Electronic Colloquium on Computational Complexity, Report No. 96 (2021)

Keep That Card in Mind: Card Guessing with Limited Memory[*]

Boaz Menuhin[†] Moni Naor[‡]

July 8, 2021

Abstract

A card guessing game is played between two players, Guesser and Dealer. At the beginning of the game, the Dealer holds a deck of n cards (labeled $1, ..., n$). For n turns, the Dealer draws a card from the deck, the Guesser guesses which card was drawn, and then the card is discarded from the deck. The Guesser receives a point for each correctly guessed card.

With perfect memory, a Guesser can keep track of all cards that were played so far and pick at random a card that has not appeared so far, yielding in expectation $\ln n$ correct guesses. With no memory, the best a Guesser can do will result in a single guess in expectation.

We consider the case of a memory bounded Guesser that has $m < n$ memory bits. We show that the performance of such a memory bounded Guesser depends much on the behavior of the Dealer. In more detail, we show that there is a gap between the static case, where the Dealer draws cards from a properly shuffled deck or a prearranged one, and the adaptive case, where the Dealer draws cards thoughtfully, in an adversarial manner. Specifically:

1. We show a Guesser with $O(\log^2 n)$ memory bits that scores a near optimal result against any static Dealer.
2. We show that no Guesser with m bits of memory can score better than $O(\sqrt{m})$ correct guesses, thus, no Guesser can score better than $\min\{\sqrt{m}, \ln n\}$, i.e., the above Guesser is optimal.
3. We show an efficient adaptive Dealer against which no Guesser with m memory bits can make more than $\ln m + 2 \ln \log n + O(1)$ correct guesses in expectation.

These results are (almost) tight, and we prove them using compression arguments that harness the guessing strategy for encoding.

[*]Research supported in part by grants from the Israel Science Foundation (no. 950/15 and 2686/20), by the Simons Foundation Collaboration on the Theory of Algorithmic Fairness and by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center

[†]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel. Email: boaz.menuhin@weizmann.ac.il.

[‡]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel. Incumbent of the Judith Kleeman Professorial Chair. Email: moni.naor@weizmann.ac.il.

ISSN 1433-8092

Contents

1 Introduction

"Those who cannot remember the past are condemned to repeat it"
—George Santayana, *The Life of Reason*, 1905 [30]

Even if you use randomness and cryptography!

There are n cards in a deck. In each turn, one player, called Dealer, selects a card and a second player, called Guesser, guesses which card was drawn. The selected cards cannot be drawn again. The quantity of interest is how many cards were guessed correctly.

A Guesser with perfect memory can guess a card that has not appeared yet. When there are i cards left, the probability of correctly guessing is $\frac{1}{i}$. By linearity of expectation, such a Guesser is expected to guess correctly about $\ln n$ times. On the other hand, at any point in time, a memoryless Guesser cannot guess with probability better than $1/n$, resulting in 1 correct guess on expectation[1]. We are interested in the case where the Guesser has $m < n$ bits of memory.

One can think about card guessing as a streaming problem where the algorithm predicts the next element in a stream, under the promise that all elements in the stream are unique. In the streaming model, a large sequence of elements is presented to an algorithm, usually one element at a time. The algorithm, which cannot store the entire input, keeps the needed information and outputs some function on the stream seen so far. As it may be impossible to output the exact value of the function without storing the entire input, it is a typical relaxation to consider an approximate value of the function as output. In this sense, prediction is a form of approximation[2].

Some streaming problems are solvable by a deterministic algorithm, while others require a randomized one (for a survey on streaming algorithms, see [25]). It is common to analyze the performance of an algorithm with respect to a worst-case stream that is chosen ahead of time and is fixed throughout the execution of the algorithm. We call such a stream *oblivious*, or *static*. A recent line of works (see Section 1.3) focuses on analyzing the performance of an algorithm as if an adversary looks at the algorithm's output in every turn and thoughtfully chooses the next element in order to make the algorithm to fail. An algorithm that performs well against such an adversary is called *adversarially robust*.

In this paper, we pinpoint the complexity of card guessing in different environments, especially with respect to the memory requirements of the Guesser.

1.1 Our Results

We study the case where the Guesser has bounded memory, and we ask how well a Guesser can perform? It turns out that the performance of a memory bounded Guesser is highly sensitive to the behavior of the Dealer.

- A **static** Dealer draws cards one by one from a prearranged deck in some specific order.
- A Dealer is called **"random shuffle"** if every ordering of the deck has equal probability. This is equivalent to drawing a card at random in every turn.

[1]These examples are taken from a textbook on algorithms by Kleinberg and Tardos [21], Chapter 13, Pages 721-722.

[2]We elaborate on prediction as a form of approximation in Section 6.4.

- An **adaptive** Dealer is allowed to change the order of the deck throughout the game.

Our results are as follows:

1. Against the random-shuffle Dealer, there exists a Guesser with $\log^2 n + \log n$ memory bits that can perform similarly to a Guesser with perfect memory, i.e., make at least $1/2 \log n$ correct guesses in expectation. This Guesser can be amplified at the cost of log factor to get closer to $\ln n$.

2. Against any static Dealer (and as a result, also against a random-shuffle Dealer), there exists a Guesser with $\log^2 n - \log n + 2$ bits of memory and $2 \log n$ random bits that scores $1/4 \ln n - O(1)$ correct guesses in expectation. This Guesser can be amplified at the cost of log factor to get closer to $\ln n$.

3. The above Guessers are optimal: Every Guesser with m bits of memory can score at most $O(\sqrt{m})$ correct guesses in expectation against the random-shuffle Dealer, regardless of the amount of randomness and cryptographic tools and assumptions that the Guesser uses.

4. For every m there exists a computationally efficient adaptive Dealer against which no Guesser with m memory bits can make more than $\ln m + 2 \ln \log n + O(1)$ correct guesses in expectation, regardless of how much randomness and what cryptographic tools and assumptions that the Guesser uses.

5. Furthermore, there exists a computationally efficient adaptive *universal* Dealer, i.e., that makes no assumption on the amount of memory of the Guesser, against which every Guesser with m bits of memory is expected to make at most $(1 + o(1)) \cdot \ln m + 8 \ln \log n + O(1)$ correct guesses.

See Table 1 for a comparison of our results.

Table 1: Partial list of results: constructive results above the line, impossibility results below.

Guessing Technique	Memory	Randomness	Dealer	Score
Subset Guesser	m	-	Any	$\ln m$
Remember last cards	m	-	Any	$\ln \frac{m}{\log n}$
Subset+Remember last	$m \leq \sqrt{n}$	-	Random	$2 \ln m - \ln \log n - \ln 2$
Following Subsets	$O(\log^2 n)$	-	Random	$1/2 \log n$
Randomized Subsets	$O(\log^2 n)$	$2 \log n$	Static	$1/4 \ln n$
Any Guesser	m	∞	Random	$O(\min\{\ln n, \sqrt{m}\})$
Any Guesser	m	∞	Adaptive	$\ln m + 2 \ln \log n + O(1)$
Any Guesser	m	∞	Adaptive-universal	$(1 + o(1)) \cdot \ln m + 8 \ln \log n + O(1)$

To summarize, the main lesson from these results is the significant impact of adaptivity of the dealer, more than any other factor.

1.2 Our Approaches and Techniques

Our results separate the required amount of memory and randomness that the Guesser requires for playing against the three types of Dealers.

Quite surprisingly, we show that against the random-shuffle Dealer, a Guesser with very limited memory, and no randomness at all, can perform similarly to a Guesser with perfect memory. More formally, the main result of Section 4.1 is:

Theorem 1.2.1. *There exists a Guesser with $\log^2 n + \log n$ memory bits and no randomness that scores $1/2 \log n$ correct guesses in expectation when playing against the random-shuffle Dealer.*

Our Guesser tracks which cards were drawn from multiple subsets of cards. Using at most $2 \log n$ bits per subset, the Guesser can recover the last card that has not appeared from each subset and guess it. Repeating that guess over and over, the Guesser can guarantee a single correct guess from each subset.

However, the last cards from the different subsets may be indistinct. The subsets are built incrementally, i.e., the ith subset is contained in the $(i+1)$-th subset, as visualized in Figure 1. It follows that the probability for the last card from each subset to appear before the last card from the following subset is fixed (and is at least 1/2). So we get that the expected number of correct guesses is proportional to the number of subsets tracked by the Guesser and the ratio between two following subsets.

With more space we can have denser subsets, getting that for $0 < \delta \leq 1$ a Guesser with $\log_{1+\delta} 2 \cdot \log^2 n + \log n$ can score $\frac{\delta}{(1+\delta)\ln(1+\delta)} \ln n$ correct guesses in expectation, when playing against the random-shuffle Dealer. For $\delta = 1$ this is the above theorem. The number of correct guesses goes to $\ln n$ as δ goes to zero.

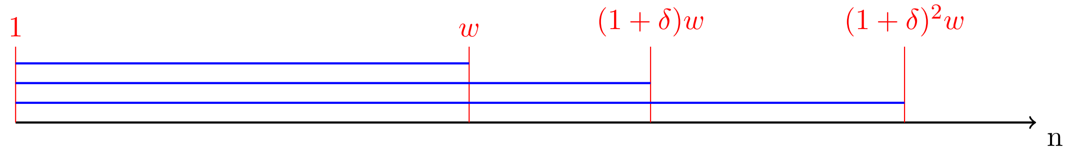

Figure 1: Following-Subsets. The blue lines represents subsets of $[n]$, the top blue line the subset $\{1, \ldots, w\}$, the one in the middle $\{1, \ldots, (1+\delta)w\}$ and so on. In the basic construction, $(1+\delta) = 2$.

Consider a static Dealer, one that fixes the sequence of cards ahead of time but may choose the worst arrangement. A simple adversarial argument shows that for every Guesser that uses no randomness, there exists an arrangement of the deck against which that deterministic Guesser scores at most 1 correct guess (guessing a single card is inevitable, even by a memoryless Guesser). In Section 4.2 we show that $2 \log n$ random bits suffice for a Guesser with $O(\log^2 n)$ bits of memory to score near perfect results when playing against any static Dealer.

Theorem 1.2.2. *There exists a Guesser with $\log^2 n - \log n + 2$ bits of memory and $2 \log n$ random bits that scores $\frac{1}{4} \ln n$ correct guesses in expectation when playing against any static Dealer.*

We use a pairwise independent permutation to split the cards to $\log n$ disjoint subsets of various sizes and track each subset similarly to the previous construction. We show that in each turn, the

Guesser recovers a correct guess from a certain subset in probability that is proportional to the number of cards left in the deck. Namely, when t cards are left in the deck, the probability of a correct guess is at least $1/4t$, resulting in $1/4 \cdot \ln n$ correct guesses in expectation.

In Section 4.2.1 we show that it is possible to amplify the above structure, and using $(\log n)$-wise independent functions for assigning cards to subsets, a Guesser can get closer to $\ln n$ correct guesses.

Theorem 1.2.3. *There exists a Guesser with $O(\log(1/\delta)\log^2 n)$ bits of memory and $O(\log(1/\delta)\log^2 n)$ bits of randomness that scores $(1-\delta)\ln n$ correct guesses on expectation against any static Dealer.*

Our deterministic and randomized Guessers are inspired by Garg and Schneider [14] and Feige's [12] algorithms for the first player in the Mirror Game in that they follow subsets of cards and track which member appeared. However, it turns out that there are fundamental differences between strategies for Mirror Game and card guessing against an adaptive dealer. We elaborate on the relation to the Mirror Game and to Feige's construction in Section 6.1.

In Section 4.3 we show that these Guessers are the best possible against the random-shuffle Dealer for $m \leq \log^2 n$. I.e., that there exists no Guessing technique that uses less memory and performs similarly.

Theorem 1.2.4. *Any Guesser with m memory bits can get* at most *$O(\min\{\ln n, \sqrt{m}\})$ correct guesses in expectation when playing against the random-shuffle Dealer.*

We show this by presenting an encoding scheme that utilizes correct guesses to encode an ordered set in an efficient manner. Using a compression argument we show that $\log^2 n$ bits of memory are actually essential for getting $O(\ln n)$ correct guesses.

Proof by Compression: This is a quite general method (see below) to prove the success of an algorithm by showing that some events allow us to compress the random bits used. Say a randomized algorithm can tolerate some number of bad events. For some specific domain (e.g. ordered sets of some size), we introduce an encoding scheme that utilizes the occurrence of certain bad events in order to achieve a shorter description of elements in the domain. We then consider the amount of bad events required to achieve a description that is shorter than the entropy of a random element in the domain. We know that for any compression method, the probability of chopping off (saving) w bits from a random string is at most 2^{-w}. We get that the probability for too many bad events is negligible.

The method has been applied in a variety of fields, for instance, to prove the success of the algorithm in the "Algorithmic Lovász Local Lemma" [24], the success probability of Cuckoo Hashing [28], lower bounds on construction of cryptographic primitives [15] and space-time trade-off for quantum algorithms [6].

To prove Theorem 1.2.4, we introduce an encoding scheme for *ordered sets* that utilizes correct guesses to achieve a shorter description of the ordered set. The idea is to simulate a game between a Guesser and a static Dealer, where the bottom of the deck is arranged according to the ordered set we wish to encode. If sufficiently many correct guesses occurred, then the encoding function stores the necessary information required to reproduce the course of the game.

Namely: the memory state of the Guesser, the set of turns at which the Guesser predicted correctly and the cards that the Dealer draws in the other turns in their respective order. By fixing the Guesser's randomness, every ordered set yields a single description by the encode function, and every description results in a single course of the game during decode. This allows the decode function to reconstruct the ordered set. A visual representation of the stored information is provided in Figure 2.

We get that we pay once for the memory state of the Guesser, and from that point on any correct guess shrinks the description of the ordered set. We then show that making too many correct guesses implies compression, i.e., a description of expected length shorter than the entropy of a random element. By doing so, we bound the expected number of correct guesses that any Guesser can make. The result holds regardless of how much randomness and what cryptographic tools and assumptions are used by the Guesser.

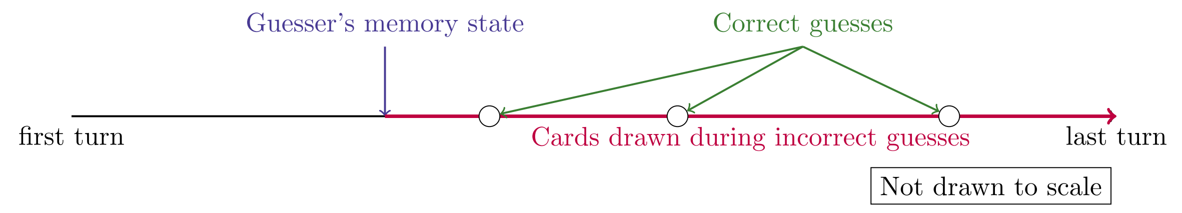

Figure 2: Correct guesses encoding. Colored - information stored by the encoding scheme.

In Section 5, we turn our attention to the adversarial adaptive Dealer. We show that if the Dealer is allowed to be adaptive, then almost any advantage gained from sophisticated memory usage vanishes. Furthermore, the adversarial dealer needs to know very little about the guesser. We show two results:

Theorem 1.2.5. *For every m there exists an efficient adaptive Dealer against which any Guesser with m bits of memory can score at most $\ln m + 2 \ln \log n + O(1)$ correct guesses in expectation.*

Theorem 1.2.6. *There exists a universal efficient adaptive Dealer against which any Guesser with m bits of memory can score at most $(1 + o(1)) \cdot \ln m + 8 \ln \log n + O(1)$ correct guesses in expectation.*

The two Dealers share the same general strategy that is parameterized differently. The Dealer's strategy is simply to refrain from drawing recently guessed cards for some turns. This result stands even if the Guesser is allowed to use unlimited randomness and cryptographic tools, while the Dealer is as simple as possible. The computational efficiency and simplicity of the Dealer, as well as the fact that the result stands even against an all-powerful Guesser with randomness and cryptography, emphasize that it is the *adaptivity that plays the key role*, rather than anything else. Recall that by Theorem 1.2.2, a mild amount of randomness suffices to achieve near optimal results against any static Dealer.

In more detail, our Dealer shuffles the deck at the beginning of the game and draws cards one by one. At some point, the Dealer begins to refrain from selecting cards guessed by the Guesser. When the Guesser guesses a card that resides in the deck, the Dealer takes that card and moves it to the back of the deck. Moving cards to the back reduces the number of cards that the Dealer may select, and as a result, makes the Dealer predictable. Therefore, at some point, the Dealer reshuffles the deck, making all cards available for drawing again, and repeats this behavior. Towards the end

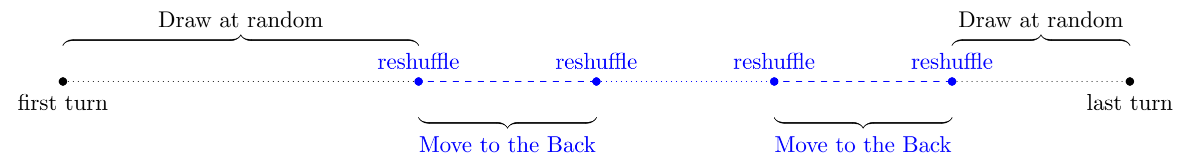

Figure 3: Adaptive Dealer general scheme.

of the game, the Dealer shuffles the deck one last time and draws cards at random. The period of turns between reshuffles is called an epoch, and the strategy is called MtBE-strategy (which stands for Move to the Back Epoch strategy).

Call a guess *reasonable* if it is a card that can be drawn, that is, a possibly correct guess. To show that moving cards to the back is an effective strategy, we show that no memory bounded Guesser is expected to make too many *reasonable* guesses during each epoch. The idea is that if the Guesser can produce many reasonable guesses, then she knows something about the set of cards in the deck and can be used to describe it efficiently. We show an encoding scheme for *sets*, and using compression argument, we bound the expected amount of reasonable guesses in every epoch.

Since our Dealer is adaptive, it is difficult to predict the order in which cards will be drawn, thus we do not use the same encoding scheme for *ordered* sets. Instead, we encode *unordered* sets using a similar encoding scheme.

As in Theorem 1.2.4, the encoding scheme works by simulating a game between the Guesser and the Dealer, where the Dealer keeps the set of cards we wish to encode for the end. If sufficiently many reasonable guesses occurred during an epoch, we use that epoch to achieve a shorter description of the set. In detail, we store the Guesser's memory state, the cards drawn by the Dealer while the Guesser guessed incorrectly, the set of turns where the Guesser guessed reasonably, and which of these guesses were actually correct. These objects allow us to simulate the same game during decoding and thus to recover the set. Visualization of the encoded information provided in Figure 4.

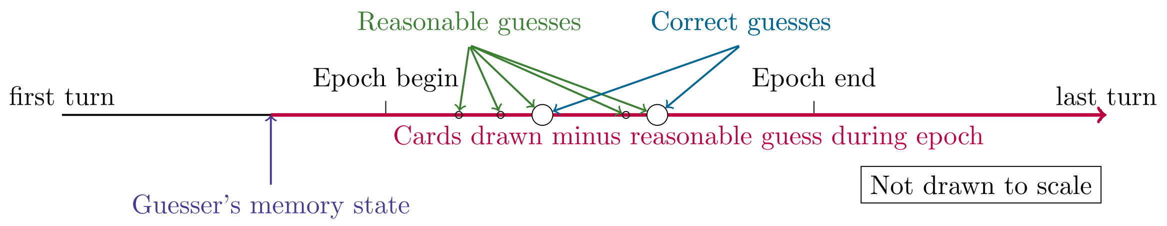

Figure 4: Reasonable guesses encoding. Colored - information stored by the encoding scheme.

Since reasonable guesses allow us to achieve shorter descriptions, we get a bound on the expected number of reasonable guesses during each epoch. By doing so, we bound the expected number of *correct* guesses in each epoch, since the cards are drawn from a large enough set. At this point, the analysis's calculations of the two Dealers vary and we analyze them differently.

Lastly, since a Guesser with m bits of memory can achieve $\ln m$ correct guesses in expectation, these results are almost tight.

1.3 Related Work

Card Guessing. An early work concerning card guessing dates back to 1924 [13], when Ronald Fisher studied the game in the context of analyzing claims of *psychic ability.* Fisher suggested and analyzed a method to measure and determine a Guesser's claim to have supernatural abilities (namely clairvoyance and telepathy) by assigning scores to the Guesser's guesses and see how much they deviates from the expectation. In 1981, Diaconis and Graham [9] studied the case where there are c_i copies of the ith card, and determined the optimal and worst strategies for some cases. They considered the cases of no feedback at all, partial feedback (was the guess correct or not) for $c_i = 1$, and full feedback (which card was drawn). Recently, Diaconis, Graham, He and Spiro [10] determined an optimal strategy for card guessing with partial feedback for $c_i \geq 2$.

