Omar Alkendi

Mark Lehr, PhD.

7 June 2020

Final Project: Mastermind AI Utilizing Knuth's Five-Guess Algorithm

Introduction:

In the development of the AI for mastermind, I came across numerous methods to break up the code. However, while some were hard to implement, others were the epitome of failure. How hard can an AI be for Mastermind? It is literally about the successful passing of possible guesses to the next generation and eliminating those that are unlikely to be the code. Yet it is the process of elimination that is difficult to come up with—I mean, on what basis should the AI eliminate a certain solution and not another? After some research, I came across Knuth's Five-Guess algorithm—which in the case of this project is not five guesses anymore because the algorithm was designed to solve a variant of the game that has only 1,296 possible solutions. The algorithm, developed by Donald Knuth in 1976, starts with a predefined initial guess, finds the surviving guesses by eliminates any combination that would not give the same result as the initial guess, applies the minimax technique by assigning every combination a score based on whether or not a surviving guess will eliminate the combination, chooses the combination with least score preferably from the list of surviving guesses, and repeats till the code is broken (Knuth 1-3).

At first, Knuth's algorithm seemed easy to implement. However, due to the conditions bestowed by the use of the stub driver, the implementation was quite challenging. As the minimax step was reached, the decision of using arrays appeared to be a poor decision, for arrays in C++ offers limited manipulation techniques and, thus, the code became cluttered and was hard to read—any attempt to troubleshoot an error felt like a neverending nightmare. Therefore, a transition to vectors and maps was rendered necessary even though vectors and maps are uncharted territory for me—hence, a great learning experience.

Summary:

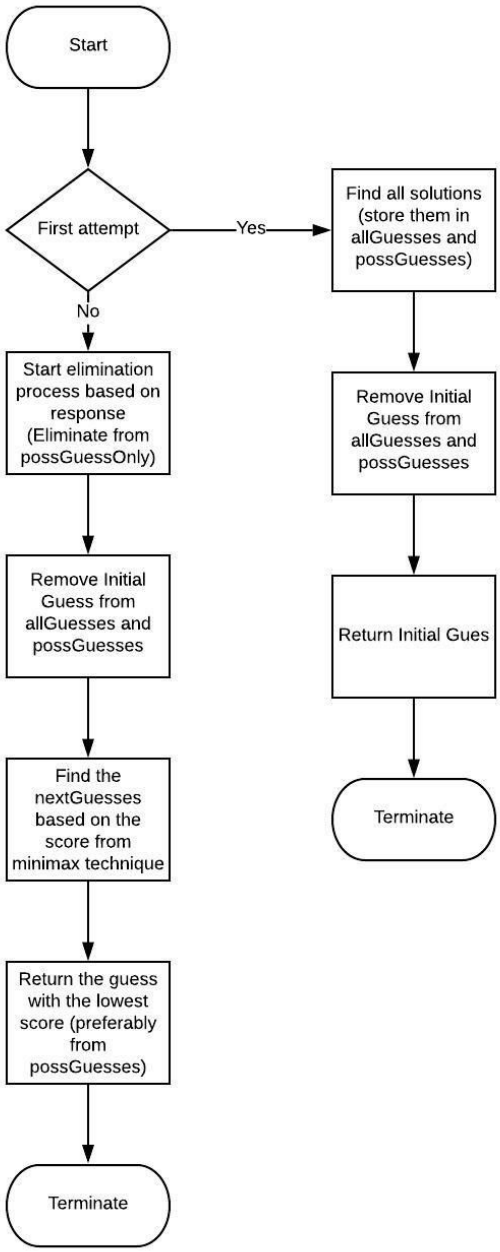
- AI size: ~180 lines.
- Number of variables: 7 (4 are vectors)
- Number of Lambda functions: 3 functions
- Number of Maps: 2

Issues:

- The minimax step loops over million times to find the best guess, but the number of iterations shrinks as the vector of possible guesses shrinks. This results in a long running time of over 1 minutes sometimes.
- There have been some cases—that unfortunately couldn't have been reproduced and troubleshoot—in which the AI would keep returning guesses for ever (occurred twice).

An Overview of the AI (The First Attempt):

The AI begins by determining whether or not a previous guess had been made. If no attempts had been made, the AI populates the **allGuesses** and the **possGuesses** vector with all possible combinations from 0000 to 9999. The combinations are stored as **integers** to simplify operations and converted to a string using a lambda function **stringify(int)** whenever the string form is needed. After successful population, the initial guess 0011 is removed from the **allGuesses** and the **possGuesses** vectors and then returned by the AI. The driver then evaluates the guess, and a response is passed to the AI.



An Overview of the AI (Any Other Attempt):

If this is not the first attempt, start the elimination process. Eliminate from **possGuesses** any combination that would not give the same response as the previous guess. This is done by comparing the response passed to the AI to the response obtained from comparing the previous guess to all **possGuesses**. This will shrink the **possGuesses** vector to guesses that are distinguishable. Next, apply the minimax technique by assigning every guess in **allGuesses** vector, which has 10,000 combinations, a score. The score is based on comparing the combination from **allGuesses** to every combination in **possGuesses**. If the result of the comparison, rr or rw, is greater than 1, then the guess from **allGuesses** receives a point. Otherwise, if the result of the comparison is zero, meaning no rr or rw between the guess from **allGuesses** and the element in **possGuesses**, then the score is just 1. The next possible guess is the guess that has the lowest score preferably from the **possGuesses** vector, but if it does not exist in the **possGuesses** vector, then the guess is obtained from the **allGuesses** vector.

Quick Peek:

```
Guess: 0011
Guess: 2345
Guess: 3267
Guess: 4672
Guess: 5736
Guess: 7536
7536
6 7536 7536

RUN SUCCESSFUL (total time: 1m 4s)
```

```
Guess: 0011
Guess: 0213
Guess: 4056
Guess: 4078
Guess: 9058
Guess: 4049
4049
6 4049 4049

RUN SUCCESSFUL (total time: 36s)
```

Work Cited

Knuth, Donald E. “THE COMPUTER AS MASTER MIND.” *JOURNAL OF RECREATIONAL MATHEMATICS*, vol. 9, no. 1, 1976, http://www.cs.uni.edu/~wallingf/teaching/cs3530/resources/knuth-mastermind.pdf.