Homework Report Week #1

Tran Van Tan Khoi

March 26, 2025

1 Introduction

This week's set of problems focuses on writing recursion functions for finding and counting solutions for various classic problems, such as finding the N-th Fibonacci number and solving the N-Queens Problem. The report goes in detail of how I approach such problems. None of the solutions presented here are revolutionary, but rather an reiteration of the most common and typical way one might go on using to solve the problems.

This report, along with C++ solutions, can be found over at this Github repo.

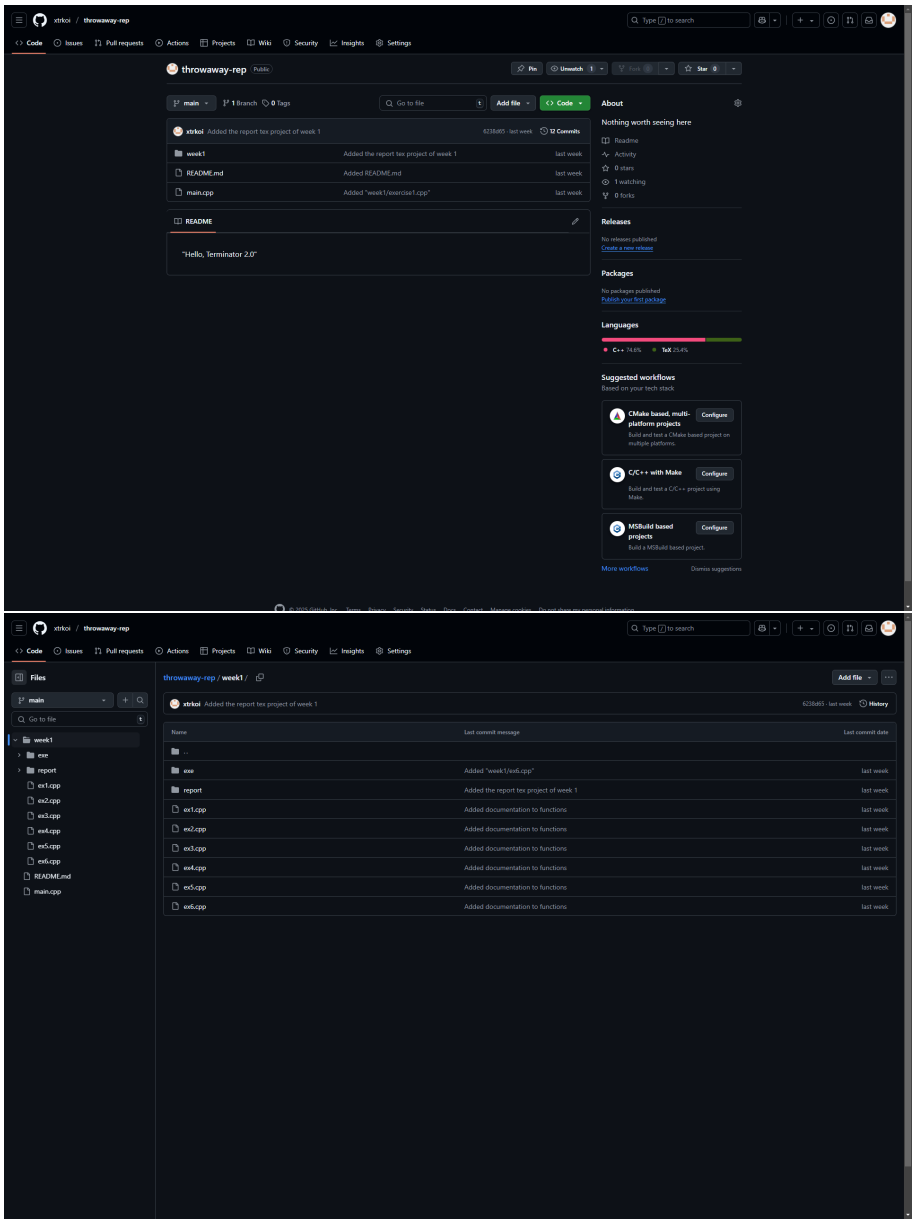

Figure 1: Online Github Repository

2 The Problems

2.1 Fibonacci Sequence

Fibonacci Sequence

The Fibonacci Sequence is a sequence defined by the recursive formula:

$$\begin{cases} F(n) = F(n-2) + F(n-1), & \forall n \geq 2 \\ F(0) = 0, F(1) = 1 \end{cases}$$

