Programming Assignment

Automata Theory Monsoon 2024, IIIT Hyderabad

September 1, 2024

Total Points: 100 Points
Deadline: 4 September 2024

General Instructions: Read the input and output formats very carefully since the evaluation will be **scripted**. There will be an evaluation/viva after the submission, and a part of the grade would depend on it.
Language constraints: The submission for question 1 must be in Python. For any further use of other libraries please refer to one of the TAs for approval. Usage of any other language will lead to no evaluation.

---

1 Pseudo Random Number Generator (PRNG)

1. [50 points] **DFA based PRNG**
Consider a pseudo-random number generator (PRNG) implemented using a modified DFA. The DFA has N states, numbered from 1 to N. There is a designated starting state S. The input alphabet consists of a single letter. The DFA outputs its current state ID while transitioning to the next state. There are no accepting states. All transitions are unique, as in a regular DFA. The PRNG generates as many random numbers as the size of the input to the modified DFA. Given such a PRNG, determine the frequency of all numbers output in the range $[1, N]$ if the size of the input to the modified DFA is X. **[CO 1, CO 2, CO 3, CO 4]**

1.1 Input Format

- The first line contains T, the number of test cases.
- For each test case:
  - The first line contains four space-separated integers N, M, S, and X, representing the number of states, number of relevant transitions in the DFA, starting state, and size of input, respectively.
  - The following M lines each contain two space-separated integers a_i and b_i, indicating that the DFA transitions from state a_i to state b_i after outputting a_i upon reading a character. All other transitions in the DFA are considered irrelevant as those states are unreachable.

1.2 Output Format

For each test case, output N space-separated integers where the i-th integer represents the frequency of the number i output by the PRNG.

1.3 Constraints

- $1 \leq N, M \leq 10^5$
- $1 \leq S \leq N$
- $0 \leq X \leq 10^{12}$

It is guaranteed that sum of N and M over all test cases does not exceed 10^5.

1.4 Sample Input

```
2
3 3 1 5
1 2
2 3
3 1
7 6 4 13
1 5
2 6
4 7
5 2
6 1
7 1
```

1.5 Sample Output

```
2 2 1
3 3 0 1 3 2 1
```

1.6 Explanation

For the first test case:

- The input itself to the DFA does not matter since they are all the same due the alphabet being a single letter, only its size which is 5 determines the number of random numbers generated. The DFA has 3 states, 3 relevant transitions and it starts from state 1.
- The sequence of outputs is 1, 2, 3, 1, 2 and hence the frequencies are 2,2,1 for the numbers 1,2,3 respectively.

2 L-Systems

L-systems, or Lindenmayer systems, are a mathematical formalism introduced by biologist Aristid Lindenmayer in 1968. They are used to model the growth processes of plants and other organisms. L-systems consist of an alphabet of symbols, production rules (grammar) that define how these symbols

can be replaced or expanded, and an initial state called the axiom. The system iteratively applies the production rules to generate complex structures, often visualized as fractals or plant-like forms.

Example:
Rule: F → F + F - F - F +F
Iteration 0: F (Axiom)
Iteration 1: F+F-F-F+F
Iteration 2: F+F-F-F+F+F+F-F-F+F-F-F+F+F+F-F-F+F

Symbol	Meaning
`F`	Move forward and draw a line
`+`	Turn right by a specified angle
`-`	Turn left by a specified angle
`[`	Save the current state (position and direction)
`]`	Restore the saved state (position and direction)
`G`	Move forward without drawing a line (gap)
`X`, `Y`	Variables used in rules (placeholders)

Table 1: Symbols and their meanings in L-systems

Look at the links spiecified below and read the documentation for further understanding.

2. [30 points] **Simulating L-systems**
Given the following production rules generate graphics for them using either the Javascript or the Haskell modelling package. Upload an image after the minimum number of iterations mentioned for each sub-part. You may also upload a gif to show the evolution. You may use the following packages:

   - Javascript
   - Haskell

   1. "I'm a mirrorball" (minimum 9 iterations)

      Axiom: G
      Angle: 30
      Rule 1: $G = X - G - X$
      Rule 2: $X = G + Y + G$
      Rule 3: $Y = [+F]F[-F]$

   2. "Is that a tree?" (minimum 4 iterations)

      Axiom: X
      Angle: $(r \bmod 30) - 15 + 10 \times (-1)^{15-(r \bmod 30)}$
      Rule 1: $X = F[-X]X[+X][+X]F - [-X] + X[-X]$
      Rule 2: $F = FF$

      Here, r stands for the last three digits of your roll number. For example, if 2019111234 is your roll number, for you $r = 234$.

