Playing Mastermind with Wordle's Feedback

Renyuan Li* Shenglong Zhu*

September 28, 2022

Abstract

In this paper, we define a new code-breaking game called Clear Mastermind, combining two similar games – Mastermind and Wordle. The only difference between Clear Mastermind and Mastermind is that the rule of giving more precise feedback in Wordle is applied in the former game. In Clear Mastermind, the feedback contains the positions of the letters the codebreaker guessed correctly and the positions of letters that appear in the answer but in different positions. Define $F(k,n)$ as the least guessing number that the codebreaker can find the answer in the Clear Mastermind with n positions and k colors. We present a proof that $F(k,n) = k$ in the case $k \le n$. In the case that $k = c \cdot n, c \in N^*$, we prove that $F(k,n) = c + n - 1$. In other cases, we give a bound of $F(k,n)$.

Keywords: Mastermind, Wordle, Combinatorial problems

1 Introduction

1.1 Background

Mastermind is a famous code-breaking game for two players. One player, the codemaker, chooses a secret color combination of 4 pegs from 6 possible colors in the game. The goal of another player, the codebreaker, is to identify the answer in as few queries as possible, exploiting information from the feedback. The query is also a color combination of 4 pegs from 6 possible colors. The feedback is the number of pegs the codebreaker guessed correctly and the number of pegs of the right color but not at the right position. The game has been studied from a computational perspective before [2].

Wordle is a single-player code-breaking game, which is similar to Mastermind. In the game, the computer chooses a five-letter word from English instead of a color combination as the answer. Like Mastermind, the player aims to find the answer in as few queries as possible and gets feedback from the computer, while the query is required to be a five-letter word from English. The feedback in Wordle is more precise than in Mastermind. In Wordle, the feedback contains the exact positions of the letters the player guessed correctly as well as the positions of letters that appear in the answer but in different positions. The online game has gained popularity and was proven to be NP-hard [1].

In this paper, we deal with *Clear Mastermind*, a combination of two code-breaking games – Mastermind and Wordle. Intuitively, Clear Mastermind is Mastermind with Wordle's feedback.

We define *Clear Mastermind* with $n \ge 1$ positions and $k \ge 1$ codes as a two-player game as follows. Let us denote by $[k]$ the set $\{1, 2, \cdots, k\}$ of positive integers. One player, the codemaker,

*Department of Mathematics, Nanjing University, China. Email: {191098106|191098372}@smail.nju.edu.cn

initially chooses a secret string $y = (y_1, y_2, \cdots, y_n) \in [k]^n$ to be the answer in the game, y_i is the code on the ith position of the answer (ith code of the answer for short). The other player, the codebreaker, aims to find the answer by guessing a sequence of strings. In each round, the codebreaker guesses a string $x \in [k]^n$. The codemaker replies with feedback $\pi(x, y) \in [\text{G,Y,B}]^n$ following specific rules. The rules will be introduced in the next section. If the codebreaker's guess equals the answer in the ith round, we say that the codebreaker finds the answer in i rounds or that the codebreaker solves the puzzle in i rounds.

1.2 Rules

The feedback after each round is given by the following algorithm.

Algorithm 1: Rules of feedback $\pi(x, y)$

Input:
guess $x = (x_1, x_2, \cdots, x_n) \in [k]^n$;
answer $y = (y_1, y_2, \cdots, y_n) \in [k]^n$.
Output:
feedback $\pi(x, y) \in [\text{G,Y,B}]^n$.

$S \leftarrow \{i \in [n] | x_i = y_i\}$.
foreach $i \in S$ **do**
  $\pi_i(x, y) \leftarrow \text{G}$
$N \leftarrow \{y_i | i \in [n] - S\}$.
foreach $j \in [n]$ **do**
  if $j \notin S$ **then**
    if $x_j \in N$ **then**
      $\pi_j(x, y) \leftarrow \text{Y}$, $S \leftarrow S \cup \{j\}, N \leftarrow \{y_i | i \in [n] - S\}$.
    if $x_j \notin N$ **then**
      $\pi_j(x, y) \leftarrow \text{B}$

Here are two examples:

- When the answer is (1,1,2,3,3), the guess is (1,1,1,2,3), then the feedback would be (G,G,B,Y,G).
- When the answer is (2,1,3,4,2), the guess is (2,2,2,2,4), then the feedback would be (G,Y,B,B,Y).

