

# Thinking Recursively

## Part I

# Outline for Today

- ***Self-Similarity***
  - Recursive patterns are everywhere!
- ***Recursive Trees***
  - Elegant structures from simple code.
- ***Information Flow***
  - How to send information around in recursion.

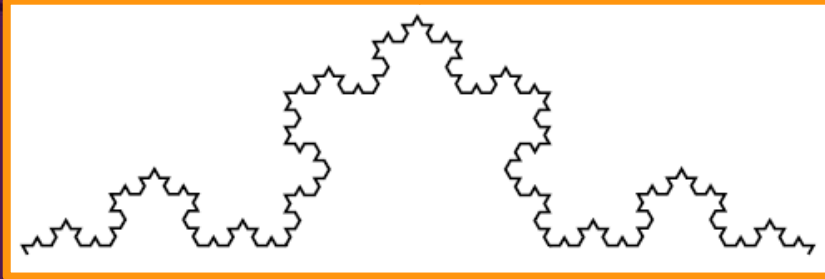
# Self-Similarity



An object is ***self-similar*** if it contains a smaller copy of itself.



Hey, it's that  
thing from  
Assignment 1!

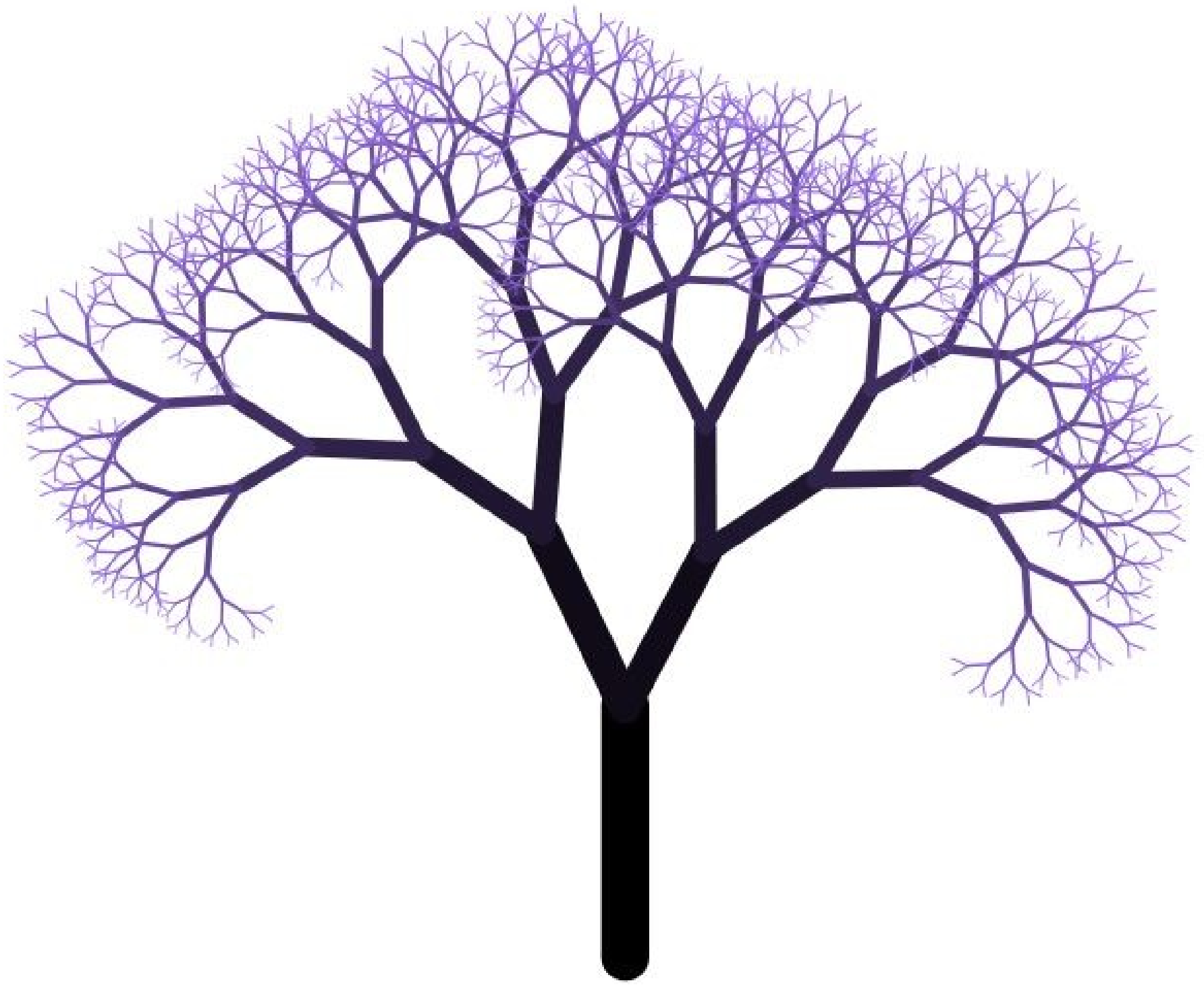


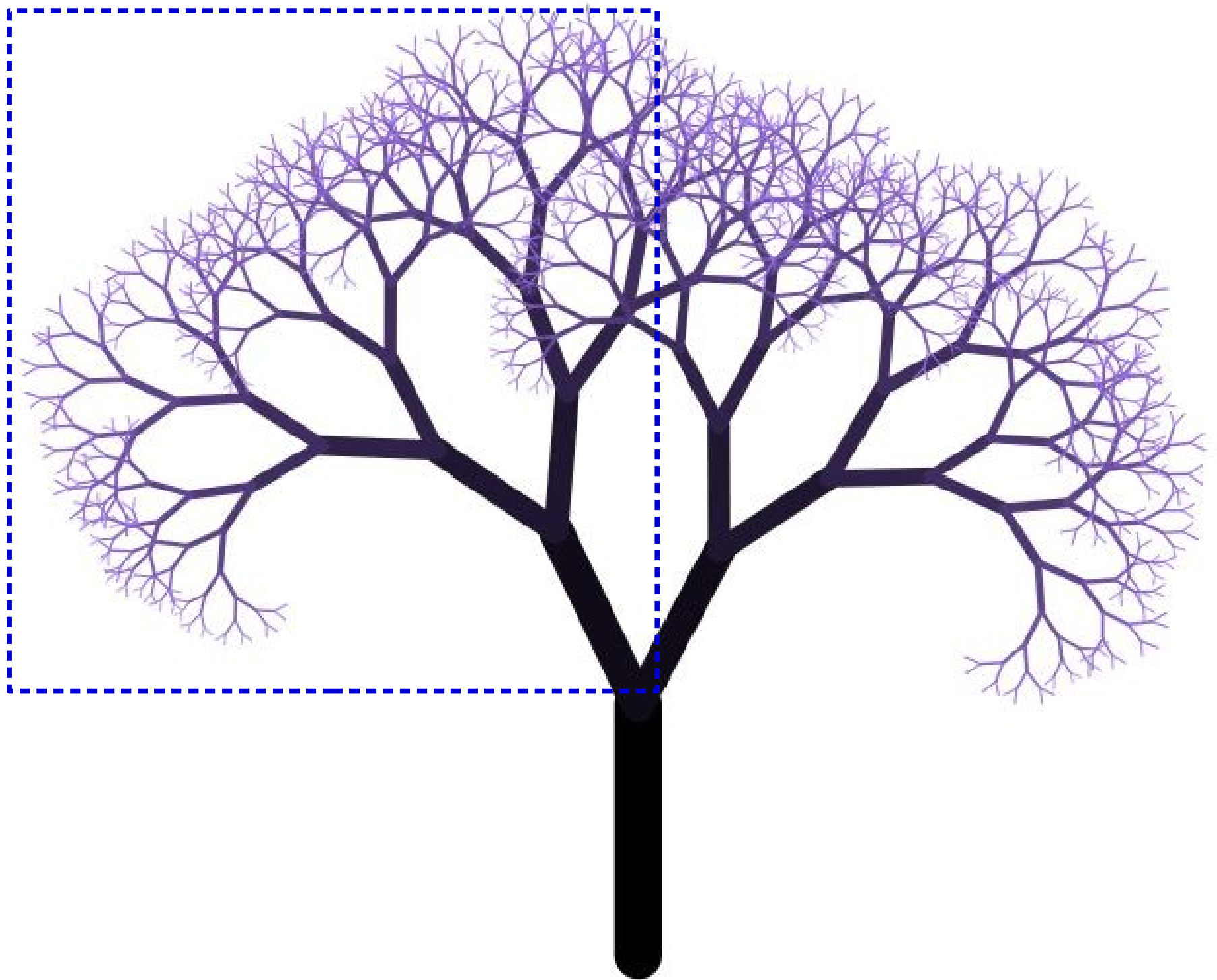
{w}

happy  
holidays  
from  
wics

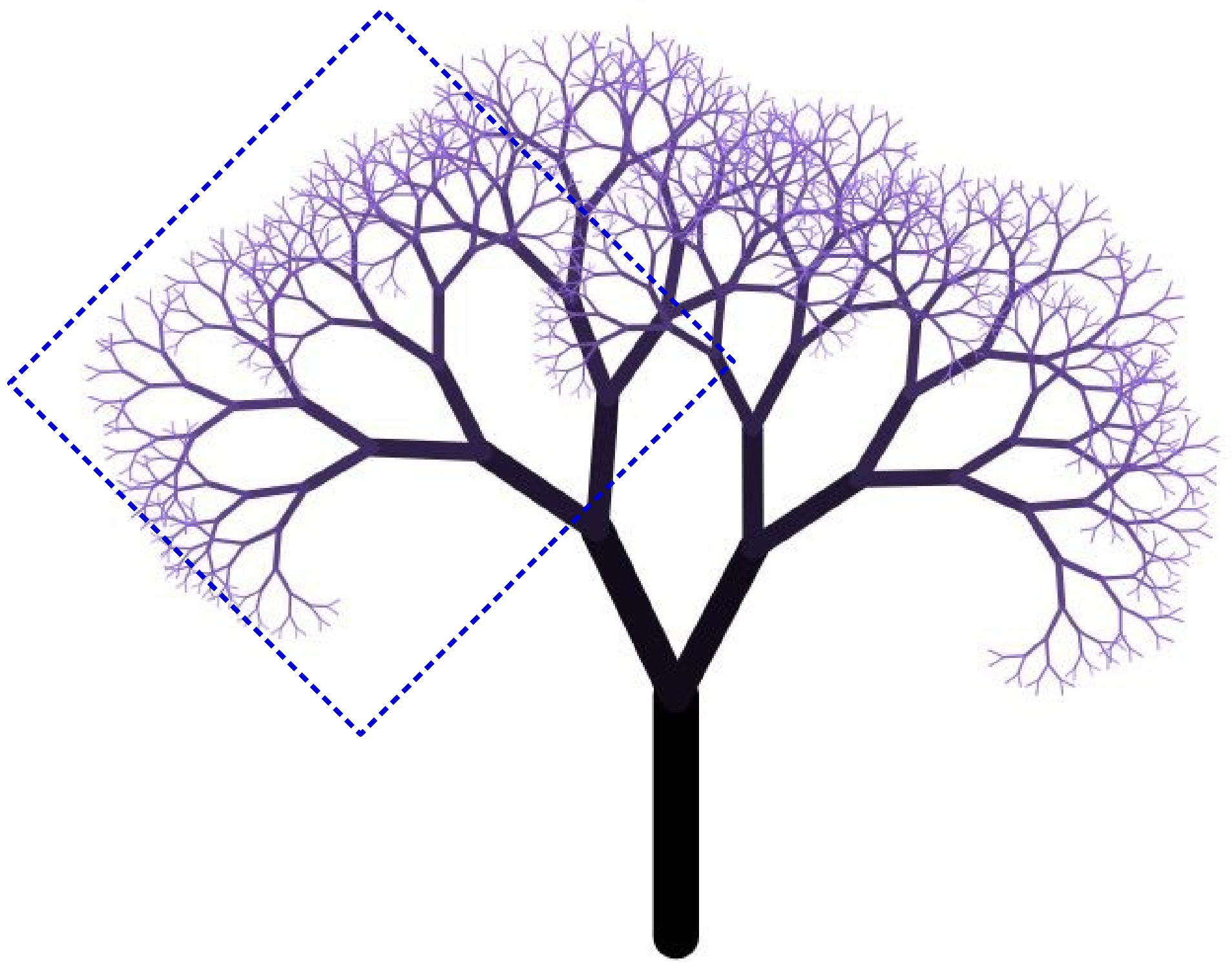
An object is ***self-similar*** if it contains a smaller copy of itself.

