

Discrete Mathematics

Programming Assignment 2
Recursion

Shin Hong

4 Nov, 2019

Assignment Overview

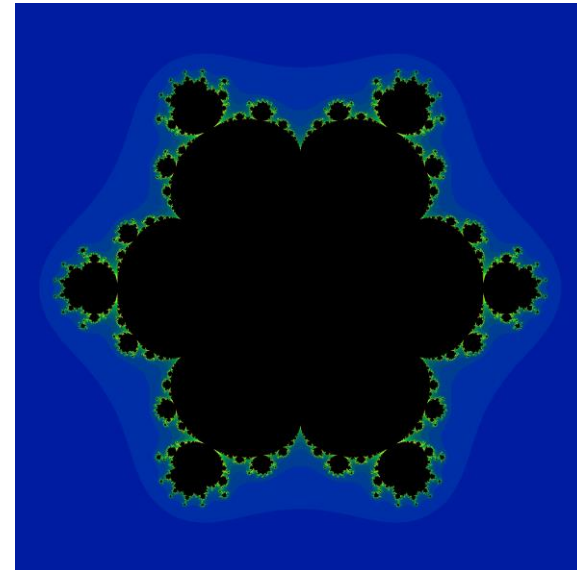
2

- Write a program for each of the following five tasks
 - Tasks 1-3. Fractals
 - Task 4. Triangulation
 - Task 5. CNF converter
- Submission (see more details at Slides XX—XX)
 - deadline: 15 Nov (Fri), 11:59 PM
 - deliverables
 - five programs (one for each task)
 - one write up for the five tasks
- Team: work with your team members to make one submission

Fractal

3

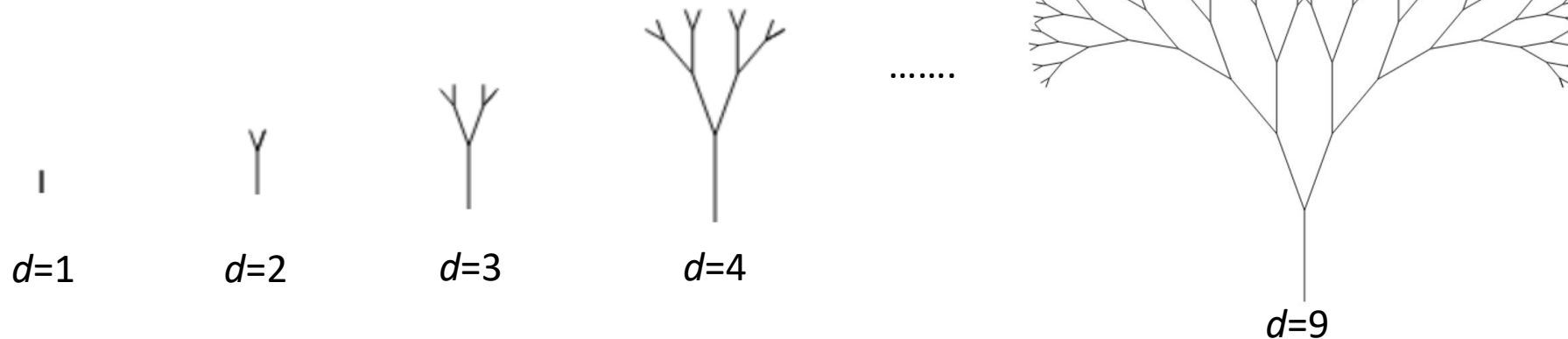
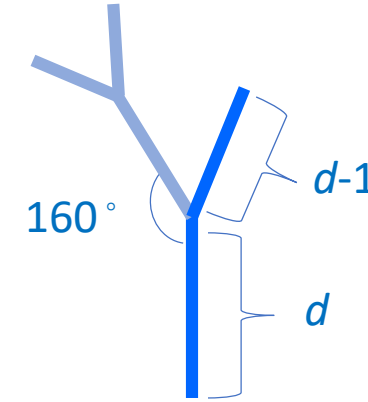
- A *fractal* is a geometric object whose structure is identical to the structures of its components.
 - a structural pattern is repeated recursively
- Many structures in nature are formed as fractals.



