Mastermind AI

CSC-7 42486
Branden Hitt
Date: 06/04/2018

Table of Contents

Introduction

Title: Mastermind AI

This is a small program that implements an AI (artificial intelligence) to play the game Mastermind.

More information on the game Mastermind can be found at: https://en.wikipedia.org/wiki/Mastermind_(board_game)

The game of Mastermind is played 1000 times with the program keeping track of certain statistics that the AI has achieved. Example Output:

Average turns to win: 14

Shortest game: 2

Longest game: 18

Games played: 1000

In addition to statistics, the program also outputs some information for each game played. Example Output:

****

Turns this game: 12

Final Guess: 2158

Key: 2158

****

Turns this game: 14

Final Guess: 3478

Key: 3478

****

Turns this game: 18

Final Guess: 1058

Key: 1058

*The program simulates all 1000 games with output in about 1 second.

Summary of AI

Ai Function size: 256 lines

For the Mastermind AI, I developed my own pattern to narrow down possibilities. I decided it would be more fun if I avoided any existing approaches and instead came up with my own original solution.

The AI begins by guessing 0011. If the response shows that there is a correct number in the correct place (CC) or a correct number in the wrong place (CW), then the next guess would be only the evens 0000. Following the response to that, the Ai then tries to store the best possibilities of where each number would go and how many would be possible inside of one string that is 16 characters long (4 possible found digits for each slot).

Following the change to the string of possibilities, the AI then moves up an index and guesses the next 2 numbers 2233. The program continues in this fashion until it knows the four digits that are in the code and some narrowed down possibilities of their individual positions. Note: the AI also keeps track of a dead index (one where no numbers are present in the key) to use later.

The final decisions made by the AI are to narrow down guesses from its string of possibilities. For example, if the string was “10xx10xx57xx57xx” with x being empty spaces, then it would start off guessing 1888. The guess used a dead index (in this case 8) to learn information about the first digit. If the first digit is incorrect then it would change the next guess to 0888. However, if the first digit was correct, then it would move on to the next digit and guess 1188. The guessing would continue to narrow down until the key is solved.

Future Improvements

My biggest obstacle with this pattern of guessing was running into codes that had an odd and even number from the same index in the same row. For example, 1058 would be harder to guess since 1 and 0 come from the same index and there isn’t much information gained from the first and second guesses 0011, 0000. If I could better narrow down the possibilities in this case, then the AI would have a lower worst case as well as a better average. From testing, I could see that this was easily adding 4 to 5 extra guesses.

Moving forward, I would either continue to attempt to solve the previous problem or I would simply start from scratch and come up with a different algorithm.

Pseudocode

```
/*
 * Author: Branden Hitt
 * Purpose: implement an AI to play mastermind
 */

/*
 * File:   main.cpp
 * Author: bhitt
 * Created on June 01, 2018, 11:34 AM
 */

//io
//rand
//regular expression
//string.length

/*
 *
 */
//Function Prototypes

    //declare variables
    //set random seed
     //variables
    //loop over the games played
        //create answer key
        //default response to create a new game
        //run through guesses
           //prompt for guess
           //increment turn count
           //check for win

        //find average
        //check for shortest or longest game

        //single game testing output
    //compute average

    //Output statistics
    //exit stage right

//function to create a key

//function to evaluate the most recent guess and provide feedback
    //Check how many are in the correct place
    //Check how many are wrong place
    //Found or not
    //static variables in ai function
    //new game reset
```

```
        //cout<<"NEW GAME START"<<endl;
    //variables
    //evaluate response
    //initial phase
        //add to counts
    //secondary phase
        //add to poss
        //only evens from the pair
            //rare case where 3 evens and one odd (5444)
                //add odds
                //some evens were in the correct place
                //some evens were in the wrong place
        //only odds from the pair
            // the odds were in the wrong place
            //the odds were in the correct place
        //both evens and odds
            //add the evens
            //add the odds
        //three evens
        //all the same even number

        //reset both holds
    //construct a new guess
        //guess half even, half odd to start
        //guess all even to narrow down
        //check for first guess during exodus phase
        //utilizing dead index, narrow down from poss string
            //guessing first digit
            //guessing second digit
            //guessing third digit
            //guessing fourth digit
    //cout<<"poss:"<<poss<<endl;  (testing output)
    //cout<<"guess:"<<guess<<endl; (testing output)

    //return the guess
```

