Cheating Computers

Daniel Yee
CIS 2168

Purpose

In this assignment, you will practice using Sets and Maps to help you cheat at playing hangman. **NOTE**: Cheating at playing Hangman for this assignment will be the ONLY time I allow, even encourage, cheating in my class.

The Scenario

Playing a game with others relies on trust. Intrinsically, we assume that the rules will be followed and we're caught off guard when someone might be cheating at a game. And when someone is caught cheating, the punishment varies from penalties to outright banishment from the gaming system.

But what about computer opponents? Do they cheat? Well, not quite. We've all had that sinking feeling that the computer has been given special powers or "omniscience" in the game that we don't possess. It may well have been their programmers who were the real cheaters.

Your goal is to write a computer program that allows a user to play a game of Hangman. If you don't know about Hangman, Google it and play a few games with your friends (because they might not want to play the game with you anymore if you let them play Hangman on your program!)

A core assumption in Hangman (and other guessing games like "I Spy") is that the person who knows the word **does not change the hidden word**. Our program is going to do just that and get away with it.

Here's an example of how it might work.

Suppose the player has made a lot of guesses and has only one wrong guess left. The player's guesses have revealed:

GOA_

Based on the player's previous guesses and knowledge of the English language, there are only two possible words left: "GOAD" and "GOAL." In a normal game of Hangman, where the person with

the hidden word is playing according to the rules, this means the player has a 50% chance of guessing the right word.

In our game, the computer will cheat. If the player guesses the letter “L,” the computer will respond with something to the effect of “Sorry, you guessed incorrectly and you’re out of guesses. You lose! The hidden word was “GOAD.”” If the player guesses “D” instead, the computer pretends the hidden word was “GOAL” all along. In other words, the player will always lose in this scenario because the computer will lie about the hidden word.

How to Cheat at Hangman (Programmer’s Edition)

Let's learn how to cheat with a more thorough example. Pretend you're playing hangman and need to choose a four-letter hidden word for someone else to guess. You also happen to have a dictionary, unlimited time, and a galling lack of morals.

Rather than choosing and sticking with a single hidden word like you're supposed to, you make a list all four-letter words. There's a lot of them and many are inappropriate, so for simplicity's sake, we'll pretend English only has the following four-letter words:

THIS TEAM LOOK GOOD EGGS ECHO EDGE

So, you gather all those words into a list and lie to the player, telling him you've chosen your hidden word.

The player starts by guessing `E,' a sound strategy, since `E' is the most common letter in the English alphabet! You now need to reveal all the E's in your word, if any. You haven't chosen a word yet though, so you have a few options. Let's underline all the E's in our list of possible hidden words to see all our options.

THIS T**E**AM LOOK GOOD **E**GGS **E**CHO **E**DG**E**

We can categorize each of these into what we'll call word families, which is grouped by what the hidden word would look like to the guesser if we had chosen that word as our hidden word.

- The player would see ____ if the hidden word was “THIS," “LOOK," or “GOOD."
- The player would see E___ if the word was “EGGS" or “ECHO."
- The player would see _E__ if the hidden word was “TEAM."
- The player would see E__E if the hidden word was “EDGE."

You need to choose one of those families and that will limit your future choices. For example, if we choose the second word family, that means we're committing to pretending we choose a hidden word that started with “E," so we can't lie about that anymore. If we go with the first group, we commit to pretending our hidden word has no “E" in it. We can't go back later and pretend it has an “E" in it, because the user will remember they guessed that letter earlier. Remember, we don't the guesser to figure out we're cheating.

Which word family should we go with then? There are many valid strategies. Maybe you decide to go with the word family with the most obscure words. Maybe you choose the word family that reveals the least letters. You can use the one below or devise your own strategy.

I will use a much simpler strategy: choose the word family with the most words. Like any programmer with a propensity for cheating, we're afraid of commitment, so we'll choose the largest word family to avoid
being boxed into a single hidden word for as long as possible. Again, this isn't necessarily the best strategy, but it is a relatively quick one and works rather well. We tell our guesser their guess is wrong, reveal ____, and now our list of hidden words is:

THIS LOOK GOOD

Next they guess “O," so your word families are:

- _OO_ containing “LOOK" and “GOOD"
- ____ containing “THIS"

So, I'll choose OO as the word family and reveal the “hidden" O's.

The game ends when the user guesses all the letters, which is unlikely given all the cheating we're doing, or if they run out of guesses. If all goes according to plan, the user will never know they've been hoodwinked!

Cheater’s Hangman

Your goal is to create Cheater's Hangman, which plays Hangman, but cheats by avoiding choosing a hidden word according to some strategy. Here's the flow of the program.

1. Read the `dictionary.txt` and store it in your program at runtime. This is the list of words your program will use.
2. Ask the user to choose the size of the hidden word they want to try to guess. Keep asking them if there are no words of that length.
3. Ask the user to choose how many wrong guesses they get to have before they lose.
4. Create an initial list of hidden words using the dictionary and choosing all the words that have the length the using asked for.
5. Play hangman using our cheating algorithm (or your own algorithm):
    a. Print out the revealed letters, their wrong guesses, and the remaining number of wrong guesses
    b. Get the user's new guess and be nice and ask them to reenter their letter if they already guessed it.
    c. Separate your list of hidden words into word families based on the input.
    d. Choose a word family using some strategy (I used the word family with the most words) and make that the new hidden word list. If this reveals letters to the user, reveal letters, otherwise the player loses a wrong guess.
    e. Keep going until:
        i. All the letters are revealed, the player wins.

ii. The player is out of guesses, randomly pick a hidden word from the hidden word list and reveal it to the user, pretending that was your hidden word all along.

6. Ask if they want to play again.

DISCLAIMER: I am not responsible for broken laptops, computer monitors, other property or bodily injury if you decide to test your program on your unsuspecting classmates. If you do make your classmates your guinea pigs, at least let them in on the joke after a while and treat them to some ice cream. By the way, using your classmates as guinea pigs is exactly what a lying, cheating programmer would do!

Note: You'll want to remove any debugging text (like the size of the hidden word list at each guess) and not calling your program "Cheating Hangman." These are dead giveaways that you're playing unfairly!

Hint

This is coming right after learning about Maps and Sets so that's probably a good place to start.

Challenge

5 points extra credit

You implement a second cheating strategy and randomly select which one to use to throw any suspecting players off the scent.

Presentation

You will only receive credit for the assignment if you present your project.