Tree Fractal (1/2)

4

- A tree of depth d has one stem and two branches.
 - The length of the stem is d
 - The angles between the stem and the two branches are -160° and $+160^\circ$, respectively.
- Each branch is in a form of a tree of depth $d - 1$.



Tree Fractal (2/2)

5

- Sample code

https://rosettacode.org/wiki/Fractal_tree#JavaScript

```
1 <html> <body>
2   <canvas id="canvas" width="600" height="500"></canvas>
3
4   <script type="text/javascript">
5     var elem = document.getElementById('canvas');
6     var context = elem.getContext('2d');
7     context.fillStyle = '#000';
8     context.lineWidth = 1;
9     var deg_to_rad = Math.PI / 180.0;
10    var depth = 9;
11
12    function drawLine(x1, y1, x2, y2, brightness){
13      context.moveTo(x1, y1);
14      context.lineTo(x2, y2);
15    }
16
17    function drawTree(x1, y1, angle, depth){
18      if (depth !== 0){
19        var x2 = x1 + (Math.cos(angle * deg_to_rad) * depth * 10.0);
20        var y2 = y1 + (Math.sin(angle * deg_to_rad) * depth * 10.0);
21        drawLine(x1, y1, x2, y2, depth);
22        drawTree(x2, y2, angle - 20, depth - 1);
23        drawTree(x2, y2, angle + 20, depth - 1);
24      }
25    }
26
27    context.beginPath();
28    drawTree(300, 500, -90, depth);
29    context.closePath();
30    context.stroke();
31  </script>
32 </body> </html>
```

PA 2.
Recursion

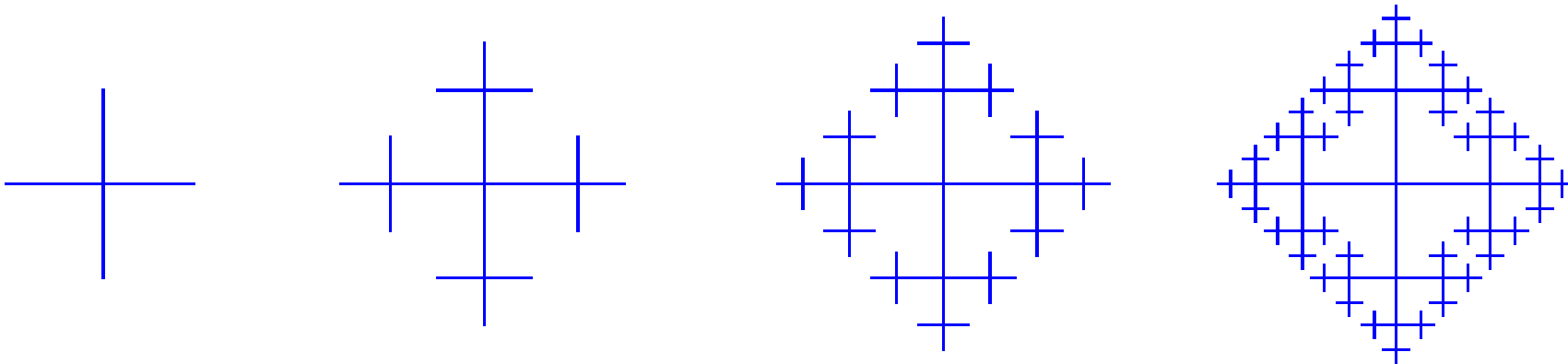
Discrete Math.

