Assignment 1

Kartik Bharadwaj CS20S020

22nd March 2021

1 Algorithm

We are given a list of 4-letter words from which an adversary can choose any word. Hence, we would like to find what word the adversary has chosen by making as minimal mistakes as possible. The adversary only provides two values, number of cats (C) and number of dogs (D). Fundamentally, C tells the number of characters common to our guess and adversary's word but not in the same position. Unlike C, D gives the number of characters common to our guess and adversary's word but *in* the same position. However, neither C nor D reveal which characters fall under its wing. This makes the problem tricky as we don't know which character in our guess is correct.

One way to find the adversary's word is to make as many random guesses as possible. However, this isn't reliable since we aren't using the information provided by the adversary for every guess. Therefore, we use the structure (# of cats and # of dogs) to design deterministic rules which will help us cut down on the word list after every guess. This will ensure that we make guesses from a shortened list of probable words to quickly narrow down on the adversary's word.

Algorithm

function FILTER_WORDS($word\_list$, $guessed\_words$)

▷ adv ⟶ adversary.

    adv_word_found = False

    $guess\_word$ = FETCH_WORDS($word\_list$)

    c, d = CALC_CATS_AND_DOGS($guess\_word$)

    if $str(c) + str(d)$ == '04' **then**
        adv_word_found = True
        return $word\_list$, # of mistakes, adv_word_found
    end if

    Append $guess\_word$ in $guessed\_words$

    if $str(c) + str(d)$ == '00' **then**
        No characters match with adversary's word. Hence, update word list by removing
        words containing any of invalid letters from our guess word.
    end if

    if $str(c) + str(d)$ in {'40', '22', '31'} **then**
        Our guess word is a permuted version of adversary's word.
        Therefore, we update our word list by keeping only other permuted versions
        of adversary's word and delete every other word from our word list.
    end if

    if $str(c) + str(d)$ in {'01', '02', '03', '10', '11', '12', '20', '21', '30'} **then**
        In these conditions, at least one of the characters from the guess word matches
        with adversary's word. Hence, we update our word list by keeping only those
        words which have at least one character same as our guess word.
    end if

    return $word\_list$, # of mistakes, adv_word_found
end function

2 Pseudo code explanation

Some analysis revealed that:

1. Possible cats and dogs pair are: {'00', '01', '02', '03', '04', '10', '11', '12', '20', '21', '22', '30', '31', '40'}, where the 1^{st} digit indicates # of cats and the 2^{nd} digit indicates # of dogs.
2. There are 38 words with 0 vowels, 2443 words with 1 vowel, 1778 words with 2 vowels, 49 words with 3 vowels, and 1 word with 4 vowels.

We can see that majority of the words contain 1 or 2 vowels. Also, there 21 consonants which we have to filter. For instance, if our adversary's word is **COLD**, then it wouldn't be optimal to have our guess word contain vowels $\{A, E, I, U\}$ or consonants other than $\{C, L, D\}$. To cut down on irrelevant characters, we split our word list into sub list: *only_consonants* and *with_vowels*. As the name suggests, *only_consonants* doesn't contain words with any vowels in it whereas *with_vowels* contain words with at least one vowel.

Since $len(\textit{with\_vowels}) > len(\textit{only\_consonants})$, almost all words will contain vowels. Our adversary will likely try to choose its word from *with_vowels* list in order for us to suffer more mistakes. To counter such a strategy, we choose our initial guesses from *only_consonants* list. Once exhausted, we move to *with_vowels*. Using these two word lists, we employ two deterministic rules:

1. If the adversary gives cats and dogs pair = '00' for our guess word, then there are no characters which match with adversary's characters. Therefore, we remove all words in our main word list.
2. If the adversary gives cats and dogs pair is in {'40', '22', '31'} for our guess word, then there the guest word is a permutation of the adversary's word. Therefore, we update our word list such that it only contains permutations of the adversary's word.
3. If the adversary gives cats and dogs pair = '04', then our guess word is exactly the same as the adversary word and we have found it.
4. If the adversary gives cats and dogs pair is in {'00', '01', '02', '03', '04', '10', '11', '12', '20', '21', '22', '30', '31', '40'}, then we do the following:
   (a) Pair {'01', '11', '21'} says that one letter from guess word matches adversary's word in the same exact position. Since there are 4 characters in a word, any word in our word list which contains at least one character in the same position will be preserved and others will be removed.
   (b) Pair {'02', '12'} says that two letters from guess word matches adversary's word in the same exact position. Given there are 4 characters in a word, there are $\binom{4}{2}$ possibilities with our guess word characters. Therefore, any possibility containing characters in the same position will be preserved.
   (c) Pair '03' uses the same principle as {'02', '12'} and {'01', '11', '21'} where the # of dogs is equal to 2 and 1 respectively.
   (d) Pair {'10', '20', '30'} tells us that at least one character is common to guess and adversary's word but is not in the same position. Hence, our word list should consists of only those words which contain at least one of the characters.

3 Results and Conclusions

1. Average # of guesses over all possible words: 15.05 ($\approx$ 16)
2. Toughest word: SADO
3. # of guesses for the toughest word: 113

In my view, the algorithm is sub-optimal but not by a big margin. Although we do eliminate irrelevant words based on each guess, it would be useful to combine # of cats and dogs from multiple guesses in order to narrow down on the adversary's word. This would remove some of the uncertainty in our guesses.