# Query Complexity of Mastermind Variants

Aaron Berger, Christopher Chute, Matthew Stone

*Yale University*

June 26, 2016

**Abstract**

We study variants of Mastermind, a popular board game in which the objective is sequence reconstruction. In this two-player game, the so-called *codemaker* constructs a hidden sequence $H = (h_1, h_2, \ldots, h_n)$ of colors selected from an alphabet $\mathcal{A} = \{1, 2, \ldots, k\}$ (*i.e.,* $h_i \in \mathcal{A}$ for all $i \in \{1, 2, \ldots, n\}$). The game then proceeds in turns, each of which consists of two parts: in turn $t$, the second player (the *codebreaker*) first submits a query sequence $Q_t = (q_1, q_2, \ldots, q_n)$ with $q_i \in \mathcal{A}$ for all $i$, and second receives feedback $\Delta(Q_t, H)$, where $\Delta$ is some agreed-upon function of distance between two sequences with $n$ components. The game terminates when $Q_t = H$, and the codebreaker seeks to end the game in as few turns as possible. Throughout we let $f(n, k)$ denote the smallest integer such that the codebreaker can determine any $H$ in $f(n, k)$ turns. We prove three main results: First, when $H$ is known to be a permutation of $\{1, 2, \ldots, n\}$, we prove that $f(n, n) \geq n - \log\log n$ for all sufficiently large $n$. Second, we show that Knuth's Minimax algorithm identifies any $H$ in at most $nk$ queries. Third, when feedback is not received until all queries have been submitted, we show that $f(n, k) = \Omega(n \log k)$.

## 1 Introduction

Mastermind is a game created by Mordechai Meirowitz in 1970. In the original game, the codebreaker's objective is to guess a hidden sequence of 4 colors, each chosen from a pool of 6. Knuth analyzed the original game in 1977, and showed that there exists a strategy that guarantees guessing the hidden vector in no more than 5 turns [10]. Mastermind is a two-player games centered around reconstruction of a hidden sequence. In all variants of the game, one player is given the role of *codemaker*, and the other is denoted the *codebreaker*. The codemaker begins the game by constructing a hidden sequence $H = (h_1, h_2, \ldots, h_n)$ where each component is selected from an alphabet $\mathcal{A} = \{1, 2, \ldots, k\}$ of $k$ colors (that is, $h_i \in \mathcal{A}$ for all $i \in \{1, 2, \ldots, n\}$). The goal of the codebreaker is to uniquely determine the hidden sequence $H$ through a series of queries, which are submissions of vectors of the form $Q_t = (q_1, q_2, \ldots, q_n)$. The codebreaker always seeks to determine $H$ with as few queries as possible, however the nature of these queries, the feedback received after a query, and the restrictions on $H$ differ between variants. Erdős and Rényi, for example, studied a 2-color version of the game, before Mastermind existed, in [5].

The variants of Mastermind which we study are defined by settings of the tuple $(n, k, \Delta, R, A)$. These parameters are defined as follows:

(i) ($n$) *Length of Sequence.* The parameter $n$ denotes the length of the hidden sequence $H$ created by the codemaker, hence $H = (h_1, h_2, \ldots, h_n)$. The codebreaker is also required to submit query vectors of length $n$, so the $t^{\text{th}}$ query vector takes the form $Q_t = (q_1, q_2, \ldots, q_n)$.

(ii) ($k$) *Size of Alphabet.* This parameter determines the number of possible values for components of $H$ and $Q_t$. Associated with each game is an alphabet $\mathcal{A} = \{1, 2, \ldots, k\}$ from which the components of $H$ and $Q_t$ are selected. That is, $h_i \in \{1, 2, \ldots, k\}$ and $q_i \in \{1, 2, \ldots, k\}$.

(iii) ($\Delta$) *Distance Function.* Each variant of Mastermind has an associated distance function $\Delta$, giving the distance between two sequences of length $n$. For each query sequence $Q_t = (q_1, q_2, \ldots, q_n)$ submitted by the codebreaker, the codemaker gives feedback $\Delta(Q_t, H)$. The information yielded by $\Delta(Q_t, H)$ clearly influences the number of queries needed to identify the hidden vector $H$. For example, if $\Delta$ is defined by $\Delta(Q_t, H) = H$ then trivially the codebreaker can determine $H$ in one query. If $\Delta(Q_t, H) = 0$, then all sequences of length $n$ must be queried to guarantee a win. We study the following distance functions:

a. *"Black-peg and white-peg."* Informally, a black peg denotes "the correct color in the correct spot," and a white peg denotes "the correct color in an incorrect spot." Formally, let $Q_t$ and $H$ be as above. The black-peg and white-peg distance function is defined by $\Delta(Q_t, H) = (b(Q_t, H), w(Q_t, H))$ where

$$b(Q_t, H) = |\{i \in [1, n] \mid q_i = h_i\}|, \tag{1}$$

and

$$w(Q_t, H) = \max_{\sigma} \; b(\sigma(Q_t), H) - b(Q_t, H),$$

where $\sigma$ iterates over all permutations of $Q_t$. We note that this is the distance function used in the original game of Mastermind.

b. *"Black-peg-only."* When $\Delta$ is the black-peg-only distance function, it is defined by $\Delta(Q_t, H) = b(Q_t, H)$, where $b$ is defined as in equation (1).