We introduce 'answer set' P_i to represent the set of strings that are potential answers for the codebreaker after i rounds. It is easy to see that $P_{i+1} \subset P_i, \ \forall i \in N$.
Here is an example to explain the answer set:

- For $k = 3, n = 3$, the codebreaker guesses (1,2,3) in the first round and the feedback is (G,G,B), then the answer set P_1 would be $\{$(1,2,1),(1,2,2)$\}$. In the second round, the codebreaker guesses (1,2,2), and if the feedback is (G,G,B), then the answer set P_2 would be $\{$(1,2,1)$\}$, which means that the answer is (1,2,1).

1.3 Our Contribution

Define $F(k, n)$ as the least guessing number that a codebreaker can find the answer in the Clear Mastermind with n positions and k codes.

In the Clear Mastermind, the codebreaker can guess by using the following strategy: In the jth ($j < k$) round, the codebreaker guesses $(j, j, \cdots, j) \in [k]^n$. After the codebreaker receives the feedback in the $(k-1)$th round, the answer set P_{k-1} contains only one string. So the codebreaker can solve the problem in the kth round, which means $F(k,n) \leq k, \forall k \in N$.

In this work, we prove the following results:

Theorem 1. *In the case that $k \leq n$,*

$$F(k,n) = k$$

Theorem 2. *In the case that $k = c \cdot n, c \in N^*$,*

$$F(k,n) = \frac{k}{n} + n - 1 = c + n - 1$$

Theorem 3. *In the case that $k = c \cdot n + b, c \in N^*, b \in \{1, 2, \cdots, n-1\}$,*

$$c + n - 1 \leq F(k,n) \leq c + n$$

2 The Case $k \leq n$

In the case $k \leq n$, in order to prove Theorem 1, it is sufficient to prove $F(k,n) \geq k$, since

$$F(k,n) \leq k, \forall k \in N$$

The feedback includes 'Y', which is challenging to deal with. However, we should note that often in this work, if the codebreaker knows the amounts of each code in the answer, feedback of 'Y' would not be needed. Giving the codebreaker the feedback $\pi(x,y) \in [\text{G,Y,B}]^n$ is equivalent to giving the codebreaker the feedback $\tau(x,y) \in \{✓,✗\}^n$ – for answer $y \in [k]^n$ and guess $x \in [k]^n$, feedback $\tau_i(x,y)$ is '✓' if $x_i = y_i$ and is '✗' if $x_i \neq y_i$. Mathematically, the codebreaker can obtain one feedback by using another one and the guess.

Denote the amounts of each code by a string $w = (w_1, w_2, \cdots, w_n) \in N^n$ with

$$\sum_{i=1}^{n} w_i = n$$

where w_i is the amount of code 'i' in the answer. It is clear that if the codebreaker knows the feedback $\pi(x,y)$, the feedback $\tau(x,y)$ can be obtained by changing 'G' to '✓' and changing 'Y' and 'B' to '✗'. Otherwise, if the codebreaker knows the feedback $\tau(x,y)$, the feedback $\pi(x,y)$ can be obtained by using **Algorithm 2**.

In the case that $n = k$ and the answer is known to be a permutation of $\{1, 2, \cdots, k\}$. The codemaker alternatively gives the codebreaker feedback containing only '✓' and '✗'. Define $G(k)$ as the least guessing number that a codebreaker can find the answer in this case with k codes.

In the case $k = n$, we know that $F(k,k) \geq G(k)$.