A more useful yet equally dubious purpose is "Card Counting" in gambling (see Wikipedia entry [33]). In 1962 Edward Thorp, a Professor of Mathematics, published a bestseller book [32] about winning strategies in the game of Black-Jack. The book covered analyses of the game from the viewpoint of the player and the Casino, as well "low-memory" strategies that increase the player's expected benefit. The idea behind card counting in this context is that by knowing the distribution of the next card(s), one can evaluate their own hand better and thus bet accordingly. Card counting has been applied to other card games, such as Bridge and Texas hold 'em Poker. In these games, evaluating the probabilities for the upcoming cards is considered essential. In the context of this paper, card counting is an applied "low-memory" card guessing technique that utilizes the structure of specific games.

Mirror Games. Garg and Schneider [14] introduced the Mirror Game, a game closely related to card guessing. In Mirror Game, there are two players, Alice and Bob, taking turns in saying numbers from $[n]$. In every turn, a player says a number that was not said before by either player. If a player says a number that was already declared, that player loses, and the other player is the winner. If there are no more numbers to say, then it is a draw. Alice is first. Bob always has a simple deterministic winning strategy that requires only $\log n$ memory bits. When Alice says x, bob says $n + 1 - x$, and hence the name's origin.

Garg and Schneider showed that every deterministic winning (drawing) strategy for Alice requires $\Omega(n)$ bits of memory. They have also presented a randomized strategy for Alice with high drawing probability that requires $O(\sqrt{n})$. Their strategy relied os access to a random matching. Using the same settings, Feige [12] showed that $O(\log^3 n)$ bits of memory suffice. Our Guessers for the static case (Sections 4.1 and 4.2) are inspired by Feige's construction.

Additional discussion on the relation and differences between card guessing and Mirror Game is provided in Section 6.1.

Adversarial streams and sampling. A streaming algorithm is called *adversarially robust* if it's performance are guaranteed to hold even when the elements in the stream are chosen adaptively in an adversarial manner. The question concerning the gap in memory consumption between the static case and the adversarially adaptive case has been the subject of recent line of works.

On the positive side, Ben-Eliezer, Jayaram, Woodruff, and Yogev [4] showed general transformations for a family of tasks, for turning a streaming algorithm to be adversarially robust, with some overhead. Woodruff and Zhou [34] suggested another set of transformation. A different approach was taken by Hassidim, Kaplan, Mansour, Matias and Stemmer[17] who showed that it may be possible to get an even smaller overhead in some cases by using differential privacy as a protection against an adaptive adversary.

On the negative side, Hard and Woodruff [16] showed that linear sketches are inherently not adversarially robust. They showed it for the task of approximating L^P norms but their technique stands for other tasks as well. In a recent result, Kaplan, Mansour, Nissim and Stemmer [20] showed a problem that requires polylogarithmic amount of memory in the static case but *any* adversarially robust algorithm for it requires exponentially larger memory.

In a similar vein, given a large enough sample from some population, then we know that the measure of any fixed sub-population is well-estimated by its frequency in the sample. The size of the sample needed is the VC dimension of the set system of the sub-populations of interest. Ben-Eliezer and Yogev [3] showed that when sampling from a stream, if the sample is public and an adversary may choose the stream based on the samples so far, then the VC Dimension may not be enough. Alon, Ben-Eliezer, Dagan, Moran, Naor and Yogev [1] showed that the Littlestone dimension, which might be much larger than the VC dimension, captures the size of the sample needed in this case.

Online Computation and Competitive Analysis. Another area where the exact power of the adversary comes up is in competitive analysis of online algorithms (see Borodin and El-Yaniv [5]). Here there are various types of adversaries, distinguished by whether they are adaptive or static and whether they decide on the movements of the competing algorithm in an online manner or an offline one. The result is a hierarchy of oblivious, adaptive online and adaptive offline adversaries. It turns out that in request-answer games (a very general form capturing issues like paging), an algorithm competitive against the adaptive offline adversary may be transformed into a deterministic one with similar competitive ratio [2]. We do not see a similar phenomenon in our setting, where one should recall that a deterministic algorithm is hopeless against a static adversary.

Distinguishing Permutations and Functions. A stream of q random elements from the domain $[n]$ is given to a memory bounded algorithm that attempts to determine whether the stream was sampled with or without repetitions. When the stream ends, the algorithm outputs its determination, and is measured by it's ability to judge better than guessing at random. If $q = \Theta(\sqrt{n})$, then by the birthday paradox, a repetition occurs with high probability. The algorithm uses $O(q \log n)$ memory bits to recognize this repetition.

Motivated by the fact that the task of distinguishing between random permutations and random functions has significant cryptographic implications, Jaeger and Tessaro [19] introduced the above problem and showed a conditional bound on the advantage of the algorithm. In particular, they showed that under an unproved combinatorial conjecture the advantage of an algorithm with m bits of memory is bounded by $\sqrt{q \cdot m/n}$. Dinur [11] showed an unconditional upper bound on the advantage of $\log q \cdot q \cdot m/n$. This was followed by the work of Shahaf, Ordentlich and Segev [31] who achieved the unconditional upper bound on the advantage of $\sqrt{q \cdot m/n}$.

2 Preliminaries

Throughout this paper we use $[n]$ and $[1-n]$ to denote the set of integers $\{1, \ldots, n\}$. We denote the collection of subsets of $[n]$ of size exactly k by $\binom{[n]}{k} = \{B \subseteq [n] \colon |B| = k\}$. We denote the set of permutations on n elements by $\mathcal{S}_n$. All logs are base 2 unless explicitly stated otherwise, ln is the natural logarithm (base e). We denote the set of binary strings of length ℓ by $\{0,1\}^\ell$. We denote the set of binary strings of any finite length by $\{0,1\}^* = \cup_{i\in\mathbb{N}}\{0,1\}^i$.

2.1 Information Theory

Definition 2.1.1 (Entropy)**.** *Given a discrete random variable X that takes values from domain $\mathcal{X}$ with probability mass function $p(x) = \Pr[X = x]$. The Binary Entropy (abbreviated Entropy) of X, denoted $H(X)$ is*

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\cdot\log p(x).$$

Fact 2.1.2. *The entropy of a random variable X drawn uniformly at random from domain $\mathcal{X}$ is $\log|\mathcal{X}|$, i.e.*

$$H(X) = \log|\mathcal{X}|.$$

Definition 2.1.3 (Prefix-free code)**.** *A set of code-words $C \subseteq \{0,1\}^*$ is prefix-free if no code-word $c \in C$ is a prefix of another code-word $c' \in C$.*

Proposition 2.1.4 (Theorem 5.3.1 in [7])**.** *The expected length L of any prefix-free binary code for a random variable X is greater than or equal to the binary entropy $H(X)$.*

Lemma 2.1.5. *Given a random variable X uniformly drawn from domain $\mathcal{X}$. For every encoding function* Encode *for X, The probability that the encoding of X is d bits less than it's entropy is at most 2^{-d}, i.e.*

$$\Pr_{X\in\mathcal{X}}[|\text{Encode}(X)| = H(X) - d] \leq 2^{-d}.$$

Proof. There are $2^{H(X)-d}$ possible descriptions of length $H(X) - d$. Thus, there are at most that many values in $\mathcal{X}$ for which a description of such length is produced. Therefor, the probability to draw one of them is at most $\frac{2^{H(X)-d}}{|\mathcal{X}|}$. The entropy of random variable X uniformly drawn from $\mathcal{X}$ is $\log(|\mathcal{X}|)$. Therefore

$$\Pr[|\text{Encode}(X)| = H(X) - d] \leq \frac{2^{H(X)-d}}{|\mathcal{X}|} = \frac{2^{\log|\mathcal{X}|-d}}{|\mathcal{X}|} = \frac{|\mathcal{X}|\cdot 2^{-d}}{|\mathcal{X}|} = 2^{-d}.$$

□

2.2 Families of k-wise Independent Functions

Definition 2.2.1. *A family of functions $\mathcal{H}_k = \{h : [n] \to [n]\}$ is called k-wise independent, if for any sequence of k different elements $x_1, \ldots, x_k \in [n]$ and every $y_1, \ldots, y_k \in [n]$ it holds that*

$$\Pr_{h\sim\mathcal{H}_k}[(h(x_1), h(x_2), \ldots, h(x_k)) = (y_1, y_2, \ldots, y_k)] = n^{-k}.$$

There exists k-wise independent families of size $O(n^k)$. In fact, the family of polynomials of degree $k-1$ over a finite field is a k-wise independent family. This family is easy to work with as we can take $k\log n$ bits of randomness and refer to them as the coefficients of the polynomial to get a random member of the family. Such k-wise independent functions are useful for when we want a function that acts as a random function on small sets. But in certain aspects they behave as random functions even for bigger sets.

Let $\mathcal{F}_{\text{rand}}$ be the set of all functions $f : [n] \to [n]$. Given a subset $B \subseteq [n]$ and a specific subset $S \subset [n]$, consider a random function $f \sim \mathcal{F}_{\text{rand}}$. The expected size of the intersection between the image of B under f and S is $\frac{|S|}{n}|B|$. Consider the event that the intersection is empty, i.e. $f(B) \cap S = \emptyset$. Compare the probability for the event when the function is chosen from $\mathcal{F}_{\text{rand}}$ vs. when the function is chosen from $\mathcal{H}_k$. Indyk [18] showed that if $\frac{|S|}{n}|B|$ is sufficiently smaller than k, then the two probabilities are not far apart.

Proposition 2.2.2 (Lemma 2.1 in [18])**.** *For every subsets $B \in \binom{[n]}{t}$, $S \in \binom{[n]}{j}$, and every family of k-independent functions $\mathcal{H}_k$, if $\frac{(j-1)t}{n} \leq \frac{k-1}{2e}$ then*

$$\left| \Pr_{h \sim \mathcal{H}_k}[h(B) \cap S = \emptyset] - \Pr_{f \sim \mathcal{F}_{\text{rand}}}[f(B) \cap S = \emptyset] \right| \leq 2^{-k+1}.$$

3 Introduction to the Card Guessing Game

A card guessing game is played between a Dealer and a Guesser. At the beginning of the game, the Dealer holds a deck of n distinct cards (labeled $1, \dots, n$). In every turn, the Dealer chooses a card from the deck, draws it, and place it face-down. The Guesser guesses which card was drawn, the card is then revealed and discarded from the Dealer's deck. The Guesser gets a point for every correct guess. The game continues, for n turns, until the Dealer has no cards to draw.

Assume that the Dealer draws cards uniformly at random from the deck. A Guesser with *perfect memory* can keep track of all cards played so far and guess cards that are still in the deck. In turn $n - t$ there are t cards in the Dealer's deck and the Guesser's probability to guess the next card correctly is $1/t$. Hence, the expected number of correct guesses in the game is

$$\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1 = H_n \approx \ln n.$$

On the other extreme, a Guesser with *no memory at all* can guess the same card over and over again without knowing whether this card was picked already or not. Such a behavior would result in 1 correct guess with probability 1.

Question. *How well can a Guesser with m memory bits play?*

A **transcript** of a card guessing game is a sequence of pairs $\{(g_t, d_t)\}_{t=1}^{n}$ that describes that at turn t the Guesser guessed the card g_t and that the Dealer drew the card d_t. The number of correct guesses during a game is the number of turns during which $g_t = d_t$. In our game, the Guesser aims to maximize the number of correct guesses.

Guesser: A Guesser consists of two probabilistic functions:

1. **State transition function** taking a memory state and a card drawn, and assigns a new memory state.
2. **Guessing function** receiving a memory state and outputs a card to guess.

A memory bounded Guesser can use only m bits to store the memory state, so we refer to each state as $\{0,1\}^m$.

Randomness: We assume that the Guesser has random bits that both of the above functions may use, e.g. to select a random element of a set as a guess. We differentiate between random bits that are used *on the fly*, i.e. "read once", and random bits that are accessed several times, i.e. *long lasting.* We charge the Guesser for the latter but not for the former. The Guesser may use her long lasting random bits for a secret permutation, seed for a pseudo-random generator, and any random object that may assist her.

We are not concerned with the number of on the fly random bits. For our constructive results, we measure the amount of long lasting random bits that the Guesser uses, as well as suggest computationally efficient solutions good against computationally powerful dealers. As for the impossibility results, we will show that they hold even if the Guesser is computationally unbounded, uses cryptography, and regardless of how much randomness, of both kinds, the Guesser uses.

Static vs. Adaptive Dealers: To show that the Guesser's performance vary with the Dealer's abilities, we present the different flavors of Dealers we consider:

- The most benign Dealer we consider is a Dealer that shuffles the deck at the beginning of the game and draws cards one by one. This is equivalent to drawing cards at random from the deck in every turn. We call this Dealer **random shuffle** as it remains with the same shuffle throughout the game, and the deck is shuffled uniformly at random.

- The second Dealer we consider may be familiar with the Guesser's behavior and fixes the deck in advance in some particular order. As the deck of the Dealer remains the same throughout the game, we call this Dealer **static**.

- The third possibility we consider is a Dealer that is adaptive and selects the cards according to past guesses made by the Guesser, thoughtfully, in an adversarial manner. For our impossibility result, we do not assume that the Dealer is familiar with the Guesser's algorithm. We call such a Dealer **adversarial adaptive**, or just **adaptive** for short. If is not even aware of the memory size of the he Guesser, we call it **adaptive universal**.

The static Dealer and the adaptive Dealer aims to minimize the number of correct guesses. For this purpose, the static Dealer fixes the deck in advance, and the adaptive Dealer uses a **choosing strategy**, a function from a transcript prefix $\{(g_t, d_t)\}_{t=1}^{k-1}$ to a distribution over a set of cards from which the Dealer samples a card d_k to draw. For example, a (silly) adaptive Dealer may look at the last guess and if the guessed card is still in the deck then draw it.

While the Guesser is limited to using m bits of memory, the Dealer remembers everything that happened since the beginning of the game and may act accordingly. This puts the Guesser at a disadvantage, as the Guesser needs to remember and maintain both a sketch of the history and in particular the parts of history that are relevant to the Dealer's strategy. In light of this, we observe that only some guesses may be fruitful. Call a guess **reasonable** if it is one of the cards that the Dealer may draw. Clearly, a correct guess is necessarily reasonable. Against a Dealer that draws cards from the deck at random, a reasonable guess is a synonym for a card that was not played yet. As remarked above, when showing the impossibility results we construct a computationally efficient Dealer. This emphasizes the role of adaptivity, especially when compared to the Guesser.

The state of the Guesser consists of m bits and we assume that they are secret, i.e. that the adversary cannot access them when choosing the next card. The only inforamtion the Delaer has is the history (transcript).

3.1 Basic guessing techniques

In this section, we describe basic guessing techniques that a Guesser with m bits of memory may use. For a comparison of these techniques, see the first three rows of Table 1.

Subset Guessing: The Guesser chooses a random (or predetermined) subset of cards $A \in \binom{[n]}{m}$ and pretends as if there are only m cards in the deck. In every turn, the Guesser guesses one of the cards from A that were not played so far. Each card requires one bit, so this strategy requires m bits in total. Counting only turns in which the Dealer draw cards from A, we get that the Guesser makes $\ln m$ correct guesses in expectation over Guesser's and Dealer's randomness.

While this technique ensures that all guesses during the game are reasonable, only on m turns a card from A will be drawn. These m cards have to be drawn at some point in the game, and the Guesser is agnostic about when exactly these cards are selected. It follows that this technique performs equally well against the different kinds of Dealers.

"Remember" the last k cards: With only $\log n$ memory bits, the Guesser can correctly guess the last card in the game: Initialize memory with $\sum_{i=1}^{n} x \pmod{n}$ and remove every drawn card from the sum. Just before the last turn, the memory will contain the one card that was not drawn yet. This technique generalizes well to k cards by storing the sums $S_p = \sum_{x=1}^{n} x^p \pmod{n}$ for $p = 1, \ldots, k$ and removing d^p from the respective sums when the card d is drawn. When k cards are left, solving the equation system reveals the missing cards (See Chapter One in [25]). Since each sum requires $\log n$ bits, a total of $k \log n$ bits are required to accurately identify the last k cards. This allows the Guesser to reasonably guess in the last k turns, and by guessing at random, the Guesser makes $\ln k$ correct guesses in expectation using $k \cdot \log n$ bits of memory. Thus, a Guesser with m bits of memory can score $\ln\lfloor \frac{m}{\log n} \rfloor$ correct guesses in expectation, when playing against any Dealer.

As we saw, these techniques works well against *any* Dealer. The two methods (Remembering last cards and subset guessing) are compatible and we can combine them against the random-shuffle Dealer (but not against the others): of the m bits, use $m/2$ for the first method and $m/2$ for second one. The last card from the subset is expected when there are $\frac{n}{1+m/2} < \frac{2n}{m}$ cards left until the end of the game. The Guesser "remembers" the last $\frac{m}{2 \log n}$ cards, so for $m \leq \sqrt{n}$ the two useful periods do not overlap. We get that a Guesser with $m \leq \sqrt{n}$ memory bits can expect to score $2 \ln m - \ln \log n - \ln 2$. For $m = \sqrt{n}$, this is near optimal.

As we will see in Section 4, it is possible to do much better.

4 Static Dealer

We first present a guessing strategy that requires low memory and no randomness, and is highly effective against the random-shuffle Dealer Section 4.1. We then show a randomized version of it that requires low memory and little randomness, and is highly effective against any static Dealer (Section 4.2). In Section 4.3 we show that these guessing techniques are optimal against the random-shuffle Dealer, and that no memory bounded Guesser with less memory can perform asymptotically better.

4.1 Following-Subsets Guesser vs. Random shuffle Dealer

We present a computationally efficient guessing technique that requires low memory, no randomness, and is highly effective against the random-shuffle Dealer. We first show that $\log^2 n + \log n$ memory bits suffice to score $1/2 \log n$ correct guesses in expectation when playing against the random-shuffle Dealer, and then we generalize this technique for Guessers with more memory.

In terms of memory usage, we use the simple idea of summing cards as we did in the "Remembering last cards" guessing technique. The general idea is to follow the cards that appeared in various subsets of $[n]$. For each such subset we store two accumulators:

1. Sum of the values of the cards from the set seen so far ("remember last card").
2. Number of cards from the set seen so far.

The memory needed for the two accumulators is $O(\log n)$ bits. In fact, for a set of size w only $2 \log w$ bits are needed, $\log w$ to count how many cards from the set appeared, and another $\log w$ to recover the last card from the set, by storing the sum of all cards $\bmod w$. At the time that all but one card appeared (as can be indicated by the number of cards accumulator), the Guesser can recover this single card, and be certain that this card wasn't played yet by the Dealer, and as a result, the Guesser can reasonably guess this card.

By tracking multiple sets, the Guesser may have more than one card to guess from. Against the random-shuffle Dealer that plays with a randomly shuffled deck, this doesn't really matter which one is guessed (at least not for the expectation).

Subset construction: We consider all the subsets of the form $[1-w]$ for $w = 2^i$. I.e. the subsets are:

$$[1-2], [1-4], [1-8], [1-16], [1-32], \ldots, [1-n]$$

If there is a subset (range) where a single card is missing, then this card is the current guess.

Observe that in this construction there cannot be competing good cards to guess. For all $k < k'$, if a card j is missing from the set $[1-k]$, then there cannot be a different one missing from the set $[1-k']$[3].

Claim 4.1.1. *There exists a Guesser with $\log^2 n + \log n$ memory bits that can score $1/2 \log n$ correct guesses in expectation when playing against the random-shuffle Dealer.*

Proof. Call a subset $[1, w]$ **useful** if the last card from it that appears is *not* the last card in the next subset $[1, 2w]$. By guessing the last missing card from each subset over and over again, by the end of the game, each subset contributed a correct guess, but it could be that several subsets contributed the same guess. However, if a subset is useful, then it is the only one to whom we attribute the correct guess. So the number of correct guesses is simply the number of useful subsets.

The probability that a subset $[1, w]$ is useful is precisely the probability that in the 'next' subset $[1, 2w]$, the last card does not come from $[1, w]$. This is $(2w - w)/2w = 1/2$. By linearity of expectation, the expected number of useful subsets is therefore $1/2 \log n$ and this is also the expected number of correct guesses.

[3]The Guesser may conclude more than one missing card in some cases. For example, if one card is missing from $[1-k]$ and exactly two cards are missing from $[1-k']$. We ignore this ability because it doesn't seem to improve the Guesser's performance.

