Query Complexity of Mastermind Variants

Aaron Berger Christopher Chute Matthew Stone

October 25, 2015

Abstract

We analyze variants of the popular board game Mastermind. In this two-player game, the codebreaker submits queries with the goal of identifying a hidden sequence, constructed at the beginning of the game by the codemaker. At each step, the codebreaker receives feedback in the form of "black" and "white" hits and incorporates the response into his next guess. We discuss asymptotics for the number of guesses needed to identify an unknown n-vector constructed from k possible colors. We look at strategies that receive two-color responses, as well as black hit-only responses. We consider both allowing and prohibiting repeated colors in the hidden sequence, and we analyze both adaptive and non-adaptive guessing strategies.

1 Introduction

Mastermind is a two-player board game that was invented in 1970, and variants on its basic four-spot, six-color structure have been studied extensively. In the original game, there are two players: the codemaker and the codebreaker. The codemaker initiates a game by constructing a 4-vector p from the six available colors, and the codebreaker attempts to guess the hidden vector in as few turns as possible. A turn consists of two parts: First the codebreaker submits a query vector q, which has the same form as a hidden vector. Second, the codemaker gives a two-part response:

1. Black Hits: The number of correct colors in the correct spot (the number of positions i such that $q_i = p_i$).

2. White Hits: The number of correct colors in the incorrect spot, and which have not been used in another hit.

In 1976, Donald Knuth presented a greedy "minimax" algorithm, and he showed via computer simulation that his algorithm always guesses the hidden vector in five turns or fewer [**DK76**]. Moreover, the minimax algorithm for Mastermind is optimal in the worst case—there is no algorithm that can guarantee to win in at most four turns. [1]

There are multiple natural extensions to the original Mastermind game. Varying the number of spots n and colors k, as well as the relationship between n and k, is one such extension. In this paper, we also consider both allowing and prohibiting repeated colors in vectors, and we

[1] Knuth shows that the worst-case response to any possible first guess leaves at least 256 possible solution vectors. After this, with 14 possible responses per turn, the trivial information-theoretic argument gives a lower bound of five turns total to win.

examine both dual-color and black-only responses. We explore another degree of freedom between adaptive strategies, in which the codebreaker receives responses and adjusts the next guess accordingly, and non-adaptive strategies, in which all guesses are submitted at the beginning of the game and the hidden vector must be uniquely determined by the sequence of responses. [2]

1.1 Previous Work

Original analysis of the game was performed by Knuth, who laid out the minimax algorithm. [**VC83**], and later [**DS13**], were able to successfully resolve the adaptive and non-adaptive games with repeated colors variant when there are fewer colors than spots. They also describe algorithms for the standard game for more colors than spots. [**KT86**], and later [**OS13**], provide algorithms for the no-repeats variant of the game.

1.2 Our Contribution

We primarily focus on three new methods for worst-case analysis of mastermind games where the number of colors k is greater than or equal to the number of spots n. Unlike the case where $k < n$, there are barely any tight bounds for any variant of the game, and some commonly analyzed variants have no upper bounds at all. We first show an upper bound of nk for any mastermind game. We perform the first ever worst-case analysis of Knuth's minimax algorithm and show that it eliminates at least a fraction of $1/nk$ of the remaining solutions at any step, for an $O(n^2 k \log k)$ upper bound. Additionally, we perform analysis on a new lower bound that models a set of best guesses such that the distribution of the responses among the solutions approaches an even distribution (the best possible distrubution) as quickly as possible. With this analysis, we obtain an improvement over the trivial lower bound (on the permutation game) with a new bound of $n - \log\log n$ turns. Next, we extend two arguments by [**DS13**]: one for the black-peg NO-REPEATS game to give an $n \log k$ lower bound, and one to the original non-adaptive game for an $O(k \log k)$ upper bound. Finally, we end with a few notes and conjectures about the different variants of the game when the number of colors far exceeds the number of spots.

2 Standard Upper Bound

The standard upper bound of nk turns for any game is derived as follows:
For a game of mastermind with n spots and k colors (with or without repeats), every hidden vector and query can be represented by a matrix $A \in \mathbb{R}^{n \times k}$, defined by $A_{ij} = 1$ if the i-th peg is the j-th color, and 0 otherwise. In that way, for a given hidden vector matrix X and query matrix Q, the codemaker's black-peg response to the query is given by $X \cdot Q$. (We are treating these matrices as vectors in $\mathbb{R}^{nk}$.) Let S be the subspace spanned by the set of possible queries. Then there must exist a basis of queries, K, with $|K| = \dim(S)$ and K spanning S. $S \subset \mathbb{R}^{n \times k}$, so $\dim(S) \leq \dim(\mathbb{R}^{n \times k}) = n \cdot k$. Now, we choose this basis as our set of guesses. We look at the black-hit responses to these guesses, which gives us the dot-products of the hidden vector X with each of the basis vectors. By distribution, we can then uniquely determine the dot-products

