Ch 5. Induction and Recursion

Recursive Definition and Structural Induction - 1

Sungwon Kang

Acknowledgement

- [Rosen 19] Kenneth H. Rosen, for Discrete Mathematics & Its Applications (8th Edition), Lecture slides
- [Hunter 11] David J. Hunter, Essentials of Discrete Mathematics, 2nd Edition, Jones & Bartlett Publishers, 2011, Lecture Slides

Ch 5. Induction and Recursion

Recursive Definitions and Structural Induction

1. Recursive Definition
2. Writing Recursive Definitions
3. Structural Induction

1. Recursive definitions

A recursive definition has two parts:

B. a base case, which defines the simplest object or objects.

R. a recursive case, which defines one or more complicated objects in terms of a simpler one or simpler ones.

☛ almost "circular" but not completely circular

A recursive definition can define

1) a set of objects, called “**a recursively generated set of objects**”, or
2) an operation, function or algorithm, called a **“recursive operation, recursive function and recursive algorithm,”** respectively.

Recursively Defined Set - Example 1

Example

A set E is defined as follows:

Basis clause 0 is a member.

Recursive clause If $n \in E$ then $n+2 \in E$.

$E = \{0, 2, 4, . . . \}$

How about $E = \{0, 1, 2, 3, 4, . . . \}$?

- ☛ Extremal clause is implicit !

Recursively Defined Set - Example 2

- This following defines a recursively generated set.

Given a set of symbols $a_1, a_2, \ldots, a_m$, a *string* of these symbols is:

Two base cases

B_1. the empty string, denoted by λ, or

B_2. any symbol a_i, or

R. xy, the *concatenation* of x and y, where x and y are strings.

Definition (String Concatenation Operation)
Given an alphabet Σ . Let x, y$\in\Sigma^*$.
If x = $a_1a_2.. a_m$ and y = $b_1b_2...b_n$, where $a_i,b_j\in \Sigma$ and m,n$\in\mathbb{N}$,
then the *concatenation of x with y*, denoted xy, is the string
$xy = a_1a_2.. a_mb_1b_2...b_n$.
If x = λ, then xy = y for any y and if y = λ then xy = x for any x.

Recursively Defined Set – Example 3

binary trees - A recursively defined data structure

B₁. The empty tree is a binary tree.

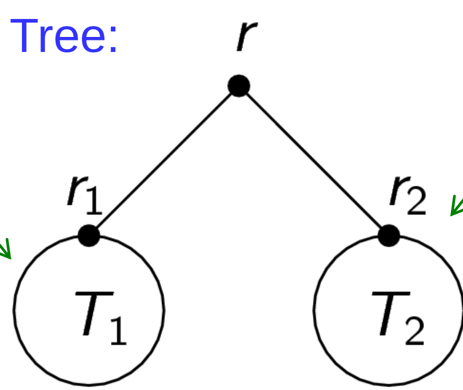
Tree: *empty tree* or *null tree*

B₂. A single vertex is a binary tree. In this case, the vertex is the root of the tree.

Tree: ● r

R. If T_1 and T_2 are binary trees with roots r_1 and r_2 respectively, then the tree

Tree:



left child of r

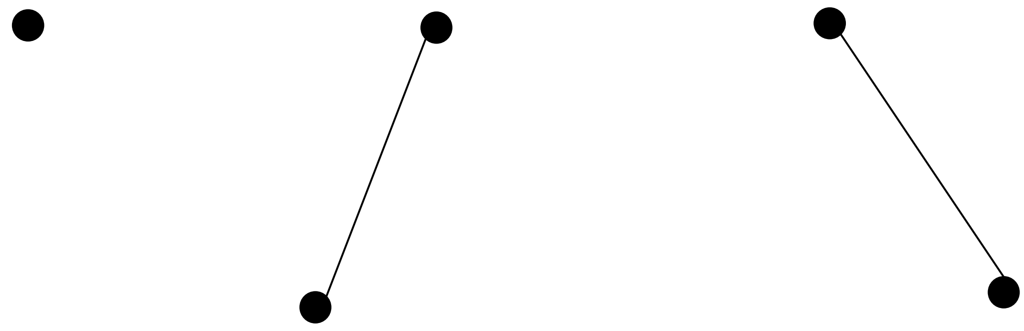
right child of r

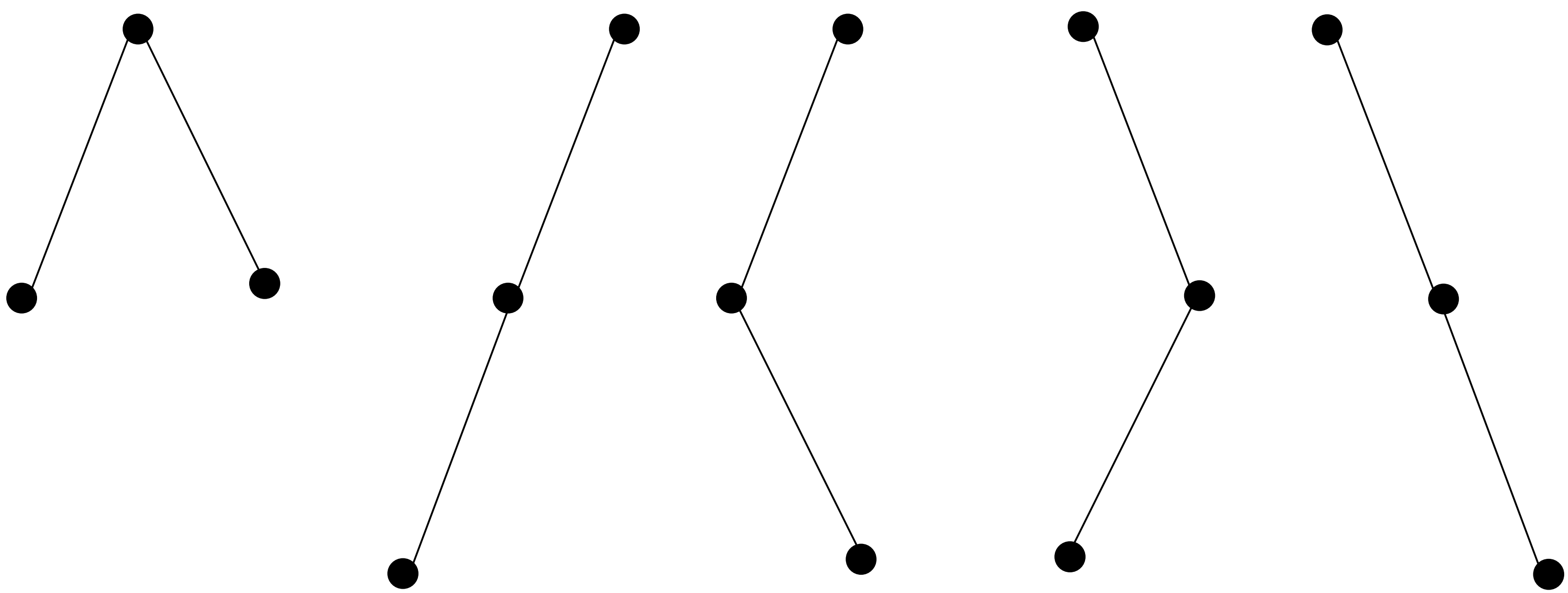
vertex = node
edge = link

is a binary tree with root r. Here the circles represent the binary trees T_1 and T_2. If either of these trees T_i ($i = 1, 2$) is the empty tree, then there is no edge from r to T_i.

All binary trees with #nodes ≤ 3

The empty tree





Recursively Defined Operation

the reverse of a string

If s is a string, define its *reverse* s^R as follows.

B. $\lambda^R = \lambda$

R. If s has one or more symbols, write $s = ta$ where a is a symbol and t is a string (possibly empty). Then
$s^R = (ta)^R = at^R$

This is a recursive operation on a recursively generated set.

[redacted] the reverse of a string

If s is a string, define its *reverse* s^R as follows.

B. $\lambda^R = \lambda$

R. If s has one or more symbols, write $s = ta$ where a is a symbol and t is a string (possibly empty). Then
$s^R = (ta)^R = at^R$

This is a recursive operation on a recursively generated set.

