Jonathan Pfefferle
Mid-Term project writeup.

1) Please analyze the time complexity of your algorithm if users enter N digits. Specifically, the time complexity for constructing the solver tree, the time complexity for dictionary searching, and the time complexity for solving the entire problem of N digits of input.

The solver-tree creation takes up the bulk of the algorithm's time, running somewhere between O(3^n) and)(4^n) depending on which numbers are entered. (Some produce only 3 children, others 4.)

Once a list of words is generated (which will have somewhere between 3^n or 4^n items) each word needs to be checked against a dictionary. Checking if it's contained in the dictionary is an O(n) due to the properties of prefix trees, or an O(1) operation using a hashtable.

Therefore, the total time complexity is O(n * n^4) or O(n^5) worse-case using a prefix tree, or O(n^4) worse-case with a hashtable.

2)Please search online references regarding the concept of Exhaustive Search. What is Exhaustive Search? Which part of this project falls into Exhaustive Search?

Exhaustive search, also known as Brute-Force search, is the process of simply generating every possible input and checking them one at a time against a given condition. For this project, the solver-tree creation falls into the realm of exhaustive search, as we simply generate every possible potential-word and check them one at a time against a dictionary.

2)Please search online references regarding the concept of Branch and Bound. What is Branch and bound?

Branch-and-Bound is an algorithmic design paradigm that tries to reduce a large number of candidate solutions to one result through use of bounds. It's similar to brute-force, but instead of going through the potential solutions one-by-one, it applies some bounding function that will exclude only false candidate (but potentially not *all* false candidates.)

For example, if checking if a string is a word, you could use a bounding function that checks if the string has at least one vowel. Any string without a vowel is not a word and can be discarded without further ado, but some strings with vowels might still not be words and need further investigation.

4) If you will use a Prefix Tree to implement the dictionary, can you use the prefix tree property to accelerate your algorithm together with Branch and Bound? and How to do that? please illustrate with a specific example and use a diagram if necessary.

When building the solver tree**,** instead of building it all up you check with the prefix tree every level to make sure a given node is a potential word. If it's not, you can discard that whole sub-tree and focus on the others exclusively.

For example, for an entered value of 2, we'll get nodes A, B and C. There's words that start with all three letters, so we can continue. But if our second number entered is also 2, we can discard the "BB" sub-tree since no English words start with "BB."

5) Please include at least 5 different runs of your program with different input sequences. Please catch any errors so that your program will not crash.

```
Loading dictionary from file...Enter any number of integers between
2 and 9. Press Q to exit
262
Prefix Words: [aoc, bob]
Hash Words: [aoc, bob]
Enter any number of integers between 2 and 9. Press Q to exit
228
Prefix Words: [act, bat, bay, cat]
Hash Words: [act, bat, bay, cat]
Enter any number of integers between 2 and 9. Press Q to exit
47277
Prefix Words: [grasp, grass]
Hash Words: [grasp, grass]
Enter any number of integers between 2 and 9. Press Q to exit
365
Prefix Words: []
Hash Words: []
Enter any number of integers between 2 and 9. Press Q to exit
292
Prefix Words: []
Hash Words: []
Enter any number of integers between 2 and 9. Press Q to exit
364
Prefix Words: [dog, emi]
Hash Words: [dog, emi]
Enter any number of integers between 2 and 9. Press Q to exit
q
```