BadStructure} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}\left(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell\right)\right] \le \sum_{i=1}^{\ell} O\left(\frac{(2kT)^{\ell+1}}{k_i^{\frac{2}{i}} N}\right)^{k_i^{\frac{1}{i}}}.$$

Proof. Consider the following algorithm $\mathcal{B}$:

1. Start with the state $\rho_{\mathsf{compressed}} \otimes |0,0,\cdots,0\rangle_{\mathbf{P}}\langle 0,0,\cdots,0|$ where $\mathbf{P}$ is a register that is used to purifying measurements.
2. Run $\Pi^{(k)}_{\mathsf{alter}}$, but for every projection (step 5(a) in Figure 4 and Π_{reset}) write the result of that projection on $\mathbf{P}$ to purify the process.

The probability

$$\Pr\left[\mathsf{BadStructure} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}\left(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell\right)\right]$$

is the probability that after running $\mathcal{B}$ on $\rho_{\mathsf{compressed}}$ one of the following happens:

1. The number of non-$\perp$ entries on $\mathbf{D}$ is more than k_0.
2. There exists $i \in 1, 2, \cdots, \ell$ and $y \in [N]$ such that the number of i-tuple of sum y is more than k_i.

Note that algorithm $\mathcal{B}$ queries the compress oracle for at most $2kT$ times and the initial state $\rho_{\mathsf{compressed}}$ has an empty database. Thus the first bullet above never happens. The second bullet for a fixed i happens with probability $O\left(\frac{(2kT)^{\ell+1}}{k_i^{\frac{2}{i}} N}\right)^{k_i^{\frac{1}{i}}}$ by Theorem 7.4. By union bound we can bound the probability by

$$\Pr\left[\mathsf{BadStructure} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}\left(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell\right)\right] \le \sum_{i=1}^{\ell} O\left(\frac{(2kT)^{\ell+1}}{k_i^{\frac{2}{i}} N}\right)^{k_i^{\frac{1}{i}}}.$$

□

Step 8(b): Now we prove if the state has a good structure, the success probability of later stages is bounded.

Lemma 8.9. *Condition on a the predicate passed and the structure being good, the success probability of the last round is bounded if $T = \Omega(K)$:*

$$\begin{aligned}
&\frac{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{GoodStructure}, \mathsf{Accept0}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell)\right]}{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{GoodStructure}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell)\right]} \\
+&\frac{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{GoodStructure}, \mathsf{Accept1}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell)\right]}{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{GoodStructure}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell)\right]} \\
\le& O\left(\frac{(kT)^2(kT+k_0)^{\ell-1}}{K_{\mathsf{sol}}^2 N}\right)^{K_{\mathsf{sol}}}.
\end{aligned}$$

for any k_0 and $k_1, \cdots, k_\ell < K$ where K_{sol} is the only real non-negative root of the following function

$$f_{K,k_1,k_2,\cdots,k_\ell}(x) = x^\ell + \sum_{i=0}^{\ell-1} k_{\ell-i}x^i - K.$$

Proof. To bound

$$\frac{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{GoodStructure}, \mathsf{Accept0}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell)\right]}{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{GoodStructure}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell)\right]}$$

in the last inequality, we consider the meaning of this probability. This probability is less or equal to the success probability of this algorithm $\mathcal{B}_0$:

1. The initial state is
$$\rho = \frac{\Pi_{\mathsf{structure}}\Pi'_{\mathsf{predicate}}\rho_{\mathsf{compressed}}\Pi'_{\mathsf{predicate}}\Pi_{\mathsf{structure}}}{\mathsf{Tr}\left(\Pi_{\mathsf{structure}}\Pi'_{\mathsf{predicate}}\rho_{\mathsf{compressed}}\Pi'_{\mathsf{predicate}}\Pi_{\mathsf{structure}}\right)}$$
where $\Pi'_{\mathsf{predicate}} = \mathsf{DecompressAll}^\dagger\Pi^{(k)}_{\mathsf{alter}}\mathsf{DecompressAll}$.
2. Controlled by the content on $\mathbf{R}$, let $Q = \bigcup_{r_{(x_1,x_2,\cdots,x_\ell)}=1} \{x_1, x_2, \cdots, x_\ell\}$ be all the points that are contained in the algorithm's output, swap them one by one to $\mathbf{X}$ and run StdDecomp every time to completely decompress the oracle into the standard oracle in these points.
3. Count the maximum number of ℓ-tuples that we can find in the compress oracle register $\mathbf{D}$. The algorithm succeeds if this number is at least K.

Note that algorithm $\mathcal{B}_0$ uses at most $O(kT)$ StdDecomp because the number of 1s on $\mathbf{R}$ is at most $O(kT)$ according to the definition of $\Pi^{(k)}_{\mathsf{alter}}$. Thus by Theorem 7.4 the success probability of $\mathcal{B}_0$ is at most $O\left(\frac{(kT)^2(kT+k_0)^{\ell-1}}{K_{\mathsf{sol}}^2 N}\right)^{K_{\mathsf{sol}}}$. Similarly,

$$\frac{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{GoodStructure}, \mathsf{Accept1}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell)\right]}{\Pr\left[\{\mathsf{PredicatePassed}, \mathsf{GoodStructure}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(k)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell)\right]}$$

is less or equal to two times (due to the 2-outcome measurement on step 3) of the success probability of this algorithm:

1. The initial state is
$$\rho = \frac{\Pi_{\mathsf{structure}}\Pi'_{\mathsf{predicate}}\rho_{\mathsf{compressed}}\Pi'_{\mathsf{predicate}}\Pi_{\mathsf{structure}}}{\mathsf{Tr}\left(\Pi_{\mathsf{structure}}\Pi'_{\mathsf{predicate}}\rho_{\mathsf{compressed}}\Pi'_{\mathsf{predicate}}\Pi_{\mathsf{structure}}\right)}$$
where $\Pi'_{\mathsf{predicate}} = \mathsf{DecompressAll}^\dagger\Pi^{(k)}_{\mathsf{alter}}\mathsf{DecompressAll}$.
2. Run $\mathcal{A}'$. All oracle queries use CPhO.
3. Controlled by the content on $\mathbf{R}$, let $Q = \bigcup_{r_{(x_1,x_2,\cdots,x_\ell)}=1} \{x_1, x_2, \cdots, x_\ell\}$ be all the points that are contained in the algorithm's output, swap them one by one to $\mathbf{X}$ and run StdDecomp every time to completely decompress the oracle into the standard oracle in these points.

4. Count the maximum number of ℓ-tuples that we can find in the compress oracle register $\mathbf{D}$. The algorithm succeeds if this number is at least K.

Note that algorithm $\mathcal{B}_1$ uses at most $O((k+1)T)$ StdDecomp because the number of 1s on $\mathbf{R}$ is at most $O((k+1)T)$ and at most $O(T)$ CPhO during step 2. Thus by Theorem 7.4 the success probability of $\mathcal{B}_0$ is at also most $O\left(\frac{(kT)^2(kT+k_0)^{\ell-1}}{K_{\mathsf{sol}}^2 N}\right)^{K_{\mathsf{sol}}}$. □

Remark 8.10. *One may wonder what if after projecting the state on $\Pi_{\mathsf{structure}}$ and run* DecompressAll, *there still exists $\perp$ entries in the oracle. This will not happen because the Hadamard basis is the eigenbasis for $\Pi_{\mathsf{structure}}$.*

Theorem 8.11 (Success probability in finding a random ℓ-tuple with advice)**.** *For all $\{y_H\}$ and all algorithms $\mathcal{A}'$ with T oracle query and T* Output *operations where $T=\Omega(S^{2\ell})$. Let $|\psi_{\mathsf{st}}\rangle$ be any S-qubit advice for the algorithm. The probability of $\mathcal{A}'$ successfully find a randomly generated ℓ-tuple of sum y_H is at most $\frac{2S^{2\ell}}{M^\ell}+O\left(\frac{(ST)^{\ell+1}}{N}\right)^S$. That is*

$$P^{(1)}_{\mathsf{FlipTest}}=\left|\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}\left(|\psi_{\mathsf{st}}\rangle|0\rangle_{\mathbf{A}}|0\rangle_{\mathbf{S}}|0\rangle_{\mathbf{B}}\right)\right|^2\le O\left(\frac{S^{2\ell}}{M^\ell}\right)+O\left(\frac{(ST)^{\ell+1}}{N}\right)^S$$

where $\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}$ is defined in Figure 4.

Proof. Set $k_0=2ST$, $k_i=S^{\ell+i}$ and $K=2S^{2\ell}$.

Claim 8.12. *Let K_{sol} be the only non-negative real root for function*

$$f(x)=x^\ell+\sum_{i=0}^{\ell-1}k_{\ell-i}x^i-K=x^\ell+\sum_{i=0}^{\ell-1}S^{2\ell-i}x^i-S^{2\ell}.$$

We have that $K_{\mathsf{sol}}\ge S$.