2019-11-04

I. Fractal I

6

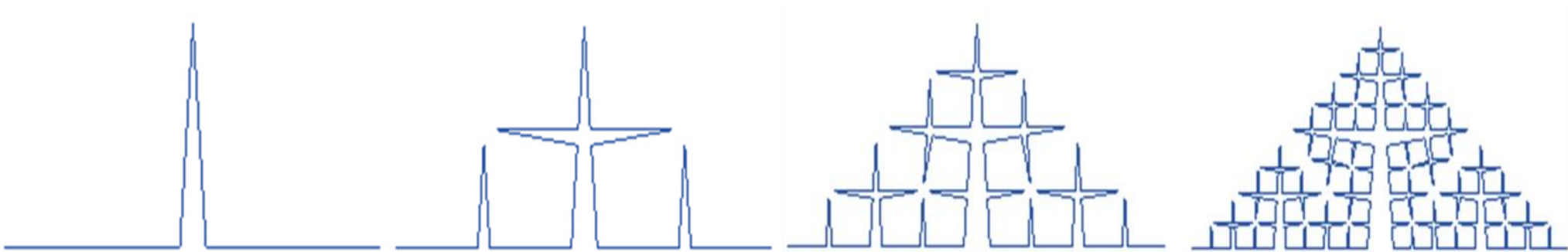
- Define the depth and the parameters of a fractal of the given example, and write a JavaScript program that draws the following fractal for given parameters



2. Fractal 2

7

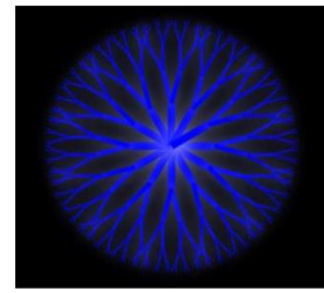
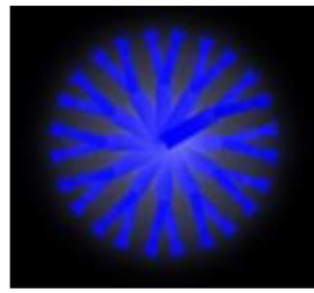
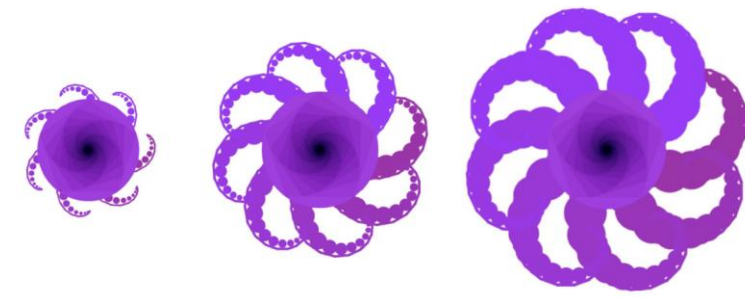
- Define the depth and the parameters of a fractal of the given example, and write a JavaScript program that draws the following fractal for given parameters



