The Perfect Optimization of "The Password Game"

Ryan King

I. Introduction

A. About "The Password Game"

"The Password Game" (neal.fun/password-game) is a short internet game made by Neal Agarwal in 2023. It is a joke on the commonality of websites that have many criteria for what is considered a "valid password". The goal of the game is to create a very complex password that satisfies 35 different rules, some of which change every time you play the game.

B. Optimization

In this paper, I intend to research the shortest possible solution to "The Password Game". I will consider a scenario where every random variable could be perfectly controlled by reloading the game for better random rules. In this scenario, we will find what length the absolute shortest a password can be while still satisfying all the constraints given by the rules of "The Password Game".

C. The Rules

In order to understand how to calculate the shortest password, we first have to categorize the rules to determine which ones are important and how they affect the problem. To get a basic understanding of the format of the game, I recommend you visit the website (neal.fun/password-game)[1] for a few minutes to help picture the format of the game to better understand the list of rules.
All 35 rules are as follows:

1) Your password must be at least 5 characters.
2) Your password must include a number.
3) Your password must include an uppercase letter.
4) Your password must include a special character.
5) The digits in your password must add up to 25.
6) Your password must include a month of they year.
7) Your password must include a roman numeral.
8) Your password must include one of our sponsors. (Pepsi, Starbucks, or Shell)
9) The roman numerals in your password should multiply to 35.
10) Your password must include this CAPTCHA. (image provided)
11) Your password must include today's Wordle answer.
12) Your password must include a two letter symbol from the periodic table.
13) Your password must include the current phase of the moon as an emoji.
14) Your password must include the name of this country. (from interactive Google Street View widget)
15) Your password must include a leap year.
16) Your password must include the best move in algebraic chess notation. (Image of chess puzzle provided)
17) 🥚 This is my chicken Paul. He hasn't hatched yet. Please put him in your password and keep him safe. (Add a egg emoji to your password.)
18) The elements in your password must have atomic numbers that add up to 200.
19) All the vowels in your password must be bolded.
20) Oh no! Your password is on fire. Quick, put it out! (Delete the first emoji from your password and replace it with the letter it initially replaced.)
21) Your password is not strong enough 🏋️ (Add 3 weightlifter emojis to your password.)
22) Your password must contain one of the following affirmations: "I am loved" or "I am worthy" or "I am enough".
23) Paul has hatched! Please don't forget to feed him. He eats three caterpillars every minute. (Paste one caterpillar emoji into the password every 20 seconds.)
24) Your password must include the URL of a YouTube video of this exact length. (number of seconds between 3:00 and 36:20 is given)
25) Pick two letters that you will no longer be able to use.
26) Your password must contain twice as many italic characters as bold.
27) At least 30% of your password must be in the Wingdings font.
28) Your password must include this color in hex. (image of a singular color provided)
29) All Roman numerals must be in Times New Roman.
30) The font size of every digit must be equal to its square.
31) Every instance of the same letter must have a different font size.
32) Your password must include the length of your password.
33) The length of your password must be a prime number.
34) This rule is skipped.
35) Your password must include the current time.

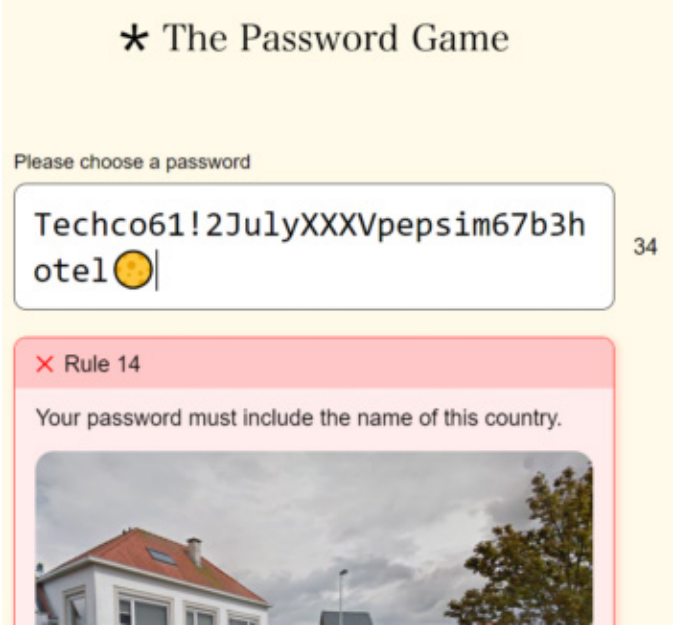

Fig. 1. A screenshot of someone attempting "The Password Game"