Program Code

```
/*
 * Author: Branden Hitt
 * Purpose: implement an AI to play mastermind
 */

/*
 * File:   main.cpp
 * Author: bhitt
 * Created on June 01, 2018, 11:34 AM
 */

#include <iostream> //io
#include <cstdlib>  //rand
#include <regex>    //regular expression
#include <string>   //string.length

using namespace std;

/*
 *
 */
//Function Prototypes
```

```
string createK( int);                       //create key (dupes allowed, four digits)
bool evaluate(string, string,char &, char &);  //check for game win
string aiGuess(char , char);                //AI guessing function

int main(int argc, char** argv) {
   //declare variables
   char cDigW= 'X', cDigC= 'X';    //correct digit in wrong place or correct digit in correct place
   short turnC=1;                  //counter for turns until win
   short upper = 0, lower = 99;    //upper and lower ranges for turn count
   int average = 0;                //average turns to win the game
   int gameC=0;                    //number of games played
   int turnT=1;                    //total turns over every game

   //set random seed
    srand(static_cast<unsigned int>(time(0)));
    //variables
   string answerK = "    ";
   string guess = "    ";
   int range = 10;  //digits allowed

   //loop over the games played
   for(int i=0;i<1000;i++){
      gameC++;
      turnC=0;
      //create answer key
      answerK = createK(range);
      //default response to create a new game
      cDigW = 'X', cDigC = 'X';
      //run through guesses
      do{
        //prompt for guess
        guess = aiGuess(cDigW,cDigC);
        //increment turn count
        turnC++;
        //check for win
      }while(evaluate(guess, answerK, cDigW, cDigC));

      //find average
      turnT += turnC;
      //check for shortest or longest game
      if(turnC < lower) lower = turnC;
      if(turnC > upper) upper = turnC;

      //single game testing output
      cout<<"**********************"<<endl;
      cout<<"Turns this game: "<<turnC<<endl;
      cout<<"Final Guess: "<<guess<<endl;
      cout<<"Key: "<<answerK<<endl;
      cout<<endl;

   }
   //compute average
   average = turnT/gameC;

   //Output statistics
   cout<<"Average Turns To Win: "<<average<<endl;
   cout<<"Shortest game: "<<lower<<endl;
   cout<<"Longest game: "<<upper<<endl;
   cout<<"Games played: "<<gameC<<endl;
   //exit stage right
```

```
    return 0;
}

//function to create a key
string createK(int range){
    string code="0000";
    for(int i=0;i<code.length();i++){
        code[i]=rand()%range+'0';
    }
    return code;
}

//function to evaluate the most recent guess and provide feedback
bool evaluate(string guess, string code, char &cDigW, char &cDigC){
    int cW=0,cC=0;
    string check="    ";
    //Check how many are in the correct place
    for(int i=0;i<code.length();i++){
        if(code[i]==guess[i]){
            cC++;
            check[i]='x';
            guess[i]='x';
        }
    }
    //Check how many are wrong place
    for(int j=0;j<code.length();j++){
        for(int i=0;i<code.length();i++){
            if((i!=j)&&(code[i]==guess[j])&&(check[i]==' ')){
                cW++;
                check[i]='x';
                break;
            }
        }
    }
    cDigC = cC + '0';
    cDigW = cW + '0';
    //Found or not
    if(cC==4)return false;
    return true;
}

string aiGuess(char cDigW, char cDigC){
    //static variables in ai function
    static string guess  = "    ";
    static string counts = "     ";
    static string poss   = "                ";
    static bool hold1 = false;
    static bool hold2 = false;
