Fall 2022 INFO6205 Project

Wordle Solver

Siddhant Kohli 002108396

6th December, 2022

Contents

Build Instructions

- You can import the project (Projects from Git option in Eclipse) using the Clone URI option. Once cloned, you can run `WordleConnectorBot.java` or `WordleTestRuns.java`
- Run the file **WordleConnectorBot** to automatically open nytimes.com/games/wordle and have the application play the game for you.
- Run **WordleTestRuns** to run 100 Wordle games on the WordleSimulator and get data for the same.

Bonus: It wasn't a requirement for a 3-member team to write a UI to interact with the New York Times Wordle server. However, we implemented it and are looking for extra credit.

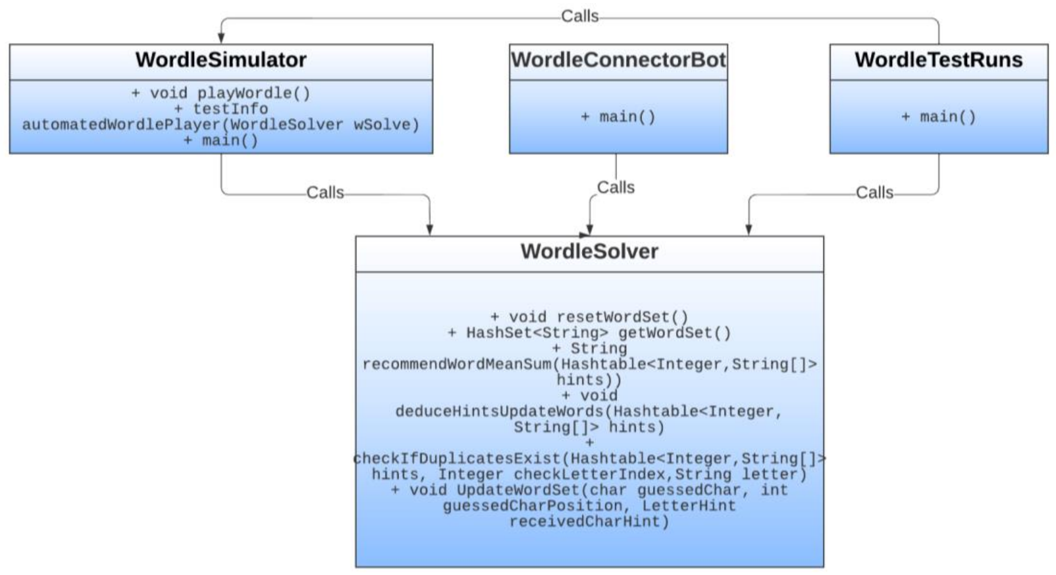
Implementation

Overview of the Application

This application consists of four classes each with having the following functionality:

- **WordleSolver**: Contains the algorithm to solve Wordle.
- **WordleConnectorBot**: Connects to nytimes.com/games/wordle and automatically solves the game based on the algorithm in WordleSolver.
- **WordleSimulator**: Code that simulates Wordle on the console.
- **WordleTestRuns**: Uses WordleSimulator to run tests and examine efficiency of the algorithm used.

The diagram below depicts the signatures of each method in each class.


Calls
WordleSimulator
+ void playWordle()
+ testInfo automatedWordlePlayer(WordleSolver wSolve)
+ main()
WordleConnectorBot
+ main()
WordleTestRuns
+ main()
Calls
Calls
Calls
WordleSolver
+ void resetWordSet()
+ HashSet<String> getWordSet()
+ String recommendWordMeanSum(Hashtable<Integer,String[]> hints))
+ void deduceHintsUpdateWords(Hashtable<Integer, String[]> hints)
+ checkIfDuplicatesExist(Hashtable<Integer,String[]> hints, Integer checkLetterIndex,String letter)
+ void UpdateWordSet(char guessedChar, int guessedCharPosition, LetterHint receivedCharHint)