II. Materials And Methods

A. Materials

The main tool I will be using is the Selenium WebDriver library for Python. It allows me quickly automate "The Password Game" so I can get to later rules to investigate them. It also allows me to quickly check all the different possible states by quickly reloading different rules. The calculations for the optimal solution will also be done in Python for convenience. I am also using basic website inspect tools to look beneath the hood of certain randomized rules.

B. Categorizing Rules and Constraints

In this section, we will be going in depth into how we can categorize all the rules of the password game. In addition, we will cover other intricacies of the such as where capitalization is required, and what addition rules to add to help make our definition of a "perfect" solution more well-defined.
There are four categories I will be sorting our set of rules into.

1) Random rules: These are the rules that will have a random state every time you play "The Password Game". The random state is either based on a random number generator that changes value every time you reload the page, or it depends on the current time and date state of your computer. For our hypothetical, these rules can be controlled by reloading the game until getting the wanted set of answers.
2) Constraint rules: These are the rules that constrain what we can have in our password. Certain characters, words, numbers, etc. are not allowed or have limits. These rules combined together define the limits of the password.
3) Required content rules: Any rule that isn't random and doesn't set a constraint, but just requires certain characters to be in the password is considered to be in this group. Individually these rules don't matter much to our algorithm, but they will change the length of the password, and so they must be considered.
4) Inconsequential rules: This group contains any rules that either do not need to be considered because they are automatically a subset of a different rule, or are purely cosmetic in terms of formatting, and thus do not change the actual contents of the password.

Random Rules (random/changing states):

- Rule 10 (the CAPTCHA solution)
- Rule 11 (Today's Wordle answer)
- Rule 13 (Current moon phase as an emoji)
- Rule 14 (Name the country from Google Street View)
- Rule 16 (Best chess move)
- Rule 24 (YouTube video link)
- Rule 28 (hex color code)
- Rule 35 (Current time in HH:MM)

Constraint Rules:

- Rule 5 (Digits must sum to 25)
- Rule 9 (Roman numerals must multiply to 35)
- Rule 18 (Atomic numbers must sum to 200)
- Rule 25 (Pick two letters you can no longer use. A.k.a., you can only use 24 unique letters.)
- Rule 32 (Password length must be contained in the password as a number)
- Rule 33 (Password length must be a prime number)

Required Content Rules (requiring certain characters):

- Rule 6 (a month)
- Rule 8 ("pepsi", "starbucks", "shell")
- Rule 15 (a leap year)
- Rule 17 (the egg emoji (later becomes chicken emoji))
- Rule 21 (3 weightlifting emojis)
- Rule 22 ("I am loved", "I am worthy", or "I am enough")

Inconsequential Rules (cosmetic or redundant):
Rules 1, 2, 3, 4, 7, 12, 19, 20, 23, 26, 27, 29, 30, 31, and 34.

Important Notes

For the YouTube video link option, we will have to make an assumption. Currently, out of all the possible YouTube links ("youtu.be/" followed by 11 base-64 characters), only 1 in 100 billion lead to valid videos that have been uploaded. In addition, for the 1 billion YouTube videos that do exist, it is impossible to find what their URLs are, and thus impossible to check for overlap between video URLs and other parts of our password.

Since there is no way to find all the YouTube videos URLs, in our theoretical examples we will have to make an assumption. We will say that any YouTube video link of the form "youtu.be/XXXXXXXXXXX" where the 11 X's are base-64 digits (YouTube video ID system), will be a valid rule 24 solution and will have a length between 180 and 2180 seconds (3:00-36:20). This is obviously not a perfect solution, as the actual subset of valid video links that satisfy the rule is much smaller. However, we have no way to determine the actual valid subset, so we will have to define rule 24 like this for our calculations.