For example,

$$
\begin{aligned}
(\mathtt{pit})^R &= \mathtt{t}(\mathtt{pi})^R && \text{by part } \mathbf{R}\\
&= \mathtt{ti}(\mathtt{p})^R && \text{by part } \mathbf{R}\\
&= \mathtt{ti}(\lambda\mathtt{p})^R && \text{(insertion of empty string)}\\
&= \mathtt{tip}\lambda^R && \text{by part } \mathbf{R}\\
&= \mathtt{tip}\lambda && \text{by part } \mathbf{B}\\
&= \mathtt{tip} && \text{(removal of empty string)}
\end{aligned}
$$

Proving Correctness of Recursively Defined Operation

The string reversal function:

B. $\lambda^R = \lambda$

R. If s has one or more symbols, write $s = ta$, where a is a symbol and t is a string (possibly empty). Then
$s^R = (ta)^R = at^R$

This is a recursive operation on a recursively generated set.

Theorem

The string reversal function works. In other words, for any $n \geq 1$, $(a_1 a_2 \cdots a_{n-1} a_n)^R = a_n a_{n-1} \cdots a_2 a_1$.

2. Writing Recursive Definitions

Using recursive definitions, we can define a set of objects (or a data structure), an operation, a function, an algorithm, and etc.

To write a recursive definition, we need to consider:

- **For the base case**, what are the simplest objects?
- **For the recursive case**, how can I construct more complicated objects from simpler ones?

Example: browsing the Internet

Suppose you start browsing the Internet at some specified page p. Let W be the set of all pages you can reach by following links, starting at p. Give a recursive definition for the set W.

Example: browsing the Internet

Suppose you start browsing the Internet at some specified page p. Let W be the set of all pages you can reach by following links, starting at p. Give a recursive definition for the set W.

Solution

Observe that if you can reach some page x, then you can reach any page to which x has a link. This gives the recursive part of the definition:

The base case is the page where you start:

B. $p \in$ W.

Example: browsing the Internet

Suppose you start browsing the Internet at some specified page p. Let W be the set of all pages you can reach by following links, starting at p. Give a recursive definition for the set W.

Solution

Observe that if you can reach some page x, then you can reach any page to which x has a link. This gives the recursive part of the definition:

R. *If* $x \in$ W *and* y *is some page such that* x *links to* y*, then* $y \in$ W.

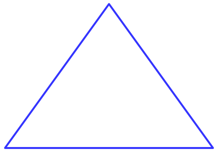
The base case is the page where you start:

B. $p \in$ W.

Koch snowflake: definition

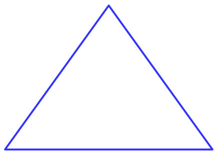
Define a sequence of shapes as follows.

B. $K(1)$ is an equilateral triangle.



Koch snowflake: definition

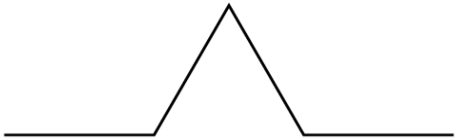
Define a sequence of shapes as follows.



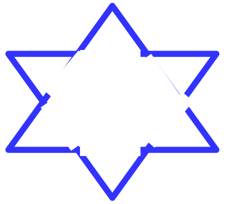
B. $K(1)$ is an equilateral triangle.

