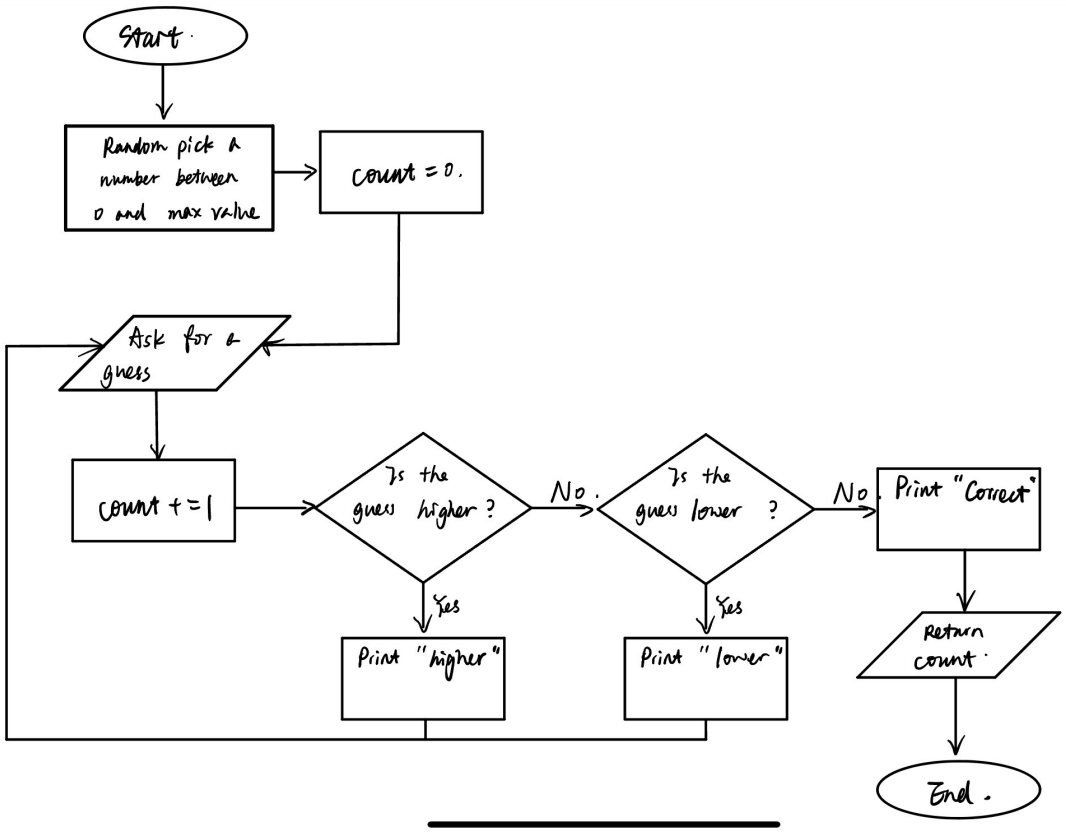
Quan Yuan
Oct 8, 2022
Project 3 Report

SUMMARY
This project contains 4 tasks, combining to form a program that random pick a number and ask the client to guess. Tell them either it is higher/lower or it is hotter/colder.

The following parts are organized by tasks, containing a brief problem description, a flowchart of the algorithm, a reflection, and an acknowledgment if applicable.

TASK 1:
This task generate a random number from given range, and ask the client to guess. Then tell them whether it is too high or too low. Then return the total number of attempts.

The flowchart is as follows:



The algorithm is to use while loops. The breaking condition is that guess is equal to the random number generated.

To make the code easier for tests and cleaner to review, I break the whole part into 2 functions, one is the basic structure of higher/lower guess. It takes in a max_value as the range, then generate a random number between 0 and max_value. Then it calls while loop to ask for client's guess until it is correct. Then it calls efficiency function to determine whether the total attempts is efficient.

The other function is to determine whether the guess is higher or lower than the generated number. The function will take in 2 parameters, the random number and the guess. Within this function, it will realize the print command to tell the client whether it is too high or too low. Also the function will return 1 if higher, -1 if lower, and 0 if correct.