Rule 32 doesn't fit super well into any one category, however, I add it to the constraint rules because it affects rule 5, and it doesn't have a state that cannot be controlled. It variably decreases the number that digits must sum to for rule 5, but it isn't as straightforward as required content, hence why I am putting it in this category.

Capitalization Rules

Out of the rules that require entering letters, some of them are case-sensitive. This is important to keep track of because it changes how we can combine different words in our algorithm.
The case-sensitive rules are:

- Rule 7 and 9: Roman numerals must be capitalized
- Rule 10: All letters found in the CAPTCHA solution must be lowercase.
- Rule 12 and 18: Periodic table symbols must be capitalized properly.
- Rule 16: Algebraic chess notation solution must be capitalized properly.

Random Sets

Many of the random options or rules that seem to have infinite possibilities, actually have a finite amount of solutions hard coded into the game. Using some clever searching methods, we can find all the rules that are actually just sets of predefined possibilities.

- Rule 10: The CAPTCHA solutions, while seemingly randomly generated, are actually just chosen from a list of 149 different CAPTCHAs. To figure this out, I created a bot that solves the first 9 rules, and then constantly refreshes the CAPTCHA thousands of times to try and encounter all the possible CAPTCHAs. Code for this program can be found on my GitHub page.
- Rule 11: To figure out all the values that the Wordle could be, we just need all the previous Wordle solutions. This can be tracked pretty easily through Wordle, or through the API on "The Password Game"'s website[1] that grabs the Wordle for any specific day. (neal.fun/api/password-game/wordle?date=YYYY-MM-DD). Using this second method, we find 927 total unique Wordle solutions, and also realize that Wordle answers are prepared 1 month into the future.
- Rule 13: By testing, we can figure out there are only 8 moon emojis that we need to consider. However, because the emojis don't interact with any other part of the password, this is unimportant as it will always take up one character.
- Rule 14: By rerunning the page many times, we can discover that only 65 countries are possible answers, much less than the number of countries in the world. A user on GitHub[2] has already compiled this list, and by we can quickly check it ourselves by running the game a few hundred times to check that all solutions are contained within the list provided.
- Rule 16: Using the same method, we find that there are 193 possible chess board randomizations. However, when getting the solutions for all 193 board layouts, we find that there are only 132 unique solutions in algebraic notation that could be in the game.
- Rule 28: For the hex color, to avoid getting a color that makes rule 5 impossible, only hex colors with digits that add up to 10 or less are possible. This means it is only possible to get 3,634,246 different colors. This substantially reduces the number of random possible hex colors, although for our optimizations, we are assuming we can reload the hex color as many times as needed anyways.
- Rule 35: For the time in HH:MM, it is worth noting it has to be in 12-hour time, and thus there are only 720 possibilities for the time, instead of double that.

C. Calculation Methods

To calculate the minimum possible length a password could be with all variables controlled for, we have to track every rule that adds characters, and all possible overlap for sharing characters that could exist between rules.
Below is a list of all the rules that take up characters in a perfect scenario.

- Rule 6 - best case scenario takes 3 characters
- Rule 8 - best case 5 characters
- Rule 9 - always 4 characters
- Rule 10 - best case 5 characters
- Rule 11 - best case 0 characters (setting PC clock ahead by a month means no wordle answer exists)
- Rule 13 - best case 1 character
- Rule 14 - best case 4 characters
- Rule 15 - best case 1 character
- Rule 16 - best case 3 characters
- Rule 17/23 - 1 character
- Rule 18 - best case 4 characters (FmFm)
- Rule 21 - 3 characters
- Rule 22 - best case 8 characters
- Rule 24 - 20 characters
- Rule 28 - 7 characters
- Rule 32 - 2 characters
- Rule 35 - best case 4 characters

Summing all of these we get 75 characters. This is of course much longer than what is possible if we combine words.

To figure out the optimal overlap, we will first calculate for each possible solution to each rule, what other rule solutions it overlaps with. We can do this by making several lists for each of the rules containing the rule's solutions. Then we check each element in the list against all the elements to find all the instances in which:

1) The start of the target solution overlaps with the end of some other solution.
2) The end of the target solution overlaps with the start of some other solution.
3) The target is contained entirely within some other solution.