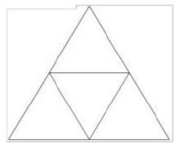
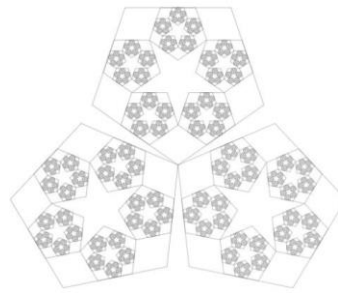
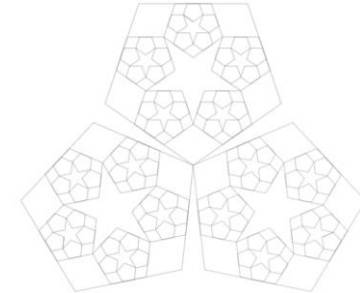
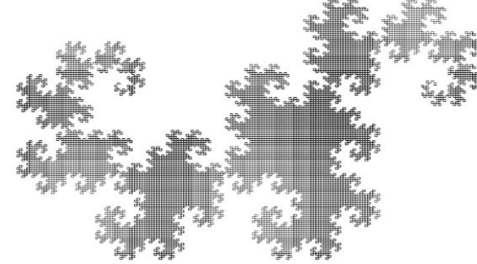
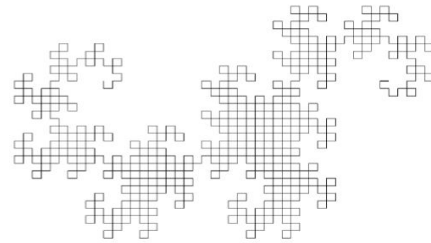
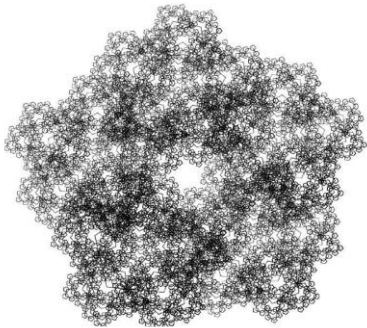
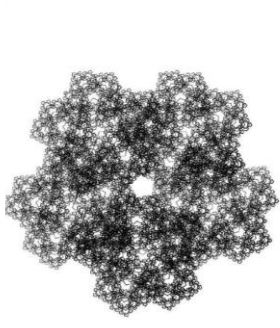
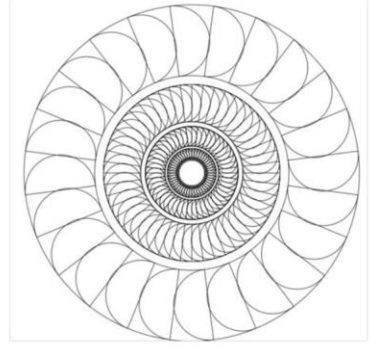
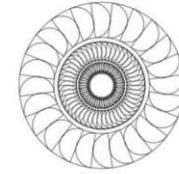
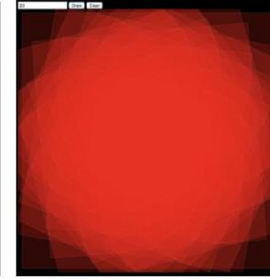
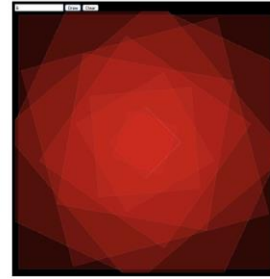
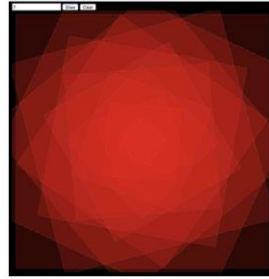
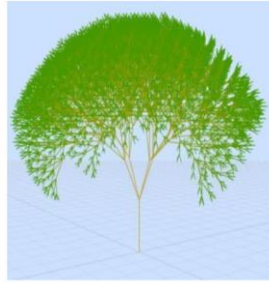
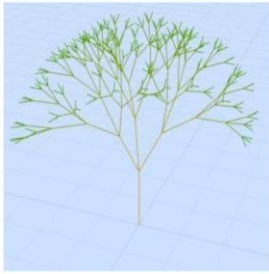
3. Fractal 3

8

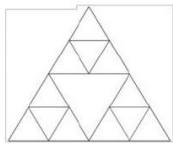
- Suggest an interesting fractal by yourself and write a program to draw the fractal in JavaScript
 - the class will vote for interesting & goodlooking fractals
 - the number of votes for your fractal will be partly counted in evaluation
 - c.f. fractal gallery of the 2018 class
- <https://github.com/hongshin/DiscreteMath/blob/master/assignments/fractal2018.pdf>