As the subset $[1-w]$ requires $2\log w = 2\log 2^i = 2i$ bits, we get that with

$$2\sum_{i=1}^{\log n} \log 2^i = 2\sum_{i=1}^{\log n} i = \log n(1+\log n) = \log^2 n + \log n$$

memory bits, a Guesser can make $1/2\log n$ correct guesses in expectation. □

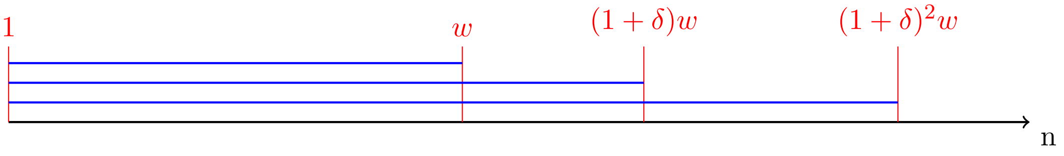

Figure 5: Following-Subsets.

Having more memory, we can have the subsets denser and have more subsets. Suppose that the ratio between two successive ranges is $1+\delta$ for $0<\delta<1$. Then there are $\log_{1+\delta} n$ such subsets. The probability of a set being useful now (i.e. that its last member arriving does not belong to a subset that contains it) is $\delta/(1+\delta)$. The expected number of useful sets is

$$\frac{\delta}{1+\delta}\log_{1+\delta} n = \frac{\delta}{(1+\delta)ln(1+\delta)}\ln n.$$

This goes to $\ln n$ as δ goes to zero.

In terms of space, the number of bits required for tracking $\log_{1+\delta} n$ buckets is

$$\begin{aligned}\sum_{i=1}^{\log_{1+\delta} n} 2i\log_2(1+\delta) &= 2\log_2(1+\delta)\sum_{i=1}^{\log_{1+\delta} n} i\\ &= 2\log_2(1+\delta)\frac{(\log_{1+\delta} n)(1+\log_{1+\delta} n)}{2}\\ &= \log_2(1+\delta)\frac{\log_2 n}{\log_2(1+\delta)}(1+\log_{1+\delta} n)\\ &= \log_2^2 n\cdot\log_{1+\delta} 2 + \log_2 n.\end{aligned}$$

Observe that the run time of the Guesser in every turn is at most $O(\log_{1+\delta} n)$, thus the Guesser is computationally efficient.

Corollary 4.1.2. *For $0<\delta\le 1$, there exists a Guesser with $\log^2 n\cdot\log_{1+\delta} 2+\log n$ memory bits that makes*

$$\frac{\delta}{(1+\delta)ln(1+\delta)}\ln n$$

correct guesses in expectation when playing against the random-shuffle Dealer.

4.2 Random-Subsets Guesser vs. Static Dealer

Consider a static Dealer such that instead of shuffling the deck uniformly at random, selects a worst case arrangement for the deck, knowing the Guesser's algorithm (but not her random bits). For example, assume that the Dealer puts the Card '1' at the top of the deck and the Card '2' at the bottom of the deck. In this case, the Following-Subsets technique yields a single correct guess. The fact that the Dealer doesn't shuffle the deck uniformly but commits to a deck arrangement as the game begins can be interpreted as a mild adversarial intent and ability.

The Guesser can defend herself against such behavior by using a secret permutation π, using her long lasting random bits. She uses π to randomize the subsets, where the subset $[1-w]$ tracks the cards $\pi(1), \ldots, \pi(w)$. The analysis and performance of the Following-Subsets technique holds as before, but $O(n \log n)$ bits of long lasting randomness are needed, which we wish to avoid.

We will show a related construction. The Guesser uses her randomness to sample a secret permutation from a family of pairwise independent permutations, for example, from the family $\mathcal{H}_{pair} = \{h(x) = ax + b \colon a \neq 0, b \geq 0\}$ over a finite field, and assigns the card x to the subset S_j if $2^{j-1} < h(x) \leq 2^j$. That is, given a function h, the subset S_j is the set of all $x \in [N]$ such that $h(x) \in \{2^{j-1}+1, \ldots, 2^j\}$.

The Guesser tracks the cards that appeared from each subset, as we did previously. In each turn, the Guesser attempts to recover a guess from a specific subset and guesses it. In detail, when $t \leq n/2$ cards are left until the end of the game, the Guesser tries to recover a guess from the subset S_j for $j = \log(n/2t)$. If all cards but one have appeared from S_j, then the Guesser knows which card it is and guesses it. For the first half of the game, the Guesser samples a random set of cards and guess cards that have not appeared from it. A procedural description of the Guesser is provided in Algorithm 1.

Algorithm 1 Randomized-Subsets

```
Sample a pairwise independent function $h \sim \mathcal{H}_{pair}$
Split the cards to subsets, such that $x \in S_j$ if $h(x) \in \{2^{j-1}+1, \ldots, 2^j\}$
Sample a set of cards $A$
while $t$ cards left for $t \in \{n, \ldots, n/2\}$ do
    Guess a random card from $A$ that has not appeared
    $d_t \leftarrow$ card drawn by Dealer
    Discard $d_t$ from the subset $S_j$ that contains it
while $t$ cards left for $t \in \{n/2+1, \ldots, 1\}$ do
    $j \leftarrow \lfloor \log(n/2t) \rfloor$                    ▷ Subset to consider
    if $|S_j| = 1$ then                             ▷ Can recover the last card
        $g_t \leftarrow$ last card in $S_j$
    else
        $g_t \leftarrow$ don't care
    Guess $g_t$
    $d_t \leftarrow$ card drawn by Dealer
    Discard $d_t$ from the subset $S_j$ that contains it
```

We will consider what are the chances that, in some turn, a specific subset yields the correct guess. That is, that the next card that the Dealer draws resides in a specific subset with a single missing card.

Theorem 4.2.1. *There exists a Guesser that uses* $\log^2 n - \log n + 2$ *memory bits and* $2\log n$ *random bits and is expected to score at least* $\frac{1}{4}\ln n$ *correct guesses in a game against any static Dealer.*

Proof. Consider a game between a Guesser that follows the Random-Subsets technique that plays against a static Dealer, i.e. one that selects an arbitrary sequence of cards to play.

For the first half of the game, the Guesser samples a random subset A and tracks it. Half of the cards in A are expected to appear in the first $n/2$ turns. By guessing cards from A at random, the probability for guessing the first card from A correctly is $\frac{1}{|A|}$, the probability to guess the second is $\frac{1}{|A|-1}$ and so forth. Resulting in roughly $\ln(|A|) - \ln(|A|/2) = \ln 2$ correct guesses in the first half of the game, with approximation error depending on the size of A. As A can be small and tracking it requires a small amount of memory that can be reused (consider storing it in the MSB of the subset counter), we neglect it from our calculations.

When $t < n/2$ cards are left, pick j such that $\lfloor \log n/2t \rfloor \le j \le \lceil \log n/2t \rceil$. Let $T_t = \{x_1, \dots, x_t\}$ be the ordered set of the remaining cards in the deck. What are the chances that the Guesser picks the card x_i? That is, what are the chances that $x_i \in S_j$ and all other cards from T_t, do not.

$$\begin{aligned}
\Pr[S_j = \{x_i\}] &= \Pr[x_i \in S_j \wedge \forall x' \in T_t \setminus \{x_i\} \colon x' \notin S_j] \\
&= \Pr[x_i \in S_j] \cdot \Pr[\forall x' \in T_t \setminus \{x_i\} \colon x' \notin S_j | x \in S_j] \\
&= \Pr[x_i \in S_j] \cdot \big(1 - \Pr[\exists x' \in T_t \setminus \{x_i\} \colon x' \in S_j | x \in S_j]\big) \\
&\ge \Pr[x_i \in S_j] \cdot \left(1 - \sum_{x' \in T_t \setminus \{x_i\}} \Pr[x' \in S_j | x \in S_j]\right).
\end{aligned}$$

Since the cards are assigned to S_j by using a pairwise independent permutation, the probability for a card x_i to be in S_j is $2^j/n$, and for the same reason $\Pr[x' \in S_j | x_i \in S_j] = \frac{2^j - 1}{n}$. It follows that the probability for x_i to be recovered by the jth subset is

$$\Pr[S_j = \{x_i\}] \ge \frac{2^j}{n} \cdot \left(1 - (t-1)\frac{2^j - 1}{n}\right) \ge \frac{1}{4t} \tag{4.1}$$

Where Inequality (4.1) is true since $j \approx \log \frac{n}{2t}$.

Since this is true for every x_i, in particular it is true for x_1. So we get that the probability to guess correctly when t cards are left is at least $1/4t$. By linearity of expectation, we get that the expected number of correct guesses throughout the $n/2$ last turns is at least $1/4 \cdot \ln(n/2)$.

By assigning each card in S_j a value in $[|S_j|]$, we get that tracking $\log n$ subsets requires $\log^2 n - \log n + 2$ memory bits. For each card the Guesser performs two operations, assign to bucket, and remove from bucket, thus the Guesser is computationally efficient. As $2\log n$ random bits are required for the pairwise independent permutation, the theorem follows. □

4.2.1 Amplification Towards $\ln n$

The above construction is simple and works well to get $1/4 \ln n$. How can we improve it and get to $\ln n$? We modify the above Guesser slightly to get an amplifiable construction.

The idea is to sample more functions $h_1, h_2, \dots, h_{2.5\log(1/\delta)}$ and for each function h_i generate its collection of sets $\{S_j^{h_i}\}_{j=1}^{\log n}$. The algorithm is now: when $t \leq n/2$ cards are left, for $j = \log(n/2t)$, the Guesser attempts to recover a reasonable guess from the jth subset in each of the collections. The Guesser takes the first collection that yields a subset with a single missing card and makes this card her guess. For the first half of the game, as previously, the Guesser sample some set of cards and guess cards that have not appeared (Subset guessing). A procedural description of this Guesser is provided in Algorithm 2.

We show that in the construction above, every collection yields a reasonable guess with constant probability. Since the collections are chosen independently, the probability of failure to recover a reasonable guess goes down with the number of functions we sampled. If we sample independently $O(1/\delta)$ functions, we can get to probability $1 - \delta$ of at least one subset succeeding in suggesting a reasonable guess.

The issue with the analysis of this process is showing that a reasonable guess is indeed correct with probability proportional to the number of cards left in the deck (and not some constant fraction of that). Since the adversary chooses the order of the cards, we need a construction of functions where we can say that if a subset yields a reasonable guess then it is also correct with the right probability. We do not know whether this is true for pairwise independent functions. As we will see, it is true for higher independence. So our Guesser samples functions from a family $\mathcal{H}_k$ of k-wise independent functions (Definition 2.2.1), and assigns cards to subsets in the same way.

Algorithm 2 Amplifed Random-Subsets

```
Parameter: $\delta \leq 1$, $k \geq 2\log n$
  Sample $h_1, h_2, \dots h_{2.5\log(1/\delta)}$ functions from a family of $k$-wise independent functions $\mathcal{H}_k$
  for $h \in H$ do
      Construct a collection of subsets $S_1^h, \dots, S_{\log n}^h$ such that $x \in S_j^h$ if $h(x) \in \{2^{j-1}+1, \dots, 2^j\}$
  Sample a set of cards $A$.
  while $t$ cards left for $t \in \{n, \dots, n/2\}$ do
      Guess a random card from $A$ that has not appeared
      $d_t \leftarrow$ card drawn by Dealer
      Discard $d_t$ from the subsets $S_j^h$ that contains it
  while $t$ cards left for $t \in \{n/2+1, \dots, 1\}$ do
      $j \leftarrow \lfloor \log(n/2t) \rfloor$                                        ▷ Subsets to consider
      if there exists $h \in \{h_1, \dots, h_{2.5\log(1/\delta)}\}$ such that $|S_j^h| = 1$ then    ▷ Can recover last card
          Let $h$ be the first such subset for which $|S_j^h| = 1$
          $g_t \leftarrow$ last card from $S_j^h$
      else
          $g_t \leftarrow$ don't care
      Guess $g_t$
      $d_t \leftarrow$ card drawn by Dealer
      Discard $d_t$ from the subsets $S_j^h$ that contains it
```

During the first half of the game, the analysis is the same as in Theorem 4.2.1.

Recall that $T_t = \{x_1, \dots, x_t\}$ is the ordered set of the remaining cards in the deck when t cards are left. We show that for $j = \lfloor \log(n/2t) \rfloor$ the probability that the jth subset of any collection yields a reasonable guess, is at least a constant. Fix some h and recall Inequality (4.1) that states that for any i the probability that S_j^h is the singleton set that consists of x_i is at least $1/4t$. Therefore, the event that S_j^h yields a reasonable guess is the disjoint union of events where S_j^h is the singleton consisting of x_i for $i \in [t]$. That is, the probability that S_j^h yields a reasonable guess is at least

$$\Pr[S_j^h \text{ yield a reasonable guess}] \geq \sum_{i=1}^{t} \frac{1}{4t} = \frac{1}{4}. \tag{4.2}$$

Let the indicator random variable R_t be the event that some subset yields a reasonable guess when t cards are left. The probability that no jth subset, of any collection, yields a reasonable guess when t cards are left is at most

$$\begin{aligned}\Pr[R_t = 0] &\leq (1 - 1/4)^{2.5\log(1/\delta)} \\ &< (3/4)^{\frac{\log 1/\delta}{\log 4/3}} \\ &= (3/4)^{\log_{4/3} 1/\delta} \\ &= (3/4)^{\log_{3/4} \delta} \\ &= \delta.\end{aligned}$$

Corollary 4.2.2. *The probability that the subsets $S_j^{h_1}, \ldots, s_j^{h_{2.5\log(1/\delta)}}$ yields a reasonable guess at turn $t > n/2$ is at least $1 - \delta$.*

Given that S_j^h yields a reasonable guess, what are the chances for it to be correct? That is, what are the chances that S_j^h is the singleton that contains the next card?

Claim 4.2.3. *When $t < n/2$ cards are left, for $j = \lfloor \log(n/2t) \rfloor$, for $k \geq 2\log n$, the probability over the choice of $h \sim \mathcal{H}_k$, that S_j^h yields a correct guess given that S_j^h yields a reasonable guess is at least*

$$\frac{1}{t + o(1)}.$$

Proof. Recall that $x \in S_j^h$ if $h(x) \in \{2^{j-1}+1, \ldots, 2^j\}$. Denote by $V_j = \{2^{j-1}+1, \ldots, 2^j\}$ the possible images of elements from S_j^h. Let B_{-i} be the set $\{x_1, \ldots, x_t\} \setminus \{x_i\}$, and denote the image of B_{-i} under h by $h(B_{-i}) = \{h(x) \colon x \in B_{-i}\}$. Let A_{-i} be the event that $h(B_{-i}) \cap V_j = \emptyset$.

The subset S_j^h yields a reasonable guess if and only if it intersects with the remaining cards at precisely one point. The event of recovering a reasonable guess is a disjoint union of the events that x_i is in the subset S_j^h and all other cards are not in S_j^h, that is, $h(B_{-i}) \cap V_j = \emptyset$, or in other words A_{-i}. Given that S_j^h yields a reasonable guess, then exactly one of the events A_{-i} occurred. Let P_i be the probability of event A_{-i} given $x_i \in S_j^h$. I.e.

$$\begin{aligned}\Pr[S_j^h \text{ yields a reasonable guess}] &= \sum_{i \in [t]} \Pr[A_{-i} \wedge x_i \in S_j^h] \\ &= \sum_{i \in [t]} \Pr[A_{-i} | x_i \in S_j^h] \cdot \Pr[x_i \in S_j^h] \\ &= \sum_{i \in [t]} P_i \cdot \Pr[x_i \in S_j^h]\end{aligned}$$

Note that given $x_i \in S_j^h$, we have $(k-1)$-wise independence of the values

$$h(x_1), h(x_2), \ldots, h(x_{i-1}), h(x_{i+1}), \ldots, h(x_t).$$

recall that $\mathcal{F}_{\text{rand}}$ is the set of all functions $\{f \colon [n] \to [n]\}$. Proposition 2.2.2 states that for every set B_{-i} of size $t-1$, the probability for the event A_{-i}, over the choice of a random function from $\mathcal{F}_{\text{rand}}$, is roughly the same as over the choice of a random function from $\mathcal{H}_k$. Applying Proposition 2.2.2 on our parameters we get that if $\frac{2^j+1}{n}(t-1) \leq \frac{k-2}{2e}$ then

$$|\Pr_{h \sim \mathcal{H}_k}[A_{-i} | x_i \in S_j^h] - \Pr_{f \sim \mathcal{F}_{\text{rand}}}[A_{-i} | x_i \in S_j^h]| \leq 2^{-k+1}.$$

In our case $j \approx \log \frac{n}{2t}$, thus $\frac{2^j+1}{n}(t-1) < 1 \leq \frac{k-2}{2e}$ and since $k \geq 2\log n$, the proposition applies.

The conditional probability for the event A_{-i} over the choice of $f \sim \mathcal{F}_{\text{rand}}$ depends solely on the size of the set B_{-i}. Since $|B_{-i}|$ is the same for every i, we get that probability over the choice of $h \sim \mathcal{H}_k$ for the event A_{-i} is about the same for every i. We conclude the distance between the probability for A_{-i} and $A_{-i'}$, that is

$$|\Pr_{h \sim \mathcal{H}_k}[A_{-i} | x_i \in S_j^h] - \Pr_{h \sim \mathcal{H}_k}[A_{-i'} | x_{i'} \in S_j^h]| \leq 2^{-k+2}. \tag{4.3}$$

Observe that for $f \sim \mathcal{F}_{\text{rand}}$ the event A_{-i} is independent from the event $x_i \in S_j^h$, and as a result

$$\begin{aligned}\Pr_{f \sim \mathcal{F}_{\text{rand}}}[A_{-i} | x_i \in S_j^h] &= \Pr_{f \sim \mathcal{F}_{\text{rand}}}[A_{-i}] \\ &= \Pr_{f \sim \mathcal{F}_{\text{rand}}}[f(B_{-i}) \cap V_j = \emptyset] \\ &= \Pr_{f \sim \mathcal{F}_{\text{rand}}}[\forall x \in B_{-i} \colon f(x) \notin \{2^{j-1}+1, \ldots, 2^j\}] \\ &= \left(1 - \frac{2^{j-1}}{n}\right)^{t-1}.\end{aligned}$$

Since $j \approx \log(n/2t)$, this is approximately some constant α which is roughly $e^{-0.5}$.

The next card in the deck is x_1. Given that S_j^h yields a reasonable guess, what is the probability for its guess to be correct? The conditional probability of guessing x_1 as the next card is

$$\begin{aligned}\frac{P_1 \cdot \Pr[x_1 \in S_j^h]}{\sum_{i=1}^t P_i \cdot \Pr[x_i \in S_j^h]} &= \frac{P_1}{\sum_{i=1}^t P_i} & (4.4)\\ &\geq \frac{P_1}{\sum_{i=1}^t (P_1 + 2^{-k+2})} & (4.5)\\ &= \frac{1}{t + t \cdot 2^{-k+2} \cdot P_1^{-1}}\end{aligned}$$

Where Equality (4.4) is true since cards are assigned to buckets using k-wise independent function, thus for every x_i the probability that $h(x_i) \in V_j$ is the same, and Inequality (4.5) follows from Inequality (4.3).

For $k \geq 2\log n$, and for $t < n/2$, the term $t \cdot 2^{-k+2} \cdot P_1^{-1} \approx t \cdot e^{1/2}/2^{k-2} = o(1)$. We conclude

$$\Pr[S_j^h \text{ yields a correct guess} | S_j^h \text{ yields a reasonable guess}] \geq \frac{1}{t + o(1)}.$$

□

The functions $\{h_1, \ldots, h_{2.5\log(1/\delta)}\}$ are chosen independently of each other, and hence so are the jth subsets. As a result, since the claim holds for an arbitrary S_j^h, it implies that for the first subset that yields a reasonable guess, the probability that its guess is correct is at least $\frac{1}{t+o(1)}$.

Recall the indicator random variable R_t that corresponds to the event that some subset yields a reasonable guess when t cards are left. Let the indicator random variable C_t be the event of a correct guess when t cards are left. We conclude that the expected number of correct guesses since turn $n/2$ is at least

$$\begin{aligned}\sum_{t=n/2-1}^{1} \mathbb{E}[C_t] &= \sum_{t=n/2-1}^{1} \Pr[R_t = 1] \cdot \mathbb{E}[C_t | R_t = 1] + \Pr[R_t = 0] \cdot \mathbb{E}[C_t | R_t = 0] \qquad (4.6)\\ &= \sum_{t=n/2-1}^{1} \Pr[R_t = 1] \cdot \mathbb{E}[C_t | R_t = 1] \qquad (4.7)\\ &> \sum_{t=n/2-1}^{1} (1-\delta) \cdot \frac{1}{t+o(1)} \qquad (4.8)\\ &\approx (1-\delta)\ln(n/2).\end{aligned}$$

Where Equality (4.6) is from law of total expectation, Equality (4.7) is true since a correct guess is necessarily reasonable, Inequality (4.8) follows from Claim (4.2.3) and from Corollary 4.2.2.

