Project Algorithm

지능기전공학부 스마트기기공학전공
18011793 이정민

- algorithm for guessing numbers(Server version)

```
#initialization
myguess = [1, 2, 3] #Answer of client predicted by server
my_SB = [0, 0] #The number of strike and ball in myguess
clientguess = [0, 0, 0] #Answer of server predicted by client
client_SB = [0, 0] #The number of strike and ball in clientguess
guess = [1, 2, 3, 4, 5, 6, 7, 8, 9]  #The numbers that can be the client's answer.
data = [] #[myguess + my_SB] append -> If flag = 1, [myguess] append
flag = -1 #Pass the first order of the server and determine if three numbers corresponding to the client's answer were found
```

The declared variables are as above (declaration and initialization of variables before starting an infinite loop).

1. When predicting numbers, the first, second, and third attempts are fixed at 123, 456, and 789, respectively. Then add [1, 2, 3, S, B], [4, 5, 6, S, B] and [7, 8, 9, S, B] in order to data.
   The reason for trying 123, 456, and 789 is to find out how many numbers each group has in the client's answer.

```
elif len(data) < 3 : # If the prediction attempt is 3 or less
      myguess = [i for i in range(3*len(data)+1,3*(len(data)+1)+1)]
      #First : [1, 2, 3], second : [4, 5, 6], third : [7, 8, 9]
```

```
#Until the third attempt, and when flag = 0, [myguess + my_SB] append
if len(data)<3 and flag == 0:
      data.append(myguess+my_SB)
```

   (The append code is in an infinite loop. After passing through the first order of the server, switch to flag = 0.)

2. From the fourth attempt, select the number randomly as the number of strike + ball in [1, 2, 3], [4, 5, 6], and [7, 8, 9] respectively. For example, if the answer is 348, one number from each of the three groups will be randomly selected.
   c2 is the intersection of c1 and guess. It plays a role in eliminating numbers that are not the correct answer. The detailed description is written on page 3.

(data = [[1, 2, 3, S, B], [4, 5, 6, S, B], [7, 8, 9, S, B]])

```
else:
      c1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
      c2 = [list(set(guess).intersection(i)) for i in c1] #c2 selects only values that overlap both c1 and guess.

      myguess = random.sample(c2[0], data[0][3]+data[0][4]) #Randomly select by the number of SB in [1, 2, 3].
      myguess = myguess+random.sample(c2[1], data[1][3]+data[1][4]) #Randomly select by the number of SB in [4, 5, 6].
      myguess = myguess+random.sample(c2[2], data[2][3]+data[2][4]) #Randomly select by the number of SB in [7, 8, 9].
```

3. Regardless of number 1 or 2, if three numbers corresponding to the correct answer are found, there are two cases.

3-1. If the predicted number is 3 balls

In other words, the positions of the predicted numbers do not all match. At this point, swap each position of the previously predicted number to the right and then predict the number. For example, if the correct answer was 512 and the predicted number was 251, 125 swapped to the right would be the next predicted value, followed by 512.

3-2. If the predicted number is not three balls

Randomly select three numbers corresponding to the correct answer and make them three digits. However, choose a number that is not included in the data list that contains the previously predicted number. That is, if you find three numbers, you avoid duplication of the predicted number.

```
if len(guess) == 3: #If the server finds three numbers corresponding to the client's answer
    if my_SB[1] == 3 : #If the number I predicted is 3 balls, swap 3 digits to the right
        tmp = myguess[0]
        for i in range(2):
            myguess[i] = myguess[i+1]
        myguess[2] = tmp
    else : #If not 3 ball
        while True : #Randomly find three digits in the guess list, excluding the previously predicted number
            myguess = random.sample(guess, 3)
            if myguess not in data :
                break
```

Among the codes described to date, they are executed in the guessNumber function to predict the number, excluding the append code. Returns the predicted myguess list.

```
def guessNumber(guess, my_SB, data, myguess): #guess number
    if len(guess) == 3: #If the server finds three numbers corresponding to the client's answer
        if my_SB[1] == 3 : #If the number I predicted is 3 balls, swap 3 digits to the right
            tmp = myguess[0]
            for i in range(2):
                myguess[i] = myguess[i+1]
            myguess[2] = tmp
        else : #If not 3 ball
            while True : #Randomly find three digits in the guess list, excluding the previously predicted number
                myguess = random.sample(guess, 3)
                if myguess not in data :
                    break
    elif len(data) < 3 : # If the prediction attempt is 3 or less
        myguess = [i for i in range(3*len(data)+1,3*(len(data)+1)+1)]
        #First : [1, 2, 3], second : [4, 5, 6], third : [7, 8, 9]
    else:
        c1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
        c2 = [list(set(guess).intersection(i)) for i in c1] #c2 selects only values that overlap both c1 and guess.

        myguess = random.sample(c2[0], data[0][3]+data[0][4]) #Randomly select by the number of SB in [1, 2, 3].
        myguess = myguess+random.sample(c2[1], data[1][3]+data[1][4]) #Randomly select by the number of SB in [4, 5, 6].
        myguess = myguess+random.sample(c2[2], data[2][3]+data[2][4]) #Randomly select by the number of SB in [7, 8, 9].
    return myguess
```

When code is executed, servers and clients predict each other's answers within an infinite loop. Even within an infinite loop, codes are executed to help predict answer by helping the function of the guessNumber.

```
#If there are no strikes or balls in any of the numbers client predict, Delete a number from guess
if my_SB[0]+my_SB[1] == 0 :
    for i in range(3):
        if myguess[i] in guess :
            del guess[guess.index(myguess[i])]
    if len(guess) == 3:
        data = [] #data list initialization
        flag = 1 #Because we found three numbers that correspond to the server's answer, change flag = 1
#If the number of strikes and balls in the numbers client predicted is 3,
if my_SB[0]+my_SB[1] == 3 and flag == 0  :
    guess = myguess
    data = [] #data list initialization
    flag = 1 #Because we found three numbers that correspond to the server's answer, change flag = 1


#Until the third attempt, and when flag = 0, [myguess + my_SB] append
if len(data)<3 and flag == 0:
    data.append(myguess+my_SB)

#[myguess] append
if flag == 1:
    data.append(myguess)
```

1. If the number predicted by the server does not have any strikes or balls

```
guess = [1, 2, 3, 4, 5, 6, 7, 8, 9]
#The numbers that can be the client's answer.
```

The guess list initializes like this at first.

If there are zero strikes and balls of the numbers I predict, they are not answers. Therefore, remove the numbers I predicted from the guess list.

2. If the number of strikes and balls of the answers predicted by the server is 3

In this case, all three numbers corresponding to the correct answer were found. Therefore, copy myguess list, the number I predicted, to guess list. Then, reset the data list. Then change the flag value to 1.

In the figure above, flag = 1 in the last if statement means that all three numbers corresponding to the correct answer have been found. At this time, the data list is appended to myguess list, a three-digit number I predicted. This process avoids duplication of the number predicted in 3-2 above.