[2] Note that in a non-adaptive guessing strategy, our guesses need to *identify* the hidden vector rather than guess it outright. That is, a non-adaptive strategy wins if the sequence of responses to the guess vectors allows the codebreaker to distinguish between all possible hidden vectors. It is not necessary that a winning set of non-adaptive guesses includes the hidden vector itself.

of X with each of the basis vectors of some arbitrary orthogonal basis of S, and we can then determne the projections of X onto each basis vector in this orthogonal basis. Adding these components together allows us to uniquely determine $X \in S$. Thus, there exists a set of at most $n \cdot k$ queries which uniquely determines every possible hidden permutation. (Note that since this strategy is non-adaptive and only uses black hits, this bound holds for every variant of the game analyzed in this paper.)

3 Adaptive Strategies

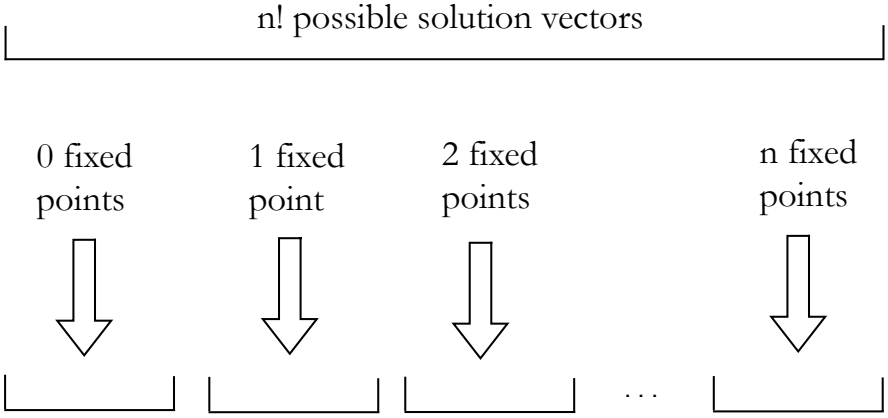
We begin by performing an analysis on the PERMUTATION game, which is the NO-REPEATS game where the number of spots n equals the number of colors k. Algorithms for this game that take $O(n \log n)$ turns were developed by [**KT86**] and [**OS13**]. The information-theoretic lower bound on this game is $n - n/\log n + O(1)$. We improve this to $n - \log n$ for small n, and $n - \log\log n$ for sufficiently large n, which constitutes, to our knowledge, the first improvement over the trivial information-theoretic lower bound for any adaptive mastermind game.

3.1 Lower Bound on the Permutation Game

1. *Trivial Lower Bound.* We have $n!$ possible hidden vectors, and each query has n possible responses.[3] Since we must distinguish between every pair of vectors, we must ask at least

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

2. *Buckets.* In order to beat the trivial lower bound, we use a more accurate picture of the distribution of the possible solutions after feedback from a given guess. For any guess, we can split the possible solutions up into groups, which we will call *buckets*; one bucket for each possible response given by the codemaker. For any guess we make, the number of possible solutions that give the response "k (black) hits" is the number of permutations with k fixed points with respect to our guess vector.



[3]The possible responses are $0, 1, \ldots, n-2$, and n, since it is impossible to get exactly $n-1$ black hits with no repeats and $n = k$.

The size of a bucket of possible solutions with k hits can be counted easily: we choose the k colors that are fixed points (with respect to our guess vector), then we permute the remaining $n-k$ colors without any fixed points, giving $D(n-k)$, the derangements on an $(n-k)$-element vector:

$$\binom{n}{k} D(n-k)$$

This tells us the initial size of each bucket (by symmetry, this is the same distribution for any guess). In general, the buckets start out large, and decrease approximately by a factor of x from the $(x-1)$ bucket to the x bucket. It is important to note that if we look at the sizes of these buckets for a particular guess part-way into the game, the size of each bucket can only decrease from its original size, as each previous guess simply eliminates some solutions from each bucket.

3. *Notation.* We let $S_n(t)$ be a lower bound on the minimum number of possible solutions remaining after t guesses for any guessing strategy. We compute this lower bound by using a hypothetical set of guesses which are each guaranteed to eliminate at least as many solutions as any other guess we could make at any point in the game. The worst-case scenario for this set of guesses therefore provides a lower bound for the number of guesses required by any guessing strategy to guarantee finding the hidden vector.

4. **Recurrence Relation on Number of Possible Solutions after t Turns:**

   (a) We will inductively prove a lower bound, $S_n(t)$. By the discussion above, we know the maximum possible size of each bucket. The best possible guess distributes the remaining solutions as evenly as possible so that the bucket containing the most remaining solutions is as small as possible. This best-possible distribution will fill some number of smaller buckets entirely and split the remaining solutions evenly into some number x larger buckets. That is, there are x incompletely filled buckets.