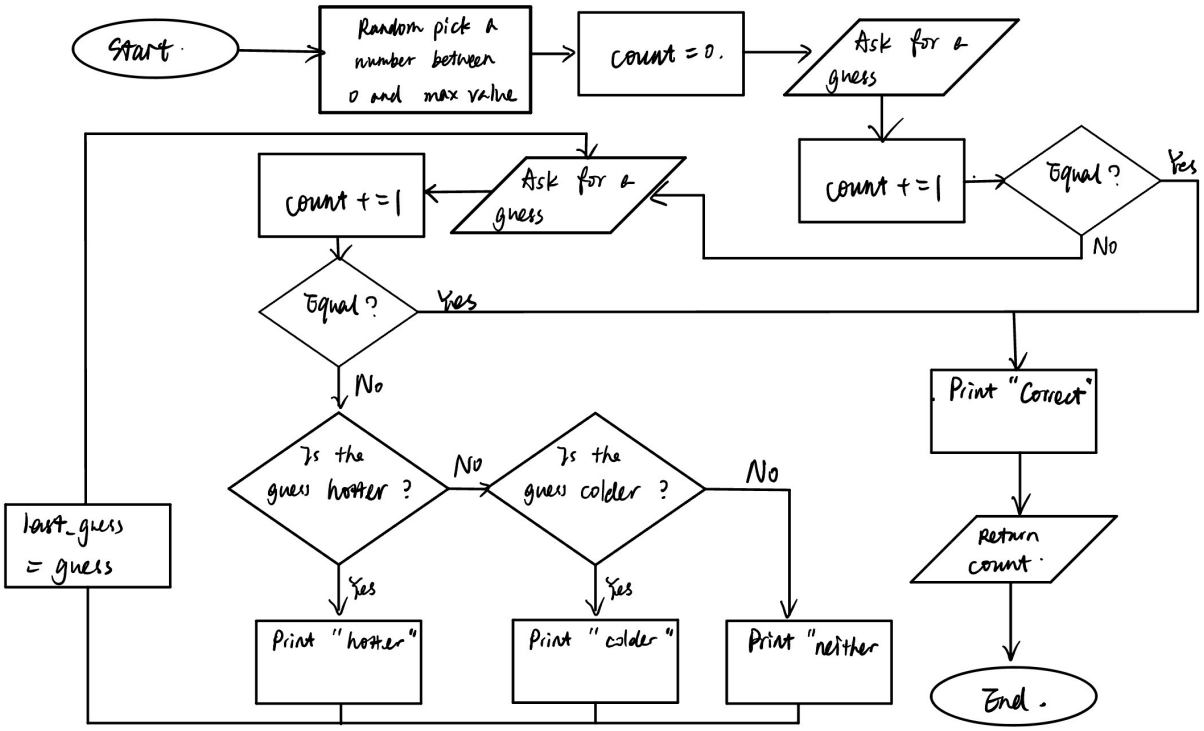
In this case, we could simply test the judging function to see if it returns the correct value, instead of the whole part which contains an input requirement.

The reason I want to create the wrapping function rather than put all the codes in main() function is that it will easier for others to review and understand the logic of the code.

Acknowledgement:
Discussion with Professor Rasika regarding how to test the function if there is an input command.

TASK 2:
This task generate a random number from given range, and ask the client to guess. Then tell them whether it is too high or too low. Then return the total number of attempts.

The flowchart is as follows:



The algorithm is also to use while loops. The breaking condition is that guess is equal to the random number generated.

Same as task 1, I break the whole part into 2 functions, a wrapping function and an inner function. The wrapping function takes in a max_value as the range, then generate a random number between 0 and max_value. Then it calls while loop to ask for client's guess until it is correct. Then it calls efficiency function to determine whether the total attempts is efficient.

One thing it is different than task 1 is that this function has to track the previous guess so that it can determine whether the current guess is closer or farther. Hence, before the while loop, we have to ask for a separate guess, saved as last_guess. And check if the it is correct. If not, we then execute the while loop to ask for more guesses and tell whether it

is hotter or colder. Finally, it calls efficiency function to determine whether the total attempts is efficient.