Main Algorithm

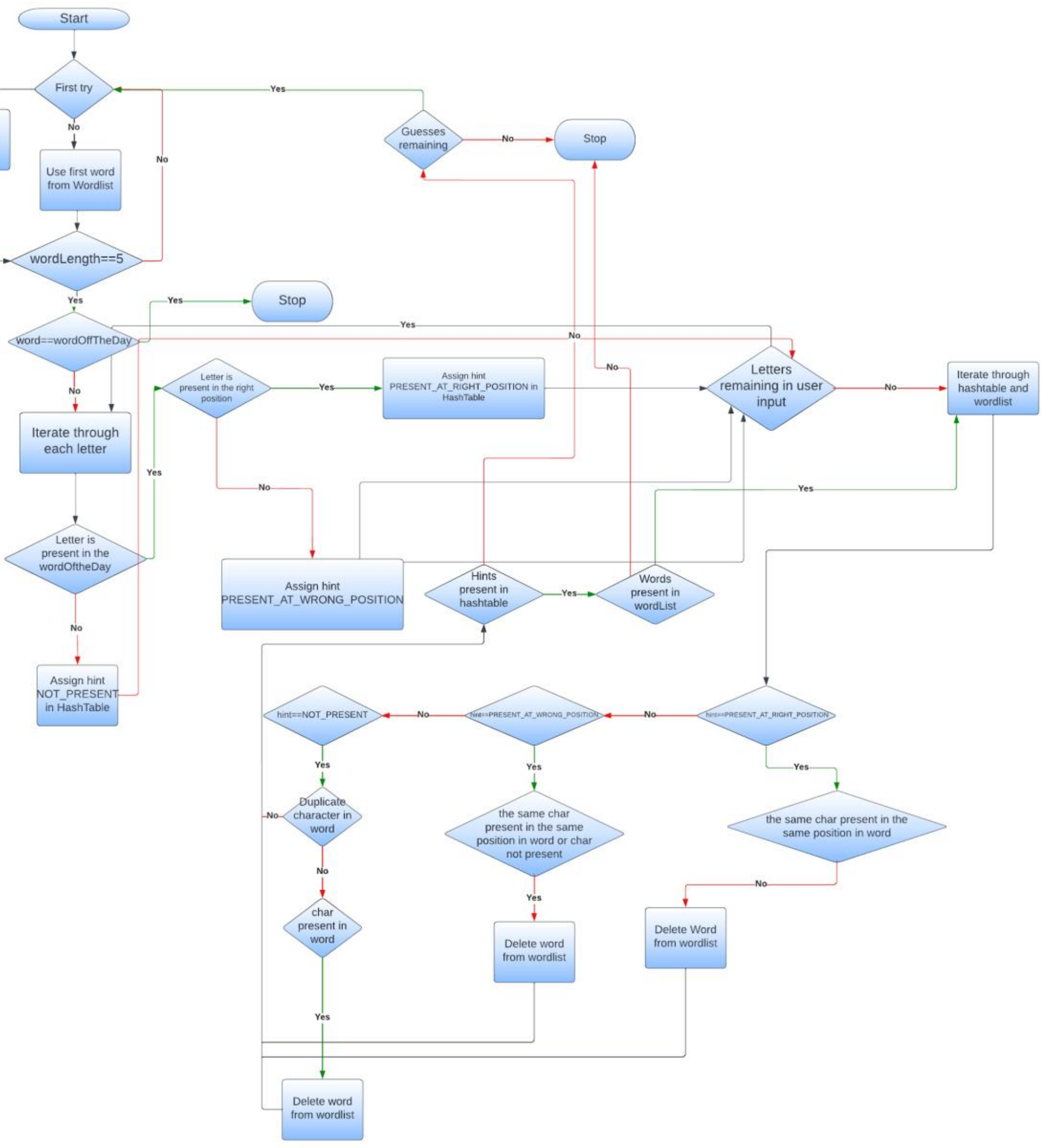

Start
First try
Yes
Use SALET
No
Use first word from Wordlist
No
wordLength==5
Yes
Guesses remaining
No
Stop
Yes
Stop
Yes
word==wordOffTheDay
Yes
No
No
Letter is present in the right position
Yes
Assign hint PRESENT_AT_RIGHT_POSITION in HashTable
No
Letters remaining in user input
No
Iterate through hashtable and wordlist
Iterate through each letter
Yes
No
Yes
Letter is present in the wordOftheDay
Assign hint PRESENT_AT_WRONG_POSITION
Hints present in hashtable
Yes
Words present in wordList
No
Assign hint NOT_PRESENT in HashTable
hint==NOT_PRESENT
No
hint==PRESENT_AT_WRONG_POSITION
No
hint==PRESENT_AT_RIGHT_POSITION
Yes
Yes
Yes
Duplicate character in word
No
the same char present in the same position in word or char not present
the same char present in the same position in word
No
char present in word
Yes
Delete word from wordlist
No
Delete Word from wordlist
Yes
Delete word from wordlist

Figure 2

The above diagram depicts the flow-chart of the entire algorithm.

Pseudocode:

Simulator

```
//Generates a wordOfTheDay from the 2309 word common-wordset

while(attempts<6){

       // User input for wordOfTheDay

       // Compare user input/guess with the wordOfTheDay

       for char in word {
              // If at the correct position, applies green color and returns hint
              If char in wordOfTheDay and wordOfDay.index(char)==word.index(char){
                     System.out.print(ANSI_GREEN + letterInCheck + ANSI_RESET);
                     Hashtable[char] = PRESENT_IN_RIGHT_POSITION
              }
              // If present in the wordOfTheDay, applies yellow color and returns hint
              If char in wordOfTheDay and  wordOfDay.index(char)!=word.index(char){
                     System.out.print(ANSI_YELLOW + letterInCheck + ANSI_RESET);
                     Hashtable[char] = PRESENT_IN_WRONG_POSITION
              }
              // If not present in the wordOfTheDay and returns hint
              If char not present in wordOfDay{
                     Hashtable[char] = NOT_PRESENT
              }


       }
```

WordleSolver

```
       //Receives a Hashtable called hints
       for key in Hashtable{
              for w in wordlist{
                     If hashtable[key]==PRESENT_IN_RIGHT_POSITION{
                            If key not in w and or not in same index{
                            //Eliminate words in wordSet that don't fit this criteria
                            }
                     }
                     If hashtable[key]==PRESENT_IN_WRONG_POSITION{
                            If key not in w and or in same index{
                            //Eliminate words in wordSet that don't fit this criteria
                            }
                     }

                     If hashtable[key]==NOT_PRESENT{
                            If duplicate character does not exist in wordOfDay{
                            //Eliminate words in wordSet that don't fit this criteria
                            }
                     }
              }

       }
}
```

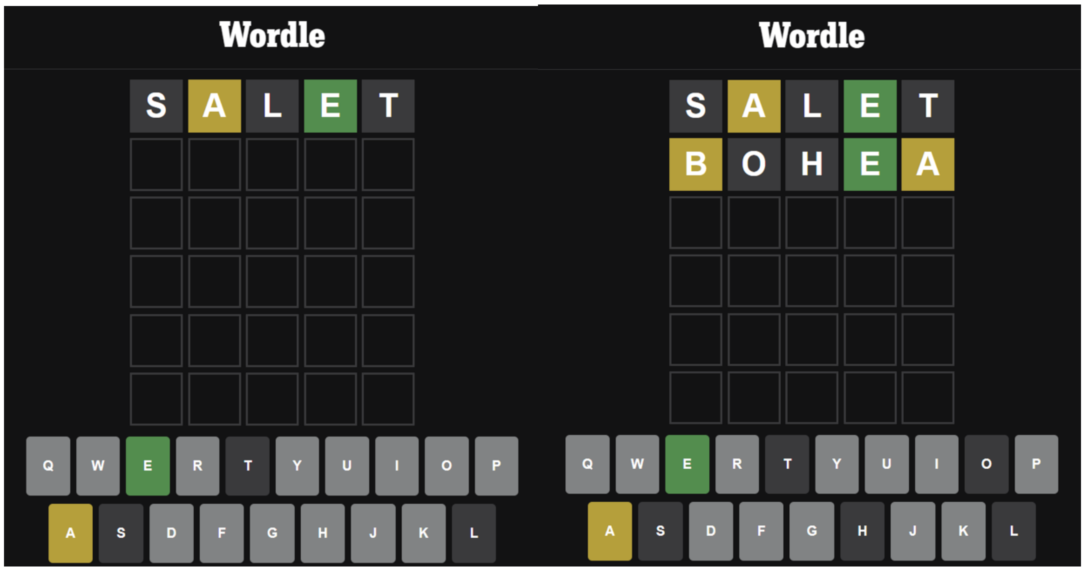
```
//Finally checks if there are repeating characters in the guessedWord

for (Character letter : letterFreq.keySet()) {
       int letterCount = 0;
       if (letterFreq.get(letter) > 1) {
              // Iterates over the wordSet
              for (key: hints.keySet()) {
                     int key = e.nextElement();
                     if (ht.get(key)[0].charAt(0) == letter && (ht.get(key)[1] ==
"PRESENT_AT_RIGHT_POSITION" || ht.get(key)[1] ==
"LetterHint.PRESENT_AT_WRONG_POSITION")) {
                                          letterCount++;
                                   }
                            }
                     if (letterFreq.get(letter) != letterCount) {
                            //Removes words with letterCount more than hints received
                            updateWordSet(letter, letterCount);
                            }
                     }
              }
```