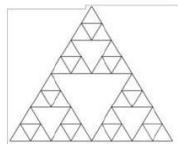
J



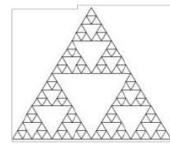
n = 1



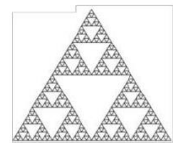
n = 2



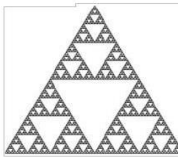
n = 3



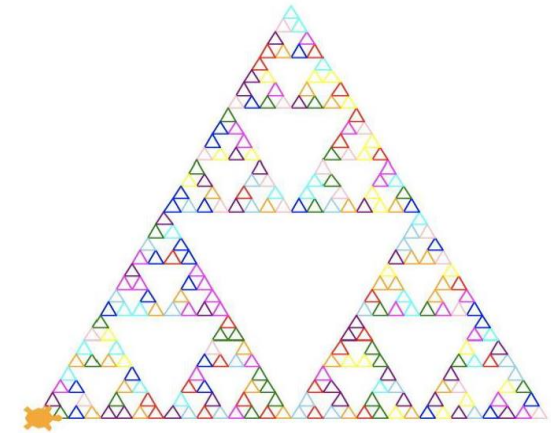
n = 4



n = 5

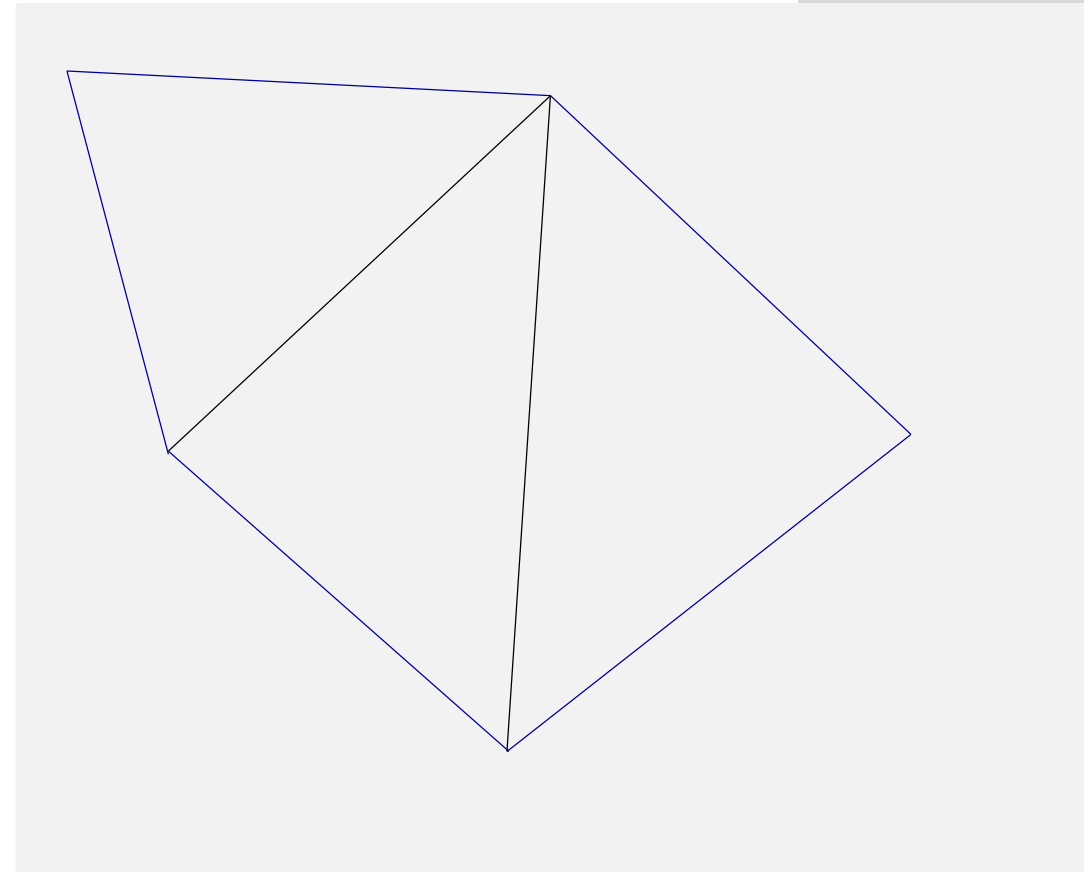


n = 6



4. Triangulation

- Write a recursive program in JavaScript that receives a simple polygon and draws a triangulation result
 - a simple polygon is given as the list of points in a 100x100 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
 - e.g.,
“(1,1) (10,40) (40, 80) (80,40), (40,5)”
 - draw nothing if the given list does not represent a simple polygon