   (b) At the beginning of the t-th turn, we have at least $S_n(t-1)$ possible solutions remaining. As we showed before, the size of each bucket is bounded by its original size of $\binom{n}{k} D(n-k)$. Thus the number of solutions remaining $(t-1)$-th turn relates to the value of $S_n(t)$ as follows:

   Let $B_i(t)$ be the exact number of possible solutions remaining in the bucket holding possible solutions with i fixed points during the t-th turn (after submitting the guess vector, but before feedback). So

$$\begin{aligned} S_n(t-1) &= \sum_{i=0}^{n} B_i(t) \\ &= \sum_{i=0}^{x-1} B_i(t) + \sum_{i=x}^{n} B_i(t) \\ &\leq \sum_{i=0}^{x-1} S_n(t) + \sum_{i=x}^{n} \binom{n}{i} D(n-i) \quad \text{for optimal } x \\ &= x \cdot S_n(t) + \sum_{i=x}^{n} \binom{n}{i} D(n-i) \end{aligned}$$

Once we know $S_n(t)$, picking x too large results in some smaller boxes overfilled. Picking x too small results in some larger boxes overfilled. So changing x to be something other than optimal always increases the RHS of this equation. Thus we have

$$S_n(t-1) \leq x \cdot S_n(t) + \sum_{i=x}^{n} \binom{n}{i} D(n-i) \quad \forall x$$

Where t is the number of turns taken, $S_n(t)$ is the number of possible solutions remaining after the t-th turn, and $D(k)$ is the derangements on a set of size k.

5. *Claim.*

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

This inequality leads to our bound on $S_n(t)$ given below.[4]

Proof.

$$\begin{aligned}
\sum_{i=x}^{n} \binom{n}{i} D(n-i) &= \frac{n!}{x!} \sum_{i=x}^{n} \frac{x!}{i!(n-i)!} D(n-i) \\
&\qquad \text{we use the fact that } \frac{x!}{i!} \leq \frac{1}{(i-x)!} \\
&\leq \frac{n!}{x!} \sum_{i=x}^{n} \frac{1}{(i-x)!(n-i)!} D(n-i) \\
&\qquad \text{let } k = i - x \\
&= \frac{n!}{x!} \sum_{k=0}^{n-x} \frac{1}{k!(n-x-k)!} D(n-x-k) \\
&= \frac{n!}{x!(n-x)!} \sum_{k=0}^{n-x} \binom{n-x}{k} D(n-x-k) \\
&\qquad \text{note the summation counts all possible permutations} \\
&= \frac{n!}{x!(n-x)!} (n-x)! \\
&= \frac{n!}{x!}
\end{aligned}$$

□

(a) Plug the above result into the inequality to get

$$S_n(t-1) \leq x \cdot S_n(t) + \frac{n!}{x!}$$

[4] A combinatorial argument can be made as follows: The LHS denotes the number of permutations of an n-element vector which have at least x fixed points. Suppose we choose x fixed points and simply permute the rest of the vector. This gives $\binom{n}{x}(n-x)! = \frac{n!}{x!}$ possible permutations. Clearly this includes all vectors with at least x fixed points (and over-counts by some margin), so the inequality holds.

So we have

$$S_n(t) \geq \frac{1}{x}\left(S_n(t-1) - \frac{n!}{x!}\right)$$

Lemma 1. *For all n, and for any constant C_n, we have*

$$\frac{S_n(t)}{n!} \geq \frac{C_n! - (H_{C_n+t} - H_{C_n})}{(C_n+t)!}$$

$$\textit{where} \quad 0 \leq t \leq n - C_n$$

Here we use H_n to denote the n-th harmonic number.[5]

Proof. While this equation holds for all x, to achieve a desirable bound, we will let $x = t + C_n$ for any positive constant C_n. We have

Base Case. When $t = 0$, we have $S_n(0) = n!$ and

$$\frac{S_n(0)}{n!} \geq \frac{C_n! - (H_{C_n} - H_{C_n})}{C_n!} = 1$$

Inductive Step. Assume the induction hypothesis holds for $t - 1$.

$$\frac{S_n(t)}{n!} \geq \frac{1}{x}\left(\frac{S_n(t-1)}{n!} - \frac{1}{x!}\right)$$

Let $x = C_n + t$.

$$\frac{S_n(t)}{n!} \geq \frac{1}{C_n+t}\left(\frac{S_n(t-1)}{n!} - \frac{1}{(C_n+t)!}\right)$$

Inductively,

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

Then

$$\begin{aligned}
S_n(n - C_n) &\geq n!\left(\frac{C_n! - (H_n - H_{C_n})}{n!}\right) \\
&\geq C_n! - (H_n - H_{C_n})
\end{aligned}$$

Thus the inequality holds for all n.

