Discrete Mathematics PA2

[Group 12]
SeoYe Kim | HyeonJi Nam | SunKi Pang | HeeJu Shin | SaeAm Choi

1. Fractal 1

Introduction

- Problem Analysis
  Problem : To Draw a fractal cross pattern using a recursion.
  Analysis
  - fractal : a geometric form in which some small pieces are similar to the whole.

  Constraints
  - Draw lines into new places without overlapping existing lines.
  - These lines' number and direction are determined by the previous direction.

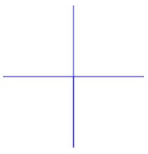
Approach

- Solution design
  - Draw the cross pattern by using recursive function
  - Decide the number according to the direction of the line and Draw three lines according to the number.
- Top-down description solution
  1. Input user values in JavaScript and save them as a Depth variable.
  2. Draw the line in four directions based on the specified point.
  3. Draw the line in three directions relative to the end point of the line drawn in each direction.
     3-1. Check the depth value. if depth value is zero, quit this function.
     3-2. The function call by numbering each direction.
     - 0 (right), 1(up), 2(left), 3(down), 4(center)

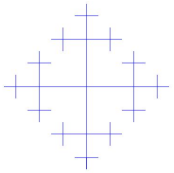
     3-3. Draw a line so that it does not overlap with the existing line.
     ex) 0 (right) - Draw the upward line, downward line, and right line at the reference point.
     3-4. Subtract 1 from the Depth value. Repeat 3-1~4 until the Depth value reaches 0.

Result

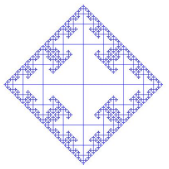
- Demonstration



<Test 1 : depth 1>



<Test 2 : depth 3>



<Test 3 : depth 7>

Discussion

- In order to prevent the lines from overlapping, when calling function drawCross(), we also passed a variable called direction with a value in the direction of the previous line.If direction is zero, we make program not to draw the left line based on (x1, y1).
  If 1, down-side, 2, right-side, 3 is not drawn above.
- The resulting shapes do not increase by more than a certain size. This is similar to the concept of convergence in mathematics.

2. Fractal 2

Introduction

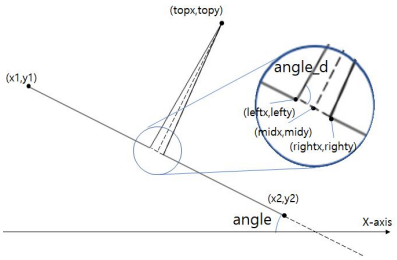
- Problem Analysis
  Problem : To Draw a needle pattern fractal using a recursion.
  Analysis
  - fractal : a geometric form in which some small pieces are similar to the whole.

  Constraints
  - Draw lines into new places without overlapping existing lines.
  - Draw one needle for every line.

Approach

- Solution design
  - Draw the needle pattern by using recursive function.

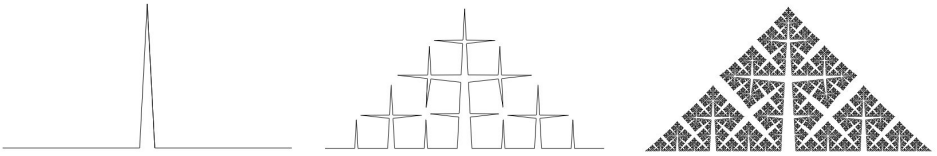
- Top-down description solution
  1. Input user values in JavaScript and save them as a Depth variable.
  2. Call drawNeedle() function for every line each depth.
     2-1. Check the depth value. if depth value is 1, draw one needle for every line.
     2-2. Call drawNeedle() function with new values for every line .
     2-3. Repeat 2-1~3 until the Depth value reaches 1.



drawNeedle(x1, y1, x2, y2, angle, depth, length) - 'length' is the distance of (topx, topy) to (midx, midy)

Result

- Demonstration



<Test 1 : depth 1> *<Test 2 : depth 3>* *<Test 3 : depth 7>*

Discussion

- At first, what we think is drawing needle step by step and erasing center of needle when step is added. However, processing of erasing the center of needle did not work well. So, we change the method. Function drawNeedle() is called continuously to save location to be drawn, and last time, when depth is to be 1, draw one needle for every line at a time.

3. Fractal 3

Introduction

- Problem Analysis
  Problem : To Draw a new creative pattern fractal using a recursion.
  Analysis
  - fractal : a geometric form in which some small pieces are similar to the whole.
  Constraints
  - Create new beautiful pattern to use fractal.