Once we have a list of options that have the most overlap, we will pick the ones that interfere with each other the least, thus allowing for the most overlap. Some of the rules listed above can be contained entirely within other rules.
These are:

- Rule 15 - Make sure one of the numbers for the time, password length, chess move, or hex code has a leap year in it.
- Rule 18 - By carefully capitalizing certain letters in our password, we should be able to make atomic numbers that add up to 200 without adding any element symbols.

Some rules we won't discuss because they cannot overlap with other rules, and just take up a set amount of space. This is because these rules require us to add emojis.
These rules are:

- Rule 13
- Rule 17/23
- Rule 21

III. Results

A. Possible Rule Solutions

First we want to ensure we aren't missing any large overlaps by only checking short words (For example, if "may" only overlaps with 1 character, but if march overlaps with 3, it would be equally good despite being 2 characters longer.) Fortunately, if we check for overlap on every single possible word in our password, we find only one overlap of 3 characters or more, between "f6ne5" (the CAPTCHA) and "Ne5+" (the chess move). This isn't even legal, since CAPTCHA letters must be lowercase, and the chess piece letter must be uppercase. This means there is no overlap greater than 2, so we only need to check words that are 2 or less characters longer than the shortest word to choose. For example, "starbucks" could never be a better sponsor than "pepsi" or "shell" since it is a whole 4 extra characters, which would require 4 character of overlap to offset.

If we list out the possible solutions to rules that are 2 (or less) characters more than the shortest possible length established for each rule in our methods section, we get the following options.

- Month: march, april, may, june, and july
- Sponsor: pepsi or shell (starbucks is too long)
- Roman numberals: Both $VIIV$ and $XXXV$ are equal in length.
- CAPTCHA: pcede, ebcbx, dbfen, cdcb3. (This is because if we write a short algorithm to find all CAPTCHAs composed entirely of hex color digits, these are the 4 possibilities.)
- Countries: Iran, Peru, Ghana, and Nepal. (There are many more 5 and 6 letter countries, but none of them have an overlap of more than 1 with other solutions, making them not good solutions.)
- Chess moves: Rh6, Ng4, Ne7, Nf3, Qc6, Kh6 (Writing a short algorithm, similar to the CAPTCHA one, we find that these are the only chess solutions possible in 3 characters that don't end in a + (because + has no possibility to overlap). I am also excluding solutions that have an atomic number in them greater than 48. Because of the roman numeral rule, we already have elements summing up to 152.)
- Affirmation: All 3 affirmations are possibly good solutions due to being similar in length.
- YouTube URL: All solutions are equal in length.
- Hex: All solutions are equal in length.
- Password length and time: As long as a time with a single digit in the hours place is used, all solutions are equal in length.

B. Overlap Results

Now we have to find overlap between all of the rules to optimize our password. I wrote a short Python script that takes in this list of CAPTCHAs, countries, sponsors, chess solutions, months, affirmations, primes, and roman numerals and finds all the pairs of words that either overlap at the ends or have one entire word containing another.

```
def lowercase_strings():
  for index, lst in enumerate(lists):
    for i in range(len(lst)):
      lists[index][i] = lists[index][i].lower()

def find_all_overlap(lists=lists):
  t_over = defaultdict(list)
  for index, lst1 in enumerate(lists):
    for lst2 in lists[:index] + lists[index+1:]:
      for str1 in lst1:
        for str2 in lst2:
          overlap = find_overlap_length(str1, str2)
          if overlap:
            t_over[overlap].append((str1, str2))
          if str1 in str2:
            t_over[len(str1)].append((str1, str2))
  return all_overlap
#lists=list of lists of strings (not shown)
lowercase_strings()
all_overlap = find_all_overlap(lists)
```

Running our code with only short words that would make sense to put in our password, we find:

1) Best Month Solutions - Rule 6:

- "juNe" overlaps 2 characters with "Ne7"
- "june" also overlaps 2 with the CAPTCHA "ebcbx"
- "may" and "july" both overlap with the youtube link. (May is better though, since it has 1 less character.)

- The 5 letter months don't have more than 1 overlap each.