Moreover, let $z = (z_1, z_2, \cdots, z_n)$ be the answer in the game in the case that $k < n$. We, as the codemaker, set $z_j = 1$ if $j > k$, and set $(z_1, z_2, \cdots, z_k)$ be a permutation of $\{1, 2, \cdots, k\}$. Let the codebreaker know the information above, and the game uses the feedback $\tau(x,y)$. It is easy to see that $F(k,n) \geq G(k)$.

Thus, Theorem 1 can be deduced from the following proposition.

Proposition 1. $G(k) \geq k, \forall k \in N^*$.

Algorithm 2: Obtain the feedback $\pi(x, y)$ by using feedback $\tau(x, y)$

Input:

feedback $\tau(x, y) \in \{✓,✗\}^n$;
the amounts of each code in the answer $w = (w_1, w_2, \cdots, w_n) \in N^n$;
guess $x = (x_1, x_2, \cdots, x_n) \in [k]^n$.

Output:

feedback $\pi(x, y) \in$ [G,Y,B]^n.

$S \leftarrow \{i \in [n] | \tau_i(x, y) =$ '✓'$\}$.
foreach $i \in S$ **do**
  $\pi_i(x, y) \leftarrow$ G;
  $q \leftarrow x_i$;
  $w_q \leftarrow w_q - 1$.
foreach $j \in [n]$ **do**
  if $j \notin S$ **then**
    $q \leftarrow x_j$;
    if $w_q > 0$ **then**
      $\pi_j(x, y) \leftarrow$ Y, $w_q \leftarrow w_q - 1$.
    if $w_q = 0$ **then**
      $\pi_j(x, y) \leftarrow$ B

Before the proof, let us have a deeper insight into the game.

For any codebreaker, we, as the codemaker, aim to find at least a string and to prove that the codebreaker can not solve the puzzle in $k-1$ rounds. In fact, as the game goes on, we do not have to choose the answer in the game because the codebreaker's guessing strategy only depends on her previous guesses and the feedback. The state of the game does not change if we choose another string in the current answer set to be the answer in this game.

As a result, the codemaker only needs to give feedback by using a specific strategy after each round and ensure the answer set is not empty. Intuitively, the specific strategy gives '✗' by the Greedy Algorithm.

Before giving the specific strategy, we extend the definition of the 'answer set'. We define 'Q-answer set' $Q_{i,j}$ as the set of strings to be potential answers considering the guesses and the feedback in the initial i rounds, and the initial j positions of the guesses and the feedback in the $(i+1)$th round. Notice that $P_i = Q_{i-1,n} = Q_{i,0}$. An example is shown below:

- In a Mastermind game with $k = 3, n = 3$, the codebreaker guesses (1,1,1) in the first round and the feedback is (B,G,B). In the second round, the codebreaker guesses (2,1,2), and if the feedback is (G,G,B), then the results are:

$$
\begin{aligned}
&P_1 = Q(1,0) = \{(2,1,2), (2,1,3), (3,1,2), (3,1,3)\} \\
&Q(1,1) = Q(1,2) = \{(2,1,2), (2,1,3)\} \\
&Q(1,3) = P_2 = \{(2,1,3)\}
\end{aligned}
$$

Now we give the specific strategy.

Strategy 1. We, the codemaker, give the feedback $\tau = (\tau_1, \tau_2, \cdots, \tau_n)$ by the following method after receiving the guess from the codebreaker in the $(i+1)$th round: Set the first position of the feedback τ_1 as '✗'. Calculate the Q-answer set $Q_{i,1}$. If $Q_{i,1}$ is empty, change the feedback's first position to '✓', and move on to the second position of the feedback τ_2; else, move on to τ_2 directly. For the rest positions of the feedback, the processes are similar. **Algorithm 3** shows the detail of the strategy.

Algorithm 3: Strategy of giving feedback

Input:

current answer set $P_r = Q_{r+1,0} \subset [k]^n$;
guess $x = (x_1, x_2, \cdots, x_n) \in [k]^n$.