R. For $n > 1$, $K(n)$ is formed by replacing each line segment

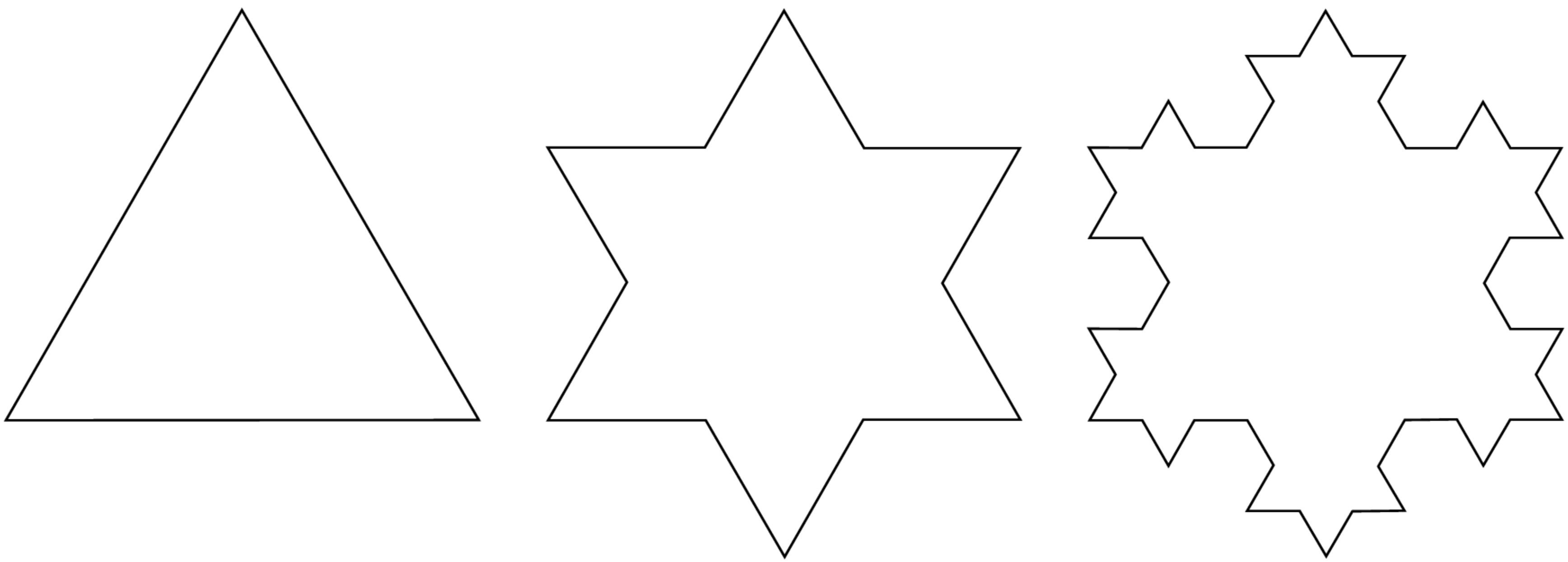
of $K(n-1)$ with the shape



such that the central vertex points outwards.