In term of memory, we allocate $2\log n$ bits per subset, and a total $2\log^2 n$ per collection of subsets. To track $2.5\log(1/\delta)$ collections a Guesser requires $O(\log(1/\delta)\log^2 n)$ bits of memory.

Sampling a function from a $(2\log n)$-wise independent family of functions requires $2\log n$ bits of long lasting randomness, e.g., consider the randomness as representing coefficients of a polynomial over a finite field. To sample $2.5\log(1/\delta)$ such functions, the Guesser requires $O(\log(1/\delta)\log^2 n)$ bits of randomness. Furthermore, the Guesser is computationally efficient: In each turn $O(1/\delta)$ functions are computed and this many subsets are updated.

Theorem 4.2.4. *There exists a Guesser with $O(\log(1/\delta)\log^2 n)$ bits of memory and $O(\log(1/\delta)\log^2 n)$ bits of randomness that scores $(1-\delta)\ln n$ correct guesses on expectation against any static Dealer.*

4.2.2 Low-memory Case

The guessing techniques seen so far assumed that the Guesser has about $\log^2 n$ bits of memory. But what can be done if m is small, say $m << log^2 n$? It is possible to fall back to the subset guessing technique and get $\ln m$ correct guesses in expectation. That would work for both the random shuffle and the static cases (also for the adaptive). But we can do better.

Our Guessers can pretend as if the domain is of size $2^{\sqrt{m}}$ and ignore all other cards! In that case, the Following-Subsets guessing technique is expected to yield about $1/2\log 2^{\sqrt{m}} = 1/2\sqrt{m}$ correct guesses against the random-shuffle Dealer. The Random-Subsets guessing technique is expected to yield about $\frac{1}{4}\ln 2^{\sqrt{m}}$ correct guesses against a static Dealer, i.e. also $O(\sqrt{m})$.

So we get that for any m, a Guesser can score at least $O(\min\{\ln n, \sqrt{m}\})$ when playing against any static Dealer.

4.3 Bounds on best possible Guesser against Random Dealer

We show that *the guessers of the previous section are the best possible low memory guessers*, up to constants.

Theorem 4.3.1. *Any Guesser using m bits of memory can get* at most $O(\min\{\ln n, \sqrt{m}\})$ *correct guesses in expectation when playing against the random-shuffle Dealer.*

Our proof will use compression argument. We will present an encoding scheme that utilizes correct guesses to achieve shorter descriptions. As the expected length of the description is bounded by the entropy of a random input, we get an upper bound on the expected number of correct guesses for every memory bounded Guesser. Our proof for an adaptive Dealer (Section 5) will follow a similar structure.

Let γ be the Guesser's randomness and π be the shuffle sampled by the Dealer's randomness. Denote by $\mathcal{G}_{(\gamma)}$ a Guesser with fixed randomness γ, and $\mathcal{D}_\pi$ a static Dealer with a deck arranged according to π. Let $\mathrm{val}(\mathcal{D}_\pi, \mathcal{G}_{(\gamma)})$ be the number of correct guesses during the last $k = n^{1-\beta}$ turns, for some $\beta > 0$. Let a random variable $C = \mathrm{val}(\mathcal{D}, \mathcal{G})$ and let c be the expected number of correct guesses during that last k turns where the expectation is taken over Guesser's and Dealer's randomness, i.e.

$$c = \mathop{\mathbb{E}}_{\gamma,\pi}[C] = \mathop{\mathbb{E}}_{\gamma,\pi}\left[\mathrm{val}(\mathcal{D}_\pi, \mathcal{G}_{(\gamma)})\right].$$

Denote by Π_B the set of all deck arrangements such that the last k cards in the deck are the ordered set B. So we can consider the expectation over the choice of the last k cards.

$$c = \mathop{\mathbb{E}}_{\gamma,\pi}[C] = \mathop{\mathbb{E}}_{\gamma,B}\mathop{\mathbb{E}}_{\pi\in\Pi_B}[C] = \sum_{\gamma}\mathop{\mathbb{E}}_{B}\mathop{\mathbb{E}}_{\pi\in\Pi_B}[C|\gamma]\cdot\Pr[\gamma].$$

In particular, we focus on bounding the term

$$\mathop{\mathbb{E}}_{B}\mathop{\mathbb{E}}_{\pi\in\Pi_B}[C|\gamma] = \mathop{\mathbb{E}}_{B}\mathop{\mathbb{E}}_{\pi\in\Pi_B}\left[\mathrm{val}(\mathcal{D}_\pi, \mathcal{G}_{(\gamma)})|\gamma\right].$$

We claim that no Guesser can expect to guess correctly too many times at the last k turns. We prove this by presenting an encoding scheme for ordered sets B (the last k cards played by the Dealer) that utilizes correct guesses to achieve shorter descriptions. The encode function works by simulating the Guesser on a deck of card, where the first $n-k$ cards are from $[n]\backslash B$ and the k cards are ordered according to B. Record the Guesser's memory (m bits) after the first $n-k$ turns and from that point on see when the Guesser gives correct guesses. These can be used to help describe B. Let the number of correct guesses be C. If $C \geq \alpha$ for some $\alpha > 0$, then to record B, we note the location of some α places with a correct guess and provide the remaining $k-\alpha$ missing values. So how many possibilities do we have? For the memory 2^m, for the correct guesses locations $\binom{k}{\alpha}$ and for the other values an ordered set of size $k-\alpha$ out of n.

Recall that Π_B is the set of all deck arrangements for which the last k cards are the ordered set B. The order of the first $n-k$ cards may lead the Guesser to different memory states; in terms of correct guesses, some of which may be more beneficial then others, especially for a Guesser with fixed randomness. Given an ordered set B and Guesser's randomness γ, let $\pi_{B,\gamma} \in \Pi_B$ be the deck arrangement for which the Guesser $\mathcal{G}_{(\gamma)}$ makes the most correct guesses in the last k turns. That is

$$\forall\pi \in \Pi_B\colon\ \mathrm{val}(\mathcal{D}_\pi, \mathcal{G}_{(\gamma)}) \leq \mathrm{val}(\mathcal{D}_{\pi_{B,\gamma}}, \mathcal{G}_{(\gamma)}).$$

The encoding function will simulate a game against a static Dealer with fixed deck order $\pi_{B,\gamma}$ to encode B. Fix some prefix free code (Definition 2.1.3) for ordered subsets. The scheme will use this code for the cases where there are not enough correct guesses to utilize.

Definition 4.3.2 (EncodeO$_{\gamma,\alpha}$)**.** *To encode B, an ordered subset of $[n]$ of size k, the function* EncodeO$_{\gamma,\alpha}$ *records and simulates a game between the Guesser $\mathcal{G}_{(\gamma)}$ and the static Dealer $\mathcal{D}_{\pi_{B,\gamma}}$.*

Let T' be the set of locations during the last k turns at which $\mathcal{G}_{(\gamma)}$ makes a correct guess, i.e.

$$T' = \{n - k_i \leq t \leq n - k_i + \ell - 1 | g_t \text{ is a correct guess}\}.$$

- *If $|T'| < \alpha$ then the code is made of an indicator bit 0 and an explicit prefix-free description of B.*
- *If $|T'| \geq \alpha$ then let T be the first α turns at which the Guesser guessed correctly.*

  The code is made of:

  1. *An indicator bit 1.*
  2. *Guesser's memory state M at turn $n - k$ (m bits).*
  3. *Description of T, the locations of the first α correct guesses made by $\mathcal{G}_{(\gamma)}$ during the last k turns ($\log \binom{k}{\alpha}$ bits).*
  4. *Description of $B \setminus \{g_t | t \in T\}$ ($\log (n(n-1)\dots(n-k+\alpha+1))$ bits).*

A visualization of the information stored is provided in Figure 6.

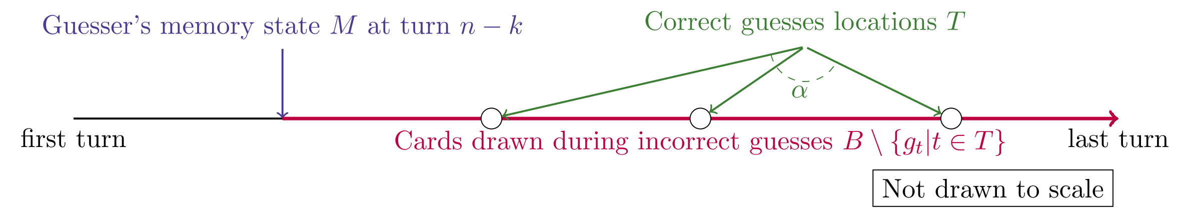
Guesser's memory state M at turn $n-k$
Correct guesses locations T
α
first turn
Cards drawn during incorrect guesses $B \setminus \{g_t | t \in T\}$
last turn
Not drawn to scale

Figure 6: EncodeO$_{\gamma,\alpha}$. Colored - information stored by the encoding scheme.

Similarly we define the decode function.

Definition 4.3.3 (DecodeO$_{\gamma,\alpha}$)**.** *If the indicator bit is 0, then decode the set in the natural way. If the indicator bit is 1, then parse the other bits as a 3-tuple (M, T, B_1) as encoded by* EncodeO$_{\gamma,\alpha}$*. The function* DecodeO$_{\gamma,\alpha}$ *works by simulating and recording a partial game between Dealer $\mathcal{D}^*_{B_1}$ and Guesser $\mathcal{G}_{(\gamma)}$:*

- *Initialize the Guesser $\mathcal{G}_{(\gamma)}$ with memory state M at turn $n - k$ and simulate k turns against the Dealer $\mathcal{D}^*_{B_1}$.*
- *If in turn $n - k \leq t \leq n$ the guess g_t is tagged as correct (by T), then $\mathcal{D}^*_{B_1}$ draws the card g_t, otherwise $\mathcal{D}^*_{B_1}$ draws the next card from B_1.*
- *Output the set of cards drawn by Dealer $\mathcal{D}^*_{B_1}$ in the order they were drawn.*

Algorithm 3 $\text{DecodeO}_{\gamma,\alpha}$

```
Parameter: $k \in [n]$, $\gamma$
Input: $M \in \{0,1\}^m, T \in \binom{[k]}{\alpha}, B_1$ an ordered subset of $[n]$ of size $k - \alpha$
  Initialize Guesser $\mathcal{G}_{(\gamma)}$ at turn $n - k$ with memory state $M$.
  $B' \leftarrow \emptyset$
  for $t \in \{n-k, \ldots, n\}$ do
      $g_t \leftarrow$ guess made by Guesser $\mathcal{G}_{(\gamma)}$
      if $g_t$ is tagged as correct (according to $T$) then
          $d_t \leftarrow g_t$
      else
          $d_t \leftarrow$ next card from $B_1$
      Append $d_t$ to $B'$
      Update $\mathcal{G}_{(\gamma)}$ memory state according to $d_t$
  return $B'$
```

A procedural description of $\text{DecodeO}_{\gamma,\alpha}$ is specified in Algorithm 3.
We assume that γ is given to us "for free" and is known during encoding and decoding of the ordered set B. We justify this assumption in two different ways:

- Fixing γ we can consider a specific encoding scheme for ordered sets $\text{EncodeO}_{\gamma,\alpha}$.
- We can assume that we encode a pair (γ, B) where γ is written explicitly in some natural way right next to $\text{EncodeO}_{\gamma,\alpha}(B)$.

Claim 4.3.4. *For every ordered set B:*

$$\text{DecodeO}_{\gamma,\alpha}(\text{EncodeO}_{\gamma,\alpha}(B)) = B.$$

Proof. If $\text{EncodeO}_{\gamma,\alpha}(B)$ encodes B explicitly, then this is trivial. Otherwise, both $\text{EncodeO}_{\gamma,\alpha}$ and $\text{DecodeO}_{\gamma,\alpha}$ works by simulating a game between a Guesser and a Dealer. Since the Guesser's randomness is fixed, the same game transcript is simulated in both simulations. Therefore, the decoder simulated Dealer $\mathcal{D}^*_{B_1}$ draws the same cards in the same order and thus recovers the same ordered set. □

Claim 4.3.5. *The code produced by* $\text{EncodeO}_{\gamma,\alpha}$ *is prefix-free.*

Proof. Let two ordered sets B, B' and denote by I the first bit of $\text{EncodeO}_{\gamma,\alpha}(B)$ and I' the first bit of $\text{EncodeO}_{\gamma,\alpha}(B')$. Observe that if $I \neq I'$ then the two descriptions cannot be a prefix of one another, and if $I = I'$ then the two descriptions are of the same length. To prove that the encoding scheme produce a prefix-free code, it suffice to show that for every B it holds that $\text{DecodeO}_{\gamma,\alpha}(\text{EncodeO}_{\gamma,\alpha}(B)) = B$. By Claim (4.3.4), this is indeed the case so the claim follows. □

Corollary 4.3.6. *If $\mathcal{G}_{(\gamma)}$ makes $C \geq \alpha$ correct guesses in the last k turns when playing against the static Dealer $\mathcal{D}_{\pi_{B,\gamma}}$ then* $\text{EncodeO}_{\gamma,\alpha}(B)$ *is of length*

$$\log\left(2 \cdot 2^m \cdot \binom{k}{\alpha} \cdot n(n-1)\cdots(n-k+\alpha+1)\right)$$

So we get that the encoding scheme saves bits for every correct guess while "paying" only m bits of memory. The contradiction comes from counting the number of the ordered sets B in two different ways:

- $n(n-1)\cdots(n-k+1)$ are all the possible options for ordered set B,
- and $\binom{k}{\alpha}\cdot n(n-1)\cdots(n-k+\alpha+1)2^{m+1}$ - upper bound on the possible options for ordered set B according to the encoding.

So we have

$$\begin{aligned} n(n-1)\cdots(n-k+1) &\le \binom{k}{\alpha}\cdot n(n-1)\cdots(n-k+\alpha+1)2^{m+1} \\ \therefore (n-k+\alpha)(n-k+\alpha-1)\cdots(n-k+1) &\le \binom{k}{\alpha}\cdot 2^{m+1} \\ \therefore (n-k)^{\alpha} &\le k^{\alpha}\cdot 2^{m+1} \end{aligned}$$

Taking logs we get

$$\alpha \le \frac{m+1}{\ln(n-k)-\ln k} \approx \frac{1}{\beta}\cdot\frac{m+1}{\ln n}.$$

As the code is prefix free, and from Lemma 2.1.5, we get that the probability over the choice of B for any correct guess beyond $\frac{m+1}{\beta\ln n}$ drops exponentially, so the expected number of correct guesses cannot be larger than that. By the above, we get that

$$\mathop{\mathbb{E}}_{B}[\mathrm{val}(\mathcal{D}_{\pi_{B,\gamma}},\mathcal{G}_{(\gamma)})|\gamma] \le \frac{m+1}{\beta\ln n}+2.$$

Recall that $\pi_{B,\gamma}$ is the deck arrangement that ends with B for which the guesser $\mathcal{G}_{(\gamma)}$ makes the most correct guesses in the last k turns. It follows that

$$\mathop{\mathbb{E}}_{B}\mathop{\mathbb{E}}_{\pi\in\Pi_B}[\mathrm{val}(\mathcal{D}_{\pi},\mathcal{G}_{(\gamma)})|\gamma] \le \mathop{\mathbb{E}}_{B}[\mathrm{val}(\mathcal{D}_{\pi_{B,\gamma}},\mathcal{G}_{(\gamma)})|\gamma]$$

Therefore, the above term upper bounds the expected number of correct guesses over the choice of B, i.e.

$$\mathop{\mathbb{E}}_{\pi}[C|\gamma] = \mathop{\mathbb{E}}_{\pi}[\mathrm{val}(\mathcal{D}_{\pi},\mathcal{G}_{(\gamma)})|\gamma] = \mathop{\mathbb{E}}_{B}\mathop{\mathbb{E}}_{\pi\in\Pi_B}[\mathrm{val}(\mathcal{D}_{\pi},\mathcal{G}_{(\gamma)})|\gamma] \le \mathop{\mathbb{E}}_{B}[\mathrm{val}(\mathcal{D}_{\pi_{B,\gamma}},\mathcal{G}_{(\gamma)})|\gamma] \le \frac{m+1}{\beta\ln n}+2.$$

Since the expected number of correct guesses over the randomness of both the Guesser and the Dealer, is a convex combination of the above, we conclude that the expected number of correct guesses in the last k turns is at most

$$c = \mathop{\mathbb{E}}_{\gamma,\pi}[C] \le \frac{m+1}{\beta\ln n}+2.$$

Now, consider the expected number of correct guesses throughout the game, where the expectation is over the deck shuffle and the Guesser's randomness. Suppose that the Guesser is perfect in the first $n-k$ steps, in the sense that all the guesses are reasonable. Then the expected number of correct guesses in the first turns is $H_n - H_k = \beta \ln n$. So we get that the total number of correct guesses is not expected to be better than

$$\beta \ln n + \frac{m+1}{\beta \ln n} + 2.$$

Taking the best β to be $\sqrt{m+1}/\ln n$, we get that this is not better than $2\sqrt{m+1}+2$.

Note that this bound still holds even if the Guesser has at it disposal a large amount of randomness that it can repeatedly access (i.e. storing the randomness is not charged to the memory). So we conclude with tight bounds up to constants:

Theorem 4.3.7. *There is a Guesser using m bits of memory that obtains* $1/2 \min\{\ln n, \sqrt{m}\}$ *correct guesses in expectation against the random-shuffle Dealer and any Guesser using m bits of memory can get* at most $O(\min\{\ln n, \sqrt{m}\})$ *correct guesses in expectation.*

5 Adaptive Dealer

We show that for every m there exists an adaptive Dealer $\mathcal{D}_m$ such that every Guesser with m memory bits is expected to make at most $\ln m + 2 \ln \log n + O(1)$ correct guesses when playing against Dealer $\mathcal{D}_m$.

Our proof is similar in structure to that in Section 4.3 in showing that a too successful Guesser can be used to compress a random set. We present our "Move-to-the-Back Dealer" in Section 5.1. We describe an encoding scheme for unordered sets (Section 5.3) that utilizes reasonable guesses made against our Dealer in order to achieve a shorter description of unordered sets. In Section 5.4 we show that having too many reasonable guesses implies compression, i.e. descriptions that are too short, and we get that the expected number of reasonable guesses is bounded. By bounding the expected number of reasonable guesses, we bound the expected number of correct guesses (Section 5.5). We analyze the performance of the entire Dealer, as a whole, in Section 5.6.

In Section 5.7 we show a *universal* adaptive Dealer that doesn't know how much memory the Guesser has, against which any Guesser with m bits of memory can score at most $(1+o(1)) \cdot \ln m + 8 \ln \log n + O(1)$.

5.1 Move-to-the-Back Dealer

Consider a game between some memory bounded Guesser and a Dealer who selects cards adaptively in an adversarial manner. Assume that at some turn the Guesser makes an incorrect guess. This guess may be incorrect because the Guesser had no luck, but it may also be incorrect because that card was played earlier and the Guesser did not recall that.

The idea is to use the Guesser's past guesses against her, and by doing so, forcing the Guesser to keep track of both past guesses and cards drawn. We achieve this by making incorrect available card guesses undrawable for some turns, i.e. "moving cards to the back of the deck". The Dealer we present begins the game with a properly shuffled deck, similarly to the random-shuffle Dealer. At a certain turn the Dealer begins to "move cards to the back" and every once in a while the Dealer reshuffles the deck, making undrawable cards available again. Towards the end of the game our Dealer makes one last reshuffle and draws cards one by one.

Epochs and the MtBE-strategy: The span of turns between reshuffles is called an **epoch** In particular, for $k \in [n]$, $\ell \in [k]$, the span of ℓ turns that begins when k cards are left, and ends when $k - \ell + 1$ cards are left, is called a (k, ℓ)-epoch. We refer to applying the strategy of "moving cards to the back" during a span of turns (epoch) by **MtBE-strategy** (which stands for Move-to-the-Back Epoch strategy).

The axis specifies the number of cards left in the deck.

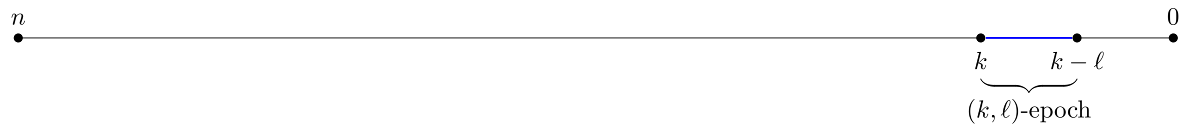

Figure 7: (k, ℓ)-epoch

Definition 5.1.1 ((k, ℓ, u)-MtBE-strategy)**.** *Given $k \in [n], \ell \in [k]$ and $u \leq \min\{k - \ell, \ell\}$, when t cards are left s.t. $k \leq t \leq k - \ell + 1$: Let A_t be the set of t available cards, let B'_t be the set of reasonable guesses made by the Guesser since when there were k cards in the back and until there are t, let $u_t = \min\{u, |B'_t|\}$ and let B_t be the set of the first u_t guesses from B'_t.*

A Dealer that follows (k, ℓ, u)-MtBE-strategy, draws a card uniformly at random from the set $A_t \setminus B_t$ when t cards are left in the deck for $k \leq t \leq k - \ell + 1$.