□

[5] $H_n = \sum_{i=1}^{n} \frac{1}{i}$

We choose C_n by the following reasoning: we want to achieve a lower bound of $n - C_n$ turns. Thus, we want $S_n(n - C_n) > 1$, which follows if

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

For $C_n = \log\log n$, this is ≥ 1 for $n \geq e^{198}$.
For $C_n = \log n$, this is true for $n \geq 10$.

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

$$\begin{aligned} S_n(n - \log\log n) &\geq (\log\log n)! - (H_n - H_{\log\log n}) \\ &> 1 \end{aligned}$$

Thus there is no strategy that can reduce the set of possible solutions to one, and guarantee finding the codemaker's hidden vector, in $n - \log\log n$ turns or fewer.

3.2 Best Guesses and the Minimax Algorithm

We now move on to an analysis of the best possible guesses and the minimax algorithm. Here, the best possible guess is defined in the same way as [**DK76**]: it is the one such that the maximum number of solutions remaining over all responses is minimized (equivalently, the minimum number of solutions eliminated over all responses is maximized). This is the guess chosen by the minimax algorithm.
We begin by analyzing the NO-REPEATS game.

Theorem 1. *In Mastermind with no repeats with n spots and k colors, for any set of possible remaining solutions, there exists a guess vector for which any response will eliminate at least $1/nk$ of the remaining solutions.*

1. *Notation.*

    i. *Basics.* Let n be the number of spots in the hidden vector. Let k be the number of colors available. Since we allow no repeats in this variant of Mastermind, we have $k \geq n$. We will use single-count responses, which are the number of spots in which our guess vector and the hidden vector have the same color in that spot.

    ii. *Remaining Solution Set.* As the game proceeds, we maintain a set S that contains all possible solution vectors that haven't been ruled out by a previous guess and response.

    iii. *Bucket.* As defined in the previous section, a guess splits the remaining possible solutions into "buckets," where a bucket B_i contains all the remaining possible solutions with i fixed points with respect to the guess vector.

    iv. *Sub-Bucket.* Within a bucket, we will divide the vectors based on a chosen component, say the first element of each vector. Then there will be k sub-buckets D_i, where D_1 contains all vectors from the bucket with a 1 in the first spot, D_2 contains all those with a 2 in the first spot, and so on.

2. *Proof.* We give a proof by induction on n.

Base Case. When $n = 1$, we have a single-element vector. So there are at most k possible solutions remaining at any step, thus $|S| \leq k$. If we guess one of the remaining solutions $v \in S$, this will eliminate 1 solution (v itself) if the response is 0 and $|S| - 1$ solutions if the response is 1. So we're guaranteed to eliminate at least $1/|S| \geq 1/(nk)$ solutions.

Inductive Step. Assume that for $n - 1$ spots and any number of colors $\widehat{k}$ (where $n \geq 2$), we are guaranteed that there is a guess that eliminates a fraction at least $1/(n\widehat{k})$ of the solutions.

Consider a guess vector $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$. We will analyze the number of solutions eliminated from S by guessing σ in the worst-case. We divide S into n buckets $B_1, B_2, \ldots, B_n$, where B_i is as defined above (i fixed points with respect to σ). Consider the response j that eliminates as few possible solutions as possible from S, so $|B_j| = \max\{|B_1|, |B_2|, \ldots, |B_n|\}$.

Suppose that

$$|B_j| \leq \frac{nk-1}{nk} \cdot |S|$$

Then the response j would eliminate at least $|S|/(nk)$ solutions from S, and since j eliminates the fewest possible solutions of any response, all responses to σ eliminate a fraction at least $1/(nk)$ of the possible solutions in S.

Otherwise we have

$$|B_j| > \frac{nk-1}{nk} \cdot |S|$$

Case 1. If $j > 0$, the total number of hits among all vectors in B_j is $j \cdot |B_j|$. These hits are divided up among n spots, therefore by the pigeonhole principle, at least one position must have at least

$$\frac{j}{n} \cdot |B_j|$$

Vectors that match the guess at that spot. WLOG, we assume this is position one. We now divide the bucket B_j into sub-buckets D_i, where permutation $s \in B_j$ is put into sub-bucket D_{s_1}. From above, we have

$$|D_{\sigma_1}| \geq \frac{j}{n} \cdot |B_j|$$

We have two sub-cases:

i. $|D_{\sigma_1}| \leq \dfrac{n-1}{n} \cdot |B_j|$:
Then we apply the pigeonhole principle on the remaining solutions in B_j to find the second-largest sub-bucket: By our assumption, there are at least $|B_j|/n$ solutions remaining in B_j, and there are $k-1$ remaining sub-buckets D_i. By the pigeonhole principle, there exists a D_α such that

$$|D_\alpha| \geq \frac{1}{n} \cdot \frac{1}{k-1} \cdot |B_j|$$

If α is an element of σ, then WLOG let the color be σ_2. Then in this case we consider the guess vector $\overline{\sigma} = \{\sigma_2, \sigma_1, \sigma_3, \ldots, \sigma_n\}$. Otherwise, consider the