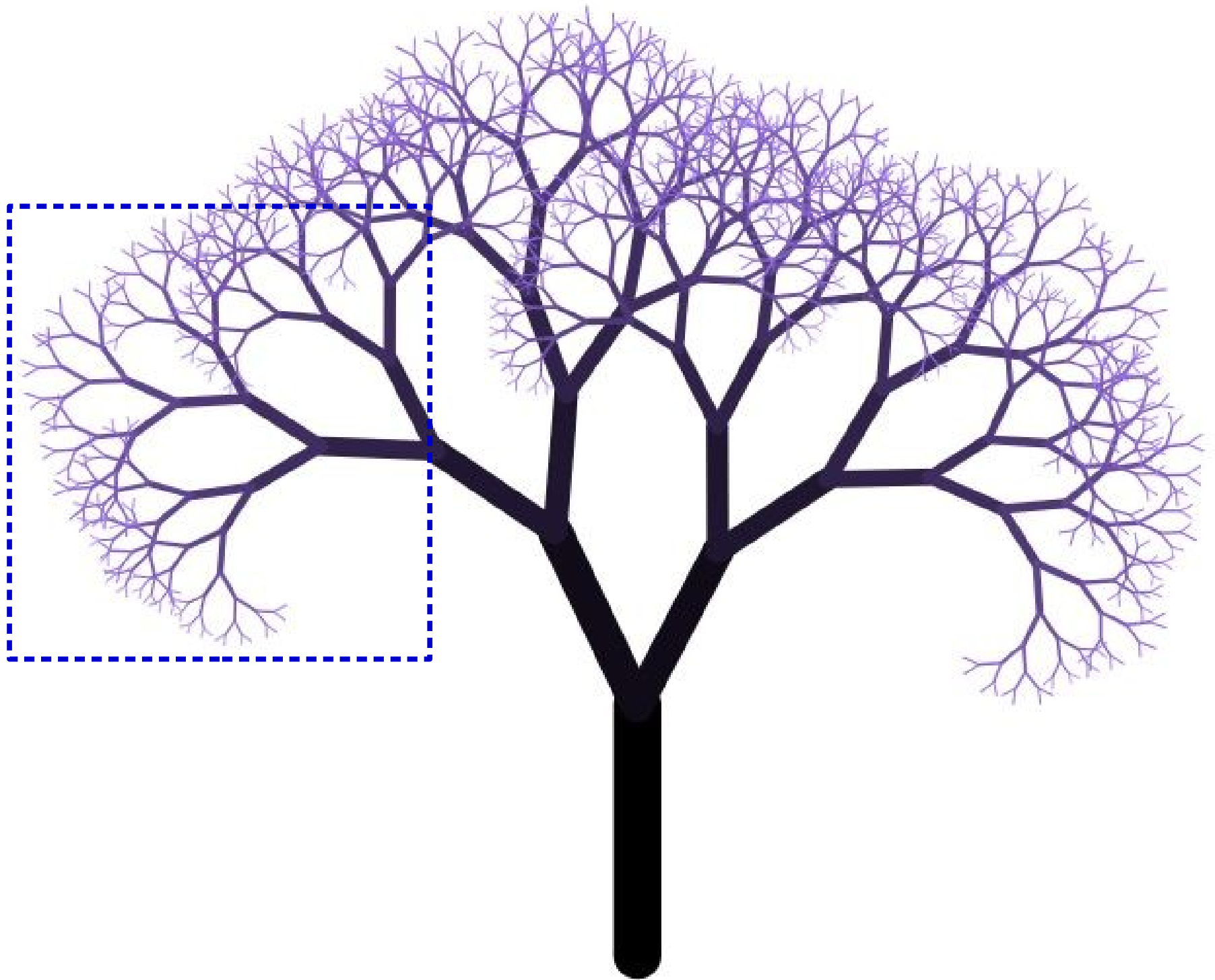
# Drawing Self-Similar