Koch snowflake: first three terms

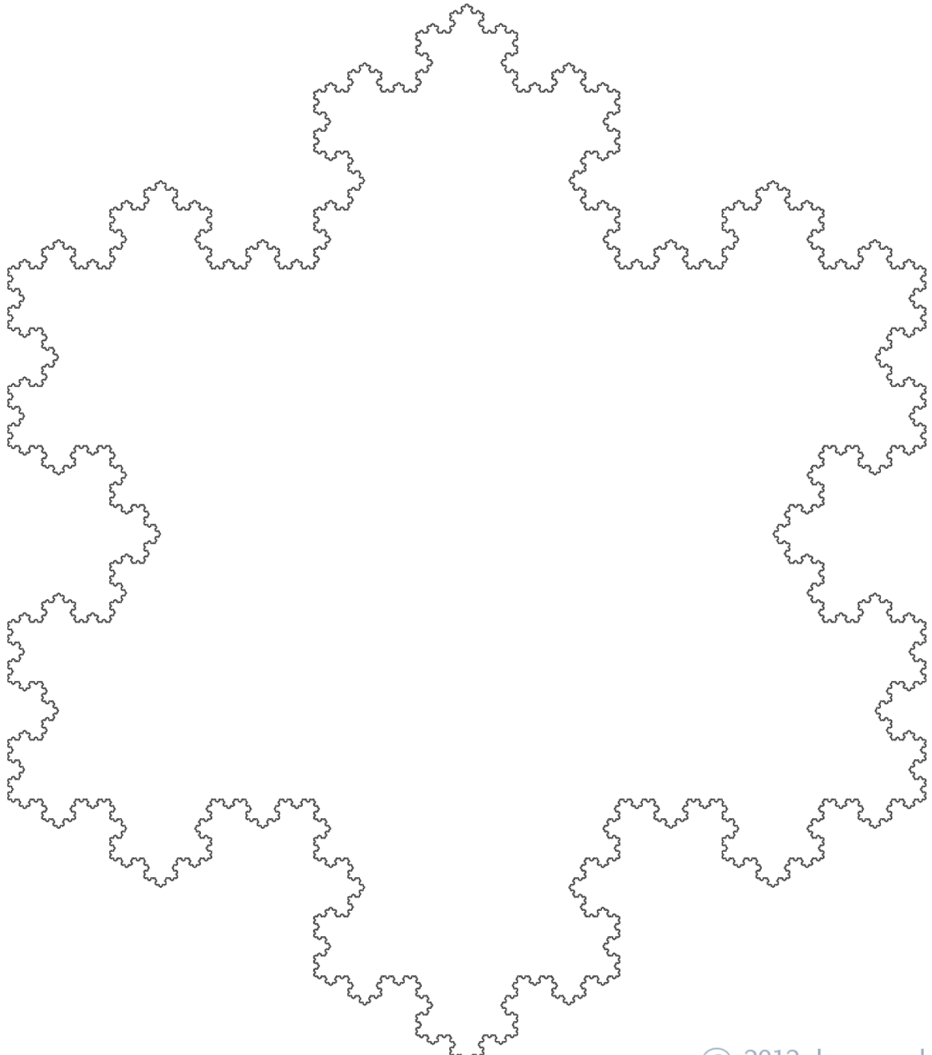


$K(1)$ $K(2)$ $K(3)$

- Recursively generated geometric objects.

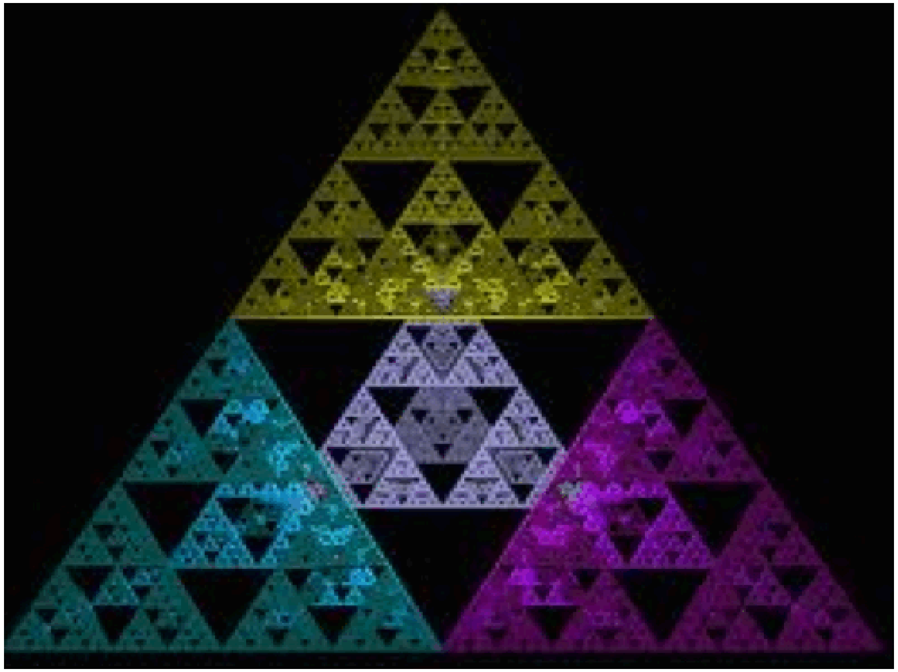
Koch snowflake: limiting fractal

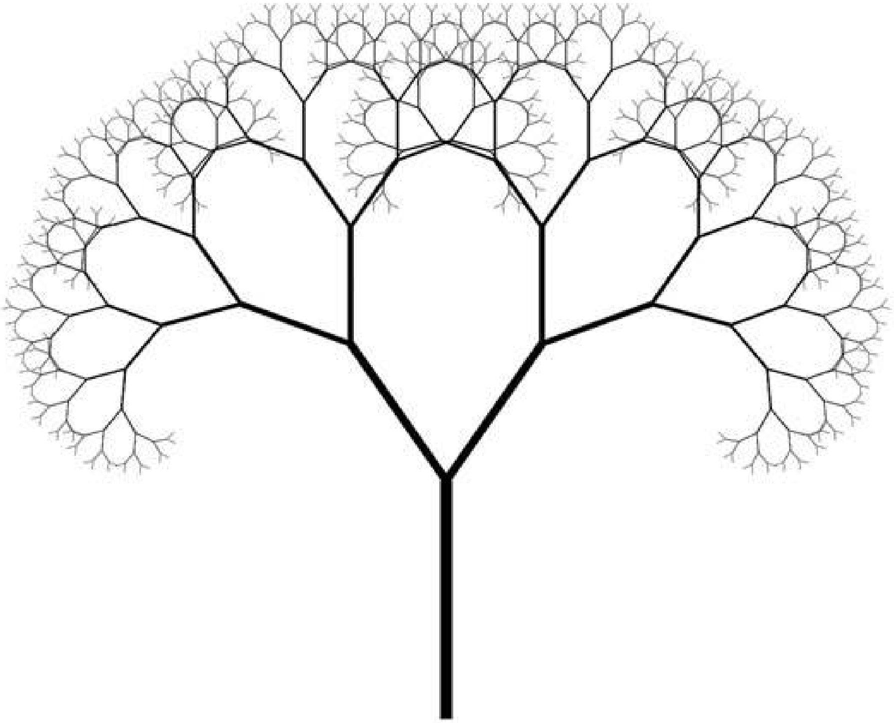
After applying Rule R infinitely many times