guess vector $\overline{\sigma} = \{\alpha, \sigma_2, \sigma_3, \ldots, \sigma_n\}$. Consider the buckets C_i formed from the guess vector $\overline{\sigma}$. In either case, the vectors in D_{σ_1} will have at least one fewer hit with $\overline{\sigma}$ than with σ, and the guess vectors in D_α will have at least one more hit. Therefore, any response to $\overline{\sigma}$ which corresponds to some vectors in D_{σ_1} corresponds to none of the vectors in D_α and vice versa. So we know that any response to $\overline{\sigma}$ will entirely eliminate at least one of those two sets. From above, we know

$$\begin{aligned}
|D_{\sigma_1}| &\geq \frac{j}{n} \cdot |B_j| \\
&\geq \frac{j}{n} \cdot \frac{nk-1}{nk} \cdot |S| \\
&\geq \frac{1}{n} \cdot \frac{n}{nk} \cdot |S| \\
&= \frac{1}{nk} \cdot |S|
\end{aligned}$$

And

$$\begin{aligned}
|D_\alpha| &\geq \frac{1}{n} \cdot \frac{1}{k-1} \cdot |B_j| \\
&\geq \frac{1}{nk-n} \cdot \frac{nk-1}{nk} \cdot |S| \\
&\geq \frac{1}{nk} \cdot |S|
\end{aligned}$$

Since $\overline{\sigma}$ will always eliminate at least one of these two sets, we also have that $\overline{\sigma}$ will eliminate at least $|S|/(nk)$ vectors from S with any response.

ii. $|D_{\sigma_1}| > \dfrac{n-1}{n} \cdot |B_j|$:
Consider the set of vectors in D_{σ_1}. By definition, all of them have σ_1 as their first entry. Their remaining $n-1$ entries are made up of the remaining $k-1$ colors, and no two of them coincide in all spots $2, 3, \ldots, n$ (otherwise they would be the same vector). Therefore, if we consider this new set of $n-1$ vectors formed by taking spots $2, 3, \ldots, n$ of each vector in D_{σ_1}, they are a set of possible solutions for this Mastermind game with $n-1$ spots and $k-1$ colors. By the induction hypothesis, there is a guess of size $n-1$ that is guaranteed to eliminate $1/((n-1)(k-1))$ of these vectors. If we add σ_1 to the beginning of this guess, it becomes a guess of size n. For any response r, any of the vectors in D_{σ_1} will be eliminated if and only if their vector of spots $2, 3, \ldots, n$ was eliminated by the response $r-1$ in the smaller game of Mastermind. Therefore, this guess still guarantees that we eliminate

$$\begin{aligned}
\frac{1}{(n-1)(k-1)}|D_{\sigma_1}| &\geq \frac{1}{(n-1)(k-1)} \cdot \frac{n-1}{n} \cdot |B_j| \\
&\geq \frac{1}{(k-1)} \cdot \frac{1}{n} \cdot \frac{nk-1}{nk} \cdot |S| \\
&\geq \frac{1}{nk-n} \cdot \frac{nk-1}{nk} \cdot |S| \\
&\geq \frac{1}{nk} \cdot |S|
\end{aligned}$$

Case 2. If $j = 0$, we consider the first component of all vectors in the bucket B_0. Since $j = 0$ and $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$, no vector in B_0 can contain σ_1 in the first position, so there are $k - 1$ possible colors in the first position. By the pigeonhole principle, there must be one color which appears in the first spot of at least

$$\frac{1}{k-1} \cdot |B_0|$$

vectors in B_0. If this color appears in σ, WLOG let the color be σ_2. Then in this case we consider the guess vector $\overline{\sigma} = \{\sigma_2, \sigma_1, \sigma_3, \ldots, \sigma_n\}$. Otherwise, call this new color τ and consider the guess vector $\overline{\sigma} = \{\tau, \sigma_2, \sigma_3, \ldots, \sigma_n\}$. In either case, $\overline{\sigma}$ matches these $|B_0|/(k-1)$ vectors in either one or two positions. Consider the buckets C_i formed by the guess $\overline{\sigma}$. We know

$$\begin{aligned}
|C_1| + |C_2| &\geq \frac{1}{k-1} \cdot |B_0| \\
&\geq \frac{1}{k-1} \cdot \frac{nk-1}{nk} \cdot |S| \\
&\geq \frac{1}{k-1} \cdot \frac{nk-n}{nk} \cdot |S| \\
&= \frac{n}{nk} \cdot |S| \\
&\geq \frac{2}{nk} \cdot |S|
\end{aligned}$$

Therefore we have either

$$|C_1| \geq \frac{1}{nk} \cdot |S| \qquad \text{or} \qquad |C_2| \geq \frac{1}{nk} \cdot |S|$$

Let this larger bucket be C_α, then suppose

$$|C_\alpha| \leq \frac{nk-1}{nk}|S|$$

Then getting the response α will eliminate at least $|S|/(nk)$ possible solutions from S, and getting any other response will eliminate at least $|C_\alpha| \geq |S|/(nk)$ possible solutions. Therefore the guess $\overline{\sigma}$ is guaranteed to eliminate at least $|S|/(nk)$ solutions from S.