Shapes

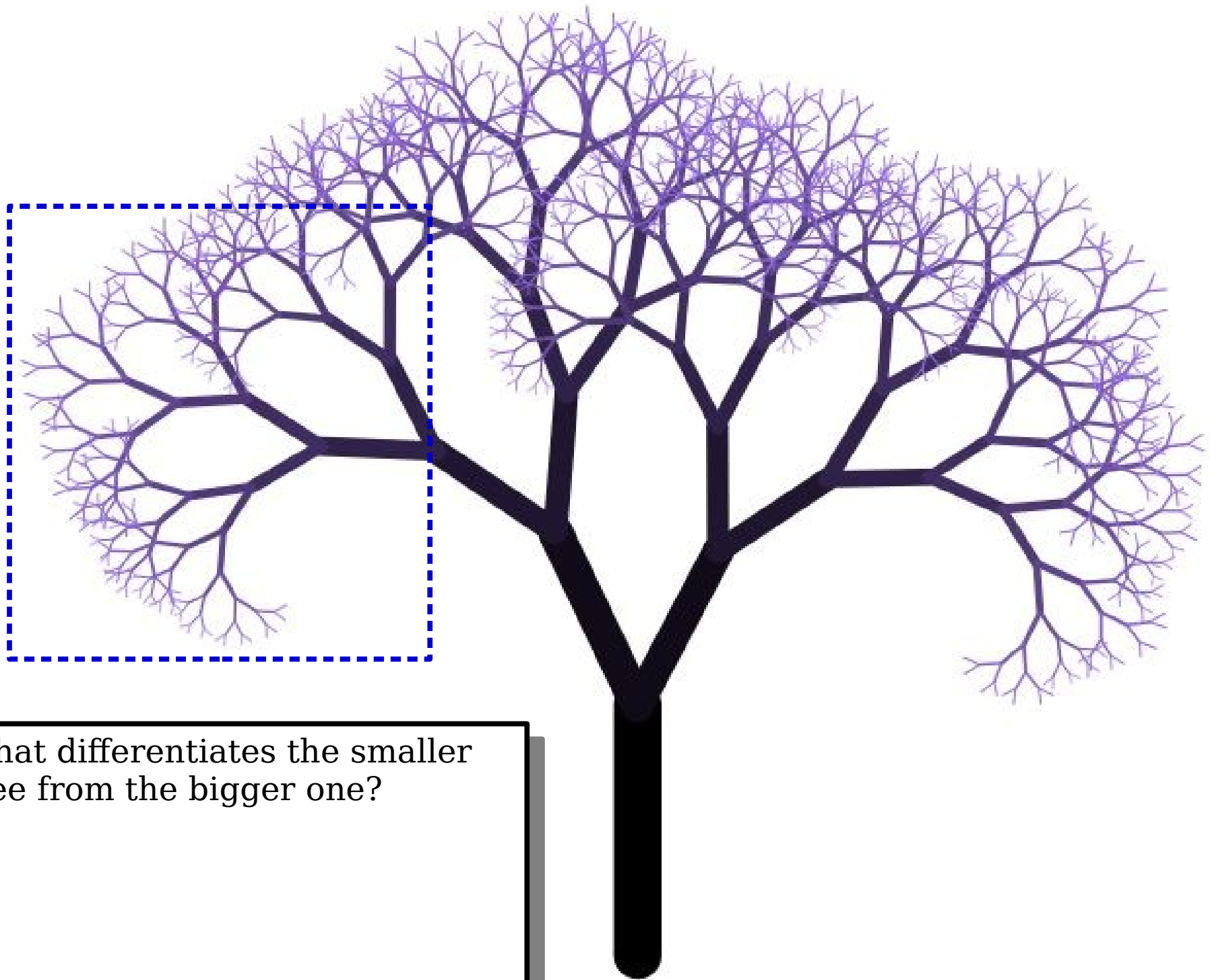