The upper bound u is necessary to make sure that during any point in (k, ℓ)-epoch the Dealer has cards to draw and that these cards are not too predictable. Though implicit, this definition describes a reshuffle, as when $t = k$ the set B_t is empty. A procedural description of this strategy is specified in Algorithm 4.

Algorithm 4 (k, ℓ, u)-MtBE-strategy

```
Parameter: $k \in [n]$, $\ell \in [k]$, $u \leq \min\{k - \ell, \ell\}$, $A \subseteq \binom{[n]}{k}$
  $B \leftarrow \emptyset$
  for $t \in \{k, \dots, k - \ell + 1\}$ do
      Draw a card $c \in_{\mathcal{R}} A \setminus B$ and discard $c$ from $A$
      $g \leftarrow$ guess made by Guesser
      if $g \in A$ and $|B| < u$ then                    ▷ Move to the back
          $B \leftarrow B \cup \{g\}$
```

We notice that, when the Dealer follows a (k, ℓ, u)-MtBE-strategy, a guess is reasonable if it is available and being guessed for the first time in the current epoch, assuming no more than u cards were moved to the back. We get that moving cards to the back works well against guessing techniques that repeat the same guess over and over. Recall that the guessing techniques that were successful against a static Dealer (namely the Following-Subsets technique from Section 4.1 and the Random-subsets technique from Section 4.2) did exactly that.

Observe that the Dealer cannot move cards to the back for too many rounds, as cards will become too predictable as $A_t \setminus B_t$ shrinks. Therefore, we apply the MtBE-strategy in a sequence and reshuffle the deck at the beginning/end of each epoch. Reshuffling the deck sets B_t to be the empty set again. This is visualized in Figure 8.

As the Dealer refrains from drawing reasonably guessed cards during an epoch, a significant portion (if not all) of these cards would reside in the deck at the beginning of the following epoch. Therefore, a Guesser can repeat her reasonable guesses from the previous epoch to get another chance, and most of these guesses will be reasonable. Repeating reasonable guesses can be done either by generating a pseudorandom sequence of guesses from which some portion would be reasonable, or by tracking cards using memory. We discuss this in detail after Lemma 5.2.1.

In blue: a sequence of d epochs of length ℓ where the ith epoch begins when k_i cards are left in the deck. The blue dots indicates a reshuffle.

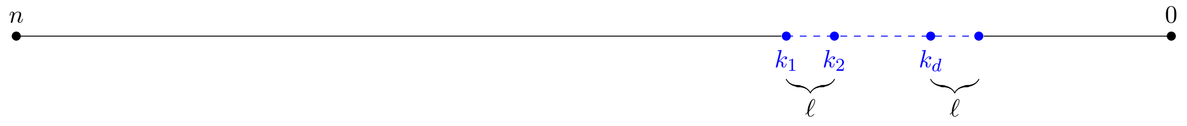

Figure 8: A sequence of MtBE-strategies in epochs of equal lengths

Finally, we present the Dealer, termed Move-to-the-Back Dealer, in all her glory.

Definition 5.1.2 (Move-to-the-Back Dealer). *Given $m \leq \frac{n}{\log^2 n}$, a Move-to-the-Back Dealer $\mathcal{D}_m$ plays according to the strategy:*

1. *Shuffle the deck uniformly at random and draw cards one by one until $\frac{n}{8e \log n}$ cards are left.*
2. *Play the MtBE-strategy d times in a sequence, where $d = (\frac{n}{8e \log n} - 2m \cdot \log n)/\ell$, each epoch for $\ell = m \cdot \log n$ turns, and move at most $u = \ell$ cards to the back during each epoch. Begin when $k_1 = \frac{n}{8e \log n}$ cards are left in the deck.*
3. *When $2m \log n$ cards left, shuffle the deck and draw cards one by one for the rest of the game.*

A visual description of $\mathcal{D}_m$ is provided in Figure 9. As per scale, consider Figure 10. Note that $\mathcal{D}_m$ is computationally efficient.

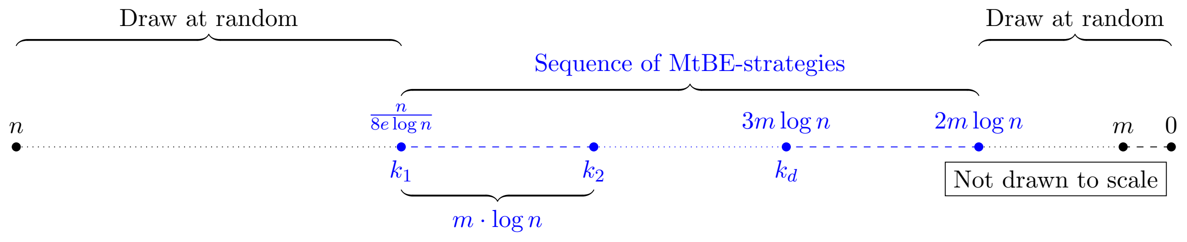

Figure 9: Our Dealer $\mathcal{D}_m$.
In blue: Play the MtBE-strategy in a sequence for d times, where $d = (\frac{n}{8e \log n} - 2m \cdot \log n)/\ell$, each epoch for $\ell = m \cdot \log n$ turns, begin when $k_1 = \frac{n}{8e \log n}$ cards are left in the deck.

We refer to the turn at which the first epoch begins as $n - k_1$. Observe that for every epoch in the sequence played by our Dealer we get that $\ell \leq k_i - \ell$ so we can set the maximal number of cards moved to the back u to ℓ.

In red, the span of turns during which the Dealer follows the MtBE-strategy.

$\frac{n}{8e\log n}$
first turn
last turn

Figure 10: Move-to-the-Back Dealer roughly to scale.

The main theorem of this section states that Move-to-the-Back Dealer works well against any memory bounded guesser.

Theorem 5.1.3. *For any m, every Guesser with m bits of memory is expected to make at most*

$$\ln m + 2\ln\log n + O(1)$$

correct guesses when playing against the Move-to-the-Back Dealer $\mathcal{D}_m$ (Definition 5.1.2).

Thinking about this theorem, it is clear that a Guesser with m memory bits can easily achieve $\ln m$ correct guesses in expectation by using the simple Subset Guessing strategy (from Section 3.1). So essentially, this theorem states that moving cards to the back and reshuffling every once in a while, is a *very* effective strategy against a memory bounded Guesser.

5.2 Towards a proof

Consider m and Move-to-the-Back Dealer $\mathcal{D}_m$. Let γ be the Guesser's randomness and let Δ be the Dealer's randomness. Let R_i denote the number of reasonable guesses made during the ith epoch. Let r_i be the expectation of R_i taken over the Guesser's and the Dealer's randomness, i.e.

$$r_i = \mathop{\mathbb{E}}_{\gamma,\Delta}\left[R_i\right].$$

To prove that our Dealer works well against any memory bounded Guesser we analyze the reasonable guesses during a single epoch. We claim that no Guesser can expect to make too many reasonable guesses during *any* of the epochs while our Dealer follows the MtBE-strategy. Consider the (k_i,ℓ)-epoch where the Dealer follows the MtBE-strategy.

Lemma 5.2.1 (Informal)**.** *A Guesser with m bits of memory that plays against a Move-to-the-Back Dealer $\mathcal{D}_m$ is expected to make at most*

$$r_i \le \max\left\{\frac{8\cdot e\cdot k_1\cdot \ell}{n}, m\right\} + 2.$$

reasonable guesses during any (k_i,ℓ)-epoch played by the $\mathcal{D}_m$.

Observe that when k_1 cards are left, the probability that a random guess is a card that is still in the deck is $\frac{k_1}{n}$. By linearity of expectation we get that guessing randomly for ℓ turns would yield at most $\frac{k_1\ell}{n}$ reasonable guesses in expectation. These cards are a reasonable guess exactly once during each epoch, as the Dealer avoids drawing them, but for the same reason it follows that a significant portion of them would still be available (and reasonable) in the next epoch. So we get that by using the same set of random guesses in each epoch the Guesser can get near the claimed upper bound of reasonable guesses. On the other hand, with carefully managed m bits, it may be possible in some cases to keep track of m cards that have not appeared (as we did in the Subset guessing technique in Section 3.1). So essentially, this lemma states that any memory bounded Guesser that plays against our Dealer cannot do much better then guessing cards at random or tracking m cards.

At any turn, the Dealer's strategy determines a distribution to sample a card from. In our case, this distribution is uniform on the available cards that were not moved to the back. We think of the Dealer as using precedence represented by a permutation π in order to make this choice:

Definition 5.2.2 (min-order)**.** *Given a permutation $\pi \in S_n$ and a set $C \subseteq [n]$ we say that a card $x \in C$ is the π-min-order card from C if x is the element of C with the smallest π value.*

We describe the randomness Δ of the Dealer in an indirect way: the Dealer has two independent parts for its randomness, (i) a sequence of permutations $\{\pi_t\}_{t=1}^{n}$ and (ii) a set $D \in \binom{[n]}{k_1}$. The way they are used is:

- For turn $1 \le t \le n - k_1$ the Dealer draws the π_t-min-order card from $A_t \setminus D$.
- For turn $n - k_1 + 1 \le t \le n$ the Dealer draws π_t-min-order card from $A_t \setminus B_t$.

As the Move-to-the-Back Dealer draws cards uniformly at random at the first $n-k_1$ turns, it follows that every set D can be kept for the last k_1 turns, so this is well defined.

Observation 5.2.3. *If the sequence of permutations $\{\pi_t\}_{t=1}^{n}$ and a set $D \in \binom{[n]}{k_1}$ are chosen uniformly at random, then this implementation is equivalent to Definition 5.1.2.*

Note that it was important to choose a permutation π_t independently for each turn t, since a common permutation π for all turns might leak information regarding the relative ranking of cards that were moved to the back at different times during an epoch. In particular, for any two reasonable guesses during some epoch, the earlier one has a higher probability of preceding the latter. As a result, the earliest reasonably guessed card has a higher probability of being drawn at the first turn in the following epoch, i.e., cards are drawn in a non-uniform manner.

We conclude that:

$$r_i = \mathop{\mathbb{E}}_{\gamma,\Delta}[R_i] = \mathop{\mathbb{E}}_{\gamma,\{\pi_t\},D}[R_i] = \sum_{\gamma,\{\pi_t\}} \mathop{\mathbb{E}}_{D}[R_i|\gamma,\{\pi_t\}] \cdot \Pr[\gamma,\{\pi_t\}].$$

The next two sections are dedicated to bounding the term

$$\mathop{\mathbb{E}}_{D}[R_i|\gamma,\{\pi_t\}]$$

for any permutations sequence $\{\pi_t\}$ and any Guesser's randomness γ.

5.3 Encoding scheme

In order to bound the expected number of reasonable guesses in a single epoch we present an encoding scheme for subsets of $[n]$ of size k_1. The encoding scheme utilizes reasonable guesses against our Dealer during any of the epochs to achieve a shorter description.

Consider some Guesser $\mathcal{G}$ that plays against the Move-to-the-Back Dealer $\mathcal{D}_m$ (Definition 5.1.2) and fix one of the epochs (k_i, ℓ)-epoch played by $\mathcal{D}_m$. For every triplet $\gamma, \{\pi_t\}_{t=1}^{n}, D$ we associate the Guesser $\mathcal{G}_{(\gamma)}$ with fixed randomness γ and the Dealer $\mathcal{D}_{m,(D,\{\pi_t\})}$ with fixed randomness that corresponds to $\{\pi_t\}_{t=1}^n$ and D as the last cards to be played.

Fix some prefix-free code (Definition 2.1.3) for sets of size k_1. The encoding scheme will use this code for the cases when there are not enough reasonable guesses to utilize.

Definition 5.3.1 (EncodeU$_{\gamma,\{\pi_t\},\alpha,i}$). *To encode a set $D \in \binom{[n]}{k_1}$, the function* EncodeU$_{\gamma,\{\pi_t\},\alpha,i}$ *simulates a game between Guesser $\mathcal{G}_{(\gamma)}$ and Dealer $\mathcal{D}_{m,(D,\{\pi_t\})}$.*

Denote by T' the set of turns during the (k_i, ℓ)-epoch at which $\mathcal{G}_{(\gamma)}$ made a reasonable guess, i.e.

$$T' = \{n - k_i \le t \le n - k_i + \ell - 1 | g_t \text{ is a reasonable guess}\}.$$

- *If $|T'| < \alpha$, then the code is made of an indicator bit set to 0 and an explicit prefix free representation of D.*
- *If $|T'| \ge \alpha$, then let T be the first α turns from T' during which the Guesser guessed reasonably during the (k_i, ℓ)-epoch. The code is made of:*
  1. *Indicator bit set to 1 (1 bit).*
  2. *Guesser's memory state M at turn $n - k_1$ (m bits).*
  3. *Description of T, the first α turns at which the Guesser guessed reasonably during the (k_i, ℓ)-epoch ($\log \binom{\ell}{\alpha}$ bits).*
  4. *Binary vector V of length α that tags which of the reasonable guesses described in T were also correct (α bits[4]).*
  5. *Description of $D \setminus \{g_t | t \in T\}$ ($\log \binom{n}{k_1 - \alpha}$ bits).*

A visual representation of the stored information is provided in Figure 11.

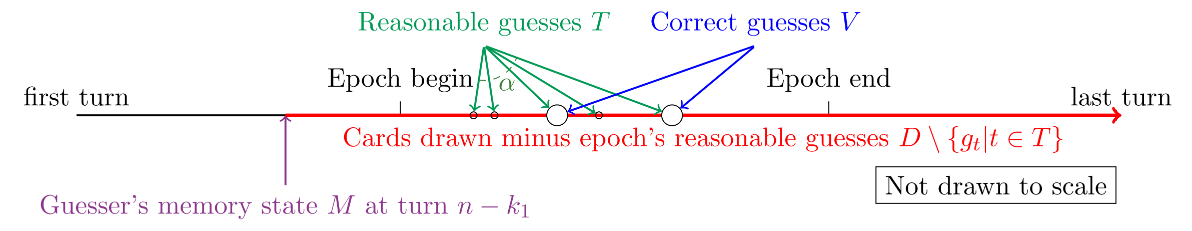
Reasonable guesses T
Correct guesses V
Epoch begin
α
Epoch end
first turn
last turn
Cards drawn minus epoch's reasonable guesses $D \setminus \{g_t | t \in T\}$
Not drawn to scale
Guesser's memory state M at turn $n - k_1$

Figure 11: EncodeU$_{\gamma,\{\pi_t\},\alpha,i}$. Colored - information stored by the encoding scheme.

We first define the Dealer we will use during the decoder simulation. Since we simulate this Dealer, we can break the usual course of the game. In particular, we will assume that the Dealer begins playing the game at the middle and with a partial deck, that the Dealer is aware of the Guesser's guess *before* placing a card and that the Dealer can place cards that do not reside in the deck.

Definition 5.3.2. *The Dealer $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ plays according to the MtBE-strategy, as configured for $\mathcal{D}_m$ (Definition 5.1.2) with few modifications:*

[4]Though a shorter representation is possible, it suffices for our purpose.

- *When t cards are left for $t \in \{k_1, \dots, k_i + 1\}$: play according to the MtBE-strategy with deck D_1 moving cards to the back when they are guessed and still in the deck.*
- *When t card are left, for $t \in \{k_i, \dots, k_i - \ell + 1\}$, and until the αth reasonable guess: if turn t is tagged both as reasonable (by T) and correct (by V), then draw the card g_t guessed by the Guesser. Otherwise draw the π_t-min-order card from the deck that was not moved to the back.*
- *Once α reasonable guesses occurred during the ith epoch, stop playing.*

A procedural description of the Dealer $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ is specified in Algorithm 5.

Algorithm 5 Behavior of $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ while simulated by $\text{DecodeU}_{\gamma,\{\pi_t\},\alpha,i}$

```
Parameter: $D_1 \subseteq \binom{[n]}{k-\alpha}$, $\{\pi_t\}_{t=1}^n$, $T$, $V$, $\alpha$, $i$
  for $j \in [i]$ do                                                    ▷ For epochs prior to $i$
    $B \leftarrow \emptyset$
    for $t \in \{k_j, \dots, k_j - \ell_j + 1\}$ do
      Draw the $\pi_t$-min-order card from $D_1 \setminus B$ and discard from $D_1$
      $g_t \leftarrow$ guess made by Guesser
      if $g_t \in D_1$ and $|B| < u$ then                               ▷ Move to the back
        $B \leftarrow B \cup \{g_t\}$
  $B \leftarrow \emptyset$                                                          ▷ $i$th epoch
  for $t \in \{k_i, \dots, k_i - \ell + 1\}$ do
    $g_t \leftarrow$ guess made by Guesser
    if $g_t$ is tagged as both reasonable and correct (according to $T$ and $V$) then
      $d_t \leftarrow g_t$
    else
      $d_t \leftarrow \pi_t$-min-order card from $D_1 \setminus B$ and discard from $D_1$
    Draw $d_t$
    if $g_t$ is tagged as reasonable (according to $T$) then            ▷ Move to the back
      $B \leftarrow B \cup \{g_t\}$
    if $|B| = \alpha$ then
      Stop playing
```

Note that after turn $n - k_1$, the Dealer simulated by the *encoder* recognizes a guess as reasonable if it is from D and wasn't drawn yet. But when the Dealer simulated by the *decoder* observes a guess of a card not in D_1, then the Dealer cannot tell whether this is a card that will turn to be reasonable at the (k_i, ℓ)-epoch or a card that has been played before turn $n - k_1$. Therefore, the Dealer simulated by the decoder recognizes a guess as reasonable (and moves it to the back) if it is from D_1. We will soon see that this behavior allows the decoder to reproduce the same game transcript, and by doing so, decode the original set.

We now define the decode function.

Definition 5.3.3 ($\text{DecodeU}_{\gamma,\{\pi_t\},\alpha,i}$). *If the indicator bit is* 0 *then decode a set from the remaining bits in the natural way. Otherwise, the function parses the remaining bits as a 4-tuple (M, V, T, D_1) as encoded by* $\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}$*, and then simulates and records a partial game between the Dealer $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ (see Definition 5.3.2) and the Guesser $\mathcal{G}_{(\gamma)}$:*

- *Initialize Guesser $\mathcal{G}_{(\gamma)}$ with memory state M at turn $n - k_1$ and simulate a game against $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ until the αth reasonable guess in the (k_i, ℓ)-epoch.*
- *Let D_2 be the set of card guesses that were tagged as reasonable (by T).*
- *Output $D' = D_1 \cup D_2$.*

A procedural description of $\mathrm{DecodeU}_{\gamma,\{\pi_t\},\alpha,i}$ is specified in Algorithm 6.

Algorithm 6 $\mathrm{DecodeU}_{\gamma,\{\pi_t\},\alpha,i}$

```
Parameter: $k_i \in [n]$, $\ell \in [k_i]$, $\{\pi_t\}_{t=1}^n$, $\gamma$
Input: $x \in \{0,1\}^*$
  if $x_1 = 0$ then
      Parse $D'$ from $x$
      return $D'$
  Parse $M \in \{0,1\}^m, V \in \{0,1\}^\alpha, T \in \binom{[\ell]}{\alpha}, D_1 \in \binom{[n]}{k_1-\alpha}$ from $x$
  Initialize Guesser $\mathcal{G}_{(\gamma)}$ at turn $n - k_1$ with memory state $M$.
  $D_2 \leftarrow \emptyset$
  for $t \in \{k_1, \ldots, k_i + 1\}$ do
      Simulate a turn between $\mathcal{G}_{(\gamma)}$ and $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ and update Guesser's memory accordingly
  for $t \in \{k_i, \ldots, k_i - \ell + 1\}$ do
      Simulate a turn between $\mathcal{G}_{(\gamma)}$ and $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ and update Guesser's memory accordingly
      $g_t \leftarrow$ guess made by Guesser $\mathcal{G}_{(\gamma)}$
      if $g_t$ is tagged as reasonable (according to $T$) then
          $D_2 \leftarrow D_2 \cup \{g_t\}$
  return $D_1 \cup D_2$
```

We assume that both the Dealer's precedence $\{\pi_t\}_{t=1}^n$ and the Guesser's randomness γ are given to us "for free" and are known during encoding and decoding of the set D. We justify this assumption in two different ways:

- Fixing γ and $\{\pi_t\}_{t=1}^n$ (i.e. fix the Dealer's random source for the order) we can consider an encoding scheme for sets $D \in \binom{[n]}{k_1}$.
- We can assume that we encode a triplet $(\gamma, \{\pi_t\}_{t=1}^n, D)$ where γ and π are written explicitly in some natural way right next to $\mathrm{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}(D)$.