We will denote the black-peg-only distance function by $\Delta = b$, and the black-white distance function by $\Delta = bw$.

(iv) ($R$) *Repetition.* The parameter $R$ is a Boolean restriction on the components of $H$. If $R$ is true, we say that the variant game is *with repeats* or that repeats are allowed. In this case, the hidden vector $H$ may have repeated colors, that is, we allow $h_i = h_j$ for any $i, j \in \{1, 2, \ldots, n\}$. When $R$ is false, we say that the variant is *no repeats*, and we require $h_i \neq h_j$ when $i \neq j$, and so we necessarily require $k \geq n$. In particular, when $R$ is false and $k = n$, we have that $H$ must be a permutation of $\{1, 2, \ldots, n\}$, and we refer to this variant as the *Permutation Game.*

(v) ($A$) *Adaptiveness.* The Boolean parameter $A$ determines whether the codebreaker receives feedback after each query. If $A$ is true, we say that the game is *adaptive.* In this case the game consists of two-part turns. On the $t^{\text{th}}$ turn, the codebreaker first submits a query sequence $Q_t$, and then receives feedback $\Delta(Q_t, H)$. The codebreaker may use the feedback to inform the query $Q_{t+1}$ made in turn $t + 1$, and the game ends in turn $s$ if and only if $Q_s = H$.

When $A$ is false, we say that the game is *non-adaptive.* In this case the codebreaker submits $m$ queries $Q_1, Q_2, \ldots, Q_m$ all at once (where the codebreaker chooses $m$). The codemaker then reports a feedback vector of the form $(\Delta(Q_1, H), \Delta(Q_2, H), \ldots, \Delta(Q_m, H))$, after which the codebreaker must submit the final query $\overline{Q}$. The codebreaker wins if and only if $\overline{Q} = H$.

Throughout we define $f(n, k, \Delta, R, A)$ to be the smallest integer such that the codebreaker can determine any hidden sequence $H$ in $f(n, k, \Delta, R, A)$ queries during a game with the corresponding

assignment of $n$, $k$, $\Delta$, $R$, and $A$. We will denote true and false by $T$ and $F$, respectively, for assignment of Boolean variables. For example, Donald Knuth's result that the original game of Mastermind (four positions, six colors, black-peg and white-peg, with repeats, and adaptive) can always be determined after four turns (and then guessed on the fifth turn) is equivalently stated as $f(4, 6, \Delta = (b, w), R = T, A = T) \leq 4$ (in fact, this bound holds with equality) [10]. The case $R = T, A = T$ is the most extensively studied in the literature [2, 3, 7, 10]. Doerr, Spöhel, Thomas, and Winzen obtain many significant results in the case $R = T$ in [4], applying techniques from [1, 8].

We focus only on analyzing the worst-case performance of query strategies for these variants of Mastermind. That is, we always consider the number of queries necessary to guarantee identification of any hidden vector.

Our first main result concerns the Permutation Game, in which $n = k$ and $R = F$, thus restricting the hidden sequence $H$ to be a permutation of the alphabet $\mathcal{A}$.

**Theorem 1.** *Consider the Permutation Game defined by $n = k$ and $R = F$. Let $\Delta = b$. Then for all sufficiently large $n$, we have*

$$f(n, k = n, \Delta = b, R = F, A) \geq n - \log\log n.$$

Explicit algorithms that take $O(n \log n)$ turns to solve this variant were developed by Ko and Teng in [11], and El Ouali and Sauerland in [12]. Ko and Teng approach the problem with an algorithm akin to binary search. The algorithm queries a sequence $Q_t$, swaps two components, and queries again, doing so repeatedly until a previously unknown component of $H$ is determined by the values of the distance function across these repeated queries. El Ouali and Sauerland improve this algorithm primarily by altering the search routine after a single component of $H$ has been identified. They also extend their results to variants with $k \geq n$. In this way, they achieve an average factor of two reduction in the number of queries needed to identify $H$. In our notation, these results state that $f(n, k = n, \Delta = b, R = F, A = T) \leq O(n \log n)$.

Via a basic information-theoretic argument, one can show that the Permutation Game satisfies $f(n, n) \geq n - n/\log n + c$ for some constant $c > 0$. We improve this lower bound to $f(n, n) \geq n - \log\log n$ for sufficiently large $n$. To our knowledge, this constitutes the first improvement over the trivial information-theoretic lower bound for the Permutation Game variant of Mastermind.