Given a positive integer n, find $F(n)$ using recursion.
Input: A positive n ($n \leq 10^6$).
Output: A list of numbers $\{F(i) | i \in [0, n)\}$.

Approach. As a formula is already given, a direct approach would be implementing it with a programming language that support recursion, with *C++* being one of them. However, to print a list of the sequence, we need to keep track of the next number in the sequence to print. Here we leverage *memoization*, a technique in which we store intermediate results, saving time and making it more convenient to output. In this case, simply saving $F(n)$ to an array $f[n]$ will do the trick.

2.2 N-th Factorial

N-th Factorial

Given the factorial of n: $n! = \prod_{x=1}^{n} x = 1 \cdot 2 \cdot \cdots \cdot (n-1) \cdot n$, and a positive integer n, find $n!$ using recursion.
Input: A positive n ($n \leq 20$).
Output: $n!$.

Approach. Define $F(n)$ as the factorial of n, with $F(0) = 1$, we obtain $F(n) = F(n-1) \cdot n$. And the rest is implementation, with a reminder that $20! \approx 2.43 \cdot 10^{18}$ can exceed the maximum value that *int* can hold, so use *long long* instead.

2.3 Generating Binary Strings

Generating Binary Strings

Define a *binary string* as a sequence of digits, each being either 0 or 1. Given a positive integer n, for each i in $[0, 2^n - 1]$, print a binary string which is the binary representation of i.
Input: A positive integer n ($n \leq 20$).
Output: 2^n seperate lines, each line i contains the binary representation of i.

Example

Input:

```
3
```

Output:

```
000
001
010
011
100
101
110
111
```

Approach. This is a fundamental application of *backtracking*. Define $B(x)$ as a binary string of length x. When $x < n$, there are only two options, adding 0 or 1 to the end of the string $B(x)$ to create $B(x+1)$. When $x = n$, simply print the string then return the function.

2.4 Tower of Hanoi

Tower of Hanoi

Three towers reside next to each other. n disks in order of decreasing radius are placed on the leftmost tower, the other two are empty. Each move consists of picking up the top most disk of a tower and move it onto another tower that is either empty or the top most of that tower has bigger radius. Find a list of moves to transport all disks from the left tower to the right tower.

Approach. Backtracking. We name the towers A, B, and C, with the disks starting on tower A. Isolate the problem by only considering two disks on tower A. To move both of them to tower C, move the smaller disk to tower B, the larger disk to tower C, then the smaller disk to tower C. All of this can be implemented by a recursive function.

2.5 Sorted Array Check

Sorted Array Check

Given an array of n integers ($n > 0$), check if the array is sorted in ascending order.

Approach. Define the function $f(i)$ to returns *true* if the array is sorted from index 1 to index i. Then $f(n)$ is *true* if and only if $f(n-1)$ is true and $a_{n-1} \leq a_n$.

2.6 N-Queens Problem

N-Queens Problem

Count the number of ways n queens can be placed on a $n \times n$ chessboard without any two queens attacking each other. Two queens are considered attacking if both of them are on the same row, column or diagonal.

Example

Input:

```
8
```

Output:

```
92
```

Approach. Backtracking. Place the queens row by row. This way, each newly placed queen don't attack any queens placed before on the same row. To check for attacking columns and diagonals, use three arrays.

3 Conclusion

This handful collection of problems help improving the logical processing of certain problems where recursion can be applied and implementing recursion using *C++* programming language. The problems range from elementary to slightly mind-bending but nothing too extreme. By having a deep understanding of how the problems arise in other applications, one can find how recursion play many roles in programming paradigms.