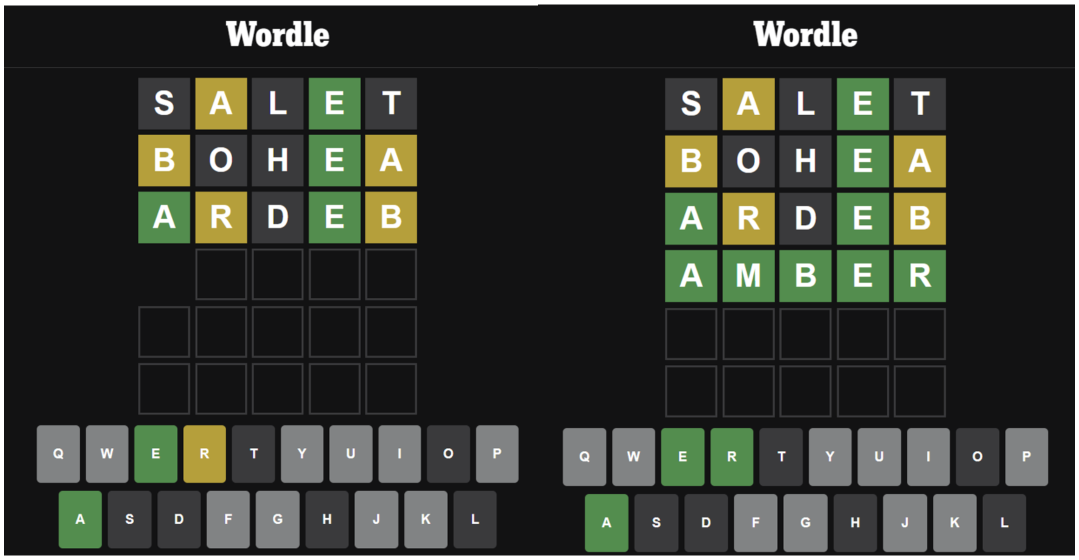
Log of Moves

The application automatically plays wordle on nytimes.com/games/wordle. The following are the screenshots for a wordle game played by the bot.

According to researchers, **SALET** is the best word to start any Wordle game with because it narrows down the words in the wordlist the fastest. Each of the letters in 'salet' are letters that occur very frequently in words and the elimination of words containing any one of these letters narrows down the list to a high degree.

Note: Although choosing SALET was based on stats of many researchers online, we completed test runs of our own to compute which first word has the best average attempts. This is detailed in the section **Statistics of Finding the Best First Word**





The algorithm was able to guess this word in 4 attempts.

Improving the Wordle Algorithm

Case 2:

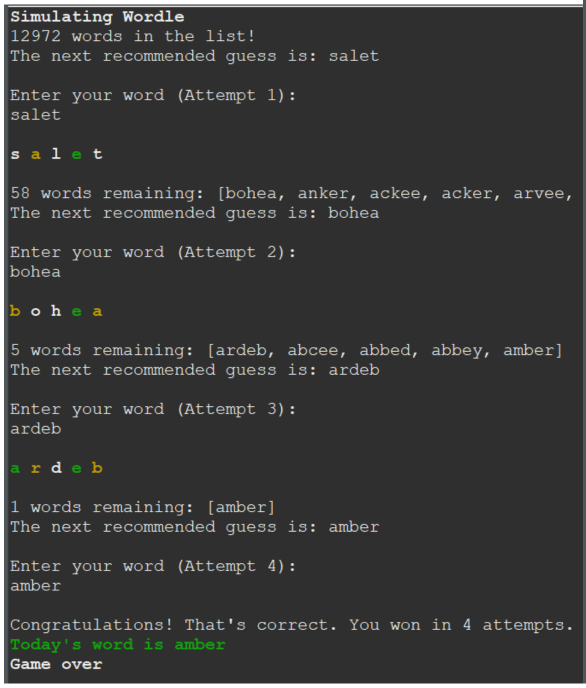
So far, the algorithm picks the next guess randomly (the first word in the remaining list of wordSet). We want to see if we can assign scores to different words and choose the next guess based on the assigned score. On the surface, assigning frequency scores seems like a good way to measure the probability of a word. In the secondary case of our algorithm, we calculate the frequency of letters at different positions and store them in HashMap. Using these scores, we create another global HashMap with a summation of all the letter frequencies. After every word guess and elimination, we calculate the word with the highest frequency score in the remaining wordSet and return it.

Entropy Table

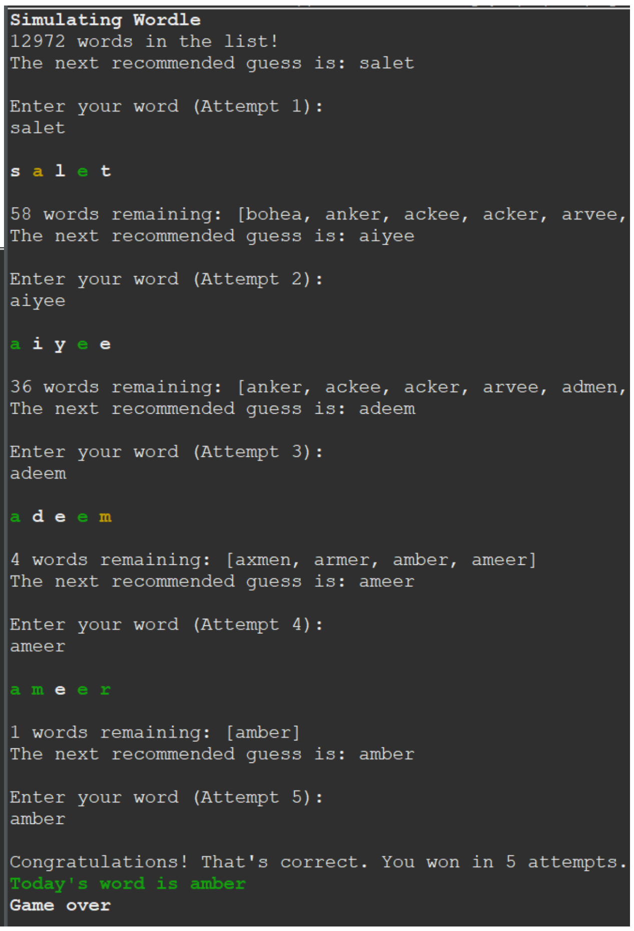
Taking the above example for the word **AMBER**, we are trying to capture how the entropy changes. We are using the following to calculate the uncertainties (or bits)

$$Bits = \log_2(Possibilities)$$

Since we have two cases of our logic, here's a screenshot of the simulator for both cases:



```
Simulating Wordle
12972 words in the list!
The next recommended guess is: salet

Enter your word (Attempt 1):
salet

s a l e t

58 words remaining: [bohea, anker, ackee, acker, arvee,
The next recommended guess is: bohea

Enter your word (Attempt 2):
bohea

b o h e a

5 words remaining: [ardeb, abcee, abbed, abbey, amber]
The next recommended guess is: ardeb

Enter your word (Attempt 3):
ardeb

a r d e b

1 words remaining: [amber]
The next recommended guess is: amber

Enter your word (Attempt 4):
amber

Congratulations! That's correct. You won in 4 attempts.
Today's word is amber
Game over
```



```
Simulating Wordle
12972 words in the list!
The next recommended guess is: salet

Enter your word (Attempt 1):
salet

s a l e t

58 words remaining: [bohea, anker, ackee, acker, arvee,
The next recommended guess is: aiyee

Enter your word (Attempt 2):
aiyee

a i y e e

36 words remaining: [anker, ackee, acker, arvee, admen,
The next recommended guess is: adeem

Enter your word (Attempt 3):
adeem

a d e e m

4 words remaining: [axmen, armer, amber, ameer]
The next recommended guess is: ameer

Enter your word (Attempt 4):
ameer

a m e e r

1 words remaining: [amber]
The next recommended guess is: amber

Enter your word (Attempt 5):
amber

Congratulations! That's correct. You won in 5 attempts.
Today's word is amber
Game over
```