Our second result concerns Donald Knuth's Minimax algorithm, which was first introduced in 1976.

**Theorem 2.** *Consider the original variant of Mastermind (allowing arbitrary $n$ and $k$), defined by $\Delta = (b, w)$, $R = T$, $A = T$. The Minimax algorithm identifies any hidden sequence $H$ in at most $nk$ queries.*

Knuth's Minimax algorithm is empirically near-optimal for solving small games of Mastermind ($n$ and $k$ less than 10) in as few guesses as possible. However, it has proven difficult to analyze the asymptotic performance of the Minimax algorithm, primarily because its behavior is determined by the distribution remaining solutions aftera series of guesses, which is difficult to analyze in general [12, 11]. To our knowledge, this is the first upper bound on the worst-case performance of the minimax algorithm, but if it performs near-optimally for large $n$ and $k$, we would expect this bound to be much smaller. We know, for example, that $f(n, k) = O(n \log k)$ for $k$ not too large [10, 11].

Our third result relates to non-adaptive variants of Mastermind, which correspond to $A = F$ in our notation.

**Theorem 3.** *Consider non-adaptive Mastermind, defined by $A = F$. Let $\Delta = b$, and let $k \geq n$. Then for either choice of $R$, we have*

$$f(n, k) = \Omega\left(n \log(k)\right).$$

The case $R = T$ was proved by Doerr, Spöhel, Thomas, and Winzen in [4]. They use an information-theoretic argument based on Shannon entropy by letting the hidden sequence be randomly chosen. Shannon entropy is discussed briefly in Section 4.1. We use a similar argument to extend this to $R = F$, and these lower bounds together cover the case $n \geq k$. For neither choice of $R$ do we have provably optimal upper bounds; when $R = T$ a corollary of a result in [4] gives an upper bound of $O(k \log k)$ guesses, and for $R = F$ no improvement over the $nk$ bound is known. On the other hand, the authors of [4] are able to extend a result of Chvátal in [3] to provide tight bounds for $n \leq k$ with $R = T$.

### 1.1 Structure of the Paper

In Section 2 we prove Theorem 1, and the proof is found in 2.2. In Section 3 we prove Theorem 2, and in Section 4 we discuss Theorem 3, with the proof in 4.2.

## 2 Adaptive Variants of Mastermind

### 2.1 The Permutation Game

We introduce notation and reasoning used in the proof of Theorem 1. We recall that this deals with the Permutation Game, defined by $n = k$ and $R = F$. We can first show that the information from white-peg responses is irrelevant. These parameters together imply $H = \overline{\sigma}(\mathcal{A})$ for some permutation $\overline{\sigma} \in S_k$. We recall the definition of the white-peg distance function as

$$w(Q_t, H) = \max_{\sigma \in S_{Q_t}} b(\sigma(Q_t), H) - b(Q_t, H).$$

Letting $Q_t = \tau(\mathcal{A})$ for some $\tau \in S_k$, we have that with $\sigma = \overline{\sigma} \circ \tau^{-1}$. So we have $\sigma(Q_t) = \overline{\sigma} \circ \tau^{-1} \circ \tau(\mathcal{A})$ which is just $H$, and then $b(\sigma(Q_t), H) = b(H, H)$ achieves the maximum possible value of $n$. Then $w(Q_t, H) = n - b(Q_t, H)$ and is uniquely determined by $b(Q_t, H)$ and so no new information is provided from white-peg responses. As such, for the permutation game we will let $\Delta = b$, as $\Delta = bw$ is precisely the same game.

As stated in the introduction, an upper bound $f(n, n) \leq c \cdot n \log n$ was proved in 1986 by Ko and Teng, and in 2013 El Ouali and Sauerland improved this constant [11, 12]. We establish lower bounds for $f(n, n) = f(n, k = n, \Delta = b, R = F, A = T)$ concerning the Permutation Game.

In this section we reference the derangements function, denoted $D(n)$, which counts the number of permutations in $S_n$ with no fixed points. That is,

$$D(n) = n! \sum_{i=0}^{n} \frac{(-1)^i}{i!}, \tag{2}$$

which is the nearest integer to $n!/e$ [9].

#### 2.1.1 Trivial Lower Bound

To motivate the proof of Theorem 1, we prove the following easier result.