Proof.

$$\begin{aligned}f(S+1)=&(S+1)^\ell+S^\ell\frac{(S+1)^\ell-S^\ell}{(S+1)-S}-2S^{2\ell}\\ \le&(S+1)^\ell S^\ell(S+1)^\ell-2S^{2\ell}<0.\end{aligned}$$

Since $f(x)$ is monotonically+increasing on $(0,+\infty)$, $K_{\mathsf{sol}}\ge S$. □

Now we assume that

$$P^{\mathsf{uniform},(S)}_{\mathsf{FlipTest}}\ge\left(\frac{8K}{M^\ell}\right)^{2S}$$

cause otherwise this theorem holds immediately by

$$P^{(1)}_{\mathsf{FlipTest}}\le 2\left(P^{\mathsf{uniform},(S)}_{\mathsf{FlipTest}}\right)^{\frac{1}{2S-1}}\le 2\left(\frac{8K}{M^\ell}\right)^{\frac{2S}{2S-1}}\le\frac{16K}{M^\ell}.$$

Now we begin the proof:

$$P^{(1)}_{\mathsf{FlipTest}} \leq 2\left(P^{\mathsf{uniform},(S)}_{\mathsf{FlipTest}}\right)^{\frac{1}{2S-1}} \leq \frac{2P^{\mathsf{uniform},(S)}_{\mathsf{FlipTest}}}{P^{\mathsf{uniform},(S-1)}_{\mathsf{FlipTest}}}$$

$$\leq \frac{2\Pr\left[\{\mathsf{PredicatePassed},\mathsf{Accepted}\} \subseteq \mathsf{Expt}^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}(S)\right]}{\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}(S)\right]}$$

$$\leq \frac{8K}{M^\ell} + \frac{4\Pr\left[\{\mathsf{PredicatePassed},\mathsf{Accept0}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(S-1)}_{\mathsf{alter}},K\right)\right]}{\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(S-1)}_{\mathsf{alter}},K\right)\right]}$$

$$+ \frac{4\Pr\left[\{\mathsf{PredicatePassed},\mathsf{Accept1}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(S-1)}_{\mathsf{alter}},K\right)\right]}{\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Standard}}\left(\Pi^{(S-1)}_{\mathsf{alter}},K\right)\right]}$$

$$\leq \frac{8K}{M^\ell} + \frac{8\Pr\left[\{\mathsf{PredicatePassed},\mathsf{GoodStructure},\mathsf{Accept0}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(S-1)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}{\Pr\left[\{\mathsf{PredicatePassed},\mathsf{GoodStructure}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(S-1)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}$$

$$+ \frac{8\Pr\left[\{\mathsf{PredicatePassed},\mathsf{GoodStructure},\mathsf{Accept1}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(S-1)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}{\Pr\left[\{\mathsf{PredicatePassed},\mathsf{GoodStructure}\} \subseteq \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}(\Pi^{(S-1)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell)\right]}$$

$$+ \frac{16\Pr\left[\mathsf{BadStructure} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}\left(\Pi^{(S-1)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell\right)\right]}{\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}\left(\Pi^{(S-1)}_{\mathsf{alter}},K,k_0,k_1,\cdots,k_\ell\right)\right]}$$

$$\leq \frac{8K}{M^\ell} + O\left(\frac{(ST)^2(ST+k_0)^{\ell-1}}{K^2_{\mathsf{sol}}N}\right)^{K_{\mathsf{sol}}} + \frac{\sum_{i=1}^{\ell} O\left(\frac{(2ST)^{i+1}}{k_i^{\frac{2}{i}}N}\right)^{k_i^{\frac{1}{i}}}}{\left|\left(\Pi_{\mathsf{reset}}\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}\right)^{S-1}\rho_{\mathsf{compressed}}\left(\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}\Pi_{\mathsf{reset}}\right)^{S-1}\right|}$$

$$\leq \frac{8K}{M^\ell} + O\left(\frac{(ST)^2(ST+k_0)^{\ell-1}}{K^2_{\mathsf{sol}}N}\right)^{K_{\mathsf{sol}}} + \frac{\sum_{i=1}^{\ell} O\left(\frac{(2ST)^{i+1}}{k_i^{\frac{2}{i}}N}\right)^{k_i^{\frac{1}{i}}}}{P^{\mathsf{uniform},(S)}_{\mathsf{FlipTest}}}$$

$$\leq \frac{8K}{M^\ell} + O\left(\frac{(ST)^2(ST+k_0)^{\ell-1}}{K^2_{\mathsf{sol}}N}\right)^{K_{\mathsf{sol}}} + \frac{\sum_{i=1}^{\ell} O\left(\frac{(2ST)^{i+1}}{k_i^{\frac{2}{i}}N}\right)^{k_i^{\frac{1}{i}}}}{\left(\frac{8K}{M^\ell}\right)^{2S}}$$

$$\leq \frac{16S^{2\ell}}{M^\ell} + O\left(\frac{(ST)^{\ell+1}}{N}\right)^{S} + \frac{O\left(\frac{(ST)^{i+1}}{S^4N}\right)^{S^2}}{\left(\frac{16S^{2\ell}}{M^\ell}\right)^{2S}}$$

$$\leq O\left(\frac{S^{2\ell}}{M^\ell}\right) + O\left(\frac{(ST)^{\ell+1}}{N}\right)^{S} + O\left(\frac{S^{\ell-3}T^{\ell+1}}{N}\right)^{S^2}$$

$$\leq O\left(\frac{S^{2\ell}}{M^\ell}\right) + O\left(\frac{(ST)^{\ell+1}}{N}\right)^S.$$

Here are the reasons on why these inequalities hold:

- The first and second inequality follows from Lemma 8.2, Lemma 8.3 and Lemma 8.4.
- The third inequality follows from a direct interpretation.
- The fourth inequality follows from Lemma 8.5.
- The fifth inequality follows from Lemma 8.7.
- The sixth and seventh inequality follows from Lemma 8.8, Lemma 8.9 and an interpretation on the term
$$\Pr\left[\mathsf{PredicatePassed} \in \mathsf{Expt}^{\mathcal{A}'}_{\mathsf{Structured}}\left(\Pi^{(S-1)}_{\mathsf{alter}}, K, k_0, k_1, \cdots, k_\ell\right)\right].$$
- The eighth inequality holds because we assume $P^{\mathsf{uniform},(S)}_{\mathsf{FlipTest}} \geq \left(\frac{8K}{M^\ell}\right)^{2S}$.

□

Corollary 8.13 (Success probability in finding a random ℓ-tuple with mixed advice)**.** *For all $\{y_H\}$ and all algorithms $\mathcal{A}'$ with T oracle query and T Output operations where $T = \Omega(S^{2\ell})$. Let*

$$\rho_{\mathsf{st}} = \frac{1}{N^M 2^{M^\ell}} \sum_{H,\hat{r}} (\rho_{H,\hat{r}})_{\mathbf{XUW}} \otimes |\hat{r}\rangle_{\mathbf{R}} \langle \hat{r}| \otimes |H\rangle_{\mathbf{D}} \langle H|,$$

be any S-qubit mixed state advice for the algorithm. The probability of $\mathcal{A}'$ successfully find a randomly generated ℓ-tuple of sum y_H is at most $\frac{2S^{2\ell}}{M^\ell} + O\left(\frac{(ST)^{\ell+1}}{N}\right)^S$.

Proof. In Theorem 8.11, the probability is linear in the input state $|\psi_{\mathsf{st}}\rangle$. By averaging, you get the same bound for any mixed state advice. □

9 The time-space bound for finding K ℓ-tuples with the same sum

9.1 The classical bound

In this section we bound the probability of a classical algorithm outputting K ℓ-tuples with the same sum using S-bit space and T oracle queries. The model we are considering is that there is a write-only M^ℓ-bit classical memory where each bit indicates whether its corresponding ℓ-tuple is added to the output (0 for no and 1 for yes). The algorithm can only access this part of the memory by making an oracle call to the $\mathsf{FLIP}(x_1, x_2, \cdots, x_\ell)$ functionality that flips the bit that corresponds to the ℓ-tuple $(x_1, x_2, \cdots, x_\ell)$. The function FLIP will also reply to the algorithm with a random generated value $g_{(x_1,x_2,\cdots,x_\ell)}$ that is only generated once for the first time that the algorithm calls $\mathsf{FLIP}(x_1, x_2, \cdots, x_\ell)$ and remains the same for later calls. We say that the algorithm finds K ℓ-tuples with the same sum iff all bits that are 1 correspond to tuples with the same sum and the number of such bits is at least K. Here we provide a statement stronger than what we need. Instead of proving a time-space bound for finding K ℓ-tuples of sum y, we prove a time-space bound for finding K ℓ-tuples with the same sum.

Theorem 9.1. *Let $H : [M] \to [N]$ be a random oracle with sufficiently large M polynomial in N. For $\ell \geq 2$ and for any classical algorithm with space $S = \Omega(\log N) = O\left(N^{\frac{1}{\ell^2-1}}\right)$ and T oracle queries on* *either H or* FLIP. *Then the expectation of the maximum number of ℓ-tuples with the same sum found by the algorithm is $O\left(\frac{S^{\frac{\ell^2-1}{\ell}}T}{N^{\frac{1}{\ell}}}\right)$.*

Proof. Divide the algorithm into $L = T/T'$ segments with each of them querying the random oracle for $T' = cS^{\frac{1}{\ell}}N^{\frac{1}{\ell}} = \Omega(S^\ell)$ times. By setting c properly, by Theorem 7.3, we can guarantee that the probability of outputting at least S^ℓ ℓ-tuples with the same sum is at most $\frac{1}{M^\ell}$ for all segments since M is a polynomial of N. This is because the advice for each segment (i.e. the internal state at the start of that segment) is of size S and thus only provides a boost of 2^S on the probability in Theorem 7.3. The exponent, $K^{\frac{1}{\ell}} = S = \Omega(\log N)$ makes it possible to absorb 2^S into any factor that is polynomial in N into the constant c. With such c, the expected number of ℓ-tuples with the same sum found in every segment is at most $S^\ell + 1$. Thus the overall expectation of the maximum number of ℓ-tuples with the same sum found by the algorithm is at most $O\left(LS^\ell\right) = O\left(\frac{S^{\frac{\ell^2-1}{\ell}}T}{N^{\frac{1}{\ell}}}\right)$. □

9.2 The quantum bound

Now we bound the expected sum-$\{y_H\}$ ℓ-tuple capacity(Definition 9.4) of the state after an algorithm with S-qubit space and T oracle queries to two oracles $H : [M] \to [N], G : [M^\ell] \to [N_0]$. That is to say, we take the imaginary process of measuring $\mathbf{D}$ to obtain H, and measure $\mathbf{R}$ to obtain $\hat{r}$, then look at the entries with indices $(x_1 \cdots, x_\ell)$ that have sum y_H, and then count the number of non-zero entries among these entries. We specifically ask that the algorithm queries G is through HaO. Originally, for $G : [M^\ell] \to [N_0]$ we are interested in $K_{\{y_H\}}$ which is how many positions are non-zero in $|\hat{G}\rangle_{\mathbf{R}}$ under standard basis. Now we turn to consider the value $K^{(i)}_{\{y_H\}}$ for $i \in [n_0]$ where $K^{(i)}_{\{y_H\}}$ is the number of entries with its i-th bit equal to 1 under standard basis. And we know that $K_{\{y_H\}} \leq \sum_{i\in[n_0]} K^{(i)}_{\{y_H\}}$ because each non-zero entry must have at least one bit being 1. To bound $K^{(i)}_{\{y_H\}}$ instead, we notice that, in this case, $G : [M^\ell] \to [N_0]$ with T HaO queries is equivalent to $G_{\mathsf{Interested}} : [M^\ell] \to \{0,1\}$ with Tn_0 HaO queries on $G_{\mathsf{Interested}}$ and another 'useless' oracle $G_{\mathsf{NotInterested}} : [(n_0-1)M^\ell] \to \{0,1\}$. Where $G_{\mathsf{Interested}}(j)$ is the i-th bit of $G(j)$ and $G_{\mathsf{NotInterested}}(j(n_0-1)+k)$ is the $(k + \mathbb{1}[k \geq i])$-th bit of $G(j)$. We store both $G_{\mathsf{Interested}}$ and $G_{\mathsf{NotInterested}}$ together as G under the Hadamard basis in $\mathbf{R}$ where we are only interested in $G_{\mathsf{Interested}}$ which is the first M^ℓ bits of G, similar to the structure of $\mathbf{R}$ in the last section. We call this process reordering.

