









A Computer Scientist Plays Wordle

Terry Sergeant

All About Wordle

- a word-guessing game with a guesser and a grader
- the grader thinks of a 5-letter word
- the guesser guesses a 5-letter word
- the grader give feedback with colors
 - green letter (right letter right spot)
 - yellow letter (right letter wrong spot)
 - gray letter (no such letter)
- guesser is given up to 6 guesses

Suppose the hidden word is "green":

Guess 1:  Guess 2:  Guess 3:  Guess 4:  Guess 5:  Guess 6: 



Why I Don't Like Word Games



Time for a Power-Up!

A boring standard laptop computer can do a billion operations a second! That's 1,000,000,000 actions in 1 second!

Getting the Computer to Play Wordle for You



You Need:

- a dictionary (orig-wordle.js)
 - check out word list!
- a programming language
- a mapping of the functionality of the game into variables / data structures

One of the major challenges of programming is finding an efficient way to represent the state of the world as variables.

Time for ICE (In-Class Exercise)

Let's tackle the problem of grading a guess.

- Given a guess and the answer word we will construct a "grade string" as follows:
 - the grade string is initially a copy of the guess
 - if the letter in the grade string should be green we replace it with the character '2'
 - if the letter in the grade string should be yellow we replace it with the character '1'
 - wrong letters are simply unchanged
- Consider: the guess, the answer, and the grade string are all strings ... but it would be more convenient if we convert them to lists to be able to access individual characters

On your marks, get set, go!

- Collaborate with your neighbor (groups of 2 or 3 ... NOT groups of 1 or 4+)
- You can write your ideas on paper or you can do this on your laptop (using the programming language of your choice)
- Create a function named `grade_word` that accepts two strings as parameters (the guess and the answer) and returns the grade string

Making Sure You Covered Your Bases

When testing your code, don't just get your code to work for the first case you thought of! Think!

Here are some examples to test:

Guess	Answer	Result
about	green	
about	abuse	
trees	abuse	

Guess	Answer	Result
abuse	trees	
amene	trees	

- Why did I pick these for testing?
- Are there others I should have picked?

Let's Explore wordle1.py



- `grade_word`
 - note wordhash
 - not really necessary when computer is playing itself (why a hash?)
- `load_dictionary`
- `main` program
 - `get_next_guess`
 - `show_grade`
- `reduce_word_list`
 - given a guess and a grade string, what rules can we apply to eliminate words as possibilities?
 - `is_possible`
- Run it!
- How v1 works:
 - randomly select a guess only from words that are still possible
 - we assume that ANY of the 13000 words can be chosen as the hidden word
- On average, how many guesses are needed?
- See: `wordle1.sorted.txt`

What is the Best Starting Word?



- Google: "finding the best starting word for wordle"
- It's an SEO jungle!
- Even if you find a bunch of sources saying the same thing, that doesn't make it true!
- Corollary: Just because Chat-GPT told you something doesn't make it true!
- The prevailing ("wrong") theory:
<https://word.tips/wordle-starting-words>
- What makes a guess "good"?



My Definition of "Best Guess"

The best guess is the one that, on average, eliminates the most words.

My Quest to Find the Best Starting Word

Assumptions I made
(beyond my "definition"):

- Using the original dictionary
- Any word from the dictionary can be the hidden word

See:

- wstats.py
-

```
for each word g (we assume g is the guess)
  for each word a (we assume a is the answer)
    grade the guess g given that a is the answer
    for each word w, keep it if it is possible given
      the guess and toss it out otherwise
    count how many words are left and keep total
  determine average number of words left
```

- Let d =size of dictionary (about 13000). This approach requires $O(d^3)$ steps or about 1.727 trillion steps.
- Divide that by a billion and we about 30 minutes (wishful thinking!)

Houston, We Have a Problem



- By spitballing, I had guessed I could calculate the average for each word in 30 minutes.
- Turns out, it took nearly 5 minutes to calculate the average for a single word!!
- Which would require about 45 days to compute.
- NOTE: Using $O()$ to estimate time required by an algorithm is a great rule of thumb. But in practice, constants do matter!

My Solution



- Estimate the goodness of a word using randomly selected words.
- Then refine through several passes
- See: trials.txt
- See: pass1.sorted.txt (sample size: 50 words, keep top 2500)
- Ultimately I did 5 passes using larger and larger sample sizes ...
- pass5.sorted.txt shows the scores for the top 20 (best) words

How Does This Compare to What Others Recommend

- SEO Jungle:
<https://word.tips/wordle-starting-words>
- CNBC/MIT:
<https://www.cnbc.com/2022/09/14/this-is-the-best-wordle-starting-word-according-to-mit-researchers.html>
- Scientific American:
<https://www.scientificamerican.com/article/information-theory-finds-the-best-wordle-starting-words1/>

Some Thoughts About the Best Guess



- Some possible reasons why my top word differs from others:
 - they are assuming the hidden word will come from the much smaller 2000+ words that people know
 - the SEO jungle folks were recommending guesses that could be potential answers (which allows for the possibility of getting it on the first try); NYT will NOT be having any of my top 20 guesses as answers!
- At what point is it best to guess possible answers rather than finding a word to minimize results (e.g., what if there are only two possible words left?)

What's the answer for this scenario:

Guess 1: agued
Guess 2: laics
Guess 3: pails
Guess 4: fails
Guess 5: kails
Guess 6: nails
Guess 7: tails
Guess 8: vails
Guess 9: mails
Guess 10: wails
Guess 11: hails

HINT: Guess 4 should be something like
(paint, faint, plant, pants)

Some Versions Using lares



- wordle2.py
- wordle3.py
- wordle4.py

Last Thoughts

- Playing around with these programs is WAY more fun than playing the game!
- Spending time in a problem space can teach you a lot!
- There's way more that can be done with this:
 - Assume the answer will be limited to the 2000+ word group
 - Find the break-even point when guessing a word from the 2000+ word group is more beneficial than the "best" next word
 - Precompute the best second guesses for all possible results

Here's my repository. Please clone and play around with it!

https://github.com/tsergeant/playing_with_wordle

