Bulls and Cows: Strategies and Techniques*

Team 4 - Bullseye†
Kolluru Mohan Sai Krishna Prasada Rao (170010022), Gautam Jagdhish M (170010031),
Swapnik Jagarlapudi (170010033), Ashrith Kumar Peddi (170010036)
Computer Science and Engineering, IIT Dharwad

November 17, 2018

Abstract

This paper describes the strategies and techniques followed by the program written by our team Team 4 - Bullseye for the *Bulls and Cows* code-breaking game condcuted as part of the course CS 211 Data Structures and Algorithms Lab.

1 The algorithm used

The numbers that the program provides as guesses are stored in an array (say S). The algorithm to guess the opponent's secret number works as follows. When the program provides a guess (say y) for the opponent's secret number, the opponent returns the number of bulls, say b, and cows, say c. Then the program goes through each element (say x) in the array and compares them with the number provided as a guess. If the x were the secret number, then the comparison would return number of bulls as b and number of cows as c. So the set of all such numbers x which return b as the number of bulls and c as the number of cows when compared with y is kept in the array and the remaining numbers are eliminated, thus narrowing down the number of possible guesses each time a number is guessed. No team-specific strategies were used, however, the method of generating random numbers has been changed over the coures of the league matches.

*This is a report on the course project for the course CS 211 Data Structures and Algorithms Lab

†Email IDs of team members:
170010022@iitdh.ac.in
170010031@iitdh.ac.in
170010033@iitdh.ac.in
170010036@iitdh.ac.in

2 Structure of the program

The program is made up of 2 functions and 3 loops. One function generates a random number and the other function clears numbers which will not be used as guesses; these numbers are decided based on the algorithm described in section 1. These functions and loops will be described in detail below.

2.1 string padeyrand()

Pseudocode for the function

```
string padeyrand(){
    srand( time(NULL) )
    random_number = S[rand() % S.size()]
    for i in {1..S.size()}
                if S[i] == random_number
                    then remove S[i];
    return give
}
```

This function provides a random guess from the set of numbers available to be used as guesses. The random number is generated using a garbage value (say r), which is a variable that is declared, but not assigned any value. The number to be provided as a guess is then chosen from the set of elements available as a guess, S, by generating an index by dividing r with the size of this vector and taking the remainder i.e. r%S.size(); the random number will be S[r%S.size()]. This works as the remainder is guaranteed to be less than the size of the number, so it will not go out of bounds. This number is provided as a guess, then subsequently removed from S. Earlier, a time-seeded pseudorandom number generated by the C++ function rand() was used to calculate the index. However, given a very small time gap of execution, it would provide numbers that are very close to each other in value, rendering it unreliable, so the strategy of using of a garbage value was implemented.

2.2 void clear(string some,int bull,int cows)

Pseudocode for the function

```
clear(string random_number, int bull, int cows){
for i in {1..S.size()}
        temp = S[i]
        b = 0, c = 0
        for j in {1..4} //this loop calculates no. of bulls
                if random_number[j] == temp[j]
                        then b++
```

```
                else for k in {1..4} //this loop calculates no. of cows
                        if(random_number[k] == temp[k])
                                then c++
        for i in {1..S.size()}
                if b != bull or c != cows
                        remove S[i]
}
```

This function implements the algorithm described in section 1. The first loop compares each element in the set with the random guess using the first function and calculates the number of bulls and cows obtained for each number, and the second loop removes all elements with a different number of bulls and cows, pruning the set.

2.3 The body of the program

The body of the program consists of three loops. The first loop creates an array of 4536 numbers, which are 4-digit numbers containing unique digits.

Pseudocode for the first loop

```
ind = 0
for i in {1..9}
  for j in {1..9}
        for z in {1..9}
          for h in {1..9}
                if i != j & i != z & i != h & j != z & j != h & z != h
                  a = 1000*i + 100*j + 10*z + 1*h
                  S[ind] = a
                  ind = ind + 1
```

After the execution of this loop, the program generates a random number using a garbage value, which is simply a variable that is declared but is not assigned any value. The second loop is initiated if the player is asked to guess first. The function padeyrand() is used to give a guess, then the clear function is used to prune the set of numbers. Then, it takes the guess of the opponent and returns the number of bulls and cows in the opponent's guess. This is an indefinite loop; it is terminated by the referee.

Pseudocode for generating secret number and the second loop

```
long long int z //this is a garbage value
sec = S[z%4536] //this is the secret number
If initial == 1 //this value indicates that the program
                                //must guess first
        while(true)
```

```
                a = padeyrand()
                output < a //the program gives the guess
                obull = 0, ocow = 0 //bulls and cows in opponent's
                                                    //guess
                input > bull > cow //the opponent gives the no. of
                                //bulls and cows in program's guess
                clear(a, bull, cow) //runs algorithm and prunes the set
                input > opponent_guess //opponent give their guess
                for i in {1..4} //this loop calculates no. of bulls
                        if sec[j] == opponent_guess[j]
                                then obull++
                        else for  in {1..4} //this calculates no. of cows
                                if sec[k] == opponent_guess[k]
                                        then ocow++
                                                break
                output < obull
                output < ocow
```

The third loop is essentially the same as the second loop, but the opponent gives their guess first and the program gives its guess later. It is initiated when the program is prompted to provide its secret number first.

The third loop

```
If initial == 0 //this value indicates that the program
                //must provide its secret number first
        while(true)
                obull = 0, ocow = 0
                input > opponent_guess
                for i in {1..4}
                        if sec[j] == opponent_guess[j]
                                then obull++
                        else for  in {1..4}
                                if sec[k] == opponent_guess[k]
                                        then ocow++
                                                break
                output < obull
                output < ocow
                a = padeyrand()
                output < a
                input > bull > cow
                clear(a, bull, cow)
```