Entropy for case 1:

Possibilities	**Uncertainties**	**State**
12972	13.663 bits	S A L E T
58	5.858 bits	B O H E A

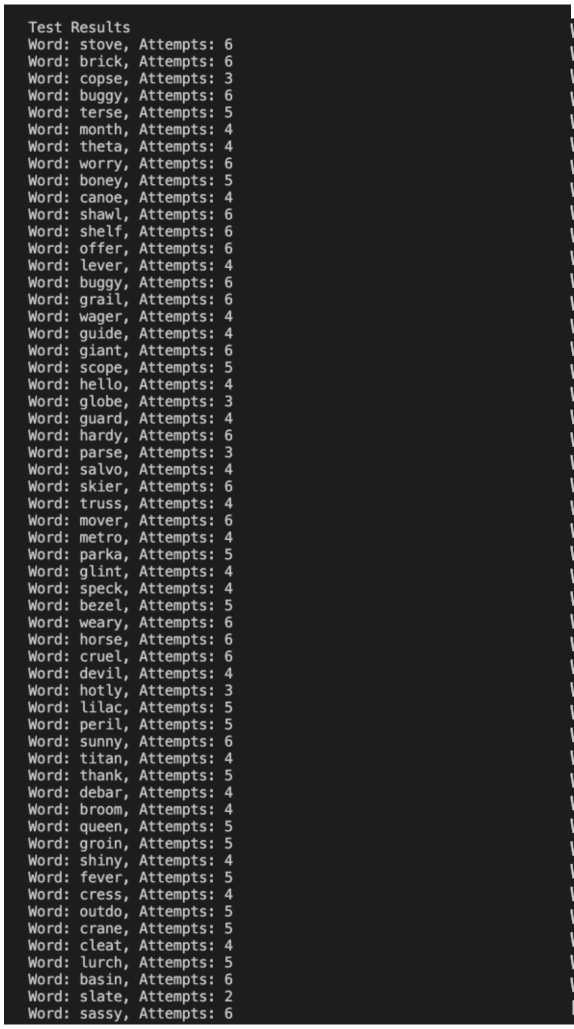
5	2.322 bits	🟩A 🟨R ⬜D 🟩E 🟨B
1	0 bits	🟩A 🟩M 🟩B 🟩E 🟩R

Entropy for case 2:

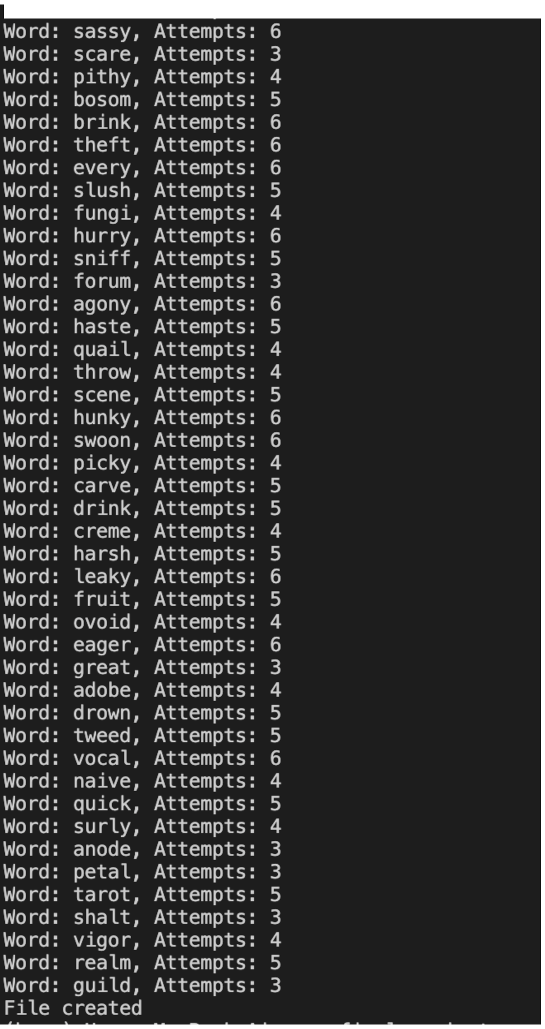
Possibilities	**Uncertainties**	**State**
12972	13.663 bits	⬜S 🟨A ⬜L 🟩E ⬜T
58	5.858 bits	🟩A ⬜I ⬜Y 🟩E ⬜E
36	5.17 bits	🟩A ⬜D ⬜E 🟩E ⬜M
4	2 bits	🟩A 🟩M ⬜E 🟩E 🟩R
1	0 bits	🟩A 🟩M 🟩B 🟩E 🟩R

Statistics of 100 Wordle Games

Case 1:



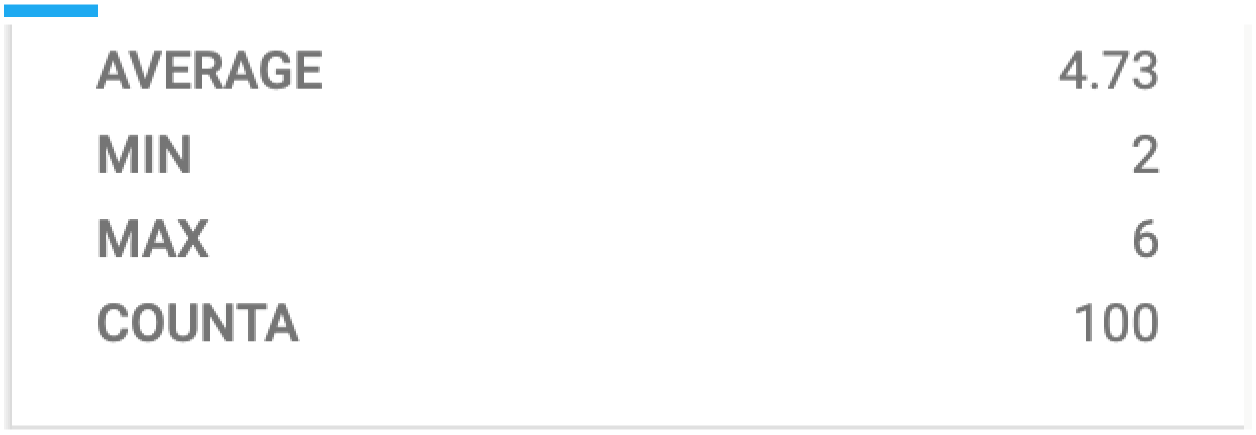
```
Test Results
Word: stove, Attempts: 6
Word: brick, Attempts: 6
Word: copse, Attempts: 3
Word: buggy, Attempts: 6
Word: terse, Attempts: 5
Word: month, Attempts: 4
Word: theta, Attempts: 4
Word: worry, Attempts: 6
Word: boney, Attempts: 5
Word: canoe, Attempts: 4
Word: shawl, Attempts: 6
Word: shelf, Attempts: 6
Word: offer, Attempts: 6
Word: lever, Attempts: 4
Word: buggy, Attempts: 6
Word: grail, Attempts: 6
Word: wager, Attempts: 4
Word: guide, Attempts: 4
Word: giant, Attempts: 6
Word: scope, Attempts: 5
Word: hello, Attempts: 4
Word: globe, Attempts: 3
Word: guard, Attempts: 4
Word: hardy, Attempts: 6
Word: parse, Attempts: 3
Word: salvo, Attempts: 4
Word: skier, Attempts: 6
Word: truss, Attempts: 4
Word: mover, Attempts: 6
Word: metro, Attempts: 4
Word: parka, Attempts: 5
Word: glint, Attempts: 4
Word: speck, Attempts: 4
Word: bezel, Attempts: 5
Word: weary, Attempts: 6
Word: horse, Attempts: 6
Word: cruel, Attempts: 6
Word: devil, Attempts: 4
Word: hotly, Attempts: 3
Word: lilac, Attempts: 5
Word: peril, Attempts: 5
Word: sunny, Attempts: 6
Word: titan, Attempts: 4
Word: thank, Attempts: 5
Word: debar, Attempts: 4
Word: broom, Attempts: 4
Word: queen, Attempts: 5
Word: groin, Attempts: 5
Word: shiny, Attempts: 4
Word: fever, Attempts: 5
Word: cress, Attempts: 4
Word: outdo, Attempts: 5
Word: crane, Attempts: 5
Word: cleat, Attempts: 4
Word: lurch, Attempts: 5
Word: basin, Attempts: 6
Word: slate, Attempts: 2
Word: sassy, Attempts: 6
```



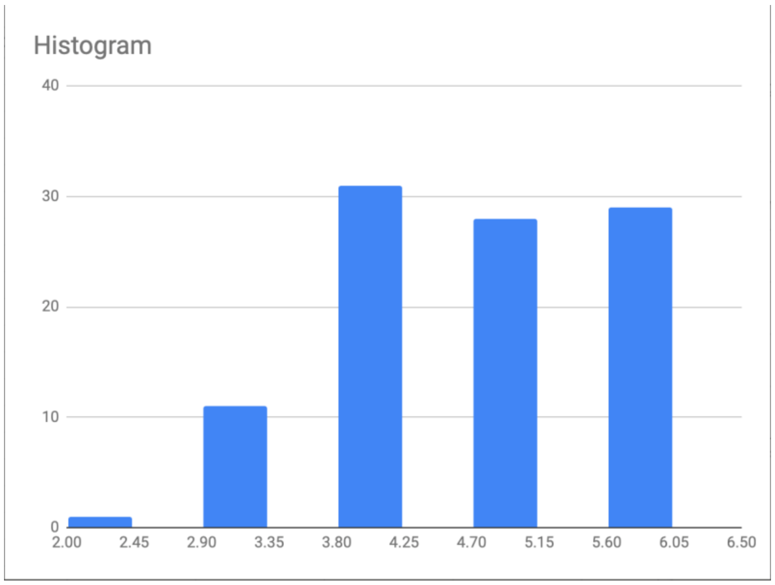
```
Word: sassy, Attempts: 6
Word: scare, Attempts: 3
Word: pithy, Attempts: 4
Word: bosom, Attempts: 5
Word: brink, Attempts: 6
Word: theft, Attempts: 6
Word: every, Attempts: 6
Word: slush, Attempts: 5
Word: fungi, Attempts: 4
Word: hurry, Attempts: 6
Word: sniff, Attempts: 5
Word: forum, Attempts: 3
Word: agony, Attempts: 6
Word: haste, Attempts: 5
Word: quail, Attempts: 4
Word: throw, Attempts: 4
Word: scene, Attempts: 5
Word: hunky, Attempts: 6
Word: swoon, Attempts: 6
Word: picky, Attempts: 4
Word: carve, Attempts: 5
Word: drink, Attempts: 5
Word: creme, Attempts: 4
Word: harsh, Attempts: 5
Word: leaky, Attempts: 6
Word: fruit, Attempts: 5
Word: ovoid, Attempts: 4
Word: eager, Attempts: 6
Word: great, Attempts: 3
Word: adobe, Attempts: 4
Word: drown, Attempts: 5
Word: tweed, Attempts: 5
Word: vocal, Attempts: 6
Word: naive, Attempts: 4
Word: quick, Attempts: 5
Word: surly, Attempts: 4
Word: anode, Attempts: 3
Word: petal, Attempts: 3
Word: tarot, Attempts: 5
Word: shalt, Attempts: 3
Word: vigor, Attempts: 4
Word: realm, Attempts: 5
Word: guild, Attempts: 3
File created
```

The above screenshots depict the number of attempts taken for 100 random words taken off of the wordlist that constitutes the most common words that New York Times uses.

A csv file was created with the same results and the statistics were calculated for it:

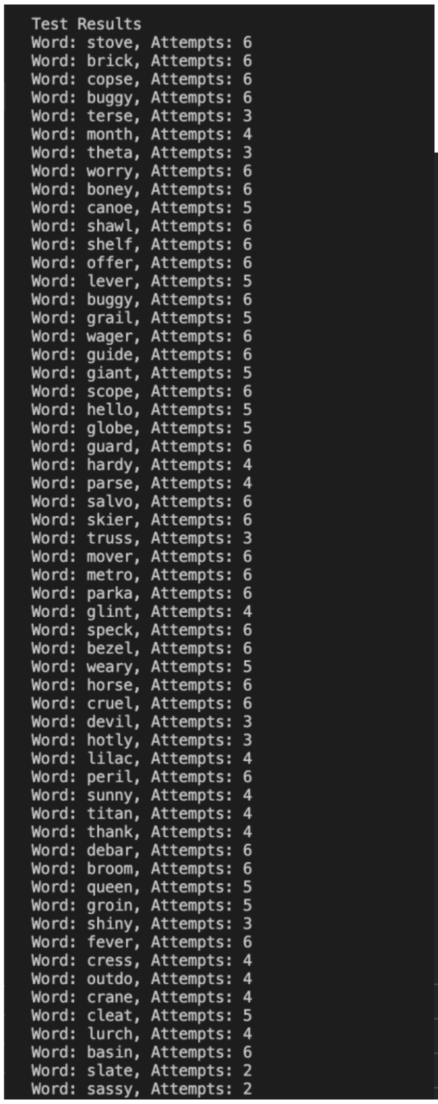


AVERAGE	4.73
MIN	2
MAX	6
COUNTA	100

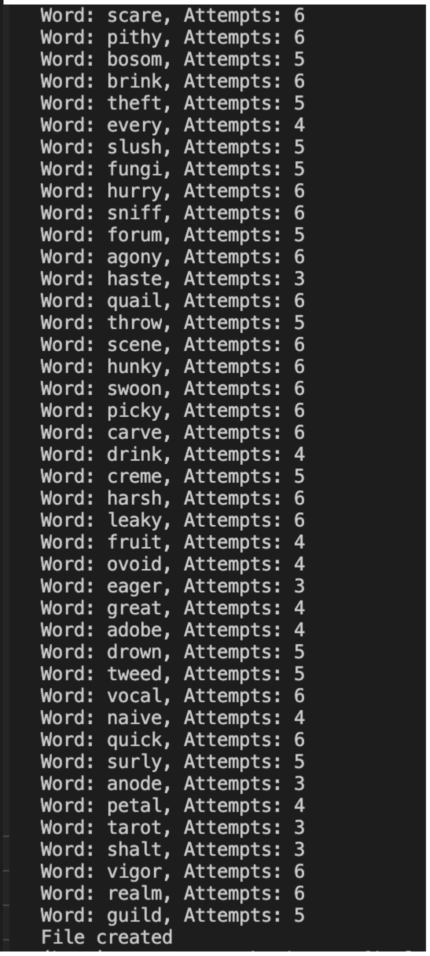


The above figure depicts the histogram of the data. The algorithm on an average takes 4 guesses to solve.

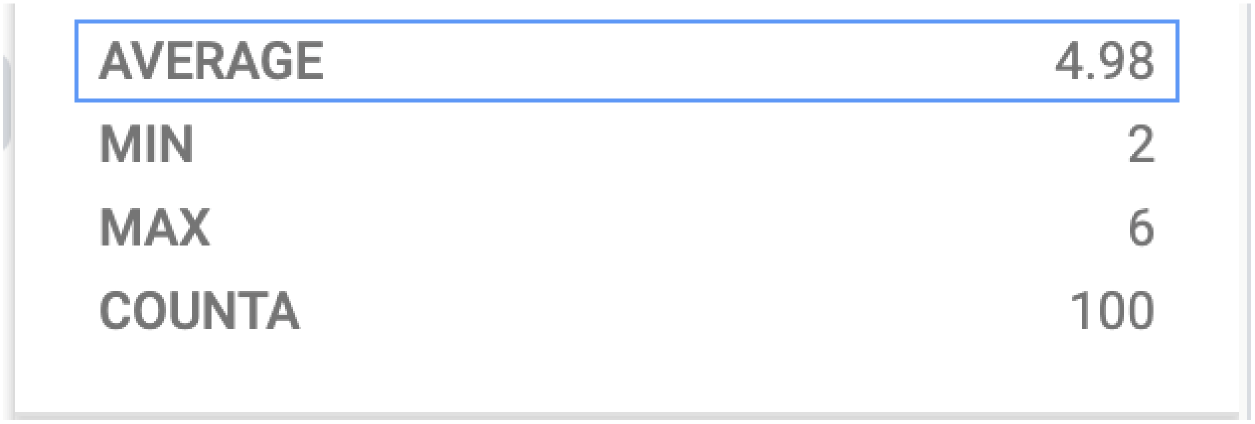
Case 2:



```
Test Results
Word: stove, Attempts: 6
Word: brick, Attempts: 6
Word: copse, Attempts: 6
Word: buggy, Attempts: 6
Word: terse, Attempts: 3
Word: month, Attempts: 4
Word: theta, Attempts: 3
Word: worry, Attempts: 6
Word: boney, Attempts: 6
Word: canoe, Attempts: 5
Word: shawl, Attempts: 6
Word: shelf, Attempts: 6
Word: offer, Attempts: 6
Word: lever, Attempts: 5
Word: buggy, Attempts: 6
Word: grail, Attempts: 5
Word: wager, Attempts: 6
Word: guide, Attempts: 6
Word: giant, Attempts: 5
Word: scope, Attempts: 6
Word: hello, Attempts: 5
Word: globe, Attempts: 5
Word: guard, Attempts: 6
Word: hardy, Attempts: 4
Word: parse, Attempts: 4
Word: salvo, Attempts: 6
Word: skier, Attempts: 6
Word: truss, Attempts: 3
Word: mover, Attempts: 6
Word: metro, Attempts: 6
Word: parka, Attempts: 6
Word: glint, Attempts: 4
Word: speck, Attempts: 6
Word: bezel, Attempts: 6
Word: weary, Attempts: 5
Word: horse, Attempts: 6
Word: cruel, Attempts: 6
Word: devil, Attempts: 3
Word: hotly, Attempts: 3
Word: lilac, Attempts: 4
Word: peril, Attempts: 6
Word: sunny, Attempts: 4
Word: titan, Attempts: 4
Word: thank, Attempts: 4
Word: debar, Attempts: 6
Word: broom, Attempts: 6
Word: queen, Attempts: 5
Word: groin, Attempts: 5
Word: shiny, Attempts: 3
Word: fever, Attempts: 6
Word: cress, Attempts: 4
Word: outdo, Attempts: 4
Word: crane, Attempts: 4
Word: cleat, Attempts: 5
Word: lurch, Attempts: 4
Word: basin, Attempts: 6
Word: slate, Attempts: 2
Word: sassy, Attempts: 2
```