We first describe the settings of all registers. Considering that a space-bounded algorithm can introduce ancilla qubits into the working registers and discard some of them on the fly, we specify an ancilla register $\mathbf{A}$ to include these qubits. $\mathbf{A}$ can be of arbitrary size, and an algorithm could apply unitary across $\mathbf{A}$ and $\mathbf{XUW}$, but it can only pass the state in working registers $\mathbf{XUW}$ to the next timestep but not $\mathbf{A}$. In other words, different steps of algorithms can only access disjoint parts of $\mathbf{A}$. The registers $\mathbf{XUWRDA}$ are initialized as:

$$|\phi_0\rangle_{\mathbf{XUWRDA}} = \frac{1}{\sqrt{N^M 2^{M^\ell}}} \sum_{H,G} |\psi^{(0)}_{H,G}\rangle_{\mathbf{XUWA}} |\hat{G}\rangle_{\mathbf{R}} |H\rangle_{\mathbf{D}}$$

where $|\psi^{(0)}_{H,G}\rangle_{\mathbf{XUWA}}$ is the purification of the advice under H, G, while the algorithm just the mixed state in $\mathbf{XUW}$. For a uniform algorithm, we set $|\psi^{(0)}_{H,G}\rangle_{\mathbf{XUWA}} = |0\rangle_{\mathbf{XUWA}}$ for all H, G. Here, $|\widehat{G}\rangle$ is the Hadamard basis state corresponding to $G \in \{0,1\}^{M^\ell}$, we can also write it as a tensor product of entry-wise Hadamard basis,

$$|\widehat{G}\rangle_{\mathbf{R}} = \frac{1}{\sqrt{2^{M^\ell}}} \sum_{R\in\{0,1\}^{M^\ell}} (-1)^{\langle F,R\rangle} |R\rangle_{\mathbf{R}} = \bigotimes_{i\in[M^\ell]} \frac{1}{\sqrt{2}} \sum_{r\in\{0,1\}} (-1)^{\langle F(i),R(i)\rangle} |R\rangle_{\mathbf{R}_i} = \bigotimes_{i\in[M^\ell]} |\widehat{G(i)}\rangle_{\mathbf{R}_i},$$

with $\mathbf{R}_i$ being the i-th entry of the register with size 2. In the following analysis of this section, we will consider the purification of the possible mixed state in the registers $\mathbf{XUWRD}$ by incorporating register $\mathbf{A}$. Notice that any projector, observable and unitary on $\mathbf{XUWRD}$ could be extended to $\mathbf{XUWRDA}$ by tensoring an identity $I_{\mathbf{A}}$, which is equivalent to just acting on the mixed state in $\mathbf{XUWRD}$.

We specify that all queries made to G need to be performed through HaO (in this section all HaO refers to Hadamard oracle query to G), because it is consistent with the setting in the last section. Recall that $\mathsf{HaO}\, |x, u, w, H\rangle_{\mathbf{XUWD}} |R\rangle_{\mathbf{R}} = |x, u, w, H\rangle_{\mathbf{XUWD}} |R \oplus (x, u)\rangle_{\mathbf{R}}$, then it is completely identical to the Output operation for outputting ℓ-tuples. The initialization of registers here is also consistent with the last section. Specifically, for uniform algorithms, $\mathbf{R}$ is both initialized as $|0, \cdots, 0\rangle = \frac{1}{\sqrt{2^{M^\ell}}} \sum_{G\in\{0,1\}^{M^\ell}} |\widehat{G}\rangle$.

Therefore, we could transform all results on algorithms with H queries and Output operations to results on algorithms with H queries and HaO G queries. We firstly define the custom version for the random entry challenge game Figure 4 in our (H, G)-two oracle setting.

We define the $\mathsf{FlipTest}$ game in the following box, where the game is parameterized by fixed oracles H, G and a sequence $\{y_H\}$. Let $Y_{H,\{y_H\}} \subseteq [M^\ell]$ denote the subset of indices which is an ℓ-tuple with sum y_H, i.e. $Y_{H,\{y_H\}} = \{i = (i_1, \cdots, i_\ell) \mid \sum_{j=1}^{\ell} H(i_j) = y_H\}$. For simplicity, we will refer to it as Y_H in the following context. Denote $\mathbf{R}_{-i} = \bigotimes_{j\neq i} \mathbf{R}_j$, and G_{-i} as the $M^\ell - 1$ entries oracle from $[M^\ell]\backslash\{i\}$ to $\{0,1\}$ copied from G. Also denote $F_{-i} \cup f_i$ as the oracle from $[M^\ell]$ to $\{0,1\}$ obtained by replacing the i-th entry of F to $f \in \{0,1\}$. We maintain the notation of ancilla register $\mathbf{A}$, parsing it into $\mathbf{A} = \mathbf{A}_0\mathbf{A}_1$, where the other half of advice purification is in $\mathbf{A}_0$, and $\mathbf{A}_1$ is the unbounded extra space for $\mathcal{A}$.

Remark 9.2. *The reason that $\mathbf{A}_0$ exists is because the S-qubit advice state can be a mixed state and $\mathbf{A}_0$ serves as its purification, though it does not participate in any computation.*

$\mathsf{FlipTest}^{\{y_H\}}_{H,G}$**:**

Player: an algorithm $\mathcal{A}$, with purified advice state $|\psi^{(0)}_{H,G}\rangle_{\mathbf{XUWA}}$.
Parameters: $H, G, \{y_H\}$.

1. Randomly sample an index $i \xleftarrow{\$} [M]^\ell$.
2. Initialize registers $\mathbf{XUWRDA}$ as: $|\psi^{(0)}_{H,G}\rangle_{\mathbf{XUWA}_0} |0\rangle_{\mathbf{A}_1} |H\rangle_{\mathbf{D}} |\widehat{G}\rangle_{\mathbf{R}}$.
3. Replace the i-th entry of $\mathbf{R}$ with $|0\rangle$, obtaining $|\psi^{(0)}_{H,G}\rangle_{\mathbf{XUWA}_0} |0\rangle_{\mathbf{A}_1} |H\rangle_{\mathbf{D}} |\widehat{G_{-i}}\rangle_{\mathbf{R}_{-i}} |0\rangle_{\mathbf{R}_i}$.
4. Run $\mathcal{A}$ on $\mathbf{XUWA}_1$, with H queries onto $\mathbf{D}$ and HaO queries onto $\mathbf{R}$.
5. Measure the i-th entry $\mathbf{R}_i$ in computational basis, obtain outcome R_i, the player wins if both are true:
   (a) $R_i \neq 0$;
   (b) $i \in Y_H$.

Figure 10: Challenge an algorithm for finding a random ℓ-tuple when H, G are fixed.

We further define game $\mathsf{AvgFlipTest}^{\{y_H\}}$ as the average-case version of $\mathsf{FlipTest}$, while it runs $\mathsf{FlipTest}$ under $\{y_H\}$ and uniformly chosen H, G. We now argue that $\mathsf{AvgFlipTest}^{\{y_H\}}$ is almost identical to the game defined in Figure 4 except the algorithm is equipped with HaO instead of Output.

Claim 9.3. *For arbitrary $\{y_H\}$, any non-uniform algorithm $\mathcal{A}$ with oracle queries to H and HaO queries to G, denote $\mathcal{A}'$ as the algorithm obtained by only substituting all Output operations to HaO in $\mathcal{A}$. define $(\rho_{\mathsf{st}})_{\mathbf{XUWRD}}$ to be the mixed state being initialized, with another half purification in $\mathbf{A}_0$, then we have*

$$\Pr\left[\mathcal{A} \text{ wins } \mathsf{AvgFlipTest}^{\{y_H\}}\right] = \mathop{\mathbb{E}}_{H,G} \Pr\left[\mathcal{A} \text{ wins } \mathsf{FlipTest}^{\{y_H\}}_{H,G}\right] = \left| \Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}} \left(\rho_{\mathsf{st}} \left|0\right\rangle_{\mathbf{A}_1\mathbf{SB}} \left\langle 0\right|\right) \left(\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}\right)^{\dagger} \right|,$$

where in the middle term we are taking expectation over uniformly sampled random oracles H, G.

Proof. We first look at the projector $\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}$ defined in Figure 4. Notice that register $\mathbf{S}$ is initialized to be the superposition over $[M^\ell]$, then controls over the choice of index in $\mathbf{R}$, then measured in standard basis. This is equivalent to classically sample $i \xleftarrow{\$} [M^\ell]$.