Looking at our options for month, we want to choose either May or June, since May is one less character but has an overlap that is 1 character shorter.

2) Best Sponsor Solutions - Rule 8:

- "shell" overlaps with nothing
- "pepsi" overlaps with all 3 affirmations and "iran"

The best sponsor solution is pepsi, since shell has no overlap with any other solutions.

3) Best Country Solutions - Rule 14:

- "iran" overlaps with Ng4, Ne7, and Nf3.
- "iran" overlaps with pepsi.
- "iran" overlaps with "VII"
- "peru" overlaps with nothing.
- "nepal" overlaps with "june" and "dbfen" (CAPTCHA solution).
- "ghana" overlaps with "iamenough".
- All other 5 or 6 letter countries only overlap 1 character with other solutions.

For the country we want to choose "iran". This is because it has overlaps with both the "i" and the "n" resulting in 2 characters potentially saved. "Peru" has 0 characters potentially saved, and although "nepal" and "ghana" also both have 2 characters saved, they are one character longer than "iran", and so are one character less efficient.

4) Best Roman Numeral - Rule 9:

- $VII * V$ - "VII" overlaps with all affirmations and "iran". "V" overlaps with "iamloved".
- $XXXV$ - overlaps with nothing.

For roman numerals, we want to use $VII * V$ because we can save two characters with overlap.

5) Best Affirmation Solution - Rule 22:

- "iamloved" overlaps with the roman numeral "V".
- "iamworthy" overlaps with "youtu.be".
- "iamenough" overlaps with "ghana".

For the affirmation, we want to choose "iamloved" because it is one character shorter than the others. "iamworthy" is not a good pick, even with an overlap, because "may" already overlaps with "youtu.be". When choosing the country, we established that "ghana" is not as optimized as "iran", and "iamenough" has no other overlaps.

6) Best Chess Move Solution - Rule 16:

- Nf3, Ng4, and Ne7 overlap with "iran" and "dbfen" (CAPTCHA).
- Ne7 overlaps 2 characters with "june".
- No other chess moves overlap in useful ways.

Nf3, Ng4, and Ne7 are all equally good solutions for chess, since they share optimizations. Ne7 is slightly better, but only if we use june as our chosen month.

7) Best Time and Hex Color Solutions - Rules 28 and 35:

For the time, which has to be included in the password, if the password length is less than 60, we can combine the length of the password rule and the time rule. In addition, the ending number of the chess move can be the hour number of the time. (Ex: "Ne7:53", for a 53 character password.)

For the hex color, we can use one of the 4 CAPTCHA solutions that are made entirely from possible hex digits. The remaining 6th digit can be used as the leap year and to make sure the digits in the password sum to 25 (to satisfy rule 5).

Summary:
For some of our rules we still haven't figured out the best possibilities.
The best solutions are:

- Month - may or june
- Sponsor - pepsi
- Country - iran
- Roman Numberal - $VII * V$
- Affirmation - iamloved
- Chess Move - Ne7, Nf3, or Ng4

C. Final Optimizations

The solutions above are the best solutions for each of there respective rules, however, certain optimizations make other optimizations impossible. If we use june as the month, we can combine it with Ne7, which saves just as many characters as combining may with "yout.be". However, this prevents us from combining iran with one of the 3 chess moves.

This means that although the month only has 2 unoptimized characters, the country goes from having 2 unoptimized characters to having 3. This is because there are no other solutions that have the ending "N" for iran to overlap with, costing us one character.

Thus, we can conclude that the best month to use is may and all 3 chess solutions are equally good (since only the N needs to match, not the "Ne").

D. Putting It Together

There are 3 sections of the password that are strung together. These sections can be arranged in any order.
The first is:

```
mayoutu.be/VIIamloVed
```
(1)

We get this part of the password by combining:

- "may" with the YouTube URL
- The roman numerals VII and V with the affirmation "IamloVed". (Note that VII has to be grouped together, but V can be separate and entirely inside the affirmation since it is part of a different number.)

The second section is:

```
pepsiraNe7:47
```
(2)

We get this part of the password by combining:

- The end of "pepsi" with the start of "iran"
- The end of "iran" with the start of any of the 3 chess moves
- The last number in any of the 3 chess moves with a time.