    static bool exodus = false;
    static bool oneTime = true;
    static short index = -1;
    static short deadInd = 0;
    static short fIndex = 0;
    static short pastCW = 0;
    static short pastCC = 0;
    static short fInd1 = 0;
    static short fInd2 = 0;
    static short fInd3 = 0;
    static short fInd4 = 0;
    if(cDigC == 'X'){ //new game reset
```

```
    //cout<<"NEW GAME START"<<endl;
    guess = "    ";
    counts = "     ";
    poss = "               ";
    hold1 = false;
    hold2 = false;
    exodus = false;
    oneTime = true;
    index = -1;
    deadInd = 0;
    fIndex = 0;
    pastCW = 0;
    pastCC = 0;
    fInd1 = 0;
    fInd2 = 0;
    fInd3 = 0;
    fInd4 = 0;
    cDigW = '0';
    cDigC = '0';
}

//variables
short cC = cDigC -'0';
short cW = cDigW - '0';
short evens = 0;
short odds = 0;
short indC = 0;


//evaluate response
if((cC + cW > 0) && !hold1) hold1=true;
if((cC + cW == 0) && !hold1 && !exodus) deadInd = index;

//initial phase
if(hold1 && !hold2 && !exodus){
    //add to counts
    if(!exodus)counts[index] = cC + cW + '0';
    //set hold2 flag
    hold2=true;
    pastCW = cW;
    pastCC = cC;
}else if(hold2 && !exodus){
//secondary phase
    indC = counts[index] - '0';
    evens = cC;
    odds = indC - evens;
    //add to poss
    if(evens == indC){    //only evens from the pair
        //rare case where 3 evens and one odd (5444)
        if(indC==3 && cC ==3){ //(4454 or 4445)
            for(int j=0;j<4;j++){
                if(poss[j]==' ' && poss[j+8]==' '){
                    poss[j] = index*2 + '0';
                    poss[j+4] = index*2 + '0';
                    poss[j+8] = index*2 + '0';
                    poss[j+12] = index*2 + '0';
                    break;
                }
            }//add odds
            for(int j=0;j<4;j++){
```

```
            if(poss[j]==' ' && poss[j+8]==' '){
                poss[j] = index*2 + 1 + '0';
                poss[j+4] = index*2 + 1 + '0';
                poss[j+8] = index*2 + 1 + '0';
                poss[j+12] = index*2 + 1 + '0';
                break;
            }
        }

    }
    else{
        if(pastCC > 0){ //some evens were in the correct place
            for(int i=0;i<pastCC;i++){
                for(int j=0;j<4;j++){
                    if(poss[j]==' '){
                        poss[j] = index*2 + '0';
                        poss[j+4] = index*2 + '0';
                        break;
                    }
                }
            }
        }
        if(pastCW >0){ //some evens were in the wrong place
            for(int i=0;i<pastCW;i++){
                for(int j=8;j<12;j++){
                    if(poss[j]==' '){
                        poss[j] = index*2 + '0';
                        poss[j+4] = index*2 + '0';
                        break;
                    }
                }
            }
        }
    }

}
if(evens == 0){    //only odds from the pair
    if(pastCW > 0){ // the odds were in the wrong place
        for(int i=0;i<pastCW;i++){
            for(int j=0;j<4;j++){
                if(poss[j]==' '&& poss[j+8]==' '){
                    poss[j] = index*2 + 1 + '0';
                    poss[j+4] = index*2 + 1 + '0';
                    break;
                }
            }
        }
    }
    if(pastCC >0){ //the odds were in the correct place
        for(int i=0;i<pastCC;i++){
            for(int j=8;j<12;j++){
                if(poss[j]==' '){
                    poss[j] = index*2 + 1 + '0';
                    poss[j+4] = index*2 + 1 + '0';
                    break;
                }
            }
        }
    }
}
```

```
