Simulation

Natural Computing Homework

Paul Blasi

April 30, 2015

Contents

List of Figures

List of Tables

List of Algorithms

Document Preparation and Updates

Current Version [1.1.0]

Prepared By:
Paul Blasi

Revision History

Date	***Author***	***Version***	***Comments***
4/29/15	*Paul Blasi*	*1.0.0*	*Wrote down problem set.*
4/29/15	*Paul Blasi*	*1.1.0*	*Finished Chapter 9 Problems*

1

Fractals - Text Chapter 7

1.1 Problem 10

Implement a bracketed OL-system and reproduce all plant-like structures of Figure 7.24 in the text. Change some derivation rules and see what happens. Make your own portfolio with at least ten plants.

The first step to solving this problem was to gather the necessary test input. The parameters from Figure 7.24 in the text were summarized into Figure 1.2. The images were created using the program developed for the problem.

After collecting the necessary data, I saw two distinct parts to this problem. One was to create the L-system strings, and the other was to interpret them using Python's Turtle graphics (as suggested).

1.1.1 L-system generation

L-systems are rather straight forward so I made a simple class to encapsulate them. I tested this class against the example in section 7.4.3 in the text. The results can be found in Figure 1.1.

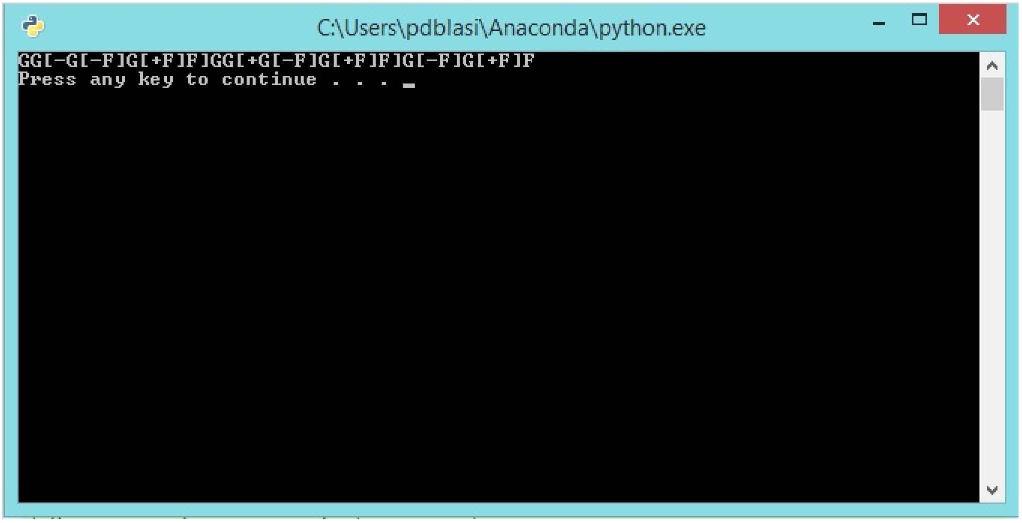


Figure 1.1: LSystem Results

1.1.2 Turtle Graphics representation

IMAGE1	IMAGE2	IMAGE3
$t = 8, \delta = 22.5°$ $\omega = G$ $G \to F + [[G] - G] - F[-FG] + G$ $F \to FF$	$t = 4, \delta = 22.5°$ $\omega = F$ $F \to FF + [+F - F - F] - [-F + F + F]$	$t = 6, \delta = 22.5°$ $\omega = G$ $G \to F[+FFG][G] - FG$ $F \to FF$
IMAGE4	IMAGE5	IMAGE6
$t = 9, \delta = 20°$ $\omega = G$ $G \to F[-G]F[+G] - G$ $F \to FF$	$t = 9, \delta = 25.7°$ $\omega = G$ $G \to F[-G][+G]FG$ $F \to FF$	$t = 5, \delta = 22.5°$ $\omega = G$ $G \to FG[-F[G] - G][G + G][+F[G] +$ $F \to FF$

Figure 1.2: Reproduction of figure 7.24 in the text

1.2 Problem 15

Implement a recursive iterated function system (RIFS) to generate all the fractals whose codes are presented in Table 7.3 in the text.