The inner function is to determine whether the guess is hotter or colder than last guess. The function will take in 3 parameters, the random number, current guess, and last guess. Within this function, it will realize the print command to tell the client whether it is hotter or colder or neither. Also the function will return 1 if hotter, -1 if colder, 2 if neither, and 0 if current guess is correct.

There is a corner case we have to consider, that is the current guess is neither hotter nor colder than last guess. For example. the random number is 10. Our last guess is 5, and current guess is 15. Both the differences are 5. So it is neither hotter nor colder. Hence, we need to reflect this situation in our code.

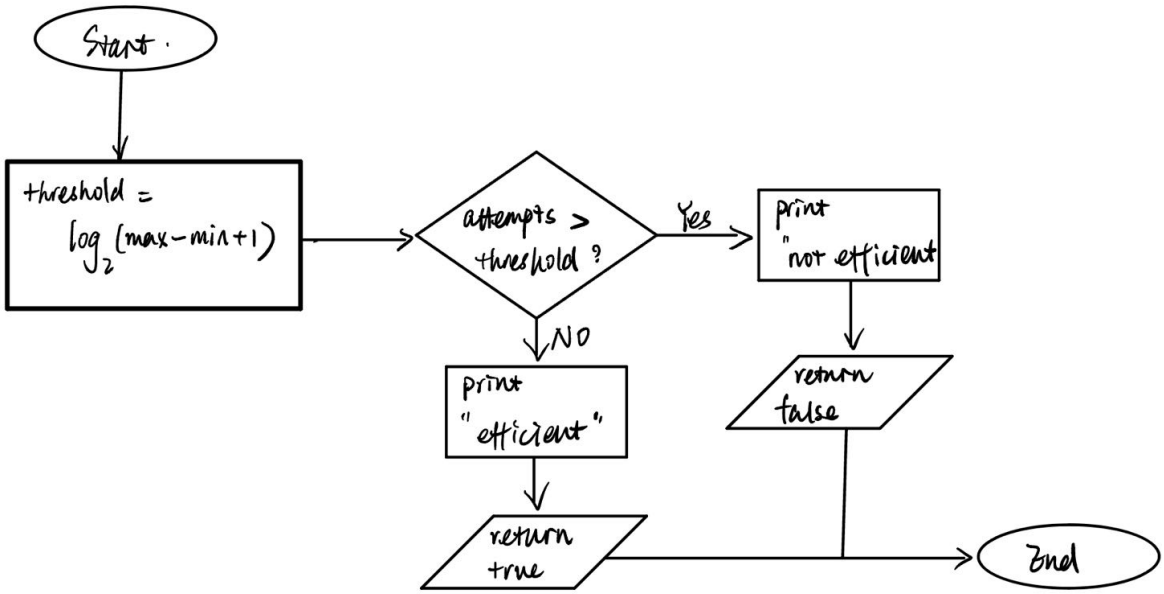
And I decide to return 2 in this situation because this case could rarely happen, and we do not use it as a judging condition. We break the loop when the current guess is correct, which will return 0. It is more clear to understand, and also align with the higher/lower function coding.

Acknowledgement:
none.

TASK 3:
This task takes in the number of guessing attempts, together with the range, and determine whether it is efficient.

So the basic algorithm is to compare the number of attempts against the number with the most efficient way. If it is lower than or equal to the target number, print the result and return true for following tests.

Flowchart is as follows:



The tricky part is how to find the most efficient way. It is relatively easy for higher/lower guess, as the program will tell you whether your guess is too high or too low. So each time when we guess the middle number of the current range, we can easily eliminate half of the range. Therefore, the maximum attempts to guess a number in this way is log base 2 of the amount of total numbers, rounding up to nearest integer.

However, there is also corner case, which is that the amount of total numbers are the power of 2. Take 2 numbers as an example, log2 of 2 is 1, but you may have to guess twice to get the number. So, we have to plus 1 when the amount of total numbers are the power of 2. Hence, the efficient attempts could be calculate as log(total numbers, base2)// 1 + 1.