    if((evens > 0) && (evens < indC)){  //both evens and odds
        //add the evens
        for(int j=0;j<4;j++){
            if(poss[j]==' '&& poss[j+8]==' '){
                poss[j] = index*2 + '0';
                poss[j+4] = index*2 + '0';
                poss[j+8] = index*2 + '0';
                poss[j+12] = index*2 + '0';
                break;
            }
        }
        //add the odds
        for(int j=0;j<4;j++){
            if(poss[j]==' '&& poss[j+8]==' '){
                poss[j] = index*2 + 1 + '0';
                poss[j+4] = index*2 + 1 + '0';
                poss[j+8] = index*2 + 1 + '0';
                poss[j+12] = index*2 + 1 + '0';
                break;
            }
        }
    }
    if(evens == 3){ //three evens
        for(int i=0;i<3;i++){
            for(int j=0;j<4;j++){
                if(poss[j]==' '&& poss[j+8]==' '){
                    poss[j] = index*2 + '0';
                    poss[j+4] = index*2 + '0';
                    poss[j+8] = index*2 + '0';
                    poss[j+12] = index*2 + '0';
                    break;
                }
            }
        }
    }
    if(evens == 4) { //all the same even number
        for(int j=0;j<4;j++){
                if(poss[j]==' '&& poss[j+8]==' '){
                    poss[j] = index*2 + '0';
                    poss[j+4] = index*2 + '0';
                    poss[j+8] = index*2 + '0';
                    poss[j+12] = index*2 + '0';
                    exodus = true;
                    break;
                }
            }
    }
    //reset both holds
    hold1=false;
    hold2=false;
    index++;
    if(index>4) index=4, exodus=true;
}else{
    index++;
    if(index>4) index=4, exodus=true;
}
//construct a new guess
if(!hold1 && !exodus){
    //guess half even, half odd to start
    guess[0] = index*2 + '0';
```

```
    guess[1] = index*2 + '0';
    guess[2] = index*2 + 1 + '0';
    guess[3] = index*2 + 1 + '0';
  }else if(hold2 && !exodus){
    //guess all even to narrow down
    guess[0] = index*2 + '0';
    guess[1] = index*2 + '0';
    guess[2] = index*2 + '0';
    guess[3] = index*2 + '0';
  }else if(exodus){
    //check for first guess during exodus phase
    if(!oneTime){
      if(cC == 1 && fIndex!=1) fIndex = 1, fInd1--;
      if(cC == 2 && fIndex!=2) fIndex = 2, fInd2--;
      if(cC == 3 && fIndex!=3) fIndex = 3, fInd3--;
    }
    //utilizing dead index, narrow down from poss string
    if(fIndex == 0){
      //guessing first digit
      while(poss[fInd1]==' '){
        fInd1++;
      };
      guess[0]=poss[fInd1];
      guess[1]=deadInd*2 + '0';
      guess[2]=deadInd*2 + '0';
      guess[3]=deadInd*2 + '0';
      fInd1++;
      oneTime=false;
    }else if(fIndex == 1){
      //guessing second digit
      while(poss[fInd2 + 4]==' '){
        fInd2++;
      };
      guess[0]=poss[fInd1];
      guess[1]=poss[fInd2 + 4];
      guess[2]=deadInd*2 + '0';
      guess[3]=deadInd*2 + '0';
      fInd2++;
    }else if(fIndex == 2){
      //guessing third digit
      while(poss[fInd3 + 8]==' '){
        fInd3++;
      };
      guess[0]=poss[fInd1];
      guess[1]=poss[fInd2 + 4];
      guess[2]=poss[fInd3 + 8];
      guess[3]=deadInd*2 + '0';
      fInd3++;
    }else{
      //guessing fourth digit
      while(poss[fInd4 + 12]==' '){
        fInd4++;
      };
      guess[0]=poss[fInd1];
      guess[1]=poss[fInd2 + 4];
      guess[2]=poss[fInd3 + 8];
      guess[3]=poss[fInd4 + 12];
      fInd4++;
    }
  }
```

```
    //cout<<"poss:"<<poss<<endl;  (testing output)
    //cout<<"guess:"<<guess<<endl; (testing output)

    //return the guess
    return guess;
}
```