Then, with random index i fixed, the unitary Store applies a generalized CNOT (we just use CNOT with a slight abuse of notation) acting on two single qubit registers $\mathbf{R}_i$ and $\mathbf{B}$ before and after running the algorithm $\mathcal{A}'$, where $\mathsf{CNOT}_{\mathbf{R}_i\mathbf{B}}$ denotes $\mathbf{R}_i$ controlling $\mathbf{B}$. Notice that $\mathsf{SWAP}_{\mathbf{R}_i\mathbf{B}} = \mathsf{CNOT}_{\mathbf{B}\mathbf{R}_i}\mathsf{CNOT}_{\mathbf{R}_i\mathbf{B}}\mathsf{CNOT}_{\mathbf{B}\mathbf{R}_i}$, also that algorithm $\mathcal{A}$ only interacts with register $\mathbf{R}_i$ by Output operations, which stabilizes states in the Hadamard basis, therefore it commutes with $\mathsf{CNOT}_{\mathbf{B}\mathbf{R}_i}$. Also notice that the initial state on $\mathbf{B}$ is $|0\rangle$, and we measure $\mathbf{B}$ in computational basis, which also commutes with $\mathsf{CNOT}_{\mathbf{B}\mathbf{R}_i}$. Thus, with fixed random index i, replacing the two $\mathsf{CNOT}_{\mathbf{R}_i\mathbf{B}}$ with two $\mathsf{SWAP}_{\mathbf{R}_i\mathbf{B}}$ does not affects the game. In this way, it is equivalent to the procedure in Figure 10 that swaps a $|0\rangle$ onto $\mathbf{R}_i$ and then measure this entry.

Then, consider the initialization of registers. In Figure 4, ρ_{st} is the reduced state of some $\frac{1}{\sqrt{N^M 2^{M^\ell}}} \sum_{H,\hat{r}} |\psi_{H,\hat{r}}\rangle_{\mathbf{XUWA}} |\hat{r}\rangle_{\mathbf{R}} |H\rangle_{\mathbf{D}}$, with r, H in uniform superposition. Since $|\widehat{r}\rangle\,|H\rangle$ is orthogonal for different $\widehat{r}, H$, it is equivalent with averaging over the same game on all possible $\widehat{r}, H$, which corresponds to $\mathsf{AvgFlipTest}^{\{y_H\}}$ as $\hat{r}, \widehat{G}$ are of same function in the process.

Therefore, we shown the equivalence of $\mathcal{A}'$ projected onto $\Pi^{\mathcal{A}',\{y_H\}}_{\mathsf{FlipTest}}$ and $\mathcal{A}$ winning $\mathsf{AvgFlipTest}^{\{y_H\}}$ with corresponding advice state.

□

Definition 9.4 (sum-$\{y_H\}$ ℓ-tuple capacity)**.** *For any state in registers $\mathbf{XUWRD}$, we define its **sum-$\{y_H\}$ ℓ-tuple capacity**:*

- *For each lazy sampling table $R \in \{0,1\}^{M^\ell}$, define its **subset-Y capacity** relative to a subset $Y \subseteq [M^\ell]$,*

$$\mathsf{Cap}_Y(R) = \sum_{i \in Y} \mathbb{1}\{R_i \neq 0\}.$$

- *Define two observables, $\widehat{\mathbf{K}}_{\{y_H\}}$ which relative to sequence $\{y_H\}$,*

$$\widehat{\mathbf{K}}_{\{y_H\}} = I_{\mathbf{XUW}} \otimes \sum_{H,R} \mathsf{Cap}_{Y_H}(R) \cdot |H\rangle\langle H|_{\mathbf{D}} \otimes |R\rangle\langle R|_{\mathbf{R}}\,,$$

while recall Y_H is the set of ℓ-tuples with sum y_H decided by $\{y_H\}$ and H.

*For any state $\rho_{\mathbf{XUWRD}}$, consider its expected value under the observables. We denote $K_{\{y_H\}} := \operatorname{Tr}\left[\rho \widehat{\mathbf{K}}_{\{y_H\}}\right]$ as its **expected sum-$\{y_H\}$ ℓ-tuple capacity** for state $|\phi\rangle$ relative to some sequence $\{y_H\}$. Specifically, when $\rho_{\mathbf{XUWRD}}$ has a purification $|\phi\rangle_{\mathbf{XUWRDA}}$, we say the expected sum-$\{y_H\}$ ℓ-tuple capacity of $|\phi\rangle$ is $\langle\phi| \widehat{\mathbf{K}}_{\{y_H\}} |\phi\rangle$, where in this term we extend the operator $\widehat{\mathbf{K}}_{\{y_H\}}$ to $\mathbf{A}$ by tensoring $I_{\mathbf{A}}$, with a little abuse of notation.*

Theorem 9.5. *Let $H : [M] \to [N]$, $G : [M^\ell] \to [N_0]$ be two random oracles. For any $\{y_H\}$ and any uniform quantum algorithm $\mathcal{A}$ with space $S = \Omega(\log N) = O\left(N^{\frac{1}{(\ell+1)(2\ell+1)}}\right)$, T queries to H, and T HaO queries to F, let $K_{\{y_H\}}$ be the expected sum-$\{y_H\}$ ℓ-tuple capacity of the final state. We have $K_{\{y_H\}}$ is upper bounded by $K_{\{y_H\}} = \widetilde{O}\left(S^{2\ell+2}T^2N^{-\frac{2}{\ell+1}}\right)$.*

Proof. We first apply the reordering process at the start of the section. And then with this new one-bit output G, we are going to bound $K^{(i)}_{\{y_H\}}$ for each i. Note that for now, the algorithm can make n_0T queries. For algorithm $\mathcal{A}$, we apply the **L-slicing** process as follows. View $\mathcal{A}$ as a sequence of local unitaries, H queries and HaO queries to G, then slice it into $L = 2n_0T/T'$ segments $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_L$ where each $\mathcal{A}_t$ has at most $T' = cS^{-1}N^{\frac{1}{\ell+1}}$ overall queries to H, G, c is some constant we will fix in later analysis. Denote $|\phi_0\rangle_{\mathbf{XUWRDA}}$ as the initial state before running algorithm, where $|\psi^{(0)}_{H,G}\rangle_{\mathbf{XUWA}}$ is all zero for any H, G. The registers $\mathbf{XUWA}$ are initialized to be all zero since $\mathcal{A}$ is a uniform algorithm. For each $t = 1, 2, \cdots, L$, denote

$$|\phi_t\rangle_{\mathbf{XUWRDA}} = \frac{1}{\sqrt{N^M 2^{M^\ell}}} \sum_{H,G} |\psi^{(t)}_{H,G}\rangle_{\mathbf{XUWA}} |H\rangle_{\mathbf{D}} |\widehat{G}\rangle_{\mathbf{R}}$$

to be the purified state after running segments from $\mathcal{A}_1, \cdots$, to $\mathcal{A}_t$.

We hereby specify the access to ancilla register $\mathbf{A} = \bigotimes_{t=0}^{L} \mathbf{A}_t$. where each sub-register $\mathbf{A}_t$ is disjoint and of arbitrary size. From the perspective of $\mathcal{A}_t$, the state before its operation has a purification across registers $\mathbf{A}_0 \cdots \mathbf{A}_{t-1}$, but it has no access to $\mathbf{A}_0 \cdots \mathbf{A}_{t-1}$ and can only apply unitaries over $\mathbf{XUW}$ and $\mathbf{A}_t$. In the following analysis, we may just use $\mathbf{A}$ in subscript for simplicity. For each induced segment $\mathcal{A}_t$, it is a non-uniform algorithm with S-qubit advice, where the advice is the mixed state in register $\mathbf{XUW}$ after running $\mathcal{A}_1, \cdots, \mathcal{A}_{t-1}$. Specifically, adopting a view back to without purifying oracles, H, G is classically sampled at the beginning, then algorithm $\mathcal{A}_t$ takes advice with purification $|\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}$ and apply some H, G-integrated local unitary $(U^{(t)}_{H,G})_{\mathbf{XUWA}_t}$ and obtain $|\psi^{(t)}_{H,G}\rangle_{\mathbf{XUWA}}$.

We then analyze the output register $\mathbf{R}$ after applying each slicing-induced $\mathcal{A}_t$. For any $t = 1, \cdots, L$, $\mathcal{A}_t$ has S-qubit mixed state advice and less than $T' = cS^{-1}N^{\frac{1}{\ell+1}} = \Omega\left(S^{2\ell}\right)$ queries due to the range upper bound of S. Then, by Corollary 8.13, any $\mathcal{A}_t$ has the following property:

$$\forall \{y_H\},\ p^{(t)}_{\{y_H\}} := \Pr\left[\mathcal{A}_t \text{ wins } \mathsf{AvgFlipTest}\right] \le p_e = O\left(\frac{(ST')^{\ell+1}}{N}\right)^S,$$

where p_e is obtained by taking a proper constant in the big-O notation. We denote such property as p_e**-bounded flipping**, namely winning the $\mathsf{AvgFlipTest}^{\{y_H\}}$ with at most probability p_e for any $\{y_H\}$. We then glue the property on each $\mathcal{A}_t$ together with the following lemma:

Lemma 9.6 (Quantum union bound on random sample game)**.** *For any sequence $\{y_H\}$, let $\mathcal{A}$ be an arbitrary algorithm with H queries and HaO queries to one-bit output G transformed from the reordering process that moves i-th bit of every point to the start, let $(\mathcal{A}_1, \cdots, \mathcal{A}_L)$ be an L-slicing of $\mathcal{A}$. Assume that*

$\forall t = 1, \cdots, L$, the non-uniform algorithm $\mathcal{A}_t$ is p_e-bounded flipping, then the expected sum-$\{y_H\}$ ℓ-tuple capacity of the final state after $\mathcal{A}$ is bounded, denoted as $K^{(i)}_{\{y_H\}}$:

$$K^{(i)}_{\{y_H\}} \leq L^2 \cdot M^{\ell} p_e.$$

Proof. For L-slicing induced non-uniform algorithms $\mathcal{A}_1, \cdots, \mathcal{A}_L$, denote $p^{(t)}_{\{y_H\}}$ as the probability of $\mathcal{A}_t$ winning the game $\mathsf{AvgFlipTest}^{\{y_H\}}$.Recall that we have specified the behavior of $\mathcal{A}_t$ under fixed H, G as follows:

- takes advice $\left(\rho^{(t-1)}_{H,G}\right)_{\mathbf{XUW}}$, which has a purification $|\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}_0\mathbf{A}_1\cdots\mathbf{A}_{t-1}}$;
- clarifies that it can only access registers $\mathbf{XUWA}_t$;
- applies H, G-integrated unitary $\left(U^{(t)}_{H,G}\right)_{\mathbf{XUWA}}$;
- obtains $|\psi^{(t)}_{H,G}\rangle_{\mathbf{XUWA}} = \left(U^{(t)}_{H,G}\right)_{\mathbf{XUWA}} |\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}$, with registers $\mathbf{A}_{t+1}\cdots\mathbf{A}_L$ still being all zeros.