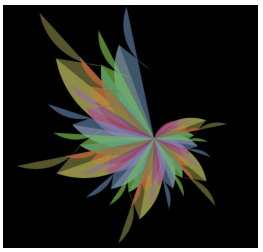
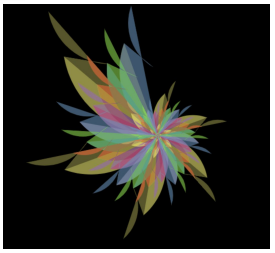
Approach

- Solution design
  - Draw the arcs of various colors counterclockwise from one reference point.
  - Draw a fractal by calling the function recursively while subtracting 1 from the Depth value.
- Top-down description solution
  1. Input user values in JavaScript and save them as a Depth variable.
  2. Call drawWing() function for drawing arcs.
     2-1. Check the depth value. if depth value is 0, quit this function.
     2-2 Call a drawWing() function with a new smaller arcs rotating counterclockwise.
     2-3 Repeat 2-1~3 until the Depth value reaches 0.

Result

- Demonstration
- Draw variety arcs of different sizes so that a special design appears when the fractals are drawn. These arcs have various colors and transparency. When the arcs overlapped, various colors were shown. And each of them

was given light and shade by adding arcs that overlapped half of the arc. The background color set to black color so that the fractal's colors and shapes were vivid. This fractal looks like a wing. So we put Tinkerbells top of the fractals. It looks like Tinkerbell has new wings.


<Test1 : depth 1> *<Test2 : depth 5>* *<Test3 : depth 25>* *<Test4 : TinkerBell>*

Discussion

- When we first try to turn the various arc counterclockwise from one reference point, result did not meet our desire because of different center point of each arc. Thus, we make arc by looking for center of circle including each arc using radius, Reference Point, Angle. To draw arc by reference point even after, we make program to calculate center of arc and draw it.
- We saw clearly the case of using mathematics, especially trigonometric functions, and we felt the need for mathematics in computer science.

4. Triangulation

Introduction

- Problem Analysis

  Problem : Develop a recursive JavaScript program that triangulates simple polygon.

  Analysis
  - Simple polygon : a polygon that does not intersect itself and has no holes.
  - Triangulation : the process of determining the location of a point by forming triangles to it from known points.

  Constraints
  - a simple polygon is given as the list of points in a 100 X 100 coordinate plane
  - each point is denoted as a pair of integers (x,y) for $1 \leq x \leq 100$ and $1 \leq y \leq 100$
  - two adjacent points make a side, and the first and the last points make a side.
  - Draw nothing if the given list does not represent a simple polygon

Approach

- Solution design

  we make points connect with each other using recursion, except for lines which cross others, and lines outside the polygon. After connecting all the points, if the number of lines drawn on the polygon with n sides is not n-3, the line will be drawn again by different starting points with recursion.
- Top-down description solution
  1. check whether a drawn polygon is a simple polygon using the definition of a simple polygon.
  2. Store the given points in the array and draw simple polygon by order of points.
  3. Make all diagonal cases using recursion that can be drawn, and determine them whether to draw or not, under the following conditions.
     (1) If the line crosses the simple polygon, except it.
     (2-1) For the first line, store the points in the new array.
     (2-2) If not, check whether if this line crosses line that connects points stored in the array above. If they cross, except it. If not, save it to the array.
     (3) Repeat step 1&2. If all points are considered, go to step 4.
     (4) Check n-3 lines or 2n-6 points are stored in the array.
     (5-1) If that's right, draw lines.
     (5-2) If not, change the starting point and repeat step 1&2 again.

Result

- Demonstration

Triangulation works well. However, there is a problem. If there is a need to connect two points and there is more than a certain number of points between them, the code will not run. It is assumed that time complexity is a problem.

Manual

It is described in the HTML file.

Discussion

1) We used a method not to connects points that does not meet the conditions out of every number of cases. However, time complexity problem occurred. So we thought there might be a way to thinks possible points from the beginning and implement triangulation. In looking into this, we came across the concept of Delaunay triangulation. This concept was very interesting, although there was a difference in the condition 'empty cycle property' from the PA2-4. The Voronoi diagram which has a 'dual image' relation with Delaunay triangulation, is a concept of dividing the plane, so that the distance between the each points and the dividing line is the same. They were also used to divide national borders and resemble tortoise shell patterns and giraffes in natural. Fractal learned in Problem 1, 2 and 3, was also found in nature. It marvels us that these mathematical concepts could be found in nature.
2) Why can't connect points there be more than a certain number of dots between them? Why time complexity problems occur. We expect time complexity to be O(n^2) because there is recursion in the for statement. But the Java script stopped working and we were frustrated. Why doesn't the code work even though it has a time-complexity of n^2 ? We looked for a reason but couldn't find an answer in time. We used the function min(8,pointP.length-1) to prevent computer being frozen as an alternative. This code is added to prevent errors when considering the case of points more than 4 distance away. If you add this code, it won't be any frozen, but the result won't come out properly.