Again, the first step was to reproduce the data needed for the problem. Table 7.3 from the text has been reproduced in Table 1.1

w	a	b	c	d[1]	e	f	p
1	0.5	0	0	0.5	1	1	0.33
2	0.5	0	0	0.5	1	50	0.33
3	0.5	0	0	0.5	50	50	0.34

Sierpinski Gasket

w	a	b	c	d	e	f	p
1	0.5	0	0	0.5	1	1	0.25
2	0.5	0	0	0.5	50	1	0.25
3	0.5	0	0	0.5	1	50	0.25
4	0.5	0	0	0.5	50	50	0.25

Square

w	a	b	c	d	e	f	p
1	0	0	0	0.16	0	0	0.01
2	0.85	0.04	-0.04	0.85	0	1.6	0.85
3	0.2	-0.26	0.23	0.22	0	1.6	0.07
4	-0.15	0.28	0.26	0.24	0	0.44	0.07

Barnsley Fern

w	a	b	c	d	e	f	p
1	0	0	0	0.5	0	0	0.05
2	0.42	-0.42	0.42	0.42	0	0.2	0.40
3	0.42	0.42	-0.42	0.42	0	0.2	0.40
4	0.1	0	0	0.1	0	0.2	0.15

Tree

Table 1.1: Reproduction of Table 7.3 from the text

1.3 Problem 21

Implement the random midpoint displacement algorithm in 3D and generate some fractal landscapes. Study the influence of H on the landscapes generated.

2

Cellular Automata - Chapter 7

2.1 Problem 1 (from slides)

Modify the heat flow example to deal with insulated conditions on the top and bottom boundary. Insulation means zero flux or u[N][j] = u[N-1][j]. This implies that instead of fixed valued ghost points on the top and bottom, you modify the CA rule using the previous relation.

2.2 Problem 2 (from slides)

Reproduce patterns theta, lambda, mu, and alpha in the Gray-Scott Model CA. You don't need to follow their color scheme.

3

ALife - Text Chapter 8

3.1 Problem 3

Choose one of the sample projects of StarLogo and solve its exploration tasks (`http://education.mit.edu/starlogo/projects.html`). Write a brief report with the results obtained including any theoretical background knowledge that may eventually be necessary to perform the exploration.

3.2 Problem 4

Implement a bi-dimensional CA following the rules of 'The Game of Life'.

4

DNA Computing - Text Chapter 9

Most of this section of the assignment was paraphrased or expanded on from Wikipedia articles.

4.1 Problem 1

Name four problems that cannot be solved by a Turing machine.

Halting Problem

Determining a busy beaver[1] champion

The Mortality Problem

Determining whether a given machine computes a partial function with a nontrivial property of partial functions.

4.2 Problem 2

Name four NP-complete and four NP-hard problems.

NP-complete problems	**NP-hard problems**
SAT problem	Subset Sum Problem
Hamiltonian Path Problem (HPP)	Traveling Salesman Problem
Knapsack Problem	K Minimum-spanning tree
Partition Problem	Graph Coloring Problem

[1] "Busy Beaver" is my favorite program name of all time.

4.3 Problem 5

The two most basic DNA sequencing techniques are known as a) Maxam-Gilbert and b) Sanger, after their proponents. Explain how each of these techniques work and contrast them.