We then look at the algorithm $\mathcal{A}$ in a whole. Under any H, G, the registers $\mathbf{XUWA}$ is initialized as $|\psi^{(0)}_{H,G}\rangle_{\mathbf{XUWA}} = |0\rangle_{\mathbf{XUWA}}$, and then the algorithm applies unitary $U_{H,G} = U^{(L)}_{H,G}\cdots U^{(2)}_{H,G}U^{(1)}_{H,G}$ over registers $\mathbf{XUWA}$, and obtain $|\psi^{(L)}_{H,G}\rangle_{\mathbf{XUWA}} = U_{H,G}|0\rangle_{\mathbf{XUWA}}$. Denote $(\sigma_L)_{\mathbf{XUWRD}}$ as the final state after applying $\mathcal{A}$, and $|\phi_L\rangle_{\mathbf{XUWRDA}}$ being the purification of it. Then the sum-$\{y_H\}$ ℓ-tuple capacity of the final state is $\langle\phi_L|\,\widehat{\mathbf{K}}_{\{y_H\}}\,|\phi_L\rangle$.

We first represent $p^{(t)}_{\{y_H\}}$ for any $1 \leq t \leq L$. For any G and $s \in \{0,1\}$, define $G^{i,s}$ to be the oracle obtained by replacing the i-th entry of G by s while others remain. We consider the process of $\mathcal{A}_t$ playing $\mathsf{FlipTest}^{\{y_H\}}_{H,G}$ for each pair of oracles H, G. On $i \xleftarrow{\$} [M^{\ell}]$ being sampled as the random index in step 1 of Figure 10, the entire state in $\mathbf{XUWRDA}$ evolves as follows:

$$\begin{aligned}
&|\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}|H\rangle_{\mathbf{D}}|\widehat{G}\rangle_{\mathbf{R}}\\
\xrightarrow{\text{replace }\mathbf{R}_i}&|\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}|H\rangle_{\mathbf{D}}|\widehat{G_{-i}}\rangle_{\mathbf{R}_{-i}}|0\rangle_{\mathbf{R}_i}\\
=&\frac{1}{\sqrt{2}}\sum_{s\in\{0,1\}}|\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}|H\rangle_{\mathbf{D}}|\widehat{G_{-i}}\rangle_{\mathbf{R}_{-i}}|\hat{s}\rangle_{\mathbf{R}_i}\\
\xrightarrow{\text{runs }\mathcal{A}_t}&\frac{1}{\sqrt{2}}\sum_{s\in\{0,1\}}U^{(t)}_{H,G^{i,s}}|\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}|H\rangle_{\mathbf{D}}|\widehat{G_{-i}}\rangle_{\mathbf{R}_{-i}}|\hat{s}\rangle_{\mathbf{R}_i}\\
=&\frac{1}{\sqrt{2}}\sum_{s\in\{0,1\}}U^{(t)}_{H,G^{i,s}}|\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}|H\rangle_{\mathbf{D}}|\widehat{G_{-i}}\rangle_{\mathbf{R}_{-i}}\frac{1}{\sqrt{2}}\sum_{r\in\{0,1\}}\omega_N^{s\cdot r}|r\rangle_{\mathbf{R}_i}\\
=&\sum_{r\in\{0,1\}}\left(\frac{1}{2}\sum_{s\in\{0,1\}}\omega_N^{s\cdot r}U^{(t)}_{H,G^{i,s}}|\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}\right)|H\rangle_{\mathbf{D}}|\widehat{G_{-i}}\rangle_{\mathbf{R}_{-i}}|r\rangle_{\mathbf{R}_i},
\end{aligned}$$

then when measuring $\mathbf{R}_i$, suppose we get r_i with probability $\Pr\left[r_i \neq 0 \mid i, \mathcal{A}_t, H, G\right]$. We have

$\Pr\big[r_i \neq 0 \mid i, \mathcal{A}_t, H, G\big] = 1 - \Pr\big[r_i = 0 \mid i, \mathcal{A}_t, H, G\big]$. Then, the overall winning probability is

$$\begin{aligned}
p^{(t)}_{H,G,\{y_H\}} =& \Pr[i \in Y_H] \cdot \Pr\Big[r_i \neq 0 \mid i, \mathcal{A}_t, H, G\Big] \\
=& \frac{1}{M^\ell} \sum_{i \in Y_H} \Big(1 - \Big|\frac{1}{2}\big(\sum_{s\in\{0,1\}} U^{(t)}_{H,G^{i,s}}\big) |\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}\Big|^2\Big) \\
=& \frac{1}{8M^\ell} \sum_{\substack{s,s' \in \{0,1\} \\ s \neq s'}} \sum_{i \in Y_H} \Big|\big(U^{(t)}_{H,G^{i,s}} - U^{(t)}_{H,G^{i,s'}}\big) |\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}\Big|^2.
\end{aligned}$$

We could then represent the p_e-bounded flipping property of $\mathcal{A}_t$ by

$$p^{(t)}_{\{y_H\}} = \frac{1}{8M^\ell} \sum_{\substack{s,s' \in \{0,1\} \\ s \neq s'}} \mathop{\mathbb{E}}_{H,G} \left[\sum_{i \in Y_H} \Big|\big(U^{(t)}_{H,G^{i,s}} - U^{(t)}_{H,G^{i,s'}}\big) |\psi^{(t-1)}_{H,G}\rangle_{\mathbf{XUWA}}\Big|^2\right] \leq p_e.$$

Then, we represent $\widehat{\mathbf{K}}^{(i)}_{\{y_H\}}$ in the similar way. For any H and $i \in [M^\ell]$, define projector $\Pi_{i,H} = I_{\mathbf{XUWA}} \otimes |H\rangle\langle H|_{\mathbf{D}} \otimes I_{\mathbf{R}_{-i}} \otimes (I - |0\rangle\langle 0|)_{\mathbf{R}_i}$, then with we notice that $\widehat{\mathbf{K}}^{(i)}_{\{y_H\}}$ could be written as a linear sum of projectors:

$$\widehat{\mathbf{K}}^{(i)}_{\{y_H\}} = I_{\mathbf{XUW}} \otimes \sum_{H,R} \mathsf{Cap}_{Y_H}(R) \cdot |H\rangle\langle H|_{\mathbf{D}} \otimes |R\rangle\langle R|_{\mathbf{R}} = \sum_{H} \sum_{i \in Y_H} \Pi_{i,H},$$

where each Y_H relative to H is induced by the fixed $\{y_H\}$. We then focus on the expected value of

$\Pi_{i,H}$ on the final state $|\phi_L\rangle_{\mathbf{XUWRDA}}$ for any H and $i \in [M^\ell]$.

$$\begin{aligned}
\langle\phi_L|\,\Pi_{i,H}\,|\phi_L\rangle =& \langle\phi_L|\,(I_{\mathbf{XUWAR}} \otimes |H\rangle\langle H|_{\mathbf{D}})\,|\phi_L\rangle - \langle\phi_L|\,\big(I_{\mathbf{XUWAR}_{-i}} \otimes |H\rangle\langle H|_{\mathbf{D}} \otimes |0\rangle\langle 0|_{\mathbf{R}_i}\big)\,|\phi_L\rangle \\
=& \frac{1}{N^M} - \left|\big(I_{\mathbf{XUWAR}_{-i}} \otimes |0\rangle\langle 0|_{\mathbf{R}_i}\big)\,|\phi_L\rangle\right|^2 \\
=& \frac{1}{N^M} - \left|\frac{1}{\sqrt{N^M 2^{M^\ell}}}\sum_G |\psi^{(L)}_{H,G}\rangle_{\mathbf{XUWA}}\,|H\rangle_{\mathbf{D}}\,|\widehat{G_{-i}}\rangle_{\mathbf{R}_{-i}}\,|0\rangle\langle 0|_{\mathbf{R}_i}\,|\widehat{G(i)}\rangle_{\mathbf{R}_i}\right|^2 \\
=& \frac{1}{N^M} - \frac{1}{N^M 2^{M^\ell}}\left|\sum_{G_{-i}}\sum_{s\in\{0,1\}} |\psi^{(L)}_{H,G^{i,s}}\rangle_{\mathbf{XUWA}}\,|H\rangle_{\mathbf{D}}\,|\widehat{G_{-i}}\rangle_{\mathbf{R}_{-i}}\,\frac{1}{\sqrt{2}}\,|0\rangle_{\mathbf{R}_i}\right|^2 \\
=& \frac{1}{N^M} - \frac{1}{N^M 2^{M^\ell}}\sum_{G_{-i}}\left|\sum_{s\in\{0,1\}} |\psi^{(L)}_{H,G^{i,s}}\rangle_{\mathbf{XUWA}}\,|H\rangle_{\mathbf{D}}\,|\widehat{G_{-i}}\rangle_{\mathbf{R}_{-i}}\,\frac{1}{\sqrt{2}}\,|0\rangle_{\mathbf{R}_i}\right|^2 \\
=& \frac{1}{N^M} - \frac{1}{N^M 2^{M^\ell}}\sum_{G_{-i}}\frac{1}{2}\left|\sum_{s\in\{0,1\}} |\psi^{(L)}_{H,G^{i,s}}\rangle_{\mathbf{XUWA}}\right|^2 \\
=& \frac{1}{N^M} - \frac{1}{N^M}\mathop{\mathbb{E}}_{G_{-i}}\left[\left|\frac{1}{2}\sum_{s\in\{0,1\}} |\psi^{(L)}_{H,G^{i,s}}\rangle_{\mathbf{XUWA}}\right|^2\right] \\
=& \frac{1}{N^M}\mathop{\mathbb{E}}_{G}\left[1 - \left|\frac{1}{2}\sum_{s\in\{0,1\}} |\psi^{(L)}_{H,G^{i,s}}\rangle_{\mathbf{XUWA}}\right|^2\right] \\
=& \frac{1}{8N^M}\sum_{\substack{s,s'\in\{0,1\}\\ s\neq s'}}\mathop{\mathbb{E}}_{G}\left|\,|\psi^{(L)}_{H,G^{i,s}}\rangle_{\mathbf{XUWA}} - |\psi^{(L)}_{H,G^{i,s'}}\rangle_{\mathbf{XUWA}}\,\right|^2.
\end{aligned}$$