**Proposition 1.**

$$f(n, n) \geq \log_n(n!) = n - \frac{n}{\ln(n)} + O(1).$$

*Proof.* Consider an instance of the Permutation Game with the black-peg-only distance function $\Delta$. To uniquely identify $H$ after $m$ turns, the query response vector $(\Delta(Q_1, H), \Delta(Q_2, H), \ldots, \Delta(Q_m, H))$ must distinguish between all $n!$ permutations of the alphabet $\mathcal{A}$. In the $t^{\text{th}}$ turn, the codebreaker submits a query sequence $Q_t$, which has $n$ possible associated responses given by $\Delta(Q_t, H) \in \{0, 1, \ldots, n-2, n\}$. We note that it is not possible to get $\Delta(Q_t, H) = n - 1$ in the Permutation Game. Therefore there are $n^m$ possible query response vectors of length $m$, hence to identify $H$ in $m$ queries requires that $n^m \geq n!$. This implies $m \geq \log_n(n!)$, and the result follows immediately. □

### 2.1.2 Solution subsets

To improve upon the trivial lower bound, we first introduce terminology which we can use to analyze the worst-case performance of a possible guess in the permutation game. Assume the codebreaker has some arbitrary fixed, deterministic guessing strategy. After $t$ turns of this strategy, the codebreaker has guessed some sequence of queries $Q_1, \ldots, Q_t$ and received responses $r_1, \ldots, r_t$. The set of valid solution vectors $h$ with $\Delta(Q_i, h) = r_i$ for all $i$ is called the *remainig solution set* $S_t$, and the codebreaker wins exactly when $|S_t| = 1$.

Now, at the beginning of turn $t$, the codebreaker is going to guess $Q_t$. This guess partitions the remaining solution set $S_{t-1}$ into subsets for each possible response $\Delta(Q_t, H)$ from 0 to $n$, where each $h \in S_{t-1}$ is in the subset corresponding to a response $\Delta(Q_t, h)$. Now say a response of $\Delta(Q_t, H) = r$ is given. Then all solutions with $\Delta(Q_t, h) \neq r$ are eliminated from the remaining solution set, and this set becomes the subset of $S_{t-1}$ corresponding to a response $r$.

We call the subset of $S_{t-1}$ corresponding to a response $r$ a *solution subset*, formally defined as

$$B_t(r) = \{h \in S_t \mid \Delta(Q_t, h) = r\}.$$

Then the above logic can be written as $S_t = B_{t-1}(\Delta(Q_t, H))$.

We now compute the size of a subset $B_0(r)$ produced in the first turn of an instance of the Permutation Game. First we choose the $s$ colors that are fixed points with respect to the query sequence $Q_1$. We then permute the remaining $n - r$ colors without any fixed points, which can be done in $D(n - r)$ ways, with $D(x)$ as defined in (2). Hence we have

$$|B_0(r)| = \binom{n}{s} D(n - r).$$

The above computation tells us the *initial* size of each subset. Now consider $|B_t(r)|$. We have

$$|B_t(r)| = \#\{h \in S_t \mid \Delta(Q_t, h) = r\} \leq \#\{h \mid \Delta(Q_t, h) = r\}$$

Noting that $Q_t$ is necessarily a permutation of $Q_1$, we see that $\#\{h \mid \Delta(Q_t, h) = r\} = \#\{h \mid \Delta(Q_1, h) = r\}$. But the latter expression is simply $|B_0(r)|$, by definition. Thus we have

$$|B_t(r)| \leq |B_0(r)| = \binom{n}{r} D(n - r).$$

## 2.2 Proof of Theorem 1

Using solution subsets, we now improve the trivial lower bound for $f(n, n)$ in the context of the Permutation Game. For the proof, we require two technical lemmas. The first bounds the sums of sizes of the subsets defined above:

**Lemma 1.** *For any positive integer $n$, we have:*

$$\sum_{i=x}^{n} \binom{n}{i} D(n-i) \leq \frac{n!}{x!}.$$

*Proof.* We give a combinatorial proof. The left-hand side denotes the number of permutations of an $n$-element vector which have at least $x$ fixed points. The right-hand side denotes the number of ways to choose $x$ fixed points and simply permute the rest of the vector. This includes all vectors with at least $x$ fixed points and over-counts by some margin, so the inequality holds. □

Using this, we will prove the following bound on $|S_n|$:

**Lemma 2.** *For a fixed deterministic guessing strategy, there is a choice of hidden vector such that*

$$\frac{|S_t|}{n!} \geq \frac{C_n! - (H_{C_n+t} - H_{C_n})}{(C_n+t)!},$$

*for any positive integer $C_n$ and all $0 \leq t \leq n - C_n$, where $H_n = \sum_{i=1}^{n} \frac{1}{i}$ is the $n^{th}$ harmonic number.*

We prove Lemma 2 at the end of the section.

*Proof of Theorem 1.* Apply Lemma 2 for $t = n - C_n$. Then we have

$$|S_{n-C_n}| \geq n! \left( \frac{C_n! - (H_n - H_{C_n})}{n!} \right) = C_n! - (H_n - H_{C_n}).$$

We choose a $C_n$ as small as possible such that the above bound gives $|S_{n-C_n}| > 1$. This would mean that after $n - C_n$ guesses and responses of any guessing strategy, there are at least 2 solutions that match every response, in the worst case. Consequently, the codebreaker would not be able to uniquely identify the hidden code after $n - C_n$ turns. Thus, we want

$$C_n! - (H_n - H_{C_n}) > 1.$$

Noting that $H_n$ is asymptotic to $\log n$ and both grow to infinity, as long as $\log n = o(C_n!)$ we will eventually have that $C_n! - H_n > 1$. Since we have, for example, that $\log x = o((\log\log x)!)$, we have that with $C_n = \lceil \log\log n \rceil$ the above inequality will eventually be satisfied.