Output:

feedback $g(P_r, x) \in \{✓,✗\}^n$.

foreach $i \in [n]$ **do**
  $S \leftarrow \{a \in Q_{r+1,i-1} |$ the ith code of a is x_i $\}$
  if $S = Q_{r+1,i-1}$ **then**
    $g_i(P_r, x) \leftarrow$ ✓
    $Q_{r+1,i} \leftarrow Q_{r+1,i-1}$
  if $S \subsetneqq Q_{r+1,i-1}$ **then**
    $g_i(P_r, x) \leftarrow$ ✗
    $Q_{r+1,i} \leftarrow Q_{r+1,i-1} - S$

Proposition 2. In a Clear Mastermind game with k positions and k colors, the codebreaker knows that the answer is a permutation of $\{1, 2, \cdots, k\}$. Then the codemaker can give feedback by using Strategy 1, so the answer set always contains more than one string after $k-2$ rounds, i.e. $|P_{k-2}| > 1$.

Remark 1. It can be inferred from Proposition 2 that any codebreaker cannot solve the puzzle in $k-1$ rounds facing Strategy 1. Consequently, Proposition 1 can be deduced by Proposition 2.

Then we are going to prove Proposition 2. We first do some preparations.

We use a binary matrix $A^{(r)} = (a_{ij}^{(r)})_{n \times n}$ to track the codebreaker's guesses and the codemaker's feedback (tracking matrix for short) in the game after r rounds. Initially, the values in the matrix are all '1', which means $a_{ij}^{(0)} = 1, \forall i, j \in [n]$.

In the rth round, the codebreaker gives a guess $x = (x_1, x_2, \cdots, x_k) \in [k]^k$. Let us denote the feedback by $w = (w_1, w_2, \cdots, w_k) \in \{✓, ✗\}^k$. For each $i \in [k]$, if w_i = '✗' , then set $a_{ix_i}^{(r)}$ as '0'. Other elements in $A^{(r)}$ remain the same as the corresponding elements in $A^{(r-1)}$.

For example, when $k = n = 4$, a codebreaker guessed (4,2,3,1), and the feedback is (✓,✓,✗,✗), then the matrix $A^{(1)}$ is

code \ position	1	2	3	4
1	1	1	1	0
2	1	1	1	1
3	1	1	0	1
4	1	1	1	1

Lemma 1. *Given a square binary matrix with a dim of $n (n \geq 2)$, if the matrix has a non-zero permanent and there are precisely two '1' in each of the columns, then the permanent of the matrix is larger than one.*

The permanent of a square binary matrix $A = (a_{ij})_{n\times n}$ is defined as:

$$\text{perm}(A) = \sum_{\sigma\in S_n}\prod_{i=1}^{n} a_{i,\sigma(i)}$$

where S_n is a symmetric group over all permutations of the numbers $1, 2, \cdots, n$, $\sigma(i)$ is the ith value in σ.

Remark 2. The permanent of the tracking matrix $A^{(r)}$ is not smaller than the number of strings in the answer set P_r. Every string in the answer set P_r maps to a unique permutation ϕ in S_n that satisfies $a^{(r)}_{i,\phi(i)} = 1, \forall i \in [n]$.

Proof. We use induction on n.

If $n = 2$, trivial.

Suppose that for $2 \leq n < n_0 (n_0 > 2)$, the lemma holds. For $n = n_0$, if there are at least two '1' in each row, then there are exactly two '1' in each row and each column. Because the permanent of the matrix is non-zero, we assume $\sigma_0 \in S_n$ that satisfies $a_{i,\sigma_0(i)} = 1$ for each $i \in [n]$. Then we find another '1' in the ith row and denote it by $a_{i,\sigma_1(i)}$. From the fact that there are exactly two '1' in each column and $\sigma_0 \in S_n$, we know $\sigma_1 \in S_n$. So $\text{perm}(A) \geq 2$. Else, we assume that in the pth row, there are less than two '1'. The permanent is a positive number. Thus, there is precisely one '1' in the pth row. Assume that $a_{p,q} = 1$, then

$$\text{perm}(A) = a_{p,q} \cdot \text{perm}(A_0) = \text{perm}(A_0)$$

where A_0 is the submatrix formed by deleting the pth row and qth column of A, a square binary matrix with a dim of $n - 1$. By induction, $\text{perm}(A) = \text{perm}(A_0) \geq 2$. □