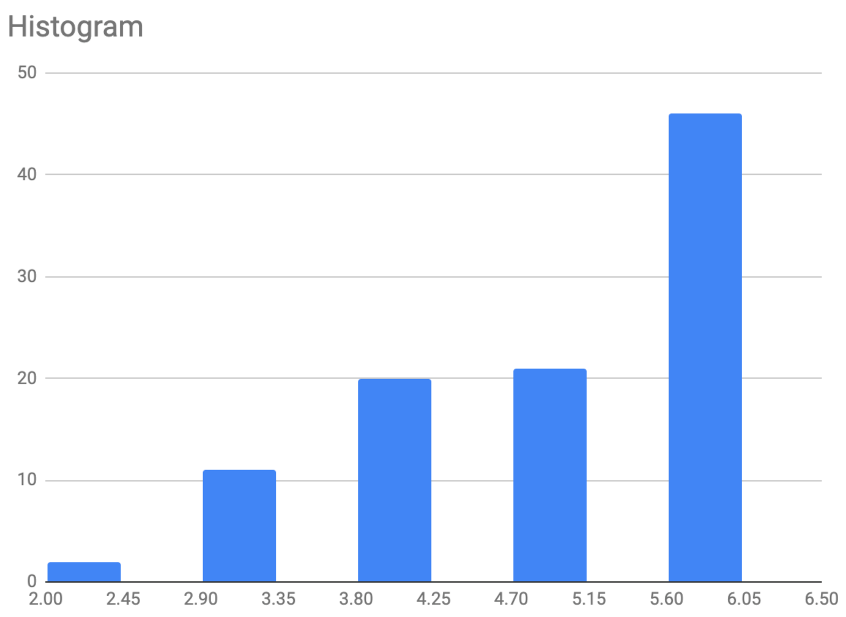


```
Word: scare, Attempts: 6
Word: pithy, Attempts: 6
Word: bosom, Attempts: 5
Word: brink, Attempts: 6
Word: theft, Attempts: 5
Word: every, Attempts: 4
Word: slush, Attempts: 5
Word: fungi, Attempts: 5
Word: hurry, Attempts: 6
Word: sniff, Attempts: 6
Word: forum, Attempts: 5
Word: agony, Attempts: 6
Word: haste, Attempts: 3
Word: quail, Attempts: 6
Word: throw, Attempts: 5
Word: scene, Attempts: 6
Word: hunky, Attempts: 6
Word: swoon, Attempts: 6
Word: picky, Attempts: 6
Word: carve, Attempts: 6
Word: drink, Attempts: 4
Word: creme, Attempts: 5
Word: harsh, Attempts: 6
Word: leaky, Attempts: 6
Word: fruit, Attempts: 4
Word: ovoid, Attempts: 4
Word: eager, Attempts: 3
Word: great, Attempts: 4
Word: adobe, Attempts: 4
Word: drown, Attempts: 5
Word: tweed, Attempts: 5
Word: vocal, Attempts: 6
Word: naive, Attempts: 4
Word: quick, Attempts: 6
Word: surly, Attempts: 5
Word: anode, Attempts: 3
Word: petal, Attempts: 4
Word: tarot, Attempts: 3
Word: shalt, Attempts: 3
Word: vigor, Attempts: 6
Word: realm, Attempts: 6
Word: guild, Attempts: 5
File created
```



AVERAGE	4.98
MIN	2
MAX	6
COUNTA	100

The above figure shows us the statistics for the algorithm. While the min and max values are the same, it can be discerned that the average number of guesses is higher for this algorithm than the previous one.

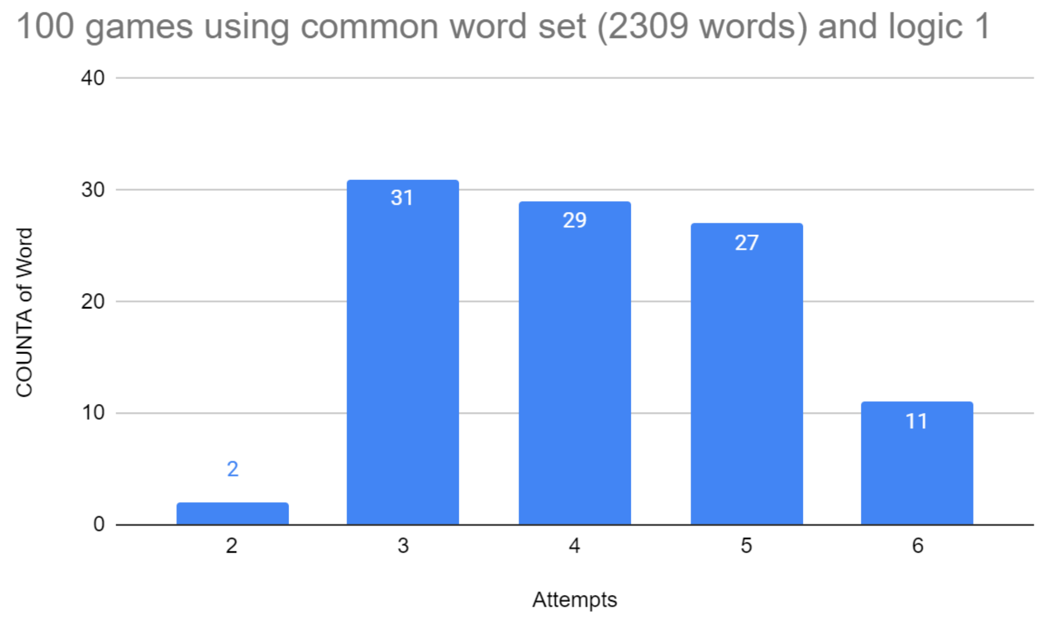


Although case 2 was intended to be an improvement of case 1, we notice that the performance hasn't improved. As a matter of fact, it has exacerbated. We can conclude that frequency mapping the letters doesn't aid in a better performing algorithm.

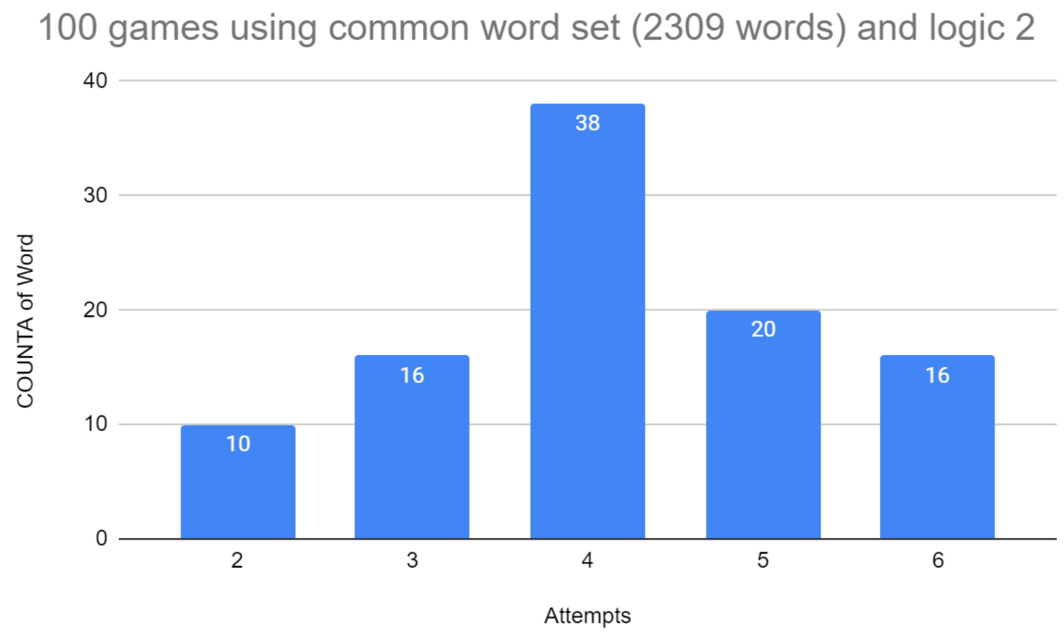
The histogram is significantly different and it is clear that the algorithm guesses more words using 6 attempts unlike the previous algorithm that had more words taking less than 6 attempts.