In conclusion, when $n$ is sufficiently large, the minimum number of remaining possible solutions after $n - \lceil \log\log n \rceil$ guesses is at least

$$S_{n-\lceil \log\log n \rceil} \geq (\log\log n)! - (H_n - H_{\log\log n}) > 1.$$

Thus there is no strategy that can identify any hidden sequence in fewer than $n - \log\log n$ turns, and so in the Permutation Game variant of Mastermind, we have $f(n,n) \geq n - \log\log n$ for all sufficiently large $n$. □

**Problem 1** (Open)**.** *In all adaptive variants of Mastermind, the best lower bounds are asymptotic to $n$. Can these lower bounds be improved?*

**Problem 2** (Open)**.** *In the variant of Mastermind with repeats, the best known strategy is $O(n \log\log k)$* [4]*, whereas in the variant without repeats, it is $O(n \log k)$* [12]*. Is it possible to extend or modify the first strategy to apply to the no-repeats game?*

## 2.3 Proof of Lemma 2

*Proof.* Recall that $S_t$ is the set of sequences that match the responses to the first $t$ questions of some fixed deterministic guessing strategy, given some hidden code $H$. Since all queries are possible when 0 questions have been asked, we have $|S_0| = n!$.

Consider the $t^{\text{th}}$ turn in an instance of the Permutation Game. There have been $t-1$ guesses made so far, and the set of remaining solutions is $S_{t-1}$. Now, the codebreaker is going to guess $Q_t$. To analyze the worst-case performance of this guess, we partition $S_{t-1}$ into subsets $B_{t-1}(r)$ for each possible response $r$. Since $S_t = B_{t-1}(\Delta(Q_t, H))$, in the worst case, $H$ will belong to the largest subset $B_t(r)$, thereby leaving a large number of solutions in the remaining set. Thus, the worst-case size of $S_t$ given $S_{t-1}$ is

$$\max_{r \in \{0,1,\ldots,n\}} |B_t(r)|.$$

The minimum possible value of this maximum occurs when the subsets partition $S_{t-1}$ as evenly as possible. We know that $|B_t(r)| \leq |B_0(r)|$, which is decreasing in $r$. In the optimal distribution of this type, some subsets of higher index will have size equal to their upper bound, while the rest will be partially filled, but to a greater amount than any of the higher index subsets. So with $|S_{t-1}|$ solutions remaining, there is an optimal $x$ such that completely filling subsets $x$ through $n$ and splitting the remaining solutions among subsets 0 through $x-1$ will give us this best distribution, and therefore a lower bound on $|S_t|$ in the worst case.

We now use this logic to bound $|S_t|$ recursively. Assume now that the codebreaker makes the query $Q_t$, receives some response $r$, and eliminates some number of possible solutions, leaving $S_t$ as the set of possible solutions remaining. We now claim that for all $x$,

$$|S_t| \geq \frac{1}{x}\left(|S_{t-1}| - \sum_{i=x}^{n} \binom{n}{i} D(n-i)\right) \tag{3}$$

By the above logic, we know this inequality holds for some optimal value of $x$, however since the subsets are decreasing in size, it is easy to see from the construction that the right-hand side is maximized for this choice of $x$, and so adjustment of $x$ away from this optimal value always preserves the inequality.

We apply Lemma 1 to bound the rightmost term and get

$$|S_t| \geq \frac{1}{x}\left(|S_{t-1}| - \frac{n!}{x!}\right). \tag{4}$$

We now exhibit a solution to this recurrence relation to prove Lemma 2.

We proceed by induction. With $t = 0$, we have $|S_0| = n!$. Then

$$1 = \frac{|S_0|}{n!} \geq \frac{C_n! - (H_{C_n} - H_{C_n})}{C_n!} = 1,$$

and the inequality is satisfied.