Corollary 1. *Given a square binary matrix with a dim of* $n(n \geq 2)$*, if the matrix has a non-zero permanent and there are no less than two '1' in each of the columns, then the permanent of the matrix is larger than one.*

Lemma 2. *If the codemaker gives feedback by using Strategy 1, then the value of the permanent of the tracking matrix is equal to the cardinal number of the answer set.*

Proof. By the remark after Lemma 1, we know that the permanent of the tracking matrix is not smaller than the number of strings in the answer set. The rest is to prove that the permanent of the matrix is not larger than the number of strings in the answer set by using Strategy 1.

Equivalently, if $\varphi \in S_n, r \in N$ satisfies that $a^{r}_{i,\varphi(i)} = 1, i = 1, 2, \cdots, n$, we are going to prove $s_\varphi = (\varphi(1), \varphi(2), \cdots, \varphi(n))$ is in the answer set P_r. The answer set $P_0 = \{s_\sigma = (\sigma(1), \sigma(2), \cdots, \sigma(n)) | \sigma \in S_n\}$ contains s_φ before the codebreaker starts making queries.

We give the proof by contradiction. If $s_\varphi \notin P_r$, we assume that s_φ is removed from the answer set in the mth round$(m \leq r)$, which means $s_\varphi \in P_{m-1}$ and $s_\varphi \notin P_m$.

Let the guess in the mth round and the feedback in the mth round be $x = (x_1, x_2, \cdots, x_n)$ and $\tau = (\tau_1, \tau_2, \cdots, \tau_n)$, respectively. Since

$$a^{(r)}_{i,\varphi(i)} = 1, i = 1, 2, \cdots, n$$

we know that

$$a^{(m)}_{i,\varphi(i)} = 1, i = 1, 2, \cdots, n \tag{1}$$

Let $S = \{i | x_i = \varphi(i)\}$, then from Equation (1), $\tau(i)$ must be '✓', $\forall i \in S$. Then τ_j is '✗' according to Strategy 1, $\forall j \in [n] - S$. So $s_\varphi \in P_m$. Contradiction! So $s_\varphi \in P_r$.

□

Finally, we give the proof of Proposition 2.

Proof. After $k-2$ rounds, the tracking matrix $A^{(k-2)}$ satisfies the conditions in Corollary 1, so the permanent of the matrix is larger than one, which means the answer set P_{k-2} contains more than one string by Lemma 2. Thus Proposition 2 is proved. □

By Remark 1, Proposition 1 is proved, then Theorem 1 holds, i.e.

$$F(k,n) \geq k, \forall k \leq n.$$

3 The Case $k \geq n$

For the case $k \geq n$, we are going to prove Theorem 2, i.e.

$$F(k,n) = \frac{k}{n} + n - 1 = c + n - 1, k = c \cdot n, c \in N^*$$

and since

$$F(m,n) \leq F(m+1,n), \quad \forall m,n \in N^*.$$

then Theorem 3 holds, i.e.

$$c + n - 1 \leq F(k,n) \leq c + n, k = c \cdot n + b, b, c \in N^*, b < n.$$

3.1 Preparation

Similar to the analysis in Section 2, the codemaker only needs to give feedback by using a specific strategy after each round and ensure the answer set is not empty.

Lemma 3. *For the Clear Mastermind with k positions and k codes, if there exists $i,j \in \{1,2,\cdots,n\}$, the codebreaker knows that the ith code is not j and that the answer is a permutation of $\{1,2,\cdots,k\}$, then the codebreaker can solve the puzzle within $k-1$ rounds.*

Proof. Without loss of generality, we assume that $i = j = 1$. Then in the rth round($r < k-1$), the codebreaker guesses $((r+1), r, r, \cdots, r) \in [k]^n$ until the first position of the feedback is 'G'. If it appears, the codebreaker does not need to consider code 'r+1' from the next round, and Lemma 3 holds. Otherwise, after $k-2$ rounds, the answer set P_{k-2} contains only one string. Lemma 3 holds. □