The fifth equation is by the orthogonality of states with different G_{-i} in $|G_i\rangle_{\mathbf{R}_{-i}}$, and the sixth equation is by ignoring the norm-1 subsystem states in registers $\mathbf{D}, \mathbf{R}_{-i}, \mathbf{R}$. Therefore, adding the expected value of these projectors together, we have

$$K^{(i)}_{\{y_H\}} = \langle\phi_L|\,\widehat{\mathbf{K}}^{(i)}_{\{y_H\}}\,|\phi_L\rangle = \frac{1}{8}\sum_{\substack{s,s'\in\{0,1\}\\ s\neq s'}}\mathop{\mathbb{E}}_{H,G}\left[\sum_{i\in Y_H}\left|\,|\psi^{(L)}_{H,G^{i,s}}\rangle_{\mathbf{XUWA}} - |\psi^{(L)}_{H,G^{i,s'}}\rangle_{\mathbf{XUWA}}\,\right|^2\right].$$

Then, we bridge the gap from $p^{(t)}_{\{y_H\}}$ to $K_{\{y_H\}}$. For any fixed H, G, we bound the $L2$-distance of the final states $|\psi^{(L)}_{H,G^{i,s}}\rangle$ and $|\psi^{(L)}_{H,G^{i,s'}}\rangle$ by the distance independently imposed by each segment $\mathcal{A}_t$. Here, for $t_1 \le t_2$, denote $U^{t_1\to t_2}_{H,G} := U^{t_2}_{H,G}U^{t_2-1}_{H,G}\cdots U^{(t_1)}_{H,G}$; for $t_1 > t_2$, denote $U^{t_1\to t_2}_{H,G} := I$. We finally obtain

$$
\begin{aligned}
&K^{(i)}_{\{y_H\}}\\
=&\frac{1}{8}\sum_{\substack{s,s'\in\{0,1\}\\ s\neq s'}}\mathop{\mathbb{E}}_{H,G}\left[\sum_{i\in Y_H}\left|\,|\psi^{(L)}_{H,G^{i,s}}\rangle-|\psi^{(L)}_{H,G^{i,s'}}\rangle\,\right|^2\right]\\
=&\frac{1}{16}\sum_{\substack{s,s'\in\{0,1\}\\ s\neq s'}}\mathop{\mathbb{E}}_{H,G}\left[\sum_{i\in Y_H}\left|\left(U^{(1\to L)}_{H,G^{i,s}}-U^{(1\to L)}_{H,G^{i,s'}}\right)|0\rangle\right|^2+\sum_{i\in Y_H}\left|\left(U^{(1\to L)}_{H,G^{i,s'}}-U^{(1\to L)}_{H,G^{i,s}}\right)|0\rangle\right|^2\right]\\
=&\frac{1}{16}\sum_{\substack{s,s'\in\{0,1\}\\ s\neq s'}}\mathop{\mathbb{E}}_{H,G}\left[\sum_{i\in Y_H}\left|\sum_{t=1}^{L}\left(U^{(t+1\to L)}_{H,G^{i,s'}}U^{(1\to t)}_{H,G^{i,s}}-U^{(t\to L)}_{H,G^{i,s'}}U^{(1\to t-1)}_{H,G^{i,s}}\right)|0\rangle\right|^2\right.\\
&\left.+\sum_{i\in Y_H}\left|\sum_{t=1}^{L}\left(U^{(t+1\to L)}_{H,G^{i,s}}U^{(1\to t)}_{H,G^{i,s'}}-U^{(t\to L)}_{H,G^{i,s'}}U^{(1\to t-1)}_{H,G^{i,s}}\right)|0\rangle\right|^2\right]\\
\leq&\frac{1}{16}\sum_{\substack{s,s'\in\{0,1\}\\ s\neq s'}}\mathop{\mathbb{E}}_{H,G}\left[\sum_{i\in Y_H}L\sum_{t=1}^{L}\left|U^{(t+1\to L)}_{H,G^{i,s'}}\left(U^{(t)}_{H,G^{i,s'}}-U^{(t)}_{H,G^{i,s}}\right)U^{(1\to t-1)}_{H,G^{i,s}}|0\rangle\right|^2\right.\\
&\left.+\sum_{i\in Y_H}L\sum_{t=1}^{L}\left|U^{(t+1\to L)}_{H,G^{i,s'}}\left(U^{(t)}_{H,G^{i,s}}-U^{(t)}_{H,G^{i,s}}\right)U^{(1\to t-1)}_{H,G^{i,s'}}|0\rangle\right|^2\right]\\
=&\frac{L}{16}\sum_{\substack{s,s'\in\{0,1\}\\ s\neq s'}}\mathop{\mathbb{E}}_{H,G}\left[\sum_{i\in Y_H}\sum_{t=1}^{L}\left|\left(U^{(t)}_{H,G^{i,s'}}-U^{(t)}_{H,G^{i,s}}\right)|\psi^{(t-1)}_{H,G^{i,s}}\rangle\right|^2+\sum_{i\in Y_H}\sum_{t=1}^{L}\left|\left(U^{(t)}_{H,G^{i,s}}-U^{(t)}_{H,G^{i,s'}}\right)|\psi^{(t-1)}_{H,G^{i,s'}}\rangle\right|^2\right]\\
=&\frac{L}{8}\sum_{\substack{s,s'\in\{0,1\}\\ s\neq s'}}\mathop{\mathbb{E}}_{H,G}\left[\sum_{i\in Y_H}\sum_{t=1}^{L}\left|\left(U^{(t)}_{H,G^{i,s'}}-U^{(t)}_{H,G^{i,s}}\right)|\psi^{(t-1)}_{H,G}\rangle\right|^2\right]\\
=&L\sum_{t=1}^{L}M^{\ell}p^{(t)}_{\{y_H\}}\leq L^2\cdot M^{\ell}p_e.
\end{aligned}
$$

where the inequality is by Cauchy-Schwarz inequality; the fourth equation is by that unitaries are norm preserving. The sixth equality is because G is a one-bit output oracle $G(i)$ is either s or s'.

□

Combining Lemma 9.6 and the bound on p_e, also carefully picking constant c, we have

$$\begin{aligned}
\mathbb{E}[K_{\{y_H\}}] \leq & \mathbb{E}\left[\sum_{i\in[n_0]} K^{(i)}_{\{y_H\}}\right] \\
\leq & n_0 O(L^2 M^\ell \cdot p_e) \\
\leq & n_0 O\left(\frac{(n_0 T)^2}{T'^2} M^\ell \left(O\left(\frac{S^{2\ell}}{M^\ell}\right) + O\left(\left(\frac{(ST')^{\ell+1}}{N}\right)^S\right)\right)\right) \\
= & \widetilde{O}\left(\frac{T^2}{(cS^{-1}N^{\frac{1}{\ell+1}})^2} M^\ell \left(\frac{S^{2\ell}}{M^\ell} + (c_3 \cdot c)^{(\ell+1)S}\right)\right) \\
= & \widetilde{O}\left(S^{2\ell+2} T^2 N^{-\frac{2}{\ell+1}}\right).
\end{aligned}$$

The second and third equation comes from the choice of c. Since $M = \mathsf{poly}(N)$, $S = \Omega(\log N)$, we choose $c_1, c_2 > 0$ such that $M \leq N^{c_1}$, $S \geq c_2 \log N$. Also choose c_3 larger than the constant in big-O notation in Theorem 8.11. Then, we pick $c = 1/c_3 \cdot 2^{-c_1/c_2}$. Therefore, we have $(c_3 \cdot c)^{(\ell+1)S} \leq \frac{1}{M^\ell}$. □

10 The time-space bound for solving ℓ-Nested Collision Finding

10.1 The classical bound

Theorem 10.1. *For $\ell \geq 2$, any classical algorithm that solves the ℓ-Nested Collision Finding problem with constant probability with space $S = \Omega(\log N) = O\left(N^{\frac{1}{\ell^2-1}}\right)$ and T oracle queries must satisfy $S^{\frac{\ell^2-1}{\ell}} T = \Omega\left(N^{\frac{1}{\ell}} N_0^{\frac{1}{2}}\right)$.*

Proof. Let K to be the expectation of the number of ℓ-tuples with the same sum one can find in all its queried tuples on G after the algorithm finishes. Then it must satisfies $TK = \Omega(N_0)$. Thus constraints are:

$$\begin{cases} K = O\left(\dfrac{S^{\frac{\ell^2-1}{\ell}} T}{N^{\frac{1}{\ell}}}\right) \\ TK = \Omega(N_0) \end{cases}$$

Combine two equations we have $S^{\frac{\ell^2-1}{2\ell}} T = \Omega\left(N^{\frac{1}{2\ell}} N_0^{\frac{1}{2}}\right)$.