Claim 5.3.4. *For all $D \in \binom{[n]}{k_1}$:*

$$\mathrm{DecodeU}_{\gamma,\{\pi_t\},\alpha,i}(\mathrm{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}(D)) = D.$$

Proof. For the case that the set is encoded explicitly, this is trivial.

Recall that both $\mathrm{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}$ and $\mathrm{DecodeU}_{\gamma,\{\pi_t\},\alpha,i}$ work by simulating a partial game between $\mathcal{G}_{(\gamma)}$ and $\mathcal{D}_{m,(D,\{\pi_t\})}$, $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ (resp.). Denote by g_t, d_t the card guessed by the Guesser and the card drawn by Dealer (resp.) at turn t during the game simulated by $\mathrm{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}$, and equivalently g'_t, d'_t those simulated by $\mathrm{DecodeU}_{\gamma,\{\pi_t\},\alpha,i}$.

We first prove by induction that the same game transcript is produced by both simulations, i.e. that for every turn $n - k_1 \leq t < n - k_i + \ell$ it holds that $g_t = g'_t$ and $d_t = d'_t$.

We highlight two facts:

1. If both simulated Guessers have the same memory state at the beginning of some turn, they guess the same card at that turn. This is true because both simulated Guessers also have the same fixed randomness γ.

2. If both simulated Guessers have the same memory state at the beginning of some turn, and both simulated Dealers draw the same card at that turn, then both Guessers begin the next turn with the same memory state.

At the beginning of turn $n - k_1$, both simulated Guessers have the same memory state. Therefore, to prove that the same transcript is simulated, it suffice to prove that both Dealers draw the same card in every turn.

Without loss of generality, assume that the (k_i, ℓ)-epoch is not the first epoch.

Base: The encoder simulated Dealer $\mathcal{D}_{m,(D,\{\pi_t\})}$ draws the min-order card d_{n-k_1} from D. Since d_{n-k_1} was drawn before the (k_i, ℓ)-epoch it is also in D_1, thus it is the min-order card from D_1 as well, so it is drawn by $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ and we get that $d_{n-k_1} = d'_{n-k_1}$.

Step for turn $n - k_1 < t < n - k_i$: Since turn $n - k_1$ the Dealer simulated by the encoder $\mathcal{D}_{m,(D,\{\pi_t\})}$ recognizes an available guess as reasonable if it is from D while the Dealer simulated by the decoder $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ recognizes it as reasonable if it is from D_1. By the induction hypothesis, the same transcript was simulated so far by the encoder and decoder, and since $D_1 \subseteq D$ it follows that the back of $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ is always a subset of the back of $\mathcal{D}_{m,(D,\{\pi_t\})}$. In particular, the *o*nly cards that are missing from the back of $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ are those who will turn to be reasonable in the (k_i, ℓ)-epoch.

Since d_t was drawn before $n - k_i$ then $d_t \in D_1 \subset D$ (it cannot be in $D \setminus D_1$, since this would mean that it could not be a candidate for reasonable guess in (k_i, ℓ)-epoch). Since d_t is the π_t-min-order card from D then d_t is also the π_t-min-order card from D_1. As a result, $d'_t = d_t$.

Step for turn $n - k_i \leq t < n - k_i + \ell$: The same argument holds as before, but we need to clarify the case where a correct guess was made. If the encoder simulated Guesser guessed correctly then $g_t = d_t$ and $d_t \notin D_1$. In that case, the turn t is tagged as both reasonable (by T) and correct (by V). Therefore, $d'_t = g'_t$ because that's how $\mathcal{D}^*_{m,(D_1,\{\pi_t\},T,V)}$ works. Since $g'_t = g_t$ we get that $d'_t = d_t$.

Finally, since the same partial game is simulated, the same guesses are tagged as reasonable during the ith epoch (by T) and $D_2 = \{g'_t | t \in T\} = \{g_t | t \in T\}$. We finish the proof by observing that

$$D = (D \setminus \{g_t | t \in T\}) \cup \{g_t | t \in T\} = D_1 \cup D_2 = \mathrm{DecodeU}_{\gamma,\{\pi_t\},\alpha,i}(\mathrm{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}(D)).$$

□

Claim 5.3.5. *The code produced by* $\mathrm{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}$ *is prefix-free.*

Proof. Let two sets $D, D' \in \binom{[n]}{k_1}$ and denote by I, I' the first bit in the encoding of D, D' (resp.). Observe that if $I \neq I'$ then the two descriptions cannot be a prefix of one another, if $I = I' = 0$ then this is trivial and if $I = I' = 1$ then the two descriptions are of the same length. Therefore, to prove that $\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}$ produce a prefix-free code, it suffice to show that for every D it holds that $\text{DecodeU}_{\gamma,\{\pi_t\},\alpha,i}(\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}(D)) = D$. By Claim (5.3.4), this is indeed the case so the claim follows. ☐

Denote by $\mathcal{X}_\alpha \subseteq \binom{[n]}{k_1}$ the collection of all sets $D \in \binom{[n]}{k_1}$ such that $\mathcal{G}_{(\gamma)}$ makes at least α reasonable guesses against $\mathcal{D}_{m,(D,\{\pi_t\})}$ during the (k_i, ℓ)-epoch. Observe that all sets in $\mathcal{X}_\alpha$ are encoded by $\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}$ to bit strings of the same length. Denote by $w(m, k_1, \ell, \alpha)$ the length in bits of $\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}(D)$ for $D \in \mathcal{X}_\alpha$.

Corollary 5.3.6. *If $\mathcal{G}_{(\gamma)}$ makes at least α reasonable guesses against the Dealer $\mathcal{D}_{m,(D,\{\pi_t\})}$ during the (k_i, ℓ)-epoch, then the length of $\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}(D)$ is*

$$w(m, k_1, \ell, \alpha) \triangleq \log\left(2 \cdot 2^m \cdot 2^\alpha \cdot \binom{n}{k_1 - \alpha} \cdot \binom{\ell}{\alpha}\right).$$

Looking at the term from this corollary, it can be seen that for every increase in α we save roughly $\log n$ bits and pay roughly $1 + \log \ell$ bits. In our Dealer (Definition 5.1.2) each epoch consists of $\ell = m \log n$ turns, so we get that we expect to save order of $\log n - \log m$ bits for every reasonable guess. We analyze this in detail in Claim (5.4.1).

5.4 Upper Bound on the Number of Reasonable Guesses

The function $\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}$ yields descriptions of lengths that varies with α. For example, for $\alpha = 0$, we get that in every simulation there are at least 0 reasonable guesses, so all descriptions are of length $\log\left(2 \cdot 2^m \binom{n}{k_1}\right)$, which is much larger than storing the set explicitly. The following two claims deals with the encoding length. We first claim that making more reasonable guesses implies shorter descriptions, and more specifically, that each additional guess saves at least a bit.

Claim 5.4.1 (Length decreases with reasonable guesses)**.** *For every $m, k_1 \in [n/4]$, $\ell \in [k_1]$, and every $\frac{4k_1\ell}{n} \leq \alpha$ it holds that*

$$w(m, k_1, \ell, \alpha + 1) \leq w(m, k_1, \ell, \alpha) - 1.$$

Proof. We will calculate $w(m, k_1, \ell, \alpha) - w(m, k_1, \ell, \alpha + 1)$ and show that it is at least 1.

$$\begin{aligned}
\log\left(2 \cdot 2^m \cdot 2^\alpha \cdot \binom{n}{k_1 - \alpha} \cdot \binom{\ell}{\alpha}\right) &- \log\left(2 \cdot 2^m \cdot 2^{\alpha+1} \cdot \binom{n}{k_1 - \alpha - 1} \cdot \binom{\ell}{\alpha + 1}\right) \\
&= \log\left(\frac{1}{2} \cdot \frac{\binom{n}{k_1-\alpha}}{\binom{n}{k_1-\alpha-1}} \cdot \frac{\binom{\ell}{\alpha}}{\binom{\ell}{\alpha+1}}\right) \\
&= \log\left(\frac{1}{2} \cdot \frac{n - k_1 + \alpha + 1}{k_1 - \alpha} \cdot \frac{\alpha + 1}{\ell - \alpha}\right) \\
&> \log\left(\frac{n\alpha}{2k_1\ell}\right).
\end{aligned}$$

By assumption $\alpha \geq \frac{4k_1\ell}{n}$ so the term inside the log is larger than 2. ☐

Corollary 5.4.2. *For every $m, k_1 \in [n/4]$, $\ell \in [k_1]$, and every $\frac{4k_1\ell}{n} \leq \alpha < \ell$ and for every $\beta \in [\ell - \alpha]$ it holds that*

$$w(m, k_1, \ell, \alpha + \beta) \leq w(m, k_1, \ell, \alpha) - \beta.$$

Consider a random set D chosen uniformly at random from $\binom{[n]}{k_1}$. The entropy of D is

$$H(D) = \log \left| \binom{[n]}{k_1} \right| = \log \binom{n}{k_1}.$$

The next lemma describes the amount of reasonable guesses required to achieve compression, i.e. descriptions of length shorter than the entropy of a random input.

Claim 5.4.3 (Encoding achieves compression)**.** *For every $m < n, k_1 \in [n/8e]$, $\ell \in [k_1]$, if*

$$\alpha \geq \max \left\{ \frac{8 \cdot e \cdot k_1 \cdot \ell}{n}, m \right\}$$

then

$$w(m, k_1, \ell, \alpha) < \log \binom{n}{k_i}.$$

Proof. We will show that $\log \binom{n}{k_1} - w(m, k_1, \ell, \alpha) \geq 0$. To do so, we will use an upper bound on binomial coefficients derived from Stirling's approximation ($\binom{b}{a} < \left(\frac{be}{a}\right)^a$) and the fact that $\binom{n}{k} / \binom{n}{k-a} = \frac{(n-k+a)\ldots(n-k+1)}{(k)\ldots(k-a+1)} > \left(\frac{n-k+a}{k}\right)^a > \left(\frac{n}{k}\right)^a$.

$$\begin{aligned}
\log \binom{n}{k_1} &- \log \left(2 \cdot 2^m \cdot 2^\alpha \cdot \binom{n}{k_1 - \alpha} \cdot \binom{\ell}{\alpha} \right) \\
&= \log \left(\frac{\binom{n}{k_1}}{\binom{n}{k_1 - \alpha}} \cdot \frac{1}{2^{m+1}} \cdot \frac{1}{2^\alpha} \cdot \frac{1}{\binom{\ell}{\alpha}} \right) \\
&> \log \left(\left(\frac{n}{k_1} \right)^\alpha \cdot \frac{1}{2^{m+1}} \cdot \frac{1}{2^\alpha} \cdot \left(\frac{\alpha}{\ell e} \right)^\alpha \right) \\
&= \log \left(\left(\frac{n\alpha}{2ek_1\ell} \right)^\alpha \cdot \frac{1}{2^{m+1}} \right) \\
&> \log \left(\left(\frac{n\alpha}{4ek_1\ell} \right)^\alpha \cdot \frac{1}{2^m} \right).
\end{aligned}$$

By assumption, $\alpha \geq \frac{8 \cdot e \cdot k_1 \cdot \ell}{n}$ and $\alpha \geq m$, so we get that the term inside the log is greater than 1, so the claim follows.

□

Combining the above claims we get that no Guesser can make too many reasonable guesses against our Dealer. Recall the random variable R_i that denotes the number of reasonable guesses that a Guesser makes during the (k_i, ℓ)-epoch. Consider the sequence of epochs $\{(k_i, \ell)\text{-epoch}\}_{i=1}^d$ played by the Move-to-the-Back Dealer (from Definition 5.1.2) $\mathcal{D}_m$.

Claim 5.4.4. *For every Guesser $\mathcal{G}$ with m bits of memory, and for every epoch $i \in [d]$, for $\beta < \ell - \max\left\{\frac{8 \cdot e \cdot k_1 \cdot \ell}{n}, m\right\}$, the probability that $\mathcal{G}$ makes more than $\beta + \max\left\{\frac{8 \cdot e \cdot k_1 \cdot \ell}{n}, m\right\}$ reasonable guesses during the (k_i, ℓ)-epoch, is at most*

$$\Pr_{D \in \binom{[n]}{k_1}} \left[R_i \geq \max\left\{\frac{8 \cdot e \cdot k_1 \cdot \ell}{n}, m\right\} + \beta \right] \leq 2^{-\beta}.$$

Proof. Fix some γ and $\{\pi_t\}_{t=1}^{n}$ and consider the probability conditioned on these choices of randomness. Let $\alpha = \beta + \max\left\{\frac{8 \cdot e \cdot k_1 \cdot \ell}{n}, m\right\}$.

If a Guesser with fixed randomness $\mathcal{G}_{(\gamma)}$ makes at least α reasonable guesses against our Move-to-the-Back Dealer with fixed randomness $\mathcal{D}_{m,(D,\{\pi_t\})}$ during the (k_i, ℓ)-epoch, then the set D is encoded by $\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}$ into a string of $w(m, k_1, \ell, \alpha)$ bits.

Since $\alpha > \max\left\{\frac{8 \cdot e \cdot k_1 \cdot \ell}{n}, m\right\}$, from Claim (5.4.3), the length of $\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}(D)$ is shorter than the entropy of a random D sampled uniformly from $\binom{[n]}{k_1}$. Since $\alpha > \frac{4k_1\ell}{n}$, from Corollary 5.4.2 we get that the length of $\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}(D)$ is at least β bits shorter than the entropy of a random D. Lemma 2.1.5 tells us that the probability to encode a random element by β bits below the entropy decays exponentially, i.e.

$$\begin{aligned}
\Pr_{D \in \binom{[n]}{k_1}} \left[R_i \geq \max\left\{\frac{8 \cdot e \cdot k_1 \cdot \ell}{n}, m\right\} + \beta \right] &\leq \Pr_{D \in \binom{[n]}{k_1}} \left[\text{EncodeU}_{\gamma,\{\pi_t\},\alpha,i}(D) = H(D) - \beta \right] \\
&\leq 2^{-\beta}.
\end{aligned}$$

As this is true to any fixed γ and $\{\pi_t\}_{t=1}^{n}$, we get that this is true unconditionally. □

What about the upper bound u? Recall that while the Dealer follows the (k, ℓ, u)-MtBE-strategy (Definition 5.1.1), only the first u reasonably guessed cards are moved to the back; therefore only the first u reasonable guesses are guaranteed to be distinct. Any reasonable guess beyond u may be useless for set encoding. Further, if the Guesser is somehow able to reach u reasonable guesses, then moving cards to the back works in the Guesser's favor (as cards become predictable), and the probability to guess reasonably grows with every guess. We address this concern by recalling that our Move-to-the-Back Dealer (Definition 5.1.2) plays the MtBE-strategy in a sequence of epochs for which $u = \ell$, therefore, it is impossible to make more than than u reasonable guesses in an epoch. In Section 5.7 we present a Dealer for which $u < \ell$, and we restate the claim to consider the upper bound u (see Claim (5.7.2)).

As a corollary we get a bound on the expected number of reasonable guesses.

Corollary 5.4.5 (Formal)**.** *For every Guesser with m bits of memory, every epoch $i \in [d]$, the expected number of reasonable guesses during (k_i, ℓ)-epoch is at most*

$$r_i \leq \max\left\{\frac{8 \cdot e \cdot k_1 \cdot \ell}{n}, m\right\} + 2.$$

For this corollary to be meaningful we require that $k_1 \leq \frac{n}{8e}$, as otherwise it implies that $r_i < \ell$, i.e. that the expected number of reasonable guesses in the epoch is less than the number of turns in the epoch, which is always true. Observe this corollary is meaningful for every epoch played by our Dealer as the first epoch begins at turn $n - \frac{n}{8e \log n}$.

5.5 Upper Bound on the Number of Correct Guesses

Determining an upper bound on the number of *reasonable* guesses, we can establish an upper bound on the number of *correct guesses* per epoch that a memory bounded guesser can make against our Dealer.

Recall that during the ith epoch our Move-to-the-Back Dealer (Definition 5.1.2) plays according to the (k_i, ℓ, u)-MtBE-strategy. Recall that r_i denotes the expected number of reasonable guesses during the ith epoch. Denote by c_i the expected number of correct guesses during the ith epoch, where the expectation is taken over the Guesser's and Dealer's randomness γ, Δ.

Lemma 5.5.1. *For every Guesser $\mathcal{G}$ with m bits of memory and every epoch $i \in [d]$, the expected number of correct guesses during the (k_i, ℓ)-epoch is at most*

$$c_i \leq \frac{r_i}{k_i - \ell - u}.$$

Proof. Let $R_{i,j}$ be an indicator random variable for the event that the jth guess during the (k_i, ℓ)-epoch is reasonable. Let $C_{i,j}$ be an indicator random variable for the event that the jth guess during the (k_i, ℓ)-epoch is correct. Let a random variable C_i denote the number of correct guesses that the Guesser $\mathcal{G}$ made during the (k_i, ℓ)-epoch.

We first bound the probability for a reasonable guess to be correct. Consider the jth turn during a (k_i, ℓ)-epoch and assume that α cards were moved to the back until the jth turn. At the beginning of the (k_i, ℓ)-epoch there are k_i cards left to play. In each turn one card is discarded, thus $j-1$ cards have been discarded since the beginning of the epoch. Each reasonable guess can be correct if it is one of the $k_i - (j-1) - \alpha$ remaining cards. Therefore

$$\begin{aligned}
\Pr\left[C_{i,j} = 1 | R_{i,j} = 1\right] &= \frac{1}{k_i - (j-1) - \alpha} \\
&\leq \frac{1}{k_i - (\ell - 1 - 1) - \alpha} && (5.1) \\
&\leq \frac{1}{k_i - \ell - u + 2} && (5.2) \\
&< \frac{1}{k_i - \ell - u}. && (5.3)
\end{aligned}$$

Where Inequality (5.1) is due to $j \leq \ell$, Inequality (5.2) is because at most u cards are moved to the back.

We can now bound the expected number of correct guesses directly

$$\begin{aligned}
c_i = \mathbb{E}[C_i] &= \sum_{j \in [\ell]} \mathbb{E}[C_{i,j}] && (5.4) \\
&= \sum_{j \in [\ell]} \Pr[R_{i,j} = 1] \cdot \mathbb{E}[C_{i,j} | R_{i,j} = 1] + \Pr[R_{i,j} = 0] \cdot \mathbb{E}[C_{i,j} | R_{i,j} = 0] && (5.5) \\
&= \sum_{j \in [\ell]} \Pr[R_{i,j} = 1] \cdot \mathbb{E}[C_{i,j} | R_{i,j} = 1] && (5.6) \\
&< \sum_{j \in [\ell]} \Pr[R_{i,j} = 1] \cdot \frac{1}{k_i - \ell - u} && (5.7) \\
&= \frac{r_i}{k_i - \ell - u}. && (5.8)
\end{aligned}$$

Where Equality (5.4) is true by definition and due to linearity of expectation, Equality (5.5) is from law of total expectation, Equality (5.6) is true since a correct guess is necessarily reasonable, Inequality (5.7) is due to Inequality (5.3), and Equality (5.8) is true since $\sum_{j\in[\ell]} \Pr[R_{i,j}=1] = r_i$. □

5.6 Analysis of the Move-to-the-Back Dealer

Having established a bound for a single epoch, we are ready to conclude the analysis of our Dealer and show its overall performance.

We recall that our Move-to-the-Back Dealer (Definition 5.1.2 and Figure 9) starts the game with a properly shuffled deck from which the Dealer draws until k_1 cards are left, the Dealer then follows the MtBE-strategy over and over again and reshuffles every ℓ turns, and when $2m \cdot \log n$ cards are left, our Dealer shuffles the deck one last time and draws cards randomly until the end of the game. In particular, the Dealer plays the MtBE-strategy in a sequence of d epochs, where $d = (\frac{n}{8e\log n} - 2m \cdot \log n)/\ell$, each epoch consists of $\ell = m \cdot \log n$ turns, and as for every epoch it holds that $\ell \le k_i - \ell$ it follows that we can set $u = \ell$.

In the upcoming lemma, we will analyze and bound the cumulative number of correct guesses that any memory bounded Guesser can expect to make throughout the sequence of epochs.