3.2 A Special Case

We consider $k = c \cdot n, c \in N^*$ in the following section. We aim to show that

$$F(k,n) = \frac{k}{n} + n - 1 = c + n - 1$$

Proof. On the one hand, in the initial $c-1$ rounds, the codemaker could give the feedback only containing 'B' because there are $c \cdot n$ kinds of codes. So the best case for the codebreaker after $c-1$ rounds is playing a Mastermind game with n positions and n codes, which needs n more rounds. So

$$F(k,n) \geq c + n - 1$$

On the other hand, the codebreaker could guess $((r-1)\cdot n+1, (r-1)\cdot n+2, \cdots, r \cdot n)$ in the rth round ($r \leq c$). Then the codebreaker knows the code that appears in the answer after c rounds.

If there are less than n kinds of codes appearing in the answer, the codebreaker needs no more than $n-1$ rounds to solve the puzzle since

$$F(k,n) \le n-1, \forall k < n$$

Else, there are n kinds of codes appearing in the answer. Without loss of generality, we assume that code '1' appears in the answer. If the first code of the answer is not '1', then by Lemma 3, the codebreaker needs no more than $n-1$ rounds to solve the puzzle. Otherwise, it is easy to check that the codebreaker also needs no more than $n-1$ rounds to solve the puzzle. □

Above all, we prove that

$$F(k,n) = \frac{k}{n} + n - 1 = c + n - 1, k = c \cdot n, c \in N^*$$

3.3 Other Cases

If $\frac{k}{n} \notin N^*$, which means that $k = c \cdot n + b, b, c \in N^*, b < n$, then we know that

$$c + n - 1 \le F(k,n) \le c + n, k = c \cdot n + b, b, c \in N^*, b < n$$

because

$$F(k,n) \le F(k+1,n), \quad \forall k, n \in N^*.$$

And

$$F(c \cdot n, n) = c + n - 1$$

so Theorem 3 proved.

4 Conclusion and Further Work

In this work, we introduce Clear Mastermind, a combination of Mastermind and Wordle. We explore the least guessing number that a codebreaker can find the answer in the game according to its two parameters – the number of codes and the length of the answer. In cases that $k \le n$ or $k = c \cdot n, c \in N^*$, we find the least guessing number by enhancing the conditions in the game and proposing a feedback strategy with a critical property. Moreover, in other cases, we give a bound of the least guessing number.

In the case $k = c \cdot n + b, b, c \in N^*, b < n$, we simulated the game by using the computer, and we got the following results:

	$F(k,n) = n$	$F(k,n) = n+1$
$n = 2$		$F(3,2) = 3$
$n = 3$	$F(4,3) = 3$	$F(5,3) = 4$
$n = 4$	$F(5/6/7,4) = 4$	
$n = 5$	$F(6/7/8/9,5) = 5$	

We speculate that

$$F(2n-1,n) = n, \forall n > 3$$

It is reasonable because when n is small, the feedback of 'G' provides little information about the other codes of the answer. If the feedback contains two types of 'G', representing whether the code exists in the other positions, then $F(3,2)$ and $F(5,3)$ would be two and three, respectively.

Another potential extension that follows from the study is the relationship between $F(k+n,n)$ and $F(k,n)$ when $k = c \cdot n + b, b, c \in N^*, b < n$. It is trivial that $F(k+n,n) \ge F(k,n)+1$. However, we do not know whether $F(k+n,n) \le F(k,n)+1$ is true.

Acknowledgments

We thank our supervisor, Professor Liang Yu, for his insightful suggestions and encouragement. He pointed out some mistakes in a previous version of this paper and guided us to find another proof.

References

[1] D. Lokshtanov and B. Subercaseaux. Wordle is np-hard. *arXiv preprint arXiv:2203.16713*, 2022.

[2] A. Martinsson and P. Su. Mastermind with a linear number of queries. *arXiv preprint arXiv:2011.05921*, 2020.