5. CNF Converter (1/3)

11

- Write a C program that receives a propositional formula and transforms it into a Conjunctive Normal Form (CNF)
 - the program uses recursion for the transformation
- Conjunctive Normal Form (CNF)
 - Two-level propositional formulas in a form of
$$(a_{11} \vee a_{12} \vee \cdots \vee a_{1n_1}) \wedge (a_{21} \vee a_{22} \vee \cdots \vee a_{2n_2}) \wedge \dots (a_{m1} \vee a_{m2} \vee \dots a_{mn_m})$$
where a_{ij} is p or $\neg p$ for a atomic propositional variable p
 - every propositional formula has an equivalent CNF form
 - e.g. $p \vee \neg(\neg(q \vee r) \vee s) \leftrightarrow (p \vee \neg s) \wedge (p \vee q \vee r)$

5. CNF Converter (2/3)

12

- A propositional formula $\phi \in P$ is specified by the following rules:
 - **a** $n \in P$ for $n \in \mathbb{N}$
 - **(and** $\phi_1 \phi_2 \dots \phi_m$) for $\phi_i \in P$
 - **(or** $\phi_1 \phi_2 \dots \phi_m$) for $\phi_i \in P$
 - **(not** ϕ) for $\phi \in P$
- A CNF is represented as a list of lines where each line represents a disjunctive clauses of atomic propositional variables and their negations
 - a positive integer n represents a propositional variable **a** n
 - a negative integer $-n$ represents a negation of a propositional variable \neg **a** n
- Example
 - Input: (or a1 (not (or (not (or a2 a3)) a4)))
 - Output
 - 1 -4
 - 1 2 3
- Your program must print out an error message if the given input does not follow the syntax

5. CNF Converter (3/3)

13

1. Define a tree-like data structure to represent a propositional formula
2. Write a recursive algorithm to translate a given text to a formula structure
3. Write a recursive algorithm that converts a given formula to a Negation Normal Form (NNF)
 - a NNF has a negation only at a leaf node (i.e., as $\neg an$)
4. Write a recursive algorithm that applies the De Morgan's law to transform a NNF into a CNF

Program Structure

14

- For Tasks 1 to 4, write each program as a HTML with JavaScript
 - Receive input from TextBox objects
 - Describe how to run the program in the same HTML page
- For Task 5, write a program as a C program running on UNIX/Linux
 - Input and output must be via the standard input and output, respectively
 - You must submit build scripts and README together with source code files
 - Build script: Bash script, Makefile, Ant, Maven, etc.
 - README: instruction/manual on how to build and run your program

PA 2.
Recursion

Discrete Math.

2019-11-04

Submission

15

- **Deadline: 15 Nov (Fri), 11:59 PM**
 - no late submission will be accepted
 - one submission per team
- Each team should submit the five programs and one write-up (report) on the program designs and results
 - Programs: source code files, build script and README
 - Write up: must not exceed 5 pages (single-sided A4)
- Submit all deliverables via Hisnet homework submission repository

PA 2.
Recursion

Discrete Math.

2019-11-04

Evaluation Criteria

16

- Write up (60 points)
 - Description (45 points)
 - check whether you found all constraints of a solution
 - check whether each constraint is correctly represented as a logic formula
 - check whether you demonstrate the correctness of your programs in a convincing way (e.g., by tests)
 - check whether all descriptions are clear and consistent
 - Discussion (15 points)
 - detailed analysis of results, interesting observations, lessons learned, suggestions, new ideas, etc.
- Tests (40 points)
 - Run each program with several inputs to see whether the results are correct

Notes

17

- Right after the deadline, an individual homework HW2 that extends PA 2 will be given
 - You will have 5 days
- Right after the deadline, peer evaluation of your team members will follow
- Your submissions may be open to the class and public
- Help desk by TAs will be offered
 - the schedule is TBA

PA 2.
Recursion

Discrete Math.

2019-11-04