Now we move to the general case. Recalling that (4) holds for all $x$, we will let $x = t + C_n$, giving

$$\frac{|S_t|}{n!} \geq \frac{1}{C_n + t}\left(\frac{|S_{t-1}|}{n!} - \frac{1}{(C_n + t)!}\right).$$

Assuming the lemma inductively for $t-1$, we obtain

$$\begin{aligned}\frac{|S_t|}{n!} &\geq \frac{1}{C_n+t}\left(\frac{C_n! - (H_{C_n+t} - H_{C_n})}{(C_n+t-1)!} - \frac{1}{(C_n+t)!}\right) \\ &\geq \left(\frac{C_n! - (H_{C_n+t-1} - H_{C_n}) - \frac{1}{C_n+t}}{(C_n+t)!}\right) \\ &\geq \left(\frac{C_n! - (H_{C_n+t} - H_{C_n})}{(C_n+t)!}\right),\end{aligned}$$

which completes the induction. □

# 3 Linear Algebra and the Minimax Algorithm

In this section we prove Theorem 2, which concerns Knuth's Minimax algorithm. We will represent an arbitrary query or hidden vector as a $(0,1)$-vector $A \in \mathbb{R}^{nk}$ in the following manner:

$$A_{in+j} = \begin{cases} 1 & \text{this guess/solution assigns the } i^{\text{th}} \text{ spot the } j^{\text{th}} \text{ color} \\ 0 & \text{otherwise,} \end{cases}$$

where $0 \leq i \leq n-1$ and $0 \leq j \leq k-1$. As such, each set of indices $A_{in}, \ldots, A_{in+k-1}$ will have exactly one 1, as the $i^{\text{th}}$ position is exactly one color.

With this notation, the black-peg distance becomes the dot product of the guess and the hidden vector, as there will be a contribution to the dot product exactly when both vectors have a one in the same spot, i.e. there is the same color in the same spot of both vectors.

The goal of Mastermind is then to find the unique valid $(0,1)$-vector such that its dot product with the hidden vector is $n$. By linearity of the dot product, once we know the value of the dot product of the hidden vector with some set of queries, we know the value of the dot product of the hidden vector with any vector in the span of these guesses. Therefore, it is an immediate consequence that a winning strategy can be found by querying a basis for the span of the valid queries, which, as a subspace of $\mathbb{R}^{nk}$, must be of size at most $nk$. Note that this strategy does not make use of adaptive feedback, of white-peg responses, or the condition on repeated colors. Thus, for any choice of $\Delta$, $A$, and $R$, we have $f(n,k) \leq nk$.

This logic also allows us to bound the minimax algorithm of [10].

## 3.1 Proof of Theorem 2

The theorem can be easily reduced to the following lemma:

**Lemma 3.** *At each turn, the minimax algorithm either guesses the hidden vector or guesses a vector that is linearly independent of the previous guesses.*

*Proof.* If the hidden vector is known, the minimax algorithm will guess it. As discussed above, this will certainly occur when the solution is in the linear span of the set of vectors queried.

Otherwise, the hidden vector is linearly independent from the previous queries. At this point, guessing a vector $q$ that is a linear combination of the previous guesses returns no new information, as the response to that guess can already be computed. Since $q$ is linearly dependent on the previous guesses, it cannot, by our assumption, be the hidden vector, and so guessing it eliminates zero vectors

and does not solve the game. If we can find a new query $r$ that is guaranteed to either eliminate at least one vector or solve the game, the minimax algorithm will choose $r$ over $q$. Since the hidden vector must exist and by our assumption is linearly independent of the queries, it is a valid, linearly independent guess, so at least one such guess must exist. Guessing such a vector will win the game with a black-peg response of $n$, and eliminate at least one vector (itself) with any other response. Thus, according to the minimax algorithm, guessing any vector linearly independent of the previous guesses will always be strictly better than guessing a linearly dependent vector, and so if the hidden vector is unknown the best guess will be necessarily independent of the previous guesses. □

Theorem 2 follows as an immediate corollary from this lemma: Since these representations are in $\mathbb{R}^{nk}$, the minimax algorithm can make at most $nk$ linearly independent guesses. After that, the minimax cannot make a linearly independent guess and by the above lemma must then guess the hidden vector, for an upper bound of $nk$ turns to determine the hidden vector and one more turn to guess it.

# 4 Non-Adaptive Variants

This section follows closely the reasoning in [4] as they analyze non-adaptive games. We perform an analysis of the Mastermind variant in which $\Delta$ is the black-peg-only distance function, $R = F$ and $A = F$ (*i.e.,* no repeats and non-adaptive). We first introduce Shannon entropy, which we then employ to give a lower bound for $f(n, k)$ when no repeats are allowed. We conclude with a small extension of one of the proofs due to Doerr, Spöhel, Thomas, and Winzen to provide an upper bound on $f(n, k)$ for non-adaptive variants when $k \geq n$.

## 4.1 Shannon Entropy

First we give a brief definition of entropy in the context of information theory. Specifically, we use *Shannon entropy*. Our presentation borrows from [6]. Shannon entropy is intuitively the "expected level of surprise" of a random variable, and hence is based upon the definition of a surprise function $S$. For random variable $X$, and possible event $X = x$, we define $S(x) = -\log_2(\mathbb{P}[X = x])$. Let $X$ and $Y$ be a random variables and let $X = x$ and $Y = y$ be events. The surprise function $S$ then satisfies $S(X = x \wedge Y = y) = S(X = x) + S(Y = y \mid X = x)$ [6].