The final section is:

```
#4cdcb3
```
(3)

The 5 characters of the CAPTCHA can be pcede, ebcbx, dbfen, or cdcb3.

In addition to these 3 sections the characters 🍦🏋️🏋️🌔 must be in the password. These 5 emojis can be anywhere, so long as they don't interrupt any words or other rule solutions.

Combining these sections, we get a combined password length of **46** characters (once we add the 5 emojis to the front)! This technically disobeys rule 33, which says the length of the password must be a prime number. For this reason, we will add a random character anywhere that does not interrupt any words. Because this character can be anything, and would not be necessary if rule 33 didn't exist, I will still consider it a 46 character password in spirit.

There are many other passwords calculated to exist for 47 characters, since taking away one character optimization opens a lot of doors for different combinations. However, I will not consider these, as they would not be possible in 46 characters if the prime number length rule was removed.

IV. Discussion

A. Caveats

There are a couple caveats to how we can arrange these sections. Because a YouTube video ID must have 11 characters, certain combinations are not possible. This is because if you look closely, section 1 only has 10 characters in the YouTube video ID. This is fine, because the URL is valid when a second section is added. However, a `#`, which section 3 starts with, is not a valid symbol in a YouTube video URL, meaning that section 2 must follow section 1. The possible arrangements of the sections are thus 1+2+3 and $3+1+2$.

B. Adding Missing Rules

You may notice that there are a few rules that are not considered in our password yet. These are rules 15 and 18 which are the rules that we can satisfy without adding any extra characters to the password.

Rule 15 - Your password must contain a leap year:

Currently there are 3 numbers in our password (4 if we use the CAPTCHA "cdcb3").

- 3, 4, or 7 depending on the chess move.
- 47, the password length, prime number, and minutes section of the time.
- The number we add to the hex code to make the digits sum to 25.

If we use chess move Ng4, we already have a valid leap year in the chess move. This means that `"Ng4:47#7cdcb3"` is valid because the digits 4+4+7+7+3 sum to 25. We cannot use the CAPTCHA solutions of `pcede, ebcbx, and dbfen`, because getting rid of the 3 at the end of "cdcb3" reduced the digit total by 3, and the most we can do to increase it is change the second 7 to a 9, which is not enough.

If we use chess moves Ne7 or Nf3, we have no leap year in the chess move anymore. Fortunately, in both of these cases the sum of digits is 8 and 4 less than 25 respectively. This means that the bonus digit in the hex code can be a leap year, giving us `"Ne7:47#4cdcb3"` and `"Nf3:47#8cdcb3"` as possible endings for the password.

In these conditions, we also cannot use the other 3 CAPTCHAs because removing the 3, would require changing the 8 and 4 to a 11 and 7 to keep the sum of 25. However, both of these changes would break the leap year rule.

As an additional note, for `"Ng4:47#7cdcb3"` it is possible to move the location of the 7 to the end of the hex code, gaining one more solution: `"Ng4:47#cdcb37"`. However, this is not possible for the other 2 chess moves because 34 and 38 are not valid leap years.

Rule 18 - All atomic numbers in your password must add up to 200:

In our password, there are 5 atomic symbols that we cannot change.
These are:

- 2x Vanadium (atomic number of 23) for the two Vs in the roman numeral
- 2x Iodine (53) for the two Is in the roman numeral
- 1x Neon (10) for Ne7 OR Nitrogen (7) for Ng4 or Nf3

In the case of Ne7 as the chess move, this sums to 162, leaving 38 as the remaining sum needed to make 200. In the case of Ng4 or Nf3 as the chess move, this sums to 159, leaving 41 as the remaining sum needed to make 200.

If we look at our password, almost any characters except for those in the CAPTCHA and the second character of the algebraic chess notation solution can be capitalized. All the elements with atomic numbers less than 41 that exist in our password as lowercase characters are:

- Be: Beryllium (4)
- B: Boron (5)
- O x2: Oxygen (8)
- Si: Silicon (14)
- P x2: Phosphorus (15)
- Y: Yttrium (39)

Beryllium and Boron share the same B. We get Boron by choosing to capitalize the E that comes after B in "youtu.BE". Thus we can only choose one of these elements.