□

Theorem 10.2. *For any $\ell \geq 2$ and any $N_0 = \Theta(N^\epsilon)$ where $\frac{1}{\ell} < \epsilon \leq \frac{2}{\ell-1}$ the corresponding ℓ-Nested Collision Finding Problem has a space-time tradeoff for classical algorithms.*

Proof. By Theorem 10.1, any classical algorithm that solves the ℓ-Nested Collision Finding Problem when $\ell \geq 2$ and $N_0 = \Theta(N^\epsilon)$ with constant probability using S bit space and T oracle queries must satisfy $S^{\frac{\ell^2-1}{2\ell}} T = \Omega\left(N^{\frac{1}{2\ell}+\frac{\epsilon}{2}}\right)$. This implies that with limited space, say $S = O(\log N)$, any classical algorithm that succeeds with constant probability should use $T = \Omega\left(N^{\frac{1}{2\ell}+\frac{\epsilon}{2}}\right)$ oracle queries. But with enough space, the classical algorithm described in Theorem 6.4 solves this problem when

$N_0 = O\left(N^{\frac{2}{\ell-1}}\right)$ with constant probability using $T = O\left(N^{\frac{1}{\ell}+\frac{\varepsilon}{2\ell}}\right)$ oracle queries which is strictly better than the bound when $\ell \geq 2$. □

10.2 The quantum bound

Theorem 10.3. *For $\ell \geq 2$, any quantum algorithm that solves the ℓ-Nested Collision Finding problem with constant probability with space $S = \Omega(\log N) = O\left(N^{\frac{1}{(\ell+1)(2\ell+1)}}\right)$ and T oracle queries must satisfy $S^{\frac{\ell+1}{2}}T = \Omega\left(N^{\frac{1}{2(\ell+1)}}N_0^{\frac{1}{4}}\right)$.*

Proof. By the definition of ℓ-Nested Collision Finding Problem, the algorithm is given a target y sampled from an arbitrary distribution $\mathcal{D}$. We will prove that for any fixed y, the algorithm solving problem will have $S^{\frac{\ell+1}{2}}T = \widetilde{\Omega}\left(N^{\frac{1}{2(\ell+1)}}N_0^{\frac{1}{4}}\right)$. Then, for any distribution $\mathcal{D}$, if an algorithm $\mathcal{A}$ solves the ℓ-Nested Collision Finding Problem with respect to $\mathcal{D}$, take y^* from the support of $\mathcal{D}$ such that $\mathcal{A}$ has largest winning probability with y^* as target, then $\mathcal{A}$ also solves the ℓ-Nested Collision Finding Problem for y^*, and therefore also satisfy the bound $S^{\frac{\ell+1}{2}}T = \widetilde{\Omega}\left(N^{\frac{1}{2(\ell+1)}}N_0^{\frac{1}{4}}\right)$.

We now consider the case when the target y is fixed, we fix $\{y_H\}$ such that $y_H = y$ for all H. We have shown that the algorithms having each of CPhO, HaO and CStO as the oracle query form are equivalent, therefore without loss of generality, we consider an arbitrary algorithm $\mathcal{A}$ interacting with random oracle G in HaO, and all registers $\mathbf{XUWRD}$ are therefore initialized as $|\Phi_0\rangle = \frac{1}{\sqrt{N^M 2^{M^\ell}}}\sum_{H,G}|0\rangle_{\mathbf{XUW}}|\widehat{G}\rangle_{\mathbf{R}}|H\rangle_{\mathbf{D}}$.

For $t = 1, \cdots, T$, we look at the state in all registers after t oracle queries, let $K^{(t)}$ be the expected sum-$\{y_H\}$ ℓ-tuple capacity of state $|\Phi_t\rangle$ with respect to our fixed $\{y_H\}$. Let $K = \frac{1}{T}\sum_{t=1}^{T}K^{(t)}$. By Theorem 9.5, algorithm up to time step t has $t \leq T$ queries, we have

$$K = \frac{1}{T}\sum_{t=1}^{T}\widetilde{O}\left(S^{2\ell+2}T^2N^{-\frac{2}{\ell+1}}\right) = \widetilde{O}\left(S^{2\ell+2}T^2N^{-\frac{2}{\ell+1}}\right). \tag{12}$$

We can also define the value of sum-$\{y_H\}$ ℓ-tuple capacity for any fixed H. Since we could view H as classically uniform-randomly sampled, then whenever we look at the state in $\mathbf{XUWRD}$ and project it to subspaces where $\mathbf{D}$ is $|H\rangle_{\mathbf{D}}$, their corresponding projected component all have the same amplitude. Also, if we denote $|\Phi_{i,H}\rangle_{\mathbf{XUWRD}}$ as the normalized state after projecting it to the subspace where $\mathbf{D}$ is $|H\rangle_{\mathbf{D}}$, $|\Phi_i\rangle = \frac{1}{\sqrt{N^M}}\sum_H|\Phi_{i,H}\rangle$. Therefore, define $K_H^{(t)}$ as the expected sum-$\{y_H\}$ ℓ-tuple capacity of $|\Phi_{i,H}\rangle$, and $K_H = \frac{1}{T}\sum_{t=1}^{T}K_H^{(t)}$, we have $K = \mathbb{E}_H[K_H]$.

Now we look at the probability of finding collisions in G with bounded expected same-sum ℓ-tuple capacity by applying Theorem 5.6. Notice that in the original setting of Theorem 5.6, G oracle is purified in register $\mathbf{D}$, initialized as $|\perp, \cdots, \perp\rangle_{\mathbf{D}}$, where the algorithm queries it using CPhO, $\mathbf{R}$ is used to output collisions, and there is no oracle H; for fixed label function, V is defined by looking at register $\mathbf{D}$ in compressed basis and count its non-$\perp$ entries among the designated label.

Here, we argue that this is equivalent to our current scenario under each fixed $H \in [N]^M$. Now output the collisions in some pre-determined positions of $\mathbf{XUW}$. We set the domain of G oracle to be $[M^\ell]$, G is purified in $\mathbf{R}$, initialized as $|0, \cdots, 0\rangle_{\mathbf{R}}$, where the algorithm queries it using HaO. Notice that HaO is essentially a CPhO under Hadamard basis of register $\mathbf{R}$, the current standard basis state $|0\rangle$ is $|\widehat{0}\rangle = |\perp\rangle$ under Hadamard basis, so it is also consistent. For our fixed H, we define its induced label function $\mathsf{Lbl}_H : [M^\ell] \to [N]$, where $\mathsf{Lbl}_H(x_1, \cdots, x_\ell) = \sum_j H(x_j) \mod N$, therefore ℓ-tuples with label y_H are ℓ-tuples with sum y_H. With fixed H, the state after t-th query becomes $|\Phi_{i,H}\rangle$, and K_H exactly characterizes the expected same-label capacity averaging over all timesteps from 1 to T.

Therefore, we could safely borrow Theorem 5.6. For any fixed H and label function Lbl_H as defined above, the probability of $\mathcal{A}$ outputting a pair of labeled-collision in G is exactly the probability of $\mathcal{A}$ outputting a pair of colliding ℓ-tuples in G of target sum y, we denote this probability as P_H, then

$$P_H = O\left(\frac{T^2 K_H}{N_0}\right). \tag{13}$$

Finally, the algorithm that solves the ℓ-Nested Collision Finding problem (defined in Definition 6.1) finds a pair of colliding ℓ-tuples in G of the target sum y with randomly sampled H, therefore $\mathbb{E}_H[P_H] = \Omega(1)$. Then we have

$$K = \mathop{\mathbb{E}}_H[K_H] = \Omega\left(T^{-2}N_0\right) \cdot \mathop{\mathbb{E}}_H[P_H] = \Omega\left(T^{-2}N_0\right). \tag{14}$$

Since the constant in the O-notation in Theorem 5.6 does not depend on H, we can take this expectation over H. Combine Equation (14) and Equation (12), we have $S^{\frac{\ell+1}{2}}T = \widetilde{\Omega}\left(N^{\frac{1}{2(\ell+1)}}N_0^{\frac{1}{4}}\right)$. □

Theorem 10.4. *For any $\ell \geq 4$ and any $N_0 = \Theta(N^\epsilon)$ where $\frac{4\ell+6}{2\ell^2-\ell-3} < \epsilon \leq \frac{3}{\ell-1}$ the corresponding ℓ-Nested Collision Finding Problem has a space-time tradeoff quantum algorithms.*

Proof. By Theorem 10.3, any quantum algorithm that solves the ℓ-Nested Collision Finding Problem when $\ell \geq 2$ and $N_0 = \Theta(N^\epsilon)$ with constant probability using S qubit space and T oracle queries must satisfy $S^{\frac{\ell+1}{2}}T = \widetilde{\Omega}\left(N^{\frac{1}{2(\ell+1)}+\frac{\varepsilon}{4}}\right)$. This implies that with limited space, say $S = O(\log N)$, any quantum algorithm that succeeds with constant probability should use $T = \widetilde{\Omega}\left(N^{\frac{1}{2(\ell+1)}+\frac{\varepsilon}{4}}\right)$ oracle queries. But with enough space, the quantum algorithm described in Theorem 6.5 solves this problem when $N_0 = O\left(N^{\frac{3}{\ell-1}}\right)$ with constant probability using $T = \Theta\left(N^{\frac{2+\varepsilon}{2\ell+1}}\right)$ oracle queries which is strictly better than the bound when $\epsilon > \frac{4\ell+6}{2\ell^2-\ell-3}$. But notice that only when $\ell \geq 4$ there exists ϵ such that $\frac{4\ell+6}{2\ell^2-\ell-3} < \epsilon \leq \frac{3}{\ell-1}$. □

References

[Aar21] Scott Aaronson. "Open problems related to quantum query complexity". In: *ACM Transactions on Quantum Computing* 2.4 (2021), pp. 1–9 (cit. on p. 3).

[BHL22] Jeremiah Blocki, Blake Holman, and Seunghoon Lee. "The parallel reversible pebbling game: Analyzing the post-quantum security of iMHFs". In: *Theory of Cryptography Conference*. Springer. 2022, pp. 52–79 (cit. on p. 3).

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. "Quantum Cryptanalysis of Hash and Claw-Free Functions". In: *LATIN: : Theoretical Informatics, Latin American Symposium*. 1998, pp. 163–169 (cit. on p. 3).

[BKW24] Paul Beame, Niels Kornerup, and Michael Whitmeyer. "Quantum Time-Space Tradeoffs for Matrix Problems". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*. 2024, pp. 596–607 (cit. on p. 3).

[Bor93] Allan Borodin. “Time space tradeoffs (getting closer to the barrier?)” In: *International Symposium on Algorithms and Computation*. Springer. 1993, pp. 209–220 (cit. on pp. 6, 10).