A Fractal

More Examples of Fractals





What is the basis case?
What is the recursive case?

Badda-Bing axiomatic system

Recall the following set of axioms.

Undefined terms: badda, bing, hit

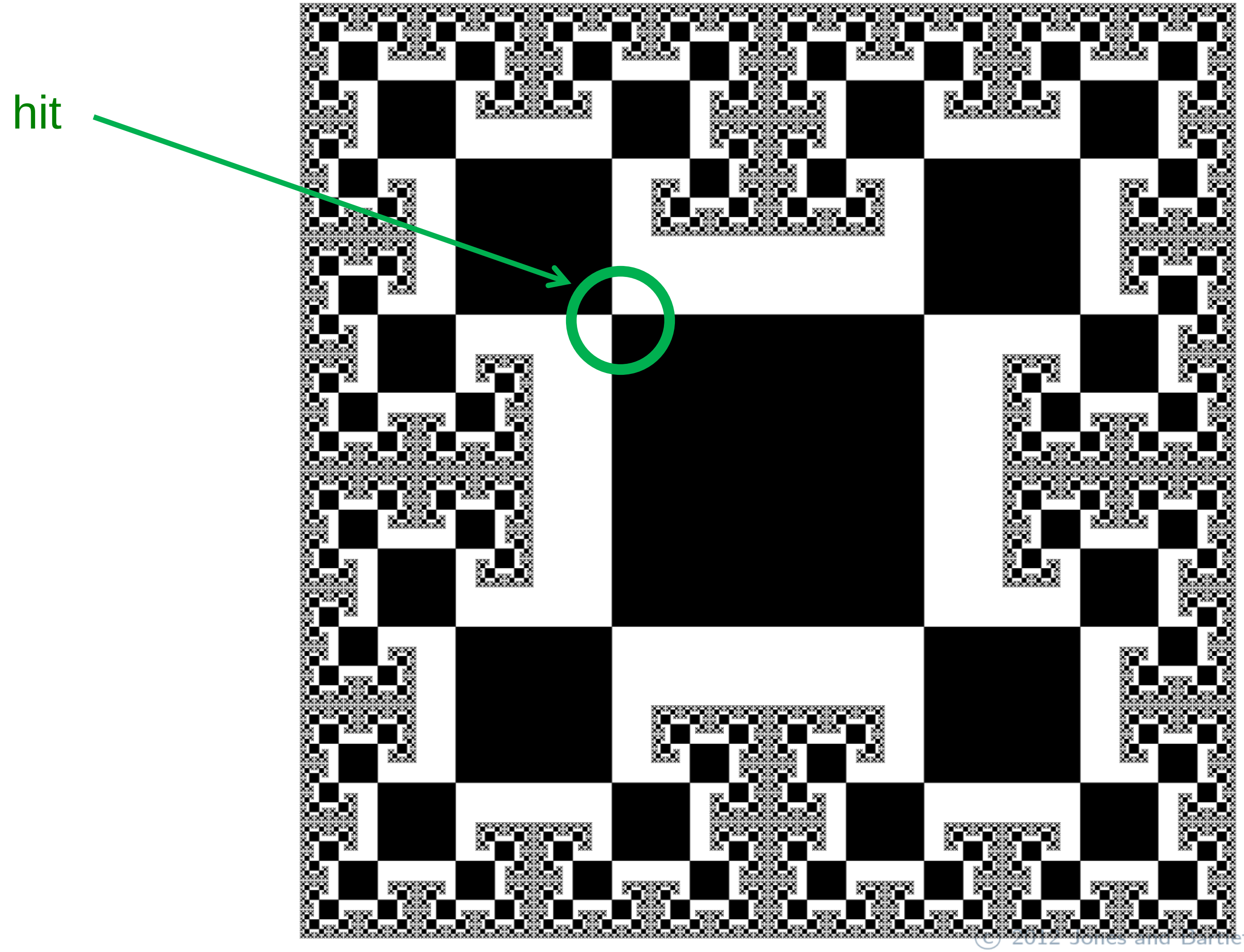
Axioms:

1. Every badda hits exactly four bings.
2. Every bing is hit by exactly two baddas.
3. If x and y are distinct baddas, each hitting bing q, then there are no other bings hit by both x and y.
4. There is at least one bing.