There is exactly one way to obtain 38 and one way to obtain 41 using these elements.

For 38, $P(15) + P(15) + O(8) = 38$.
For 41, $P(15) + Si(14) + O(8) + Be(4) = 41$.

A password using Ne7 as the chess move would look like this with correct capitalization:

🐓🐂🐂🐂🌕mayoutu.be/VIIamlOVedPePsiraNe7:47#4cdcb3 (4)

We could also use the O in "youtube", instead of in "iamloved"

A password using Ng4 or Nf3 as the chess moves would look like this:

🐓🐂🐂🐂🌕mayoutu.Be/VIIamlOVedPepSiraNg4:47#7cdcb3 (5)

We could also use the O in "youtube" or the second P in "pepsi".

C. Number of 46-character Solutions:

If we don't count the emoji placement, the placement of the random 47th character, and random capitalization, there are surprisingly few "perfect" password configurations.
We know that:

- The sections can be arranged as 1+2+3 or 3+1+2. (2 configurations)
- There are 3 different values section 2 could be depending on which chess move is used.
- If Ng4 is chosen as the chess move, there are two configurations of the hex code.

This gives us a total of 8 configurations. Of these, only 4 of them have a unique set of solutions to the rules. These 4 unique solutions are below.

🐓🐂🐂🐂🌕mayoutu.be/VIIamlOVedPePsiraNe7:47#4cdcb3
🐓🐂🐂🐂🌕mayoutu.Be/VIIamlOVedPepSiraNf3:47#8cdcb3
🐓🐂🐂🐂🌕mayoutu.Be/VIIamlOVedPepSiraNg4:47#7cdcb3
🐓🐂🐂🐂🌕mayoutu.Be/VIIamlOVedPepSiraNg4:47#cdcb37

All other possible solutions are simply made by rearranging the sections and emojis of these 4 unique solutions.

D. Visualizing the Optimizations

Below is a highlighted version of the 1st password showing the overlap between the different solutions. The labels are as follows:

- Yellow highlight: Rule 6 (month)
- Cyan highlight: Rule 24 (YouTube video link)
- Orange (looks olive green) highlight: Rule 22 (affirmation)
- Purple highlight: Rule 35 (time)
- Green highlight: Rule 28 (hex color)
- Red circles: Rule 9 (roman numerals)
- Black circles: Rule 18 (element symbols)
- Blue circle: Rule 8 (sponsor)
- Green circle: Rule 14 (country)
- Orange circle: Rule 16 (chess move)
- Yellow circle: Rule 32/33 (password length and prime number)
- Brown circle: Rule 15: (leap year)
- Purple circle: Rule 10 (CATPCHA code)

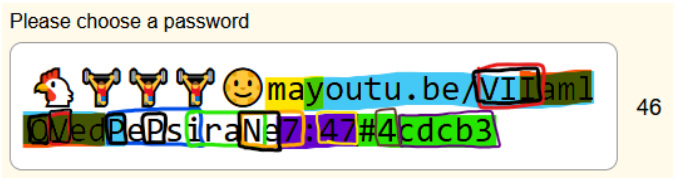

Fig. 2. A screenshot of what a (readable) version of the final password would look like

E. Probability

The odds of getting rule solutions that can be solved with a 47 character password are over 1 in 620,000,000. What really makes this impossible though, is that there is a less than 1 in 500 quadrillion chance that the URL needed for the YouTube video actually exists. For this solution to be plausible, there would need 100,000,000 times as many YouTube videos on YouTube as there currently are. So it is safe to say, a password game solution in 47 characters will only be theoretically possible.

The current record for a shortest password is 59 characters[3]. While it quite possible for a person to solve the game in 53 characters, it would require incredible (although possible) luck. The next lowest prime number before 53 is 47, which is the theoretical minimum, and below that, solving the game with fewer characters is impossible.