Let $D$ be the domain of the random variable $X$. Then the *Shannon entropy* $H(X)$ of the random variable $X$ is defined to be

$$H(X) = \mathbb{E}[S(x)] = \sum_{x \in D} \mathbb{P}[X = x] \cdot (-\log_2(\mathbb{P}[X = x])).$$

One can easily show that Shannon entropy is *subadditive* [6]. That is, if $X_1, X_2, \ldots, X_n$ are random variables, and $(X_1, \ldots, X_n)$ is the vector containing the $X_i$ as entries, then

$$H(X_1, X_2, \ldots, X_n) \leq \sum_{i=1}^{n} H(X_i). \tag{5}$$

## 4.2 Proof of Theorem 3

*Proof.* We prove Theorem 3 in the case $R = F$, noting that [4] have already proved the case $R = T$. We will be drawing substantially from the techniques they use in their proof. Consider a set $\{Q_1, Q_2, \ldots, Q_s\}$ of $s$ query sequences such that all $k!/(k-n)!$ possible hidden sequences are uniquely

determined by the responses $\Delta(Q_1, H), \Delta(Q_2, H), \ldots, \Delta(Q_s, H)$. That is, assume that if two possible hidden vectors $H$ and $H'$ satisfy $\Delta(Q_i, H) = \Delta(Q_i, H')$ for all $i$, then $H = H'$.

Let the hidden vector $Z$ be sampled uniformly at random from the set of $k!/(k-n)!$ possibilities. Then the responses $Y_i = b(Z, q_i)$ are now random variables, and the vector $Y = (Y_1, Y_2, \ldots, Y_s)$ is also a random variable. By our assumptions, $Y$ always uniquely determines, and is uniquely determined by, $Z$. This bijection of events $Z = z$ with events $Y = y$ shows that $H(Z) = H(Y)$. Since $Z$ is a random variable with $k!/(k-n)!$ outcomes of equal probability, we compute

$$H(Z) = \log_2 \left( \frac{k!}{(k-n)!} \right). \tag{6}$$

By (5), we have

$$H(Y) \leq \sum_{i=1}^{s} H(Y_i). \tag{7}$$

Now we bound $H(Y_i)$. We recall that by assumption, $\Delta = b$ so $Y_i$ is precisely the black-peg distance between $Q_i$ and $Z$. By definition,

$$H(Y_i) = -\sum_{x=0}^{n} \mathbb{P}[Y_i = x] \cdot \log_2(\mathbb{P}[Y_i = x]). \tag{8}$$

We now compute $\mathbb{P}[Y_i = x]$, namely the probability that $X$ is a solution vector with $x$ fixed points with respect to the query $q_i$. Using our previous terminology, this is the probability that $X$ is in subspace $B(x)$. The size of $B(x)$ is the number of permutations with exactly $x$ fixed points, which is fewer than the number of ways to choose $x$ fixed points, and then choose any colors for the remaining of the elements. Thus:

$$\begin{aligned}
\mathbb{P}[Y_i = x] &= \frac{|B(x)|}{\left(\frac{k!}{(k-n)!}\right)} \\
&\leq \frac{\binom{n}{x}\frac{(k-x)!}{(k-n)!}}{\left(\frac{k!}{(k-n)!}\right)} \\
&= \frac{1}{x!} \cdot \frac{n(n-1)\cdots(n-x+1)}{k(k-1)\cdots(k-x+1)} \\
&\leq \frac{1}{x!}.
\end{aligned}$$

It turns out that we cannot substitute this upper bound into (8) for all $x$, because $f(\alpha) = -\alpha \log_2 \alpha$ is an increasing function only when $\alpha < 1/e$. So we can plug in the upper bound of $\Pr[X = x] \leq 1/x!$ when $x \geq 3$ and still have an upper bound. For the first three values of $x$, we instead use the trivial upper bound $f(\alpha) \leq 1/(e \log 2)$. Then

$$H(Y_i) \leq \frac{3}{e \log 2} + \sum_{x=3}^{n} -\frac{1}{x!} \cdot \log_2 \left( \frac{1}{x!} \right) \leq \frac{3}{e \log 2} + \sum_{x=3}^{\infty} \frac{\log_2(x!)}{x!} < 3.$$

Combining this with (7) gives $H(Y) \leq 3s$, and since $H(Z) = H(Y)$ we have $H(Z) \leq 3s$. Substituting in (6) as a lower bound for $H(Z)$ and solving for $s$ gives

$$s \geq \frac{1}{3} \log_2 \left( \frac{k!}{(k-n)!} \right).$$

We can show that the right-hand side is $\Omega(n \log k)$. This is immediate for large $k$ relative to $n$ by bounding the ratio by $(k-n)^n$; for small $k$ (e.g. $k \leq 2n$) we bound the ratio by $n!$ and the claim follows from Stirling's approximation. So this gives us a lower bound of $\Omega(n \log k)$ turns for any non-adaptive strategy for Mastermind with no repeats and black-peg responses. □

#### 4.2.1 Extension to white-peg responses

We extend the $R = T$ case of Theorem 3, as proved in [4], to include $\Delta = bw$. We show that any black-white strategy must submit $\Omega(k)$ queries, and that we can convert a black and white-peg strategy into a black-peg-only strategy by adding $O(k)$ queries. Then changing a black-white strategy to a black-only strategy changes the number of queries by at most some absolute constant factor. So we have $f(\Delta = b) = O(f(\Delta = bw))$. Combining this with Theorem 3 implies the theorem for black-white responses as well.