5. CNF Converter

Introduction

- Problem Analysis

  Problem : The given propositional formula is represented in the tree structure using recursive algorithm. Than we translate the formula to CNF through NNF using recursive algorithm.

  Input, output format
  - Input format : Receive input using stdin. The input is a propositional formula in prefix notation.
  - output format : The program prints out the leaf nodes based on “and” operator. if there is “not” before a leaf node “an”, it prints “-an”.

  Analysis
  - NNF(Negation Normal Form) : Negation operator is only applied to leaf nodes.
  - CNF(Conjunctive Normal Form) : Two level propositional formula finally connected by "and" operator.

  Constraint
  - It is implemented by recursive algorithm using tree structure.
  - If incorrect syntax is entered, output error message. The error message is printed as follows.
    → The parentheses are not balanced
    → There are typo, or leaf nodes are not in the “an” form.

Approach

- Solution design

  Represent a given proposition formula as a tree structure. Next, we apply the De Morgan's laws to make it NNF form. Finally, make it CNF form through distributive law. All processes are implemented through recursive algorithm.
- Top-down description solution
  1. Make initial formula

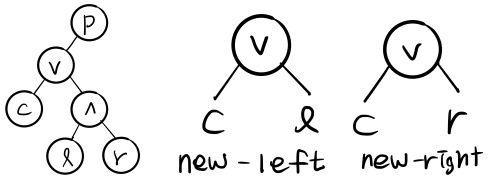
     Cut the given propositional formula, and push it in the binary tree structure. If key of current node is “and” or “or”, insert both left and right nodes recursively. If key of current node is “not”, insert only left node. if key of current node is “an”(for $n \in N$), return the address of the node.

  2. Implementation of NNF

     Applying pre-order traversal, if key of current node is “not”

(2-1) If key of the next node is “and” or “or”, remove current node and change the key of the next node (“and” to “or”, “or” to “and”). Then insert “not” to the left and right of the changed node.
(2-2) if key of the next node is “not”, two nodes are offset. In other words, delete two nodes.

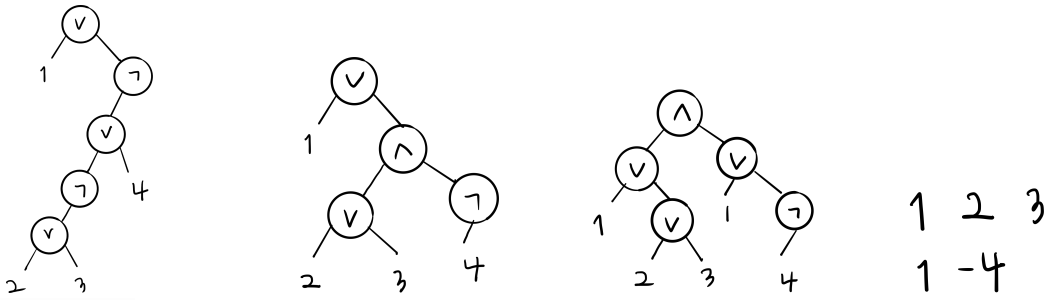
3. Implementation of CNF
   Applying pre-order traversal, if key of current node is “or” and key of the next node is “and”, perform the following.
   First, save the address of the node at pointer “temp”, and link the parent node and the next node. Then make new left and right node that will be attached to “and” node using “temp”. Lastly, link new nodes to “and” node and change the address of node to the address of “and” node. Apply these rules recursively.



4. Print the Result.
   Store the nodes in array, applying pre-order traversal. If there are multiple “and” in succession, store it only once. “or” should not be stored in the array. If there is “not”, multiply -1 to the next key and store it in the array.
   Finally, print out the arrays in order, and if there is an “and”, insert a new-link character.

Result

- Demonstration
  In case (or (a1 (not (or (not (or a2 a3)) a4)))) is given as a propositional formula



The above figure is the formula represented in binary tree, result of NNF application, result of CNF application and output. The tree is constructed as shown above.

Manual

It is described in the README file.

Limitation

- Unable to receive multi literal. For example, the following sentences cannot be received: (or a1 a2 a3 …)
- We can't find some syntax error. For example, (not a1 (or a2 a3 a4)) does not print error messages.

Discussion

detailed analysis of results, interesting observations, lessons learned, suggestions, new ideas, etc.

- If we could implement a multi-node tree, not a binary tree, the solution might be changed. There might not be some process such as translating CNF binary tree to an array to remove duplicated “and”.
- Using recursive algorithm, it was a little complicated, but it was simple to implement with short code
- First we came up with an idea, which is detecting error using stack. However, we realized that it is unnecessarily complicated because there is only one kind of parentheses.
- We tried to use a doubly linked list. It was the method that apply pre-order traversal, store all the values and eliminate unnecessary parts for the output. But this method was too complicated. Thus, we changed the method.