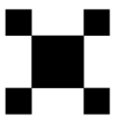
One possible interpretation:

badda: square
bing: corner of a square
hit: a square hits corner if the corner belongs to the square

Model for badda-bing system

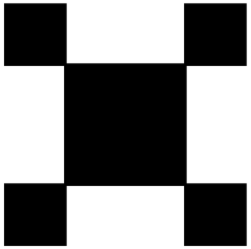


Badda-Bing fractal: definition

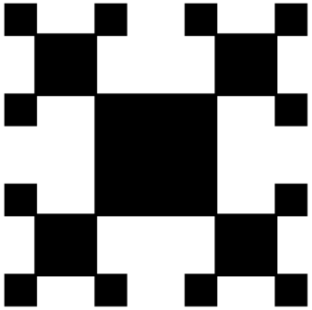
B.  is in S.

R. If  is in S (in any orientation), so are .

Badda-Bing fractal: first three terms



By Rule B

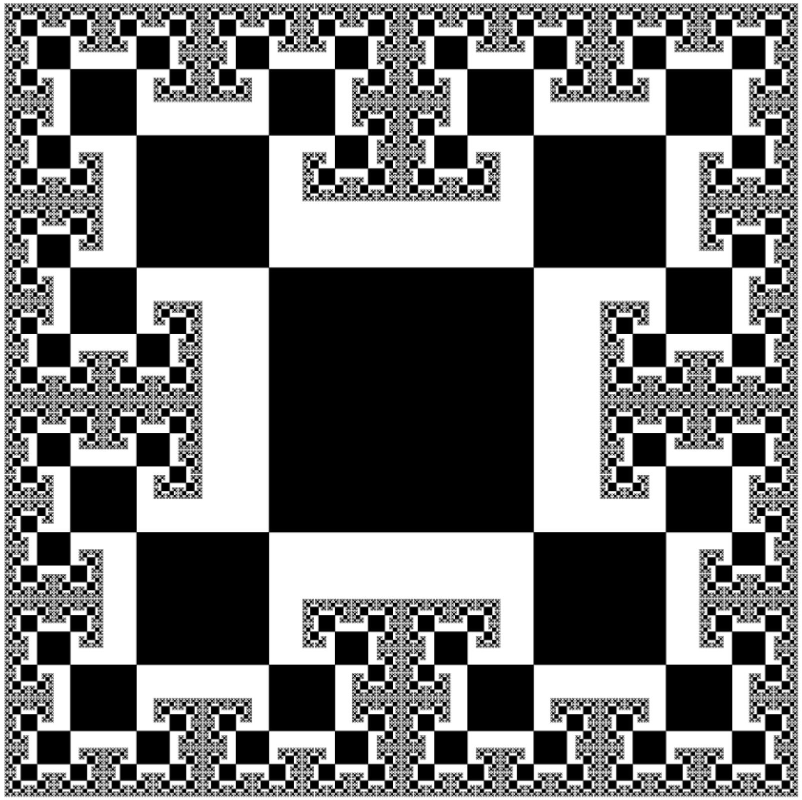


By applying
Rule R once

. . .

By applying
Rule R twice

Badda-Bing fractal: limit



After applying Rule R infinitely many times

Quiz 14-2

Which of the following is NOT true?

(a) Recursive definition can be used to define an infinite set.

(b) Recursive definition can be used to define a finite set.

(c) A recursive definition can have multiple basis clauses.

(d) A recursive definition can have multiple recursive clauses.

(e) Recursive definition may be used to define not only sets but also to prove correctness of functions.

(f) Mathematical induction may be used to prove properties of recursively generated sets and also operations and functions.

(g) It is NOT possible that when a recursive clause is applied to a given set, the resulting set remains the same as the given set.