DOLLARS AND CENTS

Amrit Lal Singh

Abstract— **Here lies the implementation of Dollars and cents in cpp. The ros part of the code was facing some consistent errors and the user had to abandon ros checker implementation. in this code the person running the code is the checker and has to output dollars and cents; Otherwise the code is perfect and works well**

You are provided with a sequence of integers of length n. You can make guesses on what that number can be. Once you make a guess in return you will receive 2 outputs i.e. number of 'Cents' and 'Dollars'.'Cents' denote the number of digits guessed which are present in the given sequence but are out of position, whereas 'Dollars' represent the number of digits guessed which are present in the sequence at the very position you guessed them at. As you might have guessed it, the game ends when you get exactly n Dollars. Your solution must be optimised enough to run for strings of length 10.

Although I was not able to implement with ROS I wrote the logic in cpp; How the code works is it outputs the order of the digits and instead of the checker function we behaving as the checker enter the dollars and cents and then this repeats till we reach the solution and congratulation message prints. The second part of the problem can be solved using the same logic as if we are given x we can convert it into dollars and cents. Let's say that we receive y as the x count this is assuming we have guessed all the digits present in the code. Then the number of imaginary dollars would be y - (number of digits) and cents would be (number of digits) - dollars. Hence converting the problem into the first one.

This code works ahead when we have guessed all the correct digits present in the number which is doable in case of the first part just by inputting each permutations and if the dollars plus cents reach the number of digits we have our correct set of digits and then the work of the code begins; However this is a fatal flaw in case of when we implement this code for the second part as guessing if the correct set of digits is present in the current set is a hard task. Let's say if we input a select permutation and one of the digit is in its correct place then its contribution to x that is 2 may shadow the no contribution of a digit that is not even in that code and may lead us to believe that we have the proper set to start trying to arrange them in the correct order. Otherwise, if the string length is 10 both the codes will work perfectly.

I. INTRODUCTION

Describe the problem statement and your approach using which you have approached the problem. Describe all the things and methods you have tried which might have failed in brief. The problem wants us to make some changes and then see if the output has increased or decreased, and then react appropriately. In turn iterating over these we have to reach the solution.

II. PROBLEM STATEMENT

This should cover one full column of page.

*The internet and to it's seeming infinite store of knowledge. The gods of stack overflow

Explain in details what your problem statement was with all the necessary images and equations required. Part 1: We at the Aerial Robotics Lab, Kharagpur have been avid fans of the popular Internet phenomenon Wordle, so we came up with our own(and more intellectual?) version of it, called Cents and Dollars. You are provided with a sequence of integers of length n. You can make guesses on what that number can be. Once you make a guess in return you will receive 2 outputs i.e. number of 'Cents' and 'Dollars'.'Cents' denote the number of digits guessed which are present in the given sequence but are out of position, whereas 'Dollars' represent the number of digits guessed which are present in the sequence at the very position you guessed them at. As you might have guessed it, the game ends when you get exactly n Dollars. Your solution must be optimised enough to run for strings of length 20. Your task is to implement the solution using a template given in ROS. Structure of the given ROS program is for you to read and understand :). Clone this git repository to find the template. Part 2: For the second part of this task, everything stays the same just that instead of 'Dollars' and 'Cents' provided to you after every interaction, you will be provided the a value X. The output of the interaction will be as follows +2 for every digit guessed that is present in the string in the correct position +1 for every digit guessed that is present in the string but is out of position. X will basically be the summation of these values. The game ends when you reach the state X=2n where n is the length of the string. Using the above interaction, you have to try to correctly guess the string

III. RELATED WORK

Wordle has been solved using methods but this problem is quite different to it. It is approached as a new problem.

IV. INITIAL ATTEMPTS

One of the initial attempt was to do brute force but as that was clearly not optimised. Next tried implementing genetic algorithm on the problem but did not led it to completion. As a method, probably it would have worked well. Then the current method evolved and worked perfectly.

V. FINAL APPROACH

This should cover both columns (full page) excluding images

We start with including bits/stdc++.h and declaring functions to print the current vector and also a finishing congratulation message. Then the function check if increased is declared which takes in the vector v the vector value at index the dollars and a boolean if increased to state true or

false. In this, V is passed as a copy and increased is passed by reference as we will need to change increased, but we do not want our testing on v to spoil the initial copy. This function swaps all allowed indexes with the given index to see if our dollars increased. also, the function has to ignore a swapping place if the correct value for that place has been found earlier. We start a loop and swap indexes and print the array; the print is equivalent to sending it to the checker function also we check ID the number of dollars are equal to the size, in that case we have our solution. we also swap back in case.

How the program works is that we input n and the main function calls the play game function with the only input as n which is the number of digits. First, the play game function guesses the correct set of integers. this could have been implemented into the code itself, but it would have made the process tedious for us as the checker because we may have to answer the dollars and cents for the code a lot of time. The first iteration of code started with ROS, but I faced a recurrent error in ROS and everything stopped working after it. then we have to output what digits are present or absent in the code. then the code makes the first guess, and we have to return the output and visible in the outputs it swaps each digit in the set one by one, skipping the ones that are fixed.

VI. Results and Observation

We as the slow checkers make the problem time-consuming. Else, the number of iteration it takes to solve is probably most optimised. The results are quite satisfying and up to the mark the problem after guessing the set seems optimally solved and if the problem ends up getting solved before the algorithm has ended then checks are included to print that the problem has been solved.

The logic of the second part of the problem is also correct but the set guessing part of the second problem stays cloudy. Brute force methods for guessing the sets can also be implemented.

VII. Future Work

Future work would involve implementing this in ROS; also designing a logic to optimally find the set of strings for the second part as it stays unsolved for now. Ahed of the code that will order the elements of the set using the input of checker is perfect

CONCLUSION

Write overall about what the problem was, how you solved it, difficulties faced and the net output with it's usefulness to ARK or in general. The difficulties faced was first in learning ROS and then inadequate knowledge of Linux as an operating system, despite using the Operating System for years, what does what and how things work. The amount of ROS and the lack of solving this sort of problems in python pushed me back to cpp in which I have the basics clear.

References

[1] There were no references