Lemma 5.6.1. *For $m \le \frac{n}{\log^2 n}$, every Guesser with m memory bits that play against the Move-to-the-Back Dealer $\mathcal{D}_m$ is expected to guess correctly at most 1 time in total throughout the sequence of epochs that follows the MtBE-strategy.*

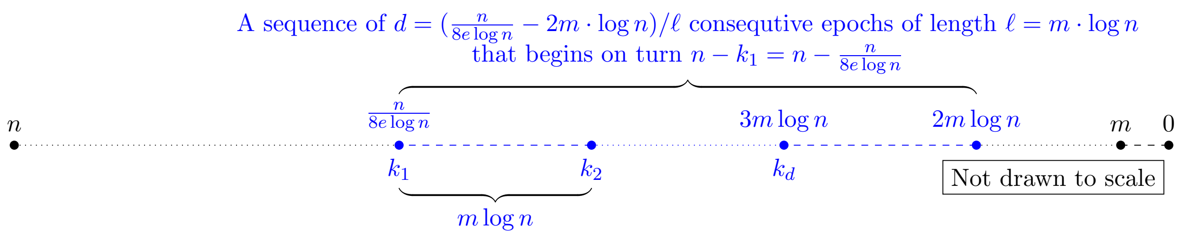
A sequence of $d = (\frac{n}{8e\log n} - 2m \cdot \log n)/\ell$ consequtive epochs of length $\ell = m \cdot \log n$
that begins on turn $n - k_1 = n - \frac{n}{8e\log n}$
n
$\frac{n}{8e\log n}$
$3m\log n$
$2m\log n$
m
0
k_1
k_2
k_d
Not drawn to scale
$m\log n$

Figure 12: Illustration of the MtBE-strategy played in a sequence of epochs.

Proof. The Dealer plays the MtBE-strategy in a sequence for $d = (\frac{n}{8e\log n} - 2m \cdot \log n)/\ell$ epochs. All epochs are of the same length $\ell = m \cdot \log n$, and during each epoch, the Dealer follows the (k_i, ℓ, u)-MtBE-strategy where $u = \ell = m \cdot \log n$.

Since the first epoch begins when there are $\frac{n}{8e\log n}$ cards left, and as a corollary from Corollary 5.4.5, we get that the expected number of reasonable guesses during any epoch in the sequence is upper bounded by $\frac{\ell}{\log n} + 2$.

$$r_i \le \max\left\{\frac{8 \cdot e \cdot k_1 \cdot \ell}{n}, m\right\} + 2 \le \frac{\ell}{\log n} + 2. \tag{5.9}$$

We now bound the expected number of correct guesses during a single epoch

$$c_i < \frac{r_i}{k_i - \ell - u} \tag{5.10}$$
$$\leq \frac{\ell/\log n + 2}{k_i - \ell - u} \tag{5.11}$$
$$= \frac{1}{\log n} \cdot \frac{\ell}{k_i - 2\ell} + \frac{2}{k_i - 2\ell} \tag{5.12}$$

Where Inequality (5.10) is by Lemma 5.5.1, Inequality (5.11) is true because of Inequality (5.9), Equality (5.12) is true since $\ell = u$.

Observe that
$$k_i = 3m \log n + (d - i) \cdot m \log n = (d - i + 3)\ell.$$

Summing over the epochs we get that

$$\begin{aligned}
\sum_{i=1}^{d} c_i &< \frac{1}{\log n} \sum_{i=1}^{d} \frac{\ell}{k_i - 2\ell} + 2 \cdot \sum_{i=1}^{d} \frac{1}{k_i - 2\ell} \\
&= \frac{1}{\log n} \sum_{i=1}^{d} \frac{\ell}{(d-i+3)\ell - 2\ell} + 2 \cdot \sum_{i=1}^{d} \frac{1}{(d-i+3)\ell - 2\ell} \\
&= \frac{1}{\log n} \sum_{i=1}^{d} \frac{1}{d-i+1} + \frac{2}{\ell} \cdot \sum_{i=1}^{d} \frac{1}{d-i+1} \\
&\approx \frac{1}{\log n} \ln d + \frac{2}{\ell} \ln d.
\end{aligned}$$

Since $d = (\frac{n}{8e \log n} - 2m \cdot \log n)/\ell$ we get that

$$\ln d = \ln \left((\frac{n}{8e \log n} - 2m \cdot \log n)/\ell \right) < \ln \left(\frac{n}{\ell} \right) < \ln n.$$

Combining the two above, we get that

$$\sum_{i=1}^{d} c_i < \frac{\ln n}{\log n} + \frac{2 \ln n}{m \cdot \log n} < 1.$$

□

With this, we can now analyze the performance of our Dealer.

Theorem 5.6.2. *For any $m \leq n$ there exists a Dealer $\mathcal{D}_m$ such that every Guesser $\mathcal{G}$ with m memory bits is expected to make at most $\ln m + 2 \ln \log n + O(1)$ correct guesses throughout the game.*

Proof. If $m > \frac{n}{\log^2 n}$ then $\ln m + 2 \ln \log n + O(1) \geq \ln(n)$ so the argument is correct against any random-shuffle Dealer. If $m \leq \frac{n}{\log^2 n}$ then consider a game played between any Guesser $\mathcal{G}$ with $m \leq \frac{n}{\log^2 n}$ memory bits and our Move-to-the-Back Dealer $\mathcal{D}_m$ (Definition 5.1.2).

- Assume that all first $n - \frac{n}{8e\log n}$ guesses are reasonable, i.e. while $\mathcal{D}_m$ draws at random, before the Dealer begins moving cards to the back. By linearity of expectation, the expected number of correct guesses in these rounds is:

$$\frac{1}{n} + \frac{1}{n-1} + ... + \frac{1}{\frac{n}{8e\log n}} \approx \ln n - \ln \frac{n}{8e\log n} = \ln\log n + \ln 8e.$$

- By Lemma 5.6.1, every Guesser $\mathcal{G}$ with m memory bits is expected to make at most 1 correct guess while the Dealer moves cards to the back, i.e., from turn $n - \frac{n}{8e\log n}$ until turn $n - 2m\log n$.
- Assume that the Guesser guesses reasonably in all remaining $2m\log n$ turns. These guesses yield $\ln 2m\log n = \ln m + \ln\log n + \ln 2$ correct guesses in expectation.

Overall, the number of correct guesses that any Guesser with $m \leq \frac{n}{\log^2 n}$ bits of memory is expected to make when playing against Move-to-the-Back Dealer $\mathcal{D}_m$ is at most $\ln m + 2\ln\log n + \ln 16e$. $\square$

5.7 Universal Move-to-the-Back Dealer

So far, we have configured our Dealer differently according to the amount of memory bits that the Guesser had. Using the building blocks and ideas seen so far in the section, we present a *universal* adaptive Dealer that works well against any Guesser with any amount of memory, without knowing how much memory the Guesser has. Albeit, with a drawback that a Guesser with m bits of memory is expected to make slightly more than $\ln m$ correct guesses. I.e. a Guesser with perfect memory is expected to achieve more then

$$(1 + o(1)) \cdot \ln n + 8\ln\log n + O(1)$$

correct guesses in expectation.

The starting point for the universal Dealer is the same as that of the Move-to-the-Back Dealer (Definition 5.1.2). Similarly, our universal Dealer separates the turns to epochs during which the Dealer follows the MtBE-strategy. However, the epochs will shrink and become shorter as more cards are drawn, and the analysis will be different.

Definition 5.7.1. *The universal Dealer $\mathcal{D}_{\text{universal}}$ plays according to the strategy:*

1. *Shuffle the deck uniformly at random, and draw cards one by one until $\frac{n}{8e\log^2 n}$ cards are left in the deck.*
2. *Play the MtBE-strategy in a sequence of d epochs, where $d = \log_{\log n}\left(\frac{n}{8e\log^6 n}\right)$, such that the ith epoch begins when $k_i = \frac{n}{8e\log^{1+i} n}$ cards are left, i.e. the length of the ith epoch is $\ell_i = k_i(1 - \frac{1}{\log n})$, and during each epoch at most $u_i = \frac{2\ell_i}{\log^2 n}$ cards are moved to the back.*
3. *When $\log^4 n$ cards left, shuffle the deck one last time and draw cards at random.*

We begin our analysis in the same way as we did for the Move-to-the-Back Dealer. We consider the same implementation of the Dealer (Section 5.2), and the same encoding scheme for sets (Section 5.3).

Recall the discussion about the maximal number of cards moved to the back during an epoch, right before Corollary 5.4.5. In that discussion we argued that we may ignore the role of the bound u since it is impossible to make more than $u = \ell$ reasonable guesses. This is not the case for the universal Dealer, as during the ith epoch at most $u_i = \frac{\ell_i}{\log^2 n}$ cards are moved to the back. We restate, without proof, Claim (5.4.4).

Claim 5.7.2 (Restate Claim (5.4.4))**.** *For every Guesser $\mathcal{G}$ with m bits of memory, and every epoch $i \in [d]$, for $\beta < u_i - \max\left\{\frac{8\cdot e\cdot k_1\cdot \ell_i}{n}, m\right\}$, the probability that $\mathcal{G}$ makes more than $\beta + \max\left\{\frac{8\cdot e\cdot k_1\cdot \ell_i}{n}, m\right\}$ reasonable guesses during the ith epoch, is at most*

$$\Pr_{D\in\binom{[n]}{k_1}}\left[R_i \geq \max\left\{\frac{8\cdot e\cdot k_1\cdot \ell_i}{n}, m\right\} + \beta\right] \leq 2^{-\beta}.$$

As the universal Dealer begins following the MtBE-strategy when $\frac{n}{8e\log^2 n}$ cards are left, we get that the probability for more than $\max\{\frac{\ell_i}{\log^2 n}, m\}$ reasonable guesses decays exponentially. It follows that the expected number of reasonable guesses per epoch depends on the amount of memory the Guesser has. In particular, this claim clarifies that we must analyze differently the epochs for which $m \geq \frac{\ell_i}{\log^2 n}$ than the other epochs.

Therefor, for every m, we separate the epochs into two eras. During the *low-memory era*, moving cards to the back works in the Dealer's favor as the MtBE-strategy guarantees that no Guesser can guess well. During the *high-memory era*, moving cards to the back works in the Guesser's favor, and we assume that the Guesser gains the maximal advantage from it.

Corollary 5.7.3. *During the low-memory era, that is for $i \in [d]$ and $m < \frac{\ell_i}{\log^2 n}$, any Guesser with m bits of memory, makes in expectation at most*

$$r_i \leq \frac{\ell_i}{\log^2 n} + 3$$

reasonable guesses during the (k_i, ℓ_i)-epoch.

Proof. Observe that since $k_1 = \frac{n}{8e\log^2 n}$ then $\frac{8\cdot e\cdot k_1 \ell_i}{n} = \frac{\ell_i}{\log^2 n}$ and by assumption $m < \frac{\ell_i}{\log^2 n}$ so it follows that $\max\left\{\frac{8\cdot e\cdot k_1\cdot \ell_i}{n}, m\right\} = \frac{\ell_i}{\log^2 n}$. To ease the analysis, we assume that the first $\frac{\ell_i}{\log^2 n}$ guesses in the epoch are reasonable. It follows that

$$\begin{aligned} r_i &= \sum_{\alpha=0}^{\ell_i} \alpha \cdot \Pr\left[R_i = \alpha\right] \\ &= \sum_{\beta=0}^{\ell_i - \frac{\ell_i}{\log^2 n}} \left(\frac{\ell_i}{\log^2 n} + \beta\right) \cdot \Pr\left[R_i = \frac{\ell_i}{\log^2 n} + \beta\right] \\ &= \frac{\ell_i}{\log^2 n} + \underbrace{\sum_{\beta=0}^{u_i - \frac{\ell_i}{\log^2 n}} \beta \cdot \Pr\left[R_i = \frac{\ell_i}{\log^2 n} + \beta\right]}_{(*)} + \underbrace{\sum_{\beta=u_i - \frac{\ell_i}{\log^2 n}+1}^{\ell_i - \frac{\ell_i}{\log^2 n}} \beta \cdot \Pr\left[R_i = \frac{\ell_i}{\log^2 n} + \beta\right]}_{(**)} \end{aligned}$$

From Claim (5.7.2), we know that the probability for reasonable guesses decays exponentially until $\beta \le u_i - \frac{\ell_i}{\log^2 n}$, so the first sum $(*)$ is upper bounded by

$$(*) < \sum_{\beta=0}^{\infty} \beta \cdot 2^{-\beta} = 2.$$

The shortest epoch consists of more than $\log^4 n$ turns, and since $u_i = 2 \cdot \frac{\ell_i}{\log^2 n}$, it follows that

$$u_i - \frac{\ell_i}{\log^2 n} = \frac{\ell_i}{\log^2 n} \ge \log^2 n.$$

From Claim (5.7.2) we get that the probabilities in the second sum $(**)$ are upper bounded by $2^{-\log^2 n} = \frac{1}{n^{\log n}}$. Which means that the second sum $(**)$ is upper bounded

$$(**) < \frac{\ell_i \cdot \ell_i}{n^{\log n}} < 1$$

. □

From Lemma 5.6.1 we conclude an upper bound on the expected number of correct guesses during an epoch in the low-memory era.

Corollary 5.7.4. *During the low-memory era, that is for $i \in [d]$ and $m < \frac{\ell_i}{\log^2 n}$, any Guesser with m bits of memory, makes on expectation at most*

$$c_i \le \frac{1}{\log(n) - 2}$$

correct guesses during the (k_i, ℓ_i)-epoch.

Proof. Observe that for any (k_i, ℓ_i)-epoch it holds that $k_i - \ell_i = \frac{k_i}{\log n}$ and since $\ell_i = k_i(1 - \frac{1}{\log n})$ then $k_i - \ell_i = \frac{\ell_i}{\log(n)-1}$. We get that

$$\begin{aligned} c_i &\le \frac{r_i}{k_i - \ell_i - u_i} && (5.13)\\ &= \frac{1}{\log^2 n} \cdot \frac{\ell_i}{k_i - \ell_i - u_i} && (5.14)\\ &= \frac{1}{\log^2 n} \cdot \frac{\ell_i}{\frac{\ell_i}{\log(n)-1} - \frac{2\ell_i}{\log^2 n}} && (5.15)\\ &= \frac{1}{\log^2 n} \cdot \frac{1}{\frac{1}{\log(n)-1} - \frac{2}{\log^2 n}}\\ &< \frac{1}{\log(n) - 2} \end{aligned}$$

Where Inequality (5.13) follows from Lemma 5.5.1, Inequality (5.14) is true since for $m < \frac{\ell_i}{\log^2 n}$ we get from Corollary 5.7.3 that $r_i \le \frac{\ell_i}{\log^2 n}$, and Inequality (5.15) is true since $k_i - \ell_i = \frac{\ell_i}{\log(n)-1}$ and $u_i = \frac{2\ell_i}{\log^2 n}$. □

The above corollary states that the MtBE-strategy works well while the Guesser has insufficient memory. However, as mentioned already, once the Guesser reaches the maximal number of cards moved to the back, the MtBE-strategy works in the Guesser's favor. We want to bound the Guesser's benefit during such an epoch.

Claim 5.7.5. *For every epoch, the expected number of correct guesses that any Guesser makes, is at most*

$$\ln\left(\log(n) + 3\right).$$

Proof. Let $n - t$ be some turn during the ith epoch (k_i, ℓ_i)-epoch. As we did in the proof of Lemma 5.5.1, since at most u_i cards can be moved to the back, it follows that the probability for a reasonable guess to be correct at turn $n - t$ is at most $\frac{1}{t-u_i}$. It follows that if all guesses in the epoch are reasonable then the expected number of correct guesses is

$$\sum_{t=k_i}^{k_i-\ell_i} \frac{1}{t-u_i} \approx \ln\left(k_i - u_i\right) - \ln\left(k_i - \ell_i - u_i\right) = \ln\left(\frac{k_i - u_i}{k_i - \ell_i - u_i}\right).$$

We bound the term inside the ln.

$$\begin{aligned}
\frac{k_i - u_i}{k_i - \ell_i - u_i} &= \frac{k_i - \frac{2\ell_i}{\log^2 n}}{\frac{k_i}{\log n} - \frac{2\ell_i}{\log^2 n}} && (5.16)\\
&< \frac{k_i - \frac{2k_i}{\log^2 n}}{\frac{k_i}{\log n} - \frac{2k_i}{\log^2 n}} && (5.17)\\
&= \log(n) + 2 + \frac{2}{\log(n) + 2} && (5.18)\\
&< \log(n) + 3.
\end{aligned}$$

Where equality (5.16) is true since $k_i - \ell_i = \frac{k_i}{\log n}$ and since $u_i = \frac{2\ell_i}{\log^2 n}$, inequality (5.17) is true since if $a > b$ and $c < d < b$ then $\frac{a-c}{b-c} < \frac{a-d}{b-d}$, and equality (5.18) is the result of division.

It follows that the expected number of correct guesses during any epoch is upper bounded by $\ln\left(\log(n) + 3\right)$. □

We therefore have two upper bounds on the number of correct guesses, one for the low-memory era (Corollary 5.7.4) and a general one (Claim (5.7.5)) that we will use for epochs during the high-memory era.

Theorem 5.7.6. *There exists an adaptive universal Dealer against which any Guesser with m bits of memory can score at most*

$$(1 + o(1)) \cdot \ln m + 8 \ln \log n + O(1)$$

correct guesses in expectation.

Proof. Consider our universal Dealer from Definition 5.7.1. Assume that all the guesses before turn $n - k_1 = n - \frac{n}{8e \log^2 n}$ were reasonable, this results in $2 \ln \log n + \ln 8e$ correct guesses.

Low-memory era: Recall that the universal Dealer plays the MtBE-strategy for $d = \log_{\log n}\left(\frac{n}{8e\log^6 n}\right)$ epochs. Observe that $d \le \log_{\log n} n = \frac{\log n}{\log\log n}$. Let d_1 be the number of epochs for which the Guesser has insufficient memory, i.e., $d_1 = |\{i \in [d] \colon m \le \frac{\ell_i}{\log^2 n}\}|$. Corollary 5.7.4 states that the expected number of correct guesses during an epoch in the low-memory era is at most $\frac{1}{\log(n)-2}$. It follows that the cumulative number of correct guesses during these epochs is less than

$$\begin{aligned} d_1 \cdot \frac{1}{\log(n)-2} &\le d \cdot \frac{1}{\log(n)-2} \\ &< \frac{\log n}{\log\log n} \cdot \frac{1}{\log(n)-2} \\ &< 1. \end{aligned}$$

High-memory era: Let $d_2 = d - d_1$ be the number of epochs in the high-memory era. Observe that $m \ge \frac{\ell_i}{\log^2 n}$ for every epoch for which $k_i \le m \cdot \log^2 n$. Therefore,

$$d_2 \le \log_{\log n}\left(m \cdot \log^2 n\right) = \frac{\ln m}{\ln\log n} + 2.$$

Claim (5.7.5) states that the expected number of correct guesses during any epoch is upper bounded by $\ln(\log(n)+3)$. It follows that total number of correct guesses during the high-memory era is

$$\begin{aligned} d_2 \cdot \ln(\log(n)+3) &\le \left(\frac{\ln m}{\ln\log n} + 2\right) \cdot \ln(\log(n)+3) \\ &= \ln m \cdot \frac{\ln(\log(n)+3)}{\ln\log n} + 2\ln(\log(n)+3) \\ &= \ln m \cdot (1+o(1)) + 2\ln(\log(n)+3). \end{aligned}$$

Assume the Guesser guesses reasonably the last $\log^4 n$ turns, the expected number of correct guesses is at most $4\ln\log n$.

Summing it all together, we get that the expected number of correct guesses that any Guesser with m bits of memory can score against our universal Dealer is at most

$$(1+o(1)) \cdot \ln m + 8\ln\log n + O(1).$$

□

6 Discussion and open problems

6.1 Relation to Mirror Game

In this section we discuss the relation between Card Guessing and Mirror Game. Recall the Mirror Game presented by Garg and Schneider [14], where Alice (the first player) and Bob take turns saying a name of a card (i.e. a number) from a deck of size $2n$, and a player loses if this card was mentioned already by either one of the players. When no more cards are left to say, then the result of the game is a draw. Bob, who plays second, has a *low memory, simple, and efficient strategy called mirroring*: Bob fixes any matching on the cards, and for every card said by Alice, Bob responds with the matched card. This allows Bob to say in every turn a card that has not appeared yet, and in our terms, to yield a reasonable guess, and *is assured not to lose.*

The question at hand is how much memory Alice needs in order not to lose (i.e. assure a draw). Garg and Schneider [14] showed that every deterministic "winning" (drawing) strategy for Alice requires space that is linear in n. They also showed a randomized strategy that draws with high probability $(1 - \frac{1}{n})$ and requires $O(\sqrt{n})$ bits of memory while relying on access to a secret random matching oracle. Using a similar setting (with respect to the secret matching), Feige [12] showed a randomized strategy for Alice that requires only $O(\log^3 n)$ bits of memory. In fact, as we will show soon, Alice can supply her own matching while being *computationally efficient* and using $O(n \log n)$ bits of long lived randomness to produce reasonable response (or alternatively using cryptography and a small amount of long lived randomness while assuming computational limitation (poly time) on Bob).