Otherwise we have

$$|C_\alpha| > \frac{nk-1}{nk}|S|$$

Then, this satisfies the conditions for Case 1 above, where we have a guess whose largest bucket is B_j with $j > 0$.

□

3. The minimax algorithm is a guessing strategy described by Knuth [**DK76**]. At each turn, the algorithm assigns each of the possible guesses a score equal to the minimum number of solutions that they would eliminate over all possible responses from the codemaker. It then picks the guess with the maximum score.

Corollary. The minimax algorithm on Mastermind with n spots, k colors, and no repeats takes at most

$$\frac{\log\left(\frac{k!}{(k-n)!}\right)}{\log\left(\frac{nk}{nk-1}\right)}$$

turns to find the hidden vector.

Proof. Each step in the minimax algorithm cuts the set of possible solutions S down by a factor of $(nk-1)/(nk)$ at each step. To complete the game, we need to cut S down by

$$\frac{1}{\left(\frac{k!}{(k-n)!}\right)}$$

Thus the required number of steps is

$$\log_{\left(\frac{nk-1}{nk}\right)}\left(\frac{1}{\frac{k!}{(k-n)!}}\right)$$

Which is equivalent to the claim. □

This gives us the following asymptotic bounds:

i. The minimax algorithm for Mastermind with n spots, k colors, and no repeats takes at most $O(n^2 k \log k)$ turns to find the hidden vector.

ii. In the same game where $n = k$, the minimax algorithm takes at most $O(n^3 \log n)$ turns to find the hidden vector.

Both of these asymptotic bounds are actually strict upper bounds, and each of them differs from the exact bound by a constant factor which approaches 1 as n approaches infinity.

3.3 Extension of Previous Proof to Repeated Colors

In order to replicate the bound of nk with repeated colors, we need to make the following assumption:[6] There exists a guess vector v such that all of the solutions in set S correspond to a single response r after making the guess v. Note that this is always satisfied if the solutions are any set of solutions remaining after any set of guesses in an actual game of Mastermind. We simply let the guess vector v be any guess vector that has already been guessed. Clearly, all solutions remaining correspond to whichever response was given to that guess vector, since guesses corresponding to any other response were eliminated. We will also prove the theorem for the special case of the very first guess separately.

Theorem 2. *In a game of Mastermind with n spots and k colors, for any set of remaining solutions that can actually be achieved during a game of Mastermind, there is always a guess for which any response will eliminate at least $1/nk$ of the remaining solutions.*

Proof. We will follow the same logic as taken in the previous proof, but will describe only the parts that differ from the previous proof.

1. Because of the problem statement, we will separately prove the base case where S is the set of all k^n possible solutions before any guesses have been made. In this case, we guess the vector $\{1, 1, \ldots, 1\}$. The bucket B_j has size

$$|B_j| = \binom{n}{j}(k-1)^{n-j}$$

So the greatest fraction of possible solutions remaining is

$$\max_j \left(\frac{\binom{n}{j}(k-1)^{n-j}}{k^n} \right)$$

When $k \geq n$, we have

$$\begin{aligned} \frac{\binom{n}{j}(k-1)^j}{k^n} &\leq \frac{n^{n-j}(k-1)^j}{k^n} \\ &\leq \frac{k^{n-j}(k-1)^j}{k^n} \\ &= \frac{(k-1)^j}{k^j} \\ &\leq \frac{k-1}{k} \end{aligned}$$

Therefore, we always eliminate a fraction at least $1/k \geq 1/nk$ of the solutions.

When $k < n$, it is difficult to find the size of one bucket compared to the entire set, so instead we will compare one bucket to the sum of itself and the next bucket. This ratio

[6] The proof in the previous section can be extended easily to general n and k: Substitute to show that there is always a guess that eliminates a factor of $1/n^2k$ possible solutions at each step. For a tighter bound with repeated colors, however, we need to make more restrictive assumptions.

will always be at least the ratio of one bucket compared to the entire set.

$$\begin{aligned}\frac{|B_{j+1}|}{|S|} &\leq \frac{|B_{j+1}|}{|B_j| + |B_{j+1}|} \\ &\leq \frac{1}{\frac{|B_j|}{|B_{j+1}|} + 1} \\ &\leq \frac{1}{\frac{\binom{n}{j}(k-1)^{n-j}}{\binom{n}{j+1}(k-1)^{n-j-1}} + 1} \\ &\leq \frac{1}{\frac{j+1}{n-j}(k-1) + 1}\end{aligned}$$