Narrowing our search..

How about we use only the 2309 common wordlist. Since we know that the word is chosen from this list, we can narrow our search and run tests with this word set. Here's how it turned out.



Average Attempts: 4.14



Average Attempts: 4.16

Time and Space Complexity

The parameters of the game are fixed. There are 26 letters in the English alphabet and 5 letters for each word. The total number of possible combinations is 11,881,376. The total number of words in the list of words that Wordle allows is 12,972 and that is also the length of the list that we are using for our inputs.

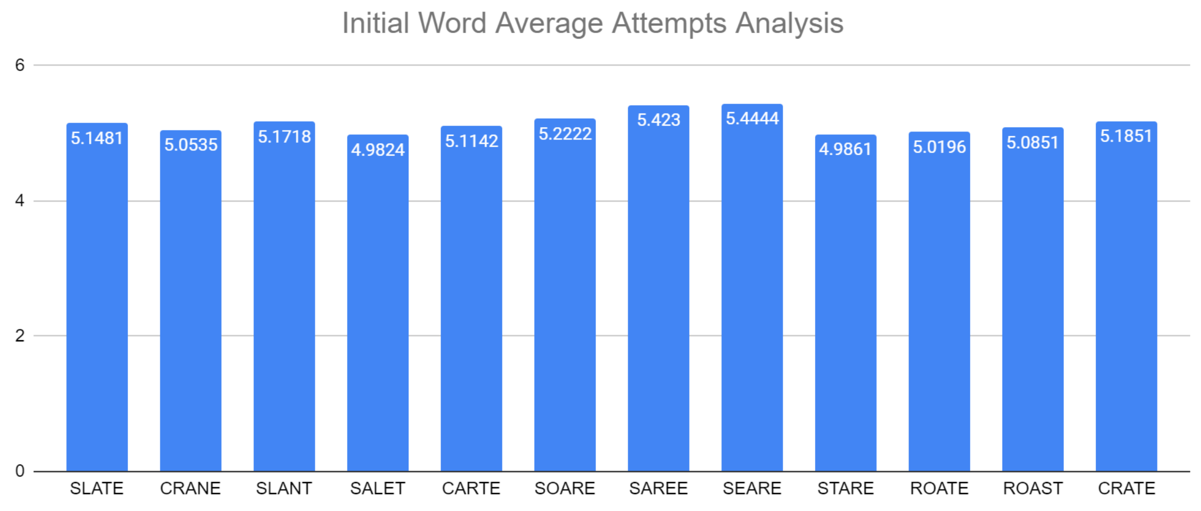
The main process in this algorithm is the deletion of words from a hashset containing the allowed input words based on hints given by Wordle. Removing an element from a hashset takes **constant time i.e O(1) time**.

The space complexity of the algorithm depends on the number of words being stored at a time in the hashset. The maximum number of words stored in the hashset is 12,972. After each iteration, words are removed from the hashset. **The space complexity of this algorithm is O(n)**.

Statistics of Finding the Best First Word

Since there are a number of different opinions on what the start word should be, multiple words were tested to obtain the best word. Using the right word makes sure that the puzzle is solved in the least number of attempts.

After perusing through many articles, it was found that 12 words were strongly debated amongst the public to be the best first word. The plot below depicts the average number of attempts required by each of these 12 words to reach the final word.



Here's an extract (20 trials) of the 366 test runs that we analyzed for the best first word:

Word	**SLATE**	**CRANE**	**SLANT**	**SALET**	**CARTE**	**SOARE**	**SAREE**	**SEARE**	**STARE**	**ROATE**	**ROAST**	**CRATE**
mured	6	7	4	7	6	7	5	7	6	7	6	6
locks	7	6	6	7	7	7	7	7	5	7	7	7

ample	5	4	5	5	5	5	5	5	5	5	5	5
begin	3	5	5	4	4	5	6	4	3	4	4	3
cadet	4	5	4	3	2	5	6	5	4	4	4	3
chuck	6	3	5	3	3	4	5	5	4	4	4	6
fishy	5	5	5	5	5	5	5	5	5	5	7	5
focal	5	5	5	5	5	5	4	4	7	7	7	5
freak	5	6	5	5	6	7	7	3	6	5	5	5
gayer	7	6	6	6	6	7	7	7	6	6	6	6
bardo	6	6	6	6	7	4	4	5	5	6	6	6
laser	3	7	3	6	6	4	4	3	4	7	4	3
hoses	7	6	7	6	6	5	7	6	7	5	7	7
samps	5	6	6	6	6	5	7	6	5	6	7	6
fundi	6	6	5	4	4	5	6	6	5	4	5	6
dawns	5	4	4	5	6	7	7	7	5	5	5	5
sinky	5	7	6	4	4	6	7	7	6	5	6	5
heeze	4	4	4	4	3	4	3	3	3	4	4	4
gross	6	7	7	7	7	7	7	7	7	6	6	6
aught	3	3	3	3	4	4	5	4	3	3	5	5

After 366 trials, the average attempts for each of the start word is in the table below:

SLATE	**CRANE**	**SLANT**	**SALET**	**CARTE**	**SOARE**	**SAREE**	**SEARE**	**STARE**	**ROATE**	**ROAST**	**CRATE**
5.1481	5.0535	5.1718	4.9824	5.1142	5.2222	5.4230	5.4444	4.9861	5.0196	5.0851	5.1851

Based on the above evidence, **SALET** takes the least number of attempts to reach the final word.

Conclusion

When we used the 12972 wordset of all allowed words for Wordle:

- Wordle could be solved with 2 guesses at best, 4.7 guesses on average and with 6 guesses at worst.
- Since the allowed number of attempts, we can always win the game!

When we used the 2309 wordset of common words for Wordle:

- Wordle could be solved with 2 guesses at best, 4.14 guesses on average and with 6 guesses at worst.
- We could beat the game every time in this case too.

We did notice that frequency mapping may not be the right improvement. However, through the course of the project we stumbled on some good ideas that could improve some cases drastically. Cases where we get 4 letters right but significant attempts are lost while trying to guess the final letter. For example we get the last four letters right: **I G H T**. The possibilities for the first letter are many(LIGHT, RIGHT, SIGHT, FIGHT, MIGHT, TIGHT, EIGHT), instead of trying each word in an attempt, we can try to maximize the information by using words like FERNS or TRIMS. We can eliminate many possibilities with just one word. Tracking such events can be difficult but a good improvement to add to the solver.

Another good idea we had while brainstorming (unimplemented however), was to use static words for the first two attempts to maximize the information retrieval and then use this information to recommend a word using our greedy algorithm (instead of using a greedy algorithm from the second attempt). We can try to go on for more attempts using the static words but I don’t think we can achieve it in the 6 attempts(of course, we don’t have tests to back this theory up).