As for the hotter/colder guess, it is much more difficult to find a most efficient way, because when you are told getting hotter, you still not sure about which direction to go,

lager or smaller. For example, the number is 20, and your last guess is 30, and your current guess is 15. Obviously you are getting hotter. But next time, you still will confuse whether the answer is larger than 15, or smaller than 15. Hence, even in the most efficient way, there are still some attempts to help determine which direction to go, instead of eliminate half of the range.

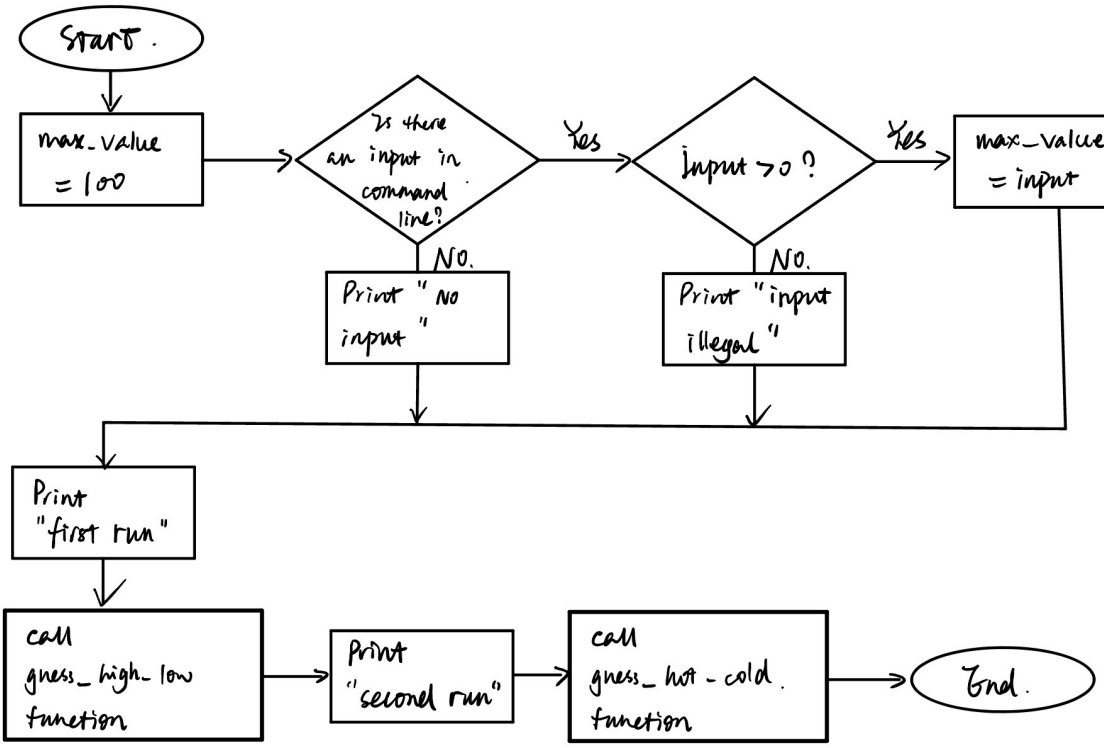
Therefore, when we still using binary search, almost half of the guesses are trying to determine the direction. So we could double the result of higher/lower guess to get the efficient attempts as 2 * log(total numbers, base2)//1.

Acknowledgement:
Guess a number knowing higher or lower:
https://stackoverflow.com/questions/27749722/fastest-way-to-guess-a-number-only-knowing-if-it-is-higher-or-lower-than-the-pre
https://www.learnhowtoprogram.com/computer-science/algorithms/binary-search-algorithm
Guess a number knowing closer or farther:
https://www.ocf.berkeley.edu/~wwu/cgi-bin/yabb/YaBB.cgi?board=riddles_cs;action=display;num=1316188034

TASK 4:
This task require to design a main function to implement all previous functions, and also realize that the client could input the range in the command line.

The logic is very simple. First, we have to import sys to get the command line inputs. We also need to set a default value in case the client doesn't input anything in the command line. In this program, I set 100 as the default value.

The flowchart is as follows:



To further improving this function, we could check how many inputs there are in the command line. If there is only 1 input, use it as the max value if it is larger than 0. If there are 2 inputs, use the larger one as max value and the smaller one as min value. Then generate a random number between max and min.

Acknowledgement:
None.