We plug in $k \geq 2$ (the entire problem is trivial when $k = 1$, as there is only 1 solution).

$$\begin{aligned}\cdots \quad &\leq \frac{1}{\frac{j+1}{n-j} + 1} \\ &\leq \frac{n-j}{n+1} \\ &\leq \frac{n}{n+1} \\ &= 1 - \frac{1}{n+1} \\ &\leq 1 - \frac{1}{nk}\end{aligned}$$

Additionally, for bucket 0, we have:

$$\begin{aligned}\frac{|B_0|}{|S|} &= \frac{(k-1)^n}{k^n} \\ &\leq \frac{k-1}{k} \\ &\leq 1 - \frac{1}{nk}\end{aligned}$$

So we have that the first guess of any game will always eliminate at least $1/nk$ of the remaining solutions.

2. The argument for the base case where $n = 1$ proceeds exactly as in the previous section.

3. The induction hypothesis remains almost the same: Assume that for $n - 1$ spots and any number of colors $\widehat{k}$ (where $n \geq 2$), we are guaranteed that there is a guess that eliminates a fraction at least $1/(n\widehat{k})$ of the solutions, given that there is a guess such that all solutions fall into the same bucket.

4. *Main Proof.*
   Let this bucket be j. From our assumptions $B_j = S$. Then we have two sub-cases as before:

   Case 1. If $j > 0$, we make the following adjustments to the previous proof.

i. For the first sub-case, we are now allowed repeats, so we can always let

$$\overline{\sigma} = \{\alpha, \sigma_2, \sigma_3, \ldots, \sigma_n\}$$

The proof of the rest of this sub-case follows.

ii. For the second sub-case, we proceed as the previous section did by considering D_{σ_1}, but now we are no longer guaranteed that the spots 2 through n of these vectors have no instance of σ_1 since colors can be repeated. Whereas before, we inducted on $n-1$ spots with $k-1$ possible colors, we now have to induct on $n-1$ spots still with k possible colors. This is fine, though, as by the induction hypothesis we can therefore eliminate

$$\begin{aligned}\frac{1}{(n-1)k} \cdot |D_{\sigma_1}| &\geq \frac{1}{(n-1)k} \cdot \frac{n-1}{n} \cdot |B_j| \\ &= \frac{1}{nk} \cdot |S|\end{aligned}$$

It is important to check that the induction hypothesis holds- namely, that there exists a guess of size $n-1$ such that all of the $(n-1)$-vectors we're considering produce the same response.

However, by our assumptions, σ is an n-vector such that all of the n-vectors fall into a single bucket j. Since every vector we're considering for induction is in the same sub-bucket D_{σ_1}, if we consider only spots 2 through n, all of these $(n-1)$-vectors will have exactly one fewer hit than they did as n-vectors, (as all of them had a hit in the first spot), so all of them fall into the same bucket (bucket $j-1$) for the guess $\{\sigma_2, \sigma_3, \ldots, \sigma_n\}$, so the condition is satisfied.

Case 2. If $j = 0$, we can no longer find a new $\overline{\sigma}$ and switch to Case 1, so instead we induct similarly to Case 1. We split the bucket up into sub-buckets as before. By pigeonhole principle, there must exist an α such that $|D_\alpha| \geq |B_j|/(k-1) \geq |S|/(nk)$ (since $j = 0$ we have no hits, so $|D_{\sigma_1}| = 0$). We then split into two subcases based on the size of D_α compared to the size of the entire bucket as we did in Case 1 and the proof proceeds in exactly the same manner.

□

4 Non-Adaptive Strategies

This section follows closely the reasoning of [**DS13**] as they analyze non-adaptive games. We perform an analysis of the Black-Peg, NO-REPEATS game to get a similar bound to their original Black-Peg lower bound using entropy. Then, we perform a small extension of one of their proofs to provide an upper bound on non-adaptive strategies for $k \geq n$.

4.1 Lower Bound on Black-Peg NO-REPEATS game

Theorem 3. *Every non-adaptive strategy for Mastermind with no repeats must submit at least $O(n \log k)$ queries to uniquely identify all possible hidden vectors.*

Proof. Our proof follows along similar lines as [**DS13**]. Let q_i be the i-th guess of our deterministic strategy. Let s be the smallest index such that all $k!/(k-n)!$ codes are uniquely determined by the responses to $q_1, q_2, \ldots, q_s$.

Consider a code Z sampled uniformly at random from the set of $k!/(k-n)!$ codes. Then consider the random variables $Y_i = \mathrm{eq}(Z, q_i)$ be the response (# black hits) to guess q_i. Then the vector $Y = (Y_1, Y_2, \ldots, Y_s)$ always uniquely determines, and is uniquely determined by, Z. So by Property 1, $H(Z) = H(Y)$. Since Z is a random variable with $k!/(k-n)!$ outcomes of equal probability, we know $H(Z) = \log_2(k!/(k-n)!)$.