First we describe the conversion. Take a black-white strategy and append, for each color, a query where every spot is that color. As a black-peg only strategy, we can deduce from these queries exactly how many times each color appears in the hidden vector, from which we can determine the white-peg responses to any of the original queries. Therefore, if the old strategy had enough information to determine the hidden vector with black-white responses, this new strategy can determine the hidden vector with just black-peg responses. Since there are $k$ colors, we have appended $k$ queries in this conversion.

Lastly, we show that any non-adaptive black-white strategy must submit $\Omega(k)$ queries. Assume that there are two colors $a$ and $b$ such that a non-adaptive strategy guesses neither $a$ nor $b$ in either spot 1 or 2. Then this strategy would not be able to distinguish between a hidden vector starting $a, b$ and one starting $b, a$. Then by contradiction any non-adaptive strategy must guess at least $k-1$ colors in these two spots. It can guess 2 colors in these spots per turn, for a total of at least $(k-1)/2 = \Omega(k)$ turns required to solve the game. Thus, by the above logic, we conclude $f(n, k) = \Omega(n \log k)$ in this case as well.

**Problem 3** (Open). *Can this argument be modified or extended to the case $R = F$, $\Delta = bw$ as well?*

### 4.3 An Upper Bound for Non-Adaptive Variants with Repeats

Consider non-adaptive variants of Mastermind in which repetitions are allowed. When $k = n$, [4] showed the existence of a set of $O(n \log n)$ queries which uniquely identify any hidden sequence $H$. When $k > n$, one can simply extend $H$ and all queries $Q_t$ by $k - n$ "auxiliary" positions. We fill these auxiliary positions with arbitrary colors, and adjust the codemaker's responses accordingly. By the results of [4], there exists a set of $O(k \log k)$ queries which will uniquely identify any hidden sequence.

**Problem 4** (Open). *In non-adaptive variants, the best lower bounds are $O(n \log k)$, whereas upper bounds are either $O(k \log k)$ or $nk$. Can these bounds be brought closer together?*

## 5 Acknowledgments

We would like to thank Daniel Montealegre for supervising and assisting our work throughout the summer, and Nathan Kaplan for both creating and guiding our project and for his invaluable assistance in editing this paper. We would also like to thank Sam Payne and the Summer Undergraduate Math Research at Yale program for organizing, funding, and supporting this project. SUMRY is supported in part by NSF grant CAREER DMS-1149054.

# References


[1] Nader H. Bshouty, *Optimal algorithms for the coin weighing problem with a spring scale*, COLT **1** (2009).

[2] Zixiang Chen, Carlos Cunha, and Steven Homer, *Finding a hidden code by asking questions*, Lecture Notes in Computer Science **1** (2005), 50–55.

[3] Václav Chvátal, *Mastermind*, Combinatorica **3** (1983), 325–329.

[4] Benjamin Doerr, Reto Spohel, Henning Thomas, and Carola Winzen, *Playing mastermind with many colors*, Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (2013), 695–704.

[5] Paul Erdős and A Rényi, *On two problems of information theory*, Magyar Tud. Akad. Mat. Kutató Int. Kőzl **8** (1963), 229–243.

[6] David Galvin, *Three tutorial lectures on entropy and counting*, First Lake Michigan Workshop on Combinatorics and Graph Theory (2014).

[7] Michael T. Goodrich, *On the algorithmic complexity of the mastermind game with black-peg results*, Information Processing Letters **109** (2009), 675–678.

[8] V Grebinsky and G Kucherov, *Optimal reconstruction of graphs under the additive model*, Algorithmica **1** (2000), 104–124.

[9] Mehdi Hassani, *Derangements and applications*, Journal of Integer Sequences **6** (2003).

[10] Donald Knuth, *The computer as master mind*, Journal of Recreational Mathematics **9** (1976), no. 1, 2–7.

[11] Ker-I Ko and Shia-Chung Teng, *On the number of queries necessary to identify a permutation*, Journal of Algorithms **7** (1986), 449–462.

[12] Mourad El Ouali and Volkmar Sauerland, *Improved approximation algorithm for the number of queries necessary to identify a permutation*, CoRR **abs/1303.5862** (2013).