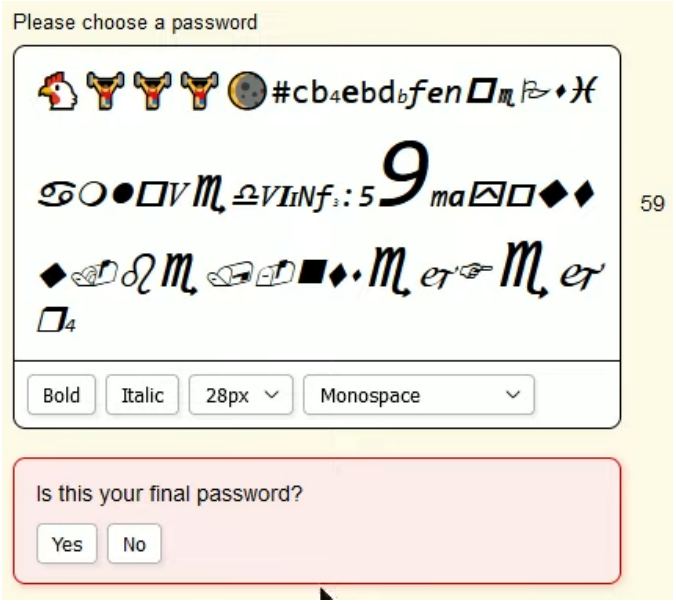

Fig. 3. A screenshot of the current world record for shortest password. The password looks so unreadable is because of the font-changing rules mentioned in the rules section.

V. Conclusion

I hope this in-depth exploration of "The Password Game" has helped you to understand the decisions made when trying to minimize the number of characters used when playing the game. Even saving one character can help because of the number of solutions that need to be considered for overlap.

Using programming tools I was able to explore millions of scenarios and combinations to ascertain the "perfect" or shortest viable solution. Using computational methods and automation is necessary for tackling many mathematical tasks, and I hope this investigation has helped show how such tools are able to provide an exhaustive search for permutative sets.

While a 46 or 47 character solution will likely never be possible for this game, finding every possible optimization can help understand what decision-making occurs when trying to actually beat the game with a several more characters. If you are interested, you should try playing the game[1] yourself. My personal record for minimum number of characters is 73, because anything less requires an extreme amount of patience and time (at least in my experience). A 53 character password, while possible, will likely never be done because of the time required.

With larger passwords, many different optimizations are possible if you don't get the perfect random rules. Since their solutions would be more than 46 characters, I did not cover most of these optimizations in my paper. I hope that if you are interested, you will be able to find some of these optimizations yourself, regardless of your luck, and that this paper helped show the methods by which a password in the game can be optimized.

VI. Works Cited

Here is my GitHub repository for a tool I made that solves the password game, and various other tools to help find optimizations.
https://github.com/RyantheKing/password-game-solver

References

[1] N. Agarwal. "The Password Game." neal.fun. https://neal.fun/password-game/ (accessed Mar. 14, 2024).

[2] pog5. "Breaking down Neal's Password Game in style" github.com. https://github.com/pog5/nealpasswordgame (accessed Mar. 14, 2024).

[3] SlashedPort. How The Password Game was beaten in 59 characters. (Nov. 18, 2023). Accessed: Mar 14, 2024. [Online Video]. Available: https://youtu.be/Tx5jPlE3Ldw

[4] Y. Adouani, M. Masmoudi, F. Jarray, B. Jarboui, "Iterative integer linear programming-based heuristic for solving the multiple-choice knapsack problem with setups", *Expert Systems with Applications*, vol. 242, no. 122835, May 15, 2024, doi: 10.1016/j.eswa.2023.122835.

[5] D. Pisinger, "A fast algorithm for strongly correlated knapsack problems", *Discrete Applied Mathematics*, vol. 89, issues 1–3, pp. 197-212, Dec. 1998, doi: 10.1016/S0166-218X(98)00127-9.

[6] H. Iida, "A simple branch-and-bound approach for the strongly correlated knapsack problem", (in Japanese), *Otaru University of Commerce*, vol. 48, no. 2/3, pp. 353-370, May 14, 2008. [Online]. Available: https://barrel.repo.nii.ac.jp/records/1336.

[7] "The Password Game - Speedrun.com" speedrun.com. https://www.speedrun.com/The$_{P}assword_{G}ame$ (accessed Mar. 20, 2024).