By Property 2, we know

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

So now we bound $H(Y_i)$. By its definition,

$$H(Y_i) = -\sum_{x=0}^{n} \Pr[Y_i = x] \cdot \log_2(\Pr[Y_i = x]). \tag{1}$$

What is $\Pr[Y_i = x]$? This is 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 bucket x. So we get[7]

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

[7]In going from the first to the second line, we make the same combinatorial argument as before: Choosing x fixed points, choosing $n-x$ colors from the remaining $k-x$ possibilities and permuting those $n-x$ positions overcounts all vectors with no repeated colors and at least x fixed points and thus overcounts the number of vectors with no repeats and exactly x fixed points (which is $|B_x|$).

We want to plug this upper bound into equation 1. For $\alpha < 1/e$, $f(\alpha) = -\alpha \log_2 \alpha$ is an increasing function. 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, we will use the upper bound $f(\alpha) \leq 1/(e \log 2)$.

$$\begin{aligned}
H(Y_i) &\leq \frac{3}{e \log 2} + \sum_{x=3}^{n} -\frac{1}{x!} \cdot \log_2(\frac{1}{x!}) \\
&= \frac{3}{e \log 2} + \sum_{x=3}^{n} \frac{\log_2(x!)}{x!} \\
&\leq \frac{3}{e \log 2} + \sum_{x=3}^{n} \frac{x \log_2 x}{x!} \\
&\leq \frac{3}{e \log 2} + \sum_{x=3}^{n} \frac{x(x-1)}{x!} \\
&= \frac{3}{e \log 2} + \sum_{x=3}^{n} \frac{1}{(x-2)!} \\
&\leq \frac{3}{e \log 2} + e - 1 \\
&< 4
\end{aligned}$$

Combining this with the summation above gives $H(Y) \leq 4s$, so we get

$$\begin{aligned}
H(X) &\leq H(Y) \\
\log_2 \left(\frac{k!}{(k-n)!} \right) &\leq 4s \\
s &\geq \frac{1}{4} \log_2 \left(\frac{k!}{(k-n)!} \right)
\end{aligned}$$

This gives us a lower bound of $O(n \log k)$ turns for any non-adaptive strategy for Mastermind with no repeats. □

4.2 Upper Bound for Original Game

First, we note that [**DS13**] showed the existence of an $O(n \log n)$ strategy for $k = n$. When $k > n$, we add 'imaginary spots' until $n = k$, and fill them in with any arbitrary colors. Then, when we make our guesses, we adjust the responses accordingly to account for the imaginary spots. By [**DS13**]'s result, we have a strategy that takes $O(k \log k)$ guesses.

5 Playing Mastermind with Many, Many Colors

5.1 Adaptive Games

For adaptive games, the analysis is relatively easy. Say a strategy makes fewer than $\lfloor k/n \rfloor$ guesses. Then there exist codes where the responses to each of these guesses are all 0 hits, because there are at least n colors left unguessed at every step. Therefore, the strategy has not guessed the secret code yet, so it needs at least $\lfloor k/n \rfloor$ guesses to find the hidden code. But we

can achieve this bound to within a constant factor for large numbers of colors. Simply partition the colors into groups of n, and guess each group. Since there are at most n colors in the hidden vector and each color is in 1 group, after this process (which takes $\lceil k/n \rceil$ guesses), there are at most n groups of n colors that got no hits, so the color pool is reduced to n^2. From [**DS13**] we have a strategy that takes $O(n^2)$ at this point, so for $k/n \geq n^2$, i.e. $k \geq n^3$, the lower bound and this upper bound agree to within a constant factor.

5.2 Non-Adaptive Games

For the original non-adaptive game, we have the upper bound of $O(k \log k)$ turns. For the NO-REPEATS version, we default to the nk upper bound. Interestingly, for $k \geq e^n$ the nk upper bound beats the $k \log k$ upper bound for the original game. As for lower bounds, we have the entropy bounds of $O(n \log k)$ but for large k relative to n these are awful bounds. Instead, we give an $O(k)$ lower bound as follows:

Assume for the sake of contradiction that in some strategy that distinguishes all possible hidden vectors, there exists a pair of colors such that neither color was guessed in the first two spots (WLOG let these colors be 1 and 2). Then it would be impossible to distinguish between the hidden vectors $(1, 2, \ldots, n)$ and $(2, 1, \ldots, n)$, which contradicts the assumptions. Therefore, for any winning strategy, at least $k - 1$ colors were guessed in these two spots. Since we can only guess two colors in those spots per turn, this gives us the lower bound of $(k-1)/2 = O(k)$ turns necessary to beat the game.

This $O(k)$ lower bound starts to beat the $O(n \log k)$ when $k > n \log n$. So we have the lower and upper bounds of k and nk, respectively, which only differ by the "constant" n, which is relatively tiny in the cases we're considering.