This stands in contrast to our impossibility result on the adaptive Dealer: In Section 5, we have bounded the number of reasonable guesses by the Guesser's memory, regardless of run time, how much randomness she holds, and what cryptography she uses. We present here a simplified computationally efficient version of Feige's construction that requires no access to a secret random matching but requires long lived randomness with random access for efficiency.

Our first point is that it is possible to construct a secret matching from more standard assumptions (long lived random bits or cryptographic ones). Ristenpart and Yilek [29] and Morris and Rogaway [23] showed a transformation from (pseudo)random functions to (pseudo)random permutations that is secure *even if all the permutation is given to the distinguisher.* Applying their constructions in our setting means using $O(n \log n)$ random bits (with random access) to construct a random permutation π so that given x it is possible to evaluate $\pi(x)$ on the fly (same for $\pi^{-1}(x)$), simply by looking at the randomness in $O(\log n)$ places and using $O(\log n)$ memory bits for intermediate calculations. Naor and Reingold [26] showed a construction that takes a permutation and its inverse and produces a permutation with any desired cycle structure[5]. In particular, we can turn π to an involution, i.e., a matching[6]. Therefore combining these two we get that Alice can have a secret matching provided she has either (i) $O(n \log n)$ secret random bits with random access or (ii) A key to a pseudorandom function and Bob is computationally limited and cannot distinguish the results from random[7].

Having a secret matching with the above properties we discuss how to use the machinery of Sections 4.2 and 4.2.1 in order to suggest a strategy for Alice. Alice will use her secret matching to imitate the mirror strategy of Bob, with an arbitrary starting point. However, from time to time she will fail in that Bob will select as a response the matched card of the starting point, leaving her with no obvious response. What she should do at this point is select a card that has not appeared yet as a new starting point. For this, she needs the moral equivalent of a reasonable guess, and in this setting, any reasonable guess is good. How many times do we expect this to happen? This is similar to card guessing with perfect memory, i.e. $\ln n$ times. Which means that she needs at least that many reasonable answers.

So in more detail, Alice uses her memory to allocate the $2n$ cards to subsets, similarly to what we did in Sections 4.2 and 4.2.1. She uses her long lived randomness to sample $O(\log^2 n)$ permutations

[5]Naor and Reingold took a permutation π, and a permutation with the structure of choice σ and returned $\pi^{-1} \circ \sigma \circ \pi$.

[6]For example, by taking σ to be the involution $\sigma(1) = 2, \sigma(3) = 4, \ldots$. The construction also gives each pair an "index" in $[n]$ that is retrievable from either member of the pair.

[7]Existentially, this is is equivalent to one-way functions.

from a family of pairwise independent permutations and splits the functions to $\log n$ collections of equal size. From each function in the jth collection, Alice produces a set of size 2^{j-1}. Resulting in a collection of sets of equal size. In detail, for every function h in the jth collection, Alice assigns the card x to the subset S_j^h if $h(x) \in \{2^{j-1}+1, \ldots, 2^j\}$. For each subset, Alice tracks the number of cards that appeared from that subset and their sum, exactly as we did for our Randomized-Subset.

As her first card, Alice chooses an arbitrary card g_1, announces g_1 as her choice and stores it in her memory. When Bob says a card x, Alice responds with $M(x)$, where M is her secret matching. She does that until Bob says $M(g_1)$. Alice cannot say $M(M(g_1)) = g_1$, as this would result in her losing. So instead, she attempts to recover a *reasonable response* from her memory. If the recovering attempt succeeds and she retrieves a card g_2, then she announces g_2 as her choice and stores it in her memory in place of g_1. She continues until Bob says $M(g_2)$ and so forth.

So we get that Alice loses the game only in case she fails to recover a reasonable card when Bob makes a correct guess. For the first half of the game, Alice stores $O(\log n)$ singletons, cards that have not appeared (i.e. subsets of size 1), and takes one of them that hasn't been declared yet. When $t \leq n/2$ turns are left for Alice, she makes a recovery attempt from the jth collection of subsets for $j = \log\lfloor n/2t \rfloor$. She checks the subsets, in some fixed order, until she finds one that yields a reasonable response. For the sake of analysis, assume that Alice accesses each subset exactly once. Afterward, that subset is deleted and ignored for the rest of the game, regardless of whether a recovery occurred.

Alice attempts to recover a reasonable guess whenever Bob guesses correctly. There is a limited number of attempts she can make, and each attempt succeeds only with some probability. So we first bound the probability that Bob makes too many correct guesses, then we bound the probability that Alice succeeds in making that many recovery attempts.

Let x_t be the indicator random variable for the event that Bob made a correct guess when t turns are left. As Alice's matching is random, and since all of Bob's guesses are reasonable, then when Bob has t turns left, the probability that Bob guesses correctly is $\frac{1}{2t-1}$. For every $j \leq n$, consider the span of $j/2$ turns that begins when Bob has j turns left. Since the matching is random, then any pair is independent of the remaining pairs. That is, figuring a single match tells nothing about the remaining pairs, so we get that every x_t, is independent of all previous ones. Let x be the number of correct guesses made by Bob in the corresponding period, i.e., $x = \sum_{t=j}^{j/2} x_t$. By linearity of expectation, the expected number of times that Bob guesses correctly is at most

$$\mu = \mathbb{E}[x] \approx 1/2 \ln j - 1/2 \ln j/2 = 1/2 \ln 2.$$

What is the probability that $x \geq 2 \log n$? By a Chernoff bound (Theorem 4.4 (3) in [22]), the probability to make more than $2 \log n$ correct guesses, i.e., $4 \log(n)$ times more than the expectation, is less then

$$\Pr\left[x \geq 2 \log n\right] \leq 2^{-2 \log n} = n^{-2}.$$

So we get that Bob makes more than $2 \log n$ correct guesses in a relatively small probability.

Alice may still lose if she fails to supply sufficiently many reasonable answers to Bob's successful guesses. We will show that this also happens with small probability. Recall that Alice samples functions independently and produces a subset from each function. Assume that for every j (and also for first half of the game), Alice samples $32 \log n$ functions. Let z_i be an indicator random variable for the event that the ith subset yields a reasonable answer. From Inequality (4.2), we know that each subset yields a card that hasn't been played with probability of at least $1/4$. We compare the probability that Alice runs out of functions to the process where we have $\{0, 1\}$ random variables $y_1, y_2, \ldots y_{32\log n}$ where each y_i is independently chosen with probability exactly 1/4, and the probability of interest is that there are less than $2\log n$ 1's. Take any configuration of the $z_1, \ldots, z_{32\log n}$ and take any configuration of the $y_1, \ldots, y_{32\log n}$ that is covered (or dominated) by the z_i's configuration in the sense that $z_i = 0$ implies $y_i = 0$. Then the probability for the configuration of the y_i's is larger than that of the z_i's, as the y_i are at least as likely to yield 0's. As a result, upper bounding the probability of less than $2\log n$ 1's of y_is (more 0's) upper bounds the probability of less than $2\log n$ 1's of z_is, and so, for Alice running out of reasonable answers. Let $y = \sum_{i=1}^{32\log n} y_i$. By linearity of expectation

$$\mu = \mathbb{E}[y] = 0.25 \cdot 32\log n = 8 \cdot \log n.$$

By a Chernoff bound (Theorem 4.5 (2) in [22]), for $\delta = 3/4$, the probability for this event is at most

$$\Pr\left[y < (1-\delta)\mu\right] \le e^{-\frac{\delta^2\mu}{2}} = e^{-\frac{9\cdot 8\cdot\log n}{32}} < n^{-2}.$$

Considering the span of turns at which Alice queries the jth collection of sets. We get that with probability at most n^{-2} Bob makes too many correct guesses during that period, and with probability at most $(1-n^{-2})n^{-2}$ Alice fails to produce sufficiently many reasonable responses at that period. It follows that Alice loses while she considers a specific j with probability at most $n^{-2} + (1-n^{-2})n^{-2} < 2n^{-2}$. By the union bound no failure occurred during any of the $\log n$ periods with probability at most $2\log n/n^4$, i.e., Alice draws (or wins) with probability at least

$$1 - \frac{2\log n}{n^2} > 1 - \frac{1}{n}.$$

We conclude that with $O(\log^3 n)$ bits of memory and $O(\log^3 n + n\log n)$ bits of long lasting randomness, Alice has a strategy against Bob, that draws or wins with probability at least $1 - \frac{1}{n}$. If we wish to go use computational assumptions, then Alice needs only to store a key to a pseudo-random function (and assume that Bob is computationally bounded, i.e. cannot distinguish between the PRF and a truly random function) and we get the desired result.

Question. *Does there exist an algorithm for Alice that requires at most polylog long lived bits of randomness and no cryptographic assumptions and gives her a reasonable chance of not losing?*

Question. *Is the $\Omega(\log^2 n)$ lower bound of Section 4.3 relevant for this setting as well?*

6.2 Card Guessing variants

Consider a the Card Guessing game with a deck that contains c copies of each card. Diaconis and Graham [9] showed that the optimal strategy against a Random Static Dealer scores $c + \theta(\sqrt{c\log n})$ correct guesses in expectation. This is achieved by tracking all cards that appeared so far and guess the one with the highest probability, which can be easily done with $n\log c$ memory bits. By tracking the deviation from the expected number of cards seen so far, it may be possible, in some cases, to achieve a slightly better memory consumption for the static case.

Note that the low-memory Guessers from Section 4 may still work if we assign all copies of a card to the same subset, however, the performance of these Guessers remains $O(\ln n)$. For $c > \ln n$ these techniques are far from optimal. In fact, with no memory at all, it is possible to get c correct guesses simply by repeating the same guess over and over again. So we get that for $c \ge \ln n$ our low memory Guessers perform worse than a Guesser with no memory at all.

Question. *How much memory and randomness does a Guesser requires to score "near-optimal" results against a Dealer with a deck containing multiple copies of each card? Is there a difference between the different kinds of Dealers?*

As for other variants of Card Guessing, the literature considers Card Guessing with partial feedback (was the guess correct or not) and no feedback at all. The optimal [10] and near-optimal [8] guessing strategies for partial feedback requires little to no memory, and so goes for the optimal strategy for no feedback [9]. We suggest the study of a general theory of when we can convert a feedback type into a low memory guessing.

6.3 Low Memory Dealer: a Conjecture

What happens when the Dealer has limited memory, say m bits, and wants to pick a permutation that is unpredictable by any Guesser, that has no limitation on the number of bits it can store or its computational power. The dealer also has a limited number of long lived random bits. It *seems* that the best such a Dealer can do is pick the next card from a set of m cards at random, and the question is how to assure that there is such a set available for as many rounds as possible. We have found such a method that makes any Guesser pick correctly only $O(n/m + \log m)$ cards in expectation. The method does not require any secrecy from the dealer. We conjecture that this bound is the best possible, at least for dealers without any secret memory.

6.4 Prediction as Approximation and Data Structures

In the streaming model of computation an algorithm observes a stream of elements and computes a function on the stream seen so far. For a memory bounded algorithm, it is a typical relaxation that the algorithm outputs an approximate value of the function. In card guessing, we ask the algorithm to predict the next card, but this prediction can be though of as a way to measure distance, and thus, as an approximation.

Consider a partial game played for some turns between some Guesser and the random-shuffle Dealer. Let $\vec{g}$ be an n-vector associated with the probability for each card to be guessed by the Guesser at that turn. Let D denote the set of cards that are still in the deck, and let $\vec{d} = \mathbb{1}_D \cdot \frac{1}{|D|}$ be the vector associated with the probability to draw each one of them at random. In general, the inner product between two normalized vectors indicates how close they are. Intuitively, if $\vec{g}$ and $\vec{d}$ are close, it means that the Guesser captured more accurately the set of cards that are still in the deck. So if the inner product of $\vec{g}$ and $\vec{d}$ is bounded, we can say that $\vec{g}$ approximate $\vec{d}$.

Now consider the chance of a correct guess. The probability to guess correctly is the probability that both the Guesser and the Dealer sampled the same card independently. As the vectors $\vec{g}$ and $\vec{d}$ represents the probability for each card to be drawn, we get that the probability for a correct guess is the inner product between $\vec{g}$ and $\vec{d}$.

$$\Pr[\text{correct guess}] = \sum_{x\in[n]} \vec{g}_x \cdot \vec{d}_x = \langle \vec{g}\,, \vec{d} \rangle$$

The idea is visualized in Table 2.

Similarly, we can think about the Guesser as holding a set-membership data structure. The Guesser guesses a card from the set of elements for which the data structure claims that are absent from the set. The prediction remains a way to measure performance. On that aspect, our work

Table 2: Each card belong to one of the four groups. The probability to guess correctly is the inner product between the vertices in vertical circle and the vertices in the horizontal one.

Card drawn / Guesser thinks the card drawn	Yes	No
Yes		Will not guess
No	Futile guess	Reasonable guess

joins that of Naor and Yogev [27] who studied the ability of an adversary to find a False Positive in a Bloom Filter, i.e., to find an element that does not reside in the set though the Bloom Filter thinks it does. In particular, they considered the advantage of the adversary to find a False-Positive, and in our terms, to upper bound the probability to guess reasonably.

6.5 Security

Unpredictability is a goal of many security mechanisms to ensure secure and reliable operation of services and authentication. Security measures are taken to increase the unpredictability of critical services and protocols.

As an example, in TCP, the source port, the sequence number and the ack number are initially randomized in order to decrease the probability of hijacking the session. Being able to predict these numbers increases the probability of a successful attack on the server in the form of connection hijack or denial of service. As the amount of source ports is finite, by opening many connections to a server it may be possible to get a better prediction the next source port to be used without actually observing all cards. In this aspect, it is interesting whether a similar approach to Move-to-the-Back may be beneficial for protecting services against such attacks.

Acknowledgments

We thank Eylon Yogev and Yotam Dikstein for many suggestions and advice. We thank Hila Dahari, Uri Feige, Tomer Grossman, and Adi Schindler for meaningful discussions and insights. We also thank Gal Vinograd for reading a preliminary version of this document.

References

[1] Noga Alon, Omri Ben-Eliezer, Yuval Dagan, Shay Moran, Moni Naor, and Eylon Yogev. *Adversarial Laws of Large Numbers and Optimal Regret in Online Classification*. 2021. arXiv: `2101.09054`.

[2] Shai Ben-David, Allan Borodin, Richard M. Karp, Gábor Tardos, and Avi Wigderson. *On the Power of Randomization in On-Line Algorithms*. Algorithmica 11.1 (1994), pp. 2–14. DOI: `10.1007/BF01294260`. URL: `https://doi.org/10.1007/BF01294260`.

[3] Omri Ben-Eliezer and Eylon Yogev. *The Adversarial Robustness of Sampling*. Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems. PODS'20. 2020, 49–62.

[4] Omri Ben-Eliezer, Rajesh Jayaram, David Woodruff, and Eylon Yogev. *A Framework for Adversarially Robust Streaming Algorithms.* PODS'20. 2020, pp. 63–80.

[5] Allan Borodin and Ran El-Yaniv. ***Online computation and competitive analysis***. Cambridge University Press, 1998.

[6] Kai-Min Chung, Tai-Ning Liao, and Luowen Qian. *Lower Bounds for Function Inversion with Quantum Advice.* 1st Conference on Information-Theoretic Cryptography (ITC 2020). Vol. 163. 2020, 8:1–8:15.

[7] Thomas M. Cover and Joy A. Thomas. ***Elements of Information Theory*** *2nd Edition.* Wiley, 2006.

[8] Persi Diaconis, Ron Graham, and Sam Spiro. *Guessing about Guessing: Practical Strategies for Card Guessing with Feedback* (2020). arXiv: `2012.04019`.

[9] Persi Diaconis and Ronald Graham. *The Analysis of Sequential Experiments with Feedback to Subjects.* The Annals of Statistics 9 (1) (Jan. 1981).

[10] Persi Diaconis, Ron Graham, Xiaoyu He, and Sam Spiro. *Card Guessing with Partial Feedback* (Oct. 2020). arXiv: 2010.05059. arXiv: `2010.05059`.

[11] Itai Dinur. *On the Streaming Indistinguishability of a Random Permutation and a Random Function.* Advances in Cryptology – EUROCRYPT 2020. Lecture Notes in Computer Science. 2020, pp. 433–460.

[12] Uriel Feige. *A randomized strategy in the mirror game.* 2019. arXiv: `1901.07809 [cs.DS]`.

[13] Ronald Fisher. *A Method of Scoring Coincidences in Tests with Playing Cards.* Proceedings of the Society for Psychical Research Volume XXXIV. Society of Psychical Research, July 1924, 181–185. URL: `http://iapsop.com/archive/materials/spr_proceedings/spr_proceedings_v34_1924.pdf`.

[14] Sumegha Garg and Jon Schneider. *The Space Complexity of Mirror Games.* 10th Innovations in Theoretical Computer Science Conference (ITCS 2019). Vol. 124. 2018, 36:1–36:14.

[15] Rosario Gennaro and Luca Trevisan. *Lower bounds on the efficiency of generic cryptographic constructions.* Proceedings 41st Annual Symposium on Foundations of Computer Science (FOCS '00). 2000, pp. 305–313.

[16] Moritz Hardt and David P. Woodruff. *How Robust Are Linear Sketches to Adaptive Inputs?* Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing. STOC '13. 2013, 121–130.

[17] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. *Adversarially Robust Streaming Algorithms via Differential Privacy.* NeurIPS 2020. 2020.

[18] Piotr Indyk. *A Small Approximately Min-Wise Independent Family of Hash Functions.* Journal of Algorithms 38.1 (2001), pp. 84–90.

[19] Joseph Jaeger and Stefano Tessaro. *Tight Time-Memory Trade-Offs for Symmetric Encryption.* Advances in Cryptology – EUROCRYPT 2019. Lecture Notes in Computer Science. 2019, pp. 467–497.

[20] Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. *Separating Adaptive Streaming from Oblivious Streaming.* 2021. arXiv: `2101.10836`.

[21] Jon Kleinberg and Éva Tardos. ***Algorithm Design***. Pearson, 2006.

[22] Michael Mitzenmacher and Eli Upfal. ***Probability and Computing***. Cambridge University Press, 2017.

[23] Ben Morris and Phillip Rogaway. *Sometimes-Recurse Shuffle*. Advances in Cryptology – EUROCRYPT 2014. Vol. 8441. 2014, pp. 311–326.

[24] Robin A. Moser and Gábor Tardos. *A Constructive Proof of the General Lovász Local Lemma*. J. ACM 57.2 (2010).

[25] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Foundations and Trends in Theoretical Computer Science 1.2 (2005), pp. 117–236.

[26] Moni Naor and Omer Reingold. *Constructing Pseudo-Random Permutations with a Prescribed Structure*. Journal of Cryptology 15.2 (Jan. 2002), pp. 97–102.

[27] Moni Naor and Eylon Yogev. *Bloom Filters in Adversarial Environments*. ACM Trans. Algorithms 15.3 (June 2019).

[28] Mihai Pătraşcu. *Cuckoo Hashing*. WebDiarios de Motocicleta. Blog post available at `http://infoweekly.blogspot.com/2010/02/cuckoo-hashing.html`. 2010.

[29] Thomas Ristenpart and Scott Yilek. *The Mix-and-Cut Shuffle: Small-Domain Encryption Secure against N Queries*. Advances in Cryptology – CRYPTO 2013. 2013, pp. 392–409.

[30] George Santayana. ***The Life of Reason*** *or The Phases of Human Progress: Introduction and Reason in Common Sense, Volume VII, Book One*. 1905.

[31] Ido Shahaf, Or Ordentlich, and Gil Segev. *An Information-Theoretic Proof of the Streaming Switching Lemma for Symmetric Encryption*. 2020 IEEE International Symposium on Information Theory (ISIT). 2020, pp. 858–863.

[32] Edward O. Thorp. ***Beat the Dealer****: A Winning Strategy for the Game of Twenty-One*. 1962.

[33] Wikipedia. *Card counting — Wikipedia, The Free Encyclopedia*. [Online; accessed 11-June-2021]. 2004. URL: `https://en.wikipedia.org/w/index.php?title=Card_counting&oldid=1016853614`.

[34] David P. Woodruff and Samson Zhou. *Tight Bounds for Adversarially Robust Streams and Sliding Windows via Difference Estimators*. 2020. arXiv: `2011.07471`.