3. “Anything that can happen will happen" (minimum 4 iterations)

   Axiom: X
   Angle: 12.5
   Rule 1: $X = F - [[-X] + X] + F[+FX] - X,\ F + [[+X] - X] - F[-FX] + X[X]$
   Rule 2: $F = F[F]F,\ F[+]F,\ F[FF]F$

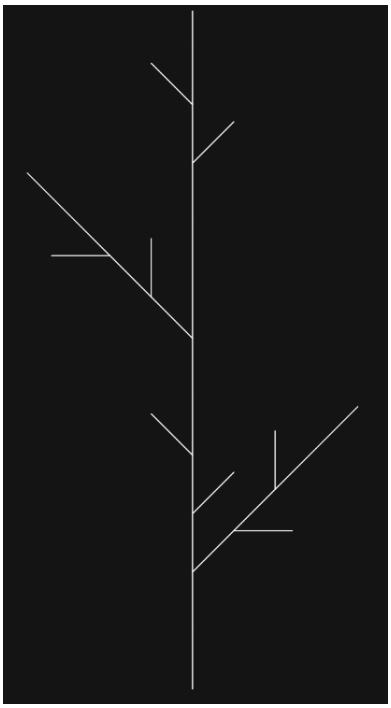
   Each comma separated rule is equally probable. Provide 5 images displaying different L-system configurations obtained from this set of axioms and rules.

4. “Content without context is noise" (minimum 9 iterations)

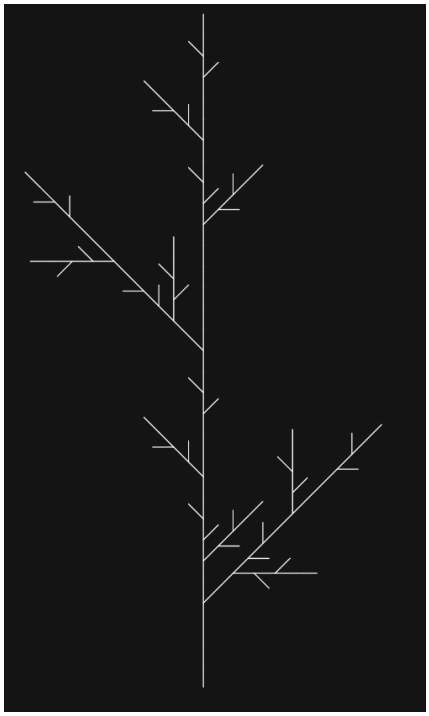
   Axiom: $F + F$
   Angle: 90
   Rule 1: $F = F - F + +F - F$
   Rule 2: $F \langle F \rangle - F = F - +F + F$

3. [20 points] **L-Systems: Find the axioms**

Given the following 2 sets of graphics or final system configurations, come up with the underlying production rules to model that system. **[CO 1, CO 2, CO 3, CO 4]**

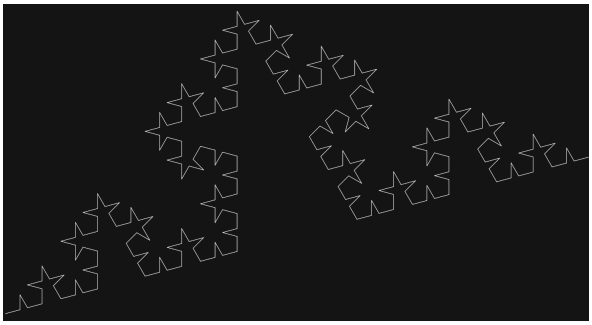


(a) after 2 iterations

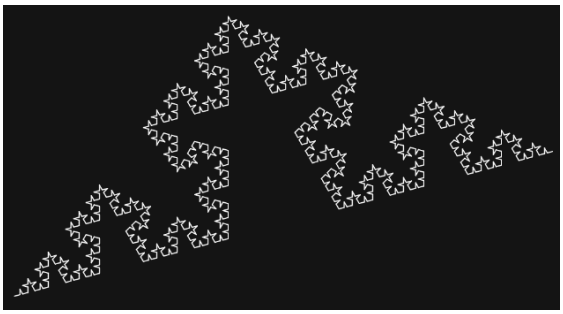


(b) after 3 iterations

Figure 1: Stick Plant



(a) after 4 iterations



(b) after 5 iterations

Figure 2: Santa K(l)osh

For each of the above sub-parts submit a JSON file in the following format. File name for sub-part i: '*i.json*':

```
{
    "axiom": ···,
    "productions": {···},
    "finals": {···}
    ···
}
```

Submission details

The submission must be a zip file containing the code and a report explaining the approach for the solutions.

```
rollnumber_prog1.zip
→ q1
    → ./prng.py
→ q2
    → ./mirrorball.xyz
    → ./tree.xyz
    → ./anything.xyz
    → ./noise.xyz
(use suitable file extensions in place of xyz)
→ q3
    → ./stick_plant.json
    → ./santa.json
→ report.pdf
```