Maxam-Gilbert Sequencing

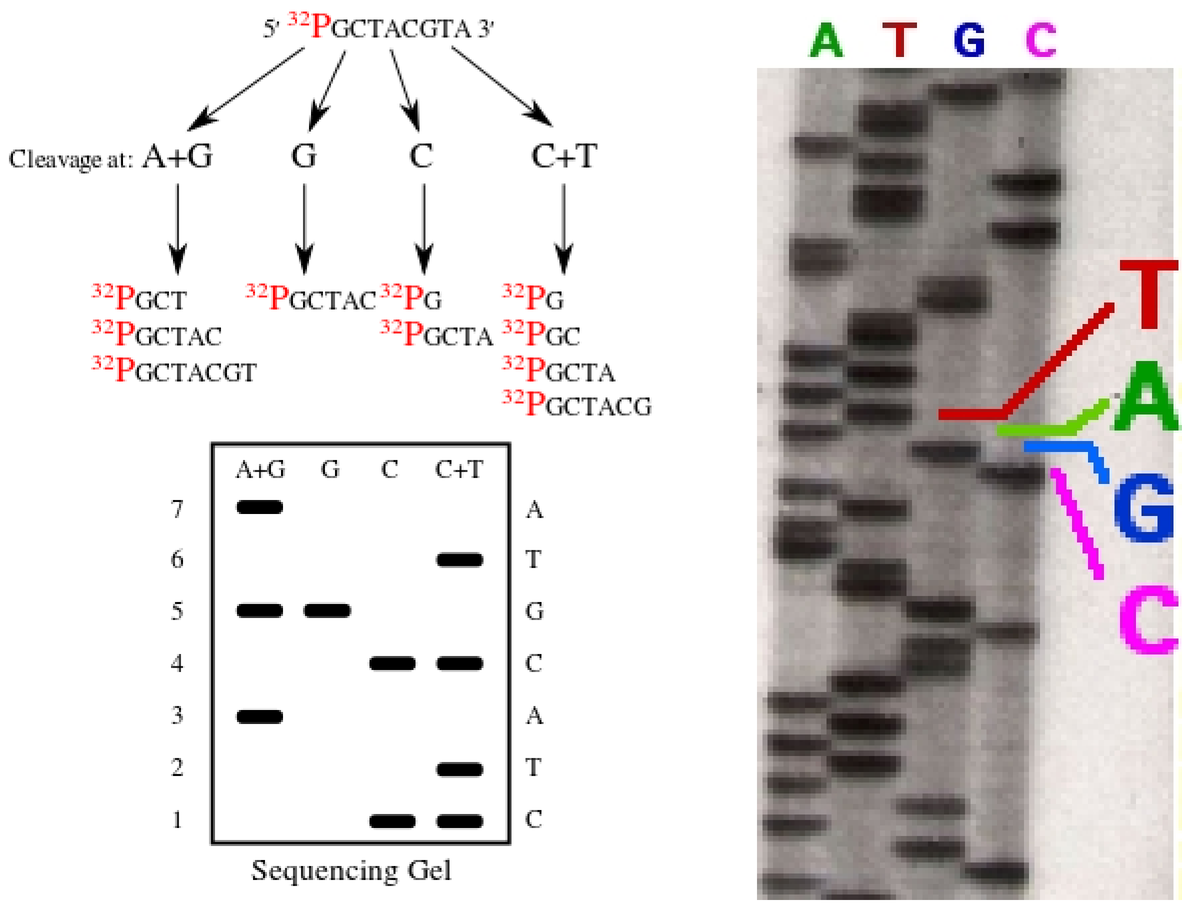
Maxam-Gilbert Sequencing works by cleaving the DNA strands via four different solutions. The solutions are balanced in such a way that each strand will, on average, only be cleaved once. The four solutions cleave at different deoxynucleotides: (A + G), G, C, & (C + T).

After electrophoresing the leftover strands to sort them by size, you are left with a distribution of sizes in each solution. Reading these from shortest to longest, you can infer the deoxynucleotide at that position. Note: in the case of the (A + G) and (C + T) bands, the presence of A and T are infered by that band showing and a lack of the G and C bands respectively. An example graphic of Maxam-Gilbert Sequencing can be seen in Figure 4.1 (a).

Sanger Sequencing

Sanger Sequencing clones a sequence of DNA in four different solutions. Each solution contains 3 of the normal deoxynucleotides that make up DNA chains. The fourth deoxynucleotide is replaced with a corresponding *di*-deoxynucleotide which inhibits chaining due to it's lack of a 3'-OH group used to form phosphodiester bonds. The di-deoxynucleotides can be labeled through various methods including florescence or radioactivity.

Once the DNA is copied, the four strands are heat denatured and separated by length using gel electrophoresis. The length of the strands was limited by the di-deoxynucleotide which means the different lengths of the strands in each solution corresponds to places where the corresponding deoxynucleotide would reside. The sequence can then be read by reading the relative positions in the four lanes. An example of this type of sequencing can be found in Figure 4.1 (b).



(a) (b)

Figure 4.1: DNA Sequencing Methods. (a)Maxam-Gilbert Sequencing (b)Sanger Sequencing

A

Supporting Materials

Supporting ...

B

Code

Listing B.1: LSystem.py

```
class LSystem:
    def __init__(self, productions_dict):
        self.productions_dict = productions_dict;

    def generate_word(self, iters, omega):
        word = omega
        for i in range(iters):
            word = self.rewrite(word)
        return word

    def rewrite(self, word):
        new = ""
        for c in word:
            if c in self.productions_dict:
                new = new + self.productions_dict[c]
            else:
                new = new + c
        return new

if __name__ == '__main__':
    system = LSystem({'F':'G[-F]G[+F]F', 'G':'GG'})
    print system.generate_word(2, 'F')
```

Listing B.2: LSystemTurtle.py

Listing B.3: RIFS.py

Listing B.4: 3d_RMD.py

Listing B.5: HeatFlowCA.py

Listing B.6: GrayScottCA.py

Listing B.7: GameOfLife.py

```
```