[CGLQ20] Kai-Min Chung, Siyao Guo, Qipeng Liu, and Luowen Qian. “Tight quantum time-space tradeoffs for function inversion”. In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2020, pp. 673–684 (cit. on pp. 3, 13, 16).

[CLL24] Sitan Chen, Jerry Li, and Allen Liu. “An optimal tradeoff between entanglement and copy complexity for state tomography”. In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*. 2024, pp. 1331–1342 (cit. on p. 3).

[Cob66] Alan Cobham. “The recognition problem for the set of perfect squares”. In: *7th Annual Symposium on Switching and Automata Theory (swat 1966)*. IEEE. 1966, pp. 78–87 (cit. on p. 3).

[Din20] Itai Dinur. “Tight time-space lower bounds for finding multiple collision pairs and their applications”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2020, pp. 405–434 (cit. on pp. 3, 6, 10).

[FLVV05] Lance Fortnow, Richard Lipton, Dieter Van Melkebeek, and Anastasios Viglas. “Time-space lower bounds for satisfiability”. In: *Journal of the ACM (JACM)* 52.6 (2005), pp. 835–865 (cit. on p. 3).

[Gao15] Jingliang Gao. “Quantum union bounds for sequential projective measurements”. In: *Physical Review A* 92.5 (2015), p. 052331 (cit. on p. 11).

[GLLZ21] Siyao Guo, Qian Li, Qipeng Liu, and Jiapeng Zhang. “Unifying Presampling via Concentration Bounds”. In: *Theory of Cryptography Conference*. Springer. 2021, pp. 177–208 (cit. on p. 12).

[Gro96] Lov K Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219 (cit. on p. 13).

[HM23] Yassine Hamoudi and Frédéric Magniez. “Quantum Time–Space Tradeoff for Finding Multiple Collision Pairs”. In: *ACM Transactions on Computation Theory* 15.1–2 (2023), pp. 1–22. DOI: 10.1145/3589986. URL: http://dx.doi.org/10.1145/3589986 (cit. on pp. 3, 6, 10–12, 17).

[Jen06] J. L. W. V. Jensen. “Sur les fonctions convexes et les inégalités entre les valeurs moyennes”. In: *Acta Mathematica* 30.none (1906), pp. 175–193. DOI: 10.1007/BF02418571. URL: https://doi.org/10.1007/BF02418571 (cit. on p. 13).

[KMW19] Samad Khabbazi Oskouei, Stefano Mancini, and Mark M Wilde. “Union bound for quantum information processing”. In: *Proceedings of the Royal Society A* 475.2221 (2019), p. 20180612 (cit. on p. 11).

[KŠW07] Hartmut Klauck, Robert Špalek, and Ronald de Wolf. “Quantum and classical strong direct product theorems and optimal time-space tradeoffs”. In: *SIAM Journal on Computing* 36.5 (2007), pp. 1472–1493 (cit. on pp. 6, 10).

[Liu22] Qipeng Liu. *Non-uniformity and Quantum Advice in the Quantum Random Oracle Model*. 2022. arXiv: 2210.06693 [quant-ph]. URL: https://arxiv.org/abs/2210.06693 (cit. on pp. 12, 38, 40).

[LMY25] Jiahui Liu, Saachi Mutreja, and Henry Yuen. "QMA vs QCMA and Pseudorandomness". In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*. 2025, pp. 1520–1531 (cit. on p. 5).

[LRZ23] Qipeng Liu, Ran Raz, and Wei Zhan. "Memory-sample lower bounds for learning with classical-quantum hybrid memory". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 1097–1110 (cit. on pp. 3, 5).

[LZ19] Qipeng Liu and Mark Zhandry. "On finding quantum multi-collisions". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2019, pp. 189–218 (cit. on p. 12).

[MW05] Chris Marriott and John Watrous. "Quantum arthur–merlin games". In: *computational complexity* 14.2 (2005), pp. 122–152 (cit. on pp. 38, 39).

[NABT14] Aran Nayebi, Scott Aaronson, Aleksandrs Belovs, and Luca Trevisan. "Quantum lower bound for inverting a permutation with advice". In: *CoRR* abs/1408.3193 (2014). arXiv: `1408.3193`. URL: `http://arxiv.org/abs/1408.3193` (cit. on p. 3).

[OV22] Ryan O'Donnell and Ramgopal Venkateswaran. "The quantum union bound made easy". In: *Symposium on Simplicity in Algorithms (SOSA)*. SIAM. 2022, pp. 314–320 (cit. on p. 11).

[Pol78] John M Pollard. "Monte Carlo methods for index computation". In: *Mathematics of computation* 32.143 (1978), pp. 918–924 (cit. on p. 3).

[Zha19] Mark Zhandry. "How to record quantum queries, and applications to quantum indifferentiability". In: *Annual International Cryptology Conference*. Springer. 2019, pp. 239–268 (cit. on pp. 9, 10, 12–15, 17).

A Proof for Lemma 8.2, Lemma 8.3 and Lemma 8.4

A.1 Proof for Lemma 8.2

Proof. Let $\{(|u_i\rangle, |v_i\rangle\}_{i\in[D]}$ be the Jordan basis of projectors $\Pi^{\mathcal{A},\{y_H\}}_{\mathsf{FlipTest}}$ and Π_{reset}. We can rewrite the initial state as:

$$|\psi_{\mathsf{st}}\rangle |0\rangle_{\mathbf{S}} |0\rangle_{\mathbf{B}} = \sum_{i\in[D]} c_i |v_i\rangle .$$

Then the final state is

$$\left(\Pi^{\mathcal{A},\{y_H\}}_{\mathsf{FlipTest}}\Pi_{\mathsf{reset}}\right)^k (|\psi_{\mathsf{st}}\rangle |0\rangle_{\mathbf{S}} |0\rangle_{\mathbf{B}}) = \sum_{i\in[D]} c_i p_i^{\frac{2k-1}{2}} |u_i\rangle$$

where $p_i = |\langle u_i|v_i\rangle|^2$. By Jensen's inequality Lemma 3.3, we have

$$\begin{aligned} P^{(k)}_{\mathsf{FlipTest}} &= \left|\left(\Pi^{\mathcal{A},\{y_H\}}_{\mathsf{FlipTest}}\Pi_{\mathsf{reset}}\right)^k (|\psi_{\mathsf{st}}\rangle |0\rangle_{\mathbf{S}} |0\rangle_{\mathbf{B}})\right|^2 \\ &= \sum_{i\in[D]} c_i^2 p_i^{2k-1} \\ &\geq \left(\sum_{i\in[D]} c_i^2 p_i\right)^{2k-1} \end{aligned}$$

$$= \left(P^{(1)}_{\mathsf{FlipTest}}\right)^{2k-1}.$$

□

A.2 Proof for Lemma 8.3

Proof. Replacement of the advice state to a maximally mixed state can only affect the success probability by $\frac{1}{2^S}$ since the advice is a S-qubit state.

$$\begin{aligned}
&\left(P^{\mathsf{uniform},(S)}_{\mathsf{FlipTest}}\right)^{\frac{1}{2S-1}}\\
\geq &\left(\frac{1}{2^S}P^{(S)}_{\mathsf{FlipTest}}\right)^{\frac{1}{2S-1}}\\
\geq &\frac{1}{2}\left(\frac{1}{2^S}P^{(S)}_{\mathsf{FlipTest}}\right)^{\frac{1}{2S-1}}\\
\geq &\frac{1}{2}P^{(1)}_{\mathsf{FlipTest}}.
\end{aligned}$$

□

A.3 Proof for Lemma 8.4

Proof. It is equivalent to prove for $k \geq 2$:

$$S_{k-1} := \frac{\sum_{i\in[D]} c_i^2 p_i^{2k-1}}{\sum_{i\in[D]} c_i^2 p_i^{2k-3}} \leq S_k := \frac{\sum_{i\in[D]} c_i^2 p_i^{2k+1}}{\sum_{i\in[D]} c_i^2 p_i^{2k-1}}.$$

Define $\alpha_i = \frac{c_i^2 p_i^{2k-3}}{\sum_{i\in[D]} c_i^2 p_i^{2k-3}}$, we have $S_k = \sum_{i\in[D]} \alpha_i p_i^2$. Let $\beta_i = \frac{\alpha_i p_i^2}{\mu}$ where $\mu = \sum_{i\in[D]} \alpha_i p_i^2$.

$$\begin{aligned}
\beta_i =&\frac{\alpha_i p_i^2}{\mu}\\
=&\frac{c_i^2 p_i^{2k-1}}{\sum_{i\in[D]} c_i^2 p_i^{2k-3}\cdot\mu}\\
=&\frac{c_i^2 p_i^{2k-1}}{\sum_{i\in[D]} c_i^2 p_i^{2k-3}\cdot\left(\sum_{i\in[D]} c_i^2 p_i^{2k-1}/\left(\sum_{i\in[D]} c_i^2 p_i^{2k-3}\right)\right)}\\
=&\frac{c_i^2 p_i^{2k-1}}{\sum_{i\in[D]} c_i^2 p_i^{2k-1}}.
\end{aligned}$$

Thus $S_k = \sum_i \beta_i p_i^2$. Now it we prove that $\sum_{i\in[D]} \beta_i p_i^2 \geq \sum_{i\in[D]} \alpha_i p_i^2$ instead, which is

$$\sum_{i\in[D]} \alpha_i p_i^4 \geq \left(\sum_{i\in[D]} \alpha_i p_i^2\right)^2 \tag{15}$$

when multiple μ on both sides. Let $\mathbf{P}$ be a random variable that takes value p_i^2 w.p. α_i. We have $\mathbb{E}[\mathbf{P}] = \sum_{i\in[D]} \alpha_i p_i^2$ and $\mathbb{E}[\mathbf{P}^2] = \sum_{i\in[D]} \alpha_i p_i^4$. Equation (15) follows from $\mathbf{Var}[\mathbf{P}] = \mathbb{E}[\mathbf{P}^2] - \mathbb{E}[\mathbf{P}]^2 \geq 0$. $\square$