CS5001 Project 7 Report

Shang Xiao

March 22, 2022

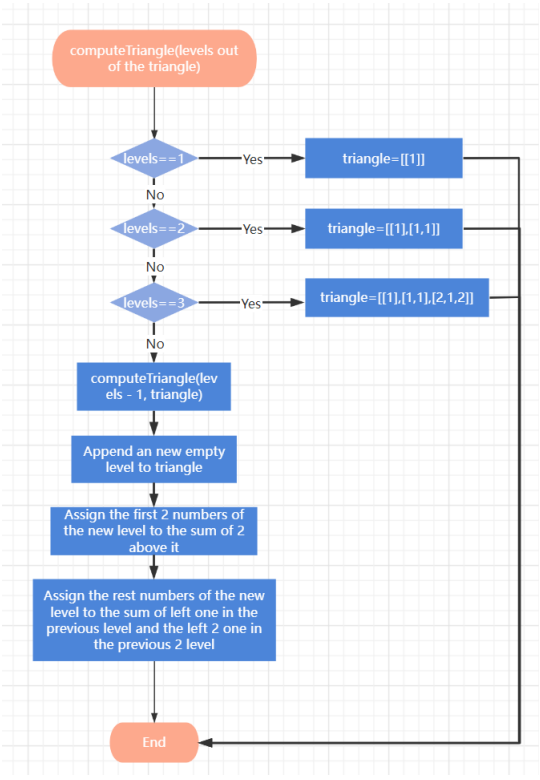
1 Problem Description

1.1 Also known as the "Fibonacci triangle," the Hosoya triangle is a set of arrangements of Fibonacci numbers based on the shape of a triangle. In this project, we are required to program a left-aligned version of this triangle. We start by asking the user to input the triangle level they want to see. Then, the program should print out the corresponding level of the triangle, followed by the rule of having the Fibonacci sequence on both the vertical side and the diagonal. An example of this type of the Hosoya triangle is shown in Excel below:

A	B	C	D	E	F	G	H	I	J	K
1										
1	1									
2	1	2								
3	2	2	3							
5	3	4	3	5						
8	5	6	6	5	8					
13	8	10	9	10	8	13				
21	13	16	15	15	16	13	21			
34	21	26	24	25	24	26	21	34		
55	34	42	39	40	40	39	42	34	55	

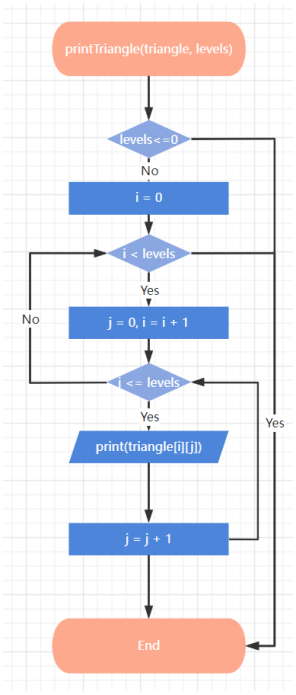
2 Activity Diagrams

2.1 computeTriangle(levels) takes the number of levels to be printed from the user and then computes the levels of the triangle as a list of lists. In this case, the first three rows of the triangle is set with particular arrangements as base cases, then we use the 'append()' method to present the logic of each row below them.



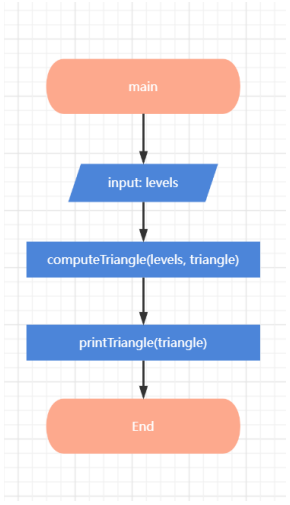
Flowchart for computeTriangle()

2.2 printTriangle(triangle, levels) takes two parameters. Each new level of the triangle will print out 'i+1' numbers given the previous level of 'i' numbers, and for each number in every level, separate them with space, ' '.



Flowchart for printTriangle()

2.3 The main() function prompt the user to enter the level of triangle to be printed and print the triangle.



Flowchart for main()

2.4 Some of the most helpful information which contributes a lot to my planning documents are the coding activities completed during the module 6 lecture, where we completed the codes for all four types of triangles using 'for loop' and the completed code for both 'recur_fact(x)' and 'fibonacci(x)' functions during lecture 7.

3 Reflection

3.1 Recursion is one of the more challenging programming topics with the fundamental idea of calling the function itself with a function. This repetition thinking is extremely useful when tackling complex math problems such as sequences and classical number games. However, we also need to note that recursion has its drawbacks as we finish the quiz for this module, which is stack growth. When dealing with large numbers of repetitions, recursive algorithms might take much longer to return output, which can be better solved by loops.

This project allows me to combine what I had written before in the 'for loop' module, where we successfully printed both the left- and right-aligned triangle with the recursive algorithms introduced in this module.

During the practice of writing the code this time, I was able to implement the 'append()' method with complete understanding and the 'assert' statement first introduced by TA during the test function for the previous simulation project.

4 Acknowledgements

4.1 Website consulted (lecture notes and documentations can be found in these links):

`https://docs.python.org/3/tutorial/datastructures.html` - Official Documentation for 'append()'

`https://www.programiz.com/python-programming/assert-statement` - The 'assert' statement

`https://northeastern.instructure.com/courses/102943` - Module videos from professor John Park

4.2 Website used for flowchart: `https://www.processon.com/`

Website used for debugging codes and loops for each line execution:
`https://pythontutor.com/visualize.html#mode=edit`