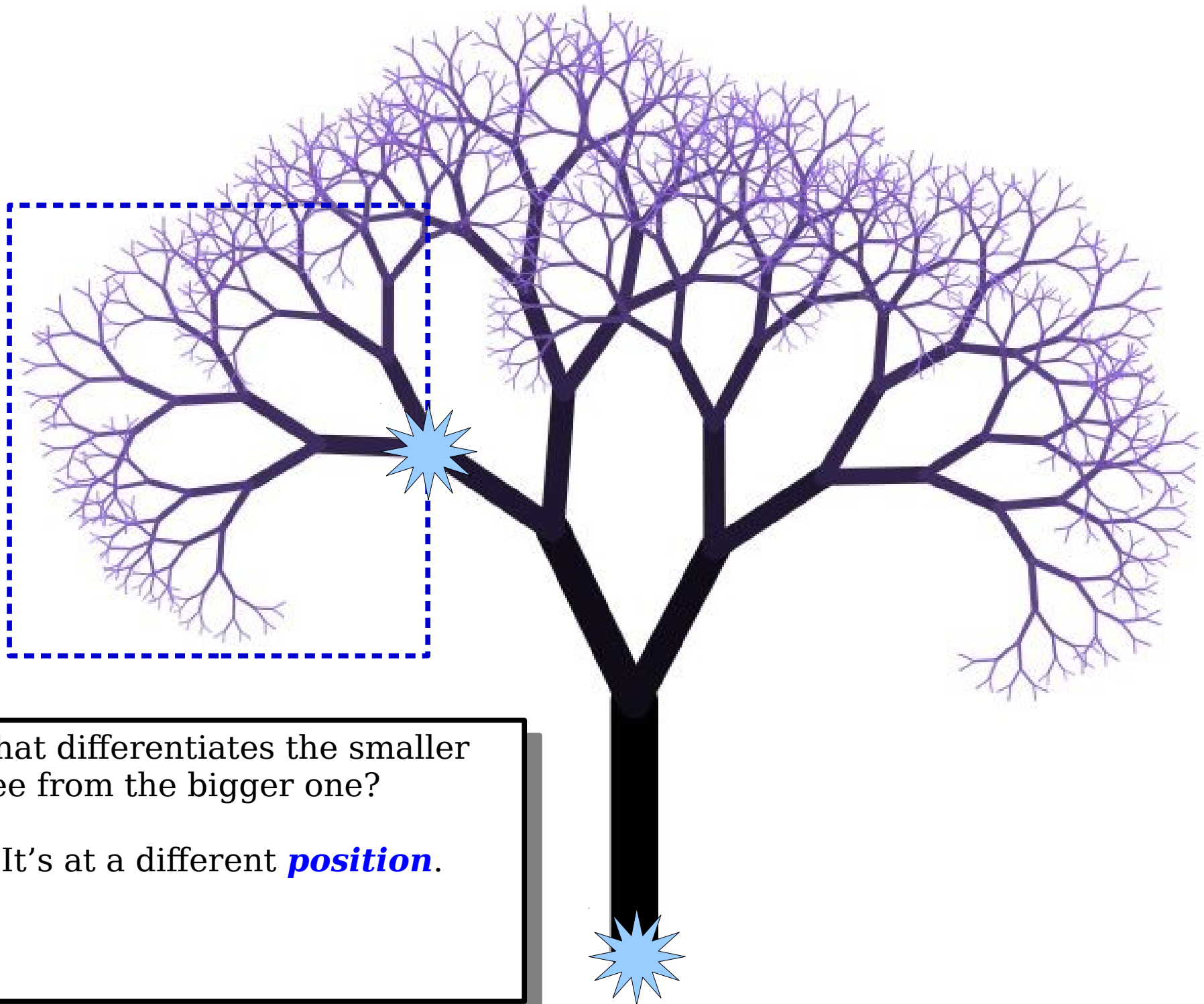






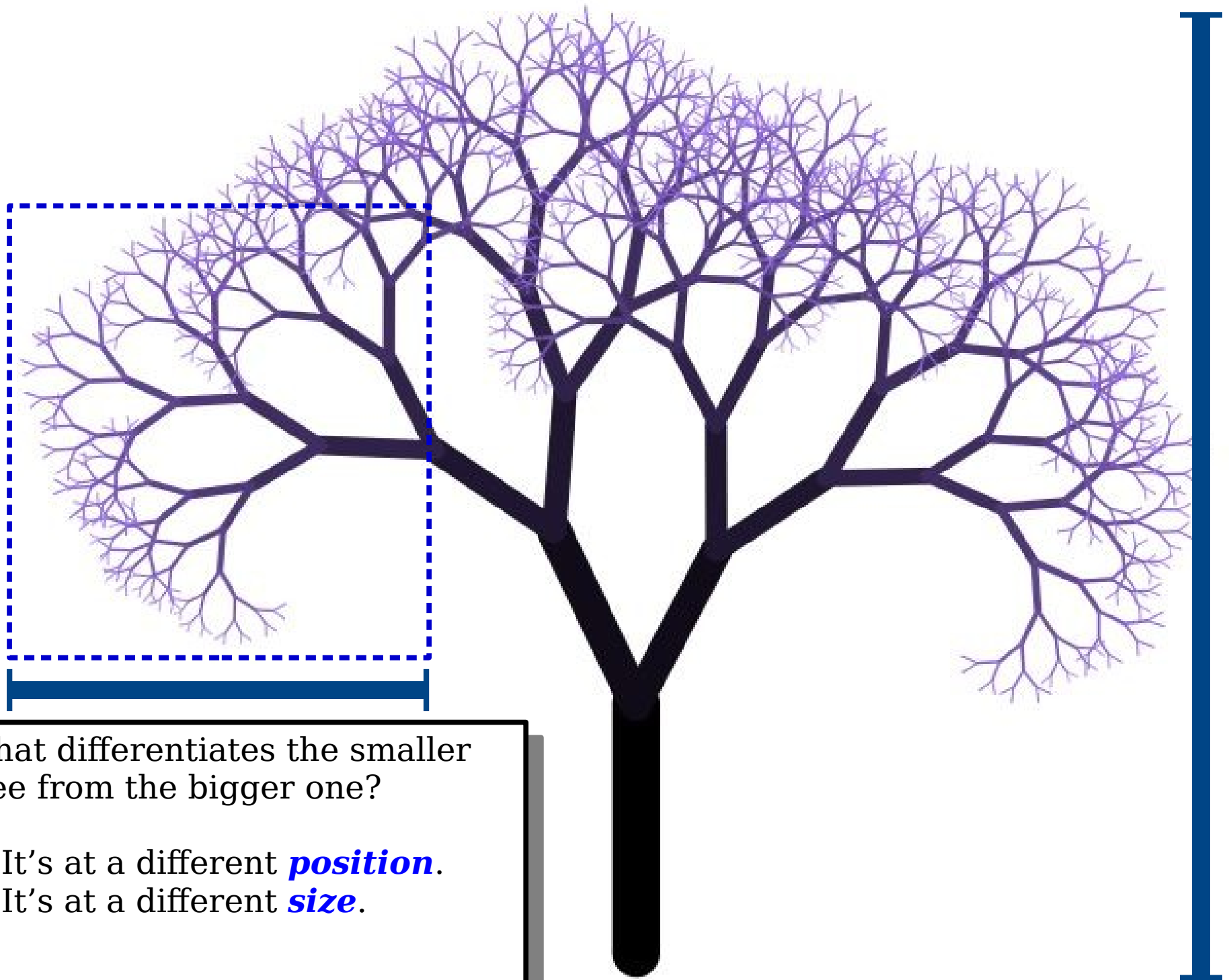


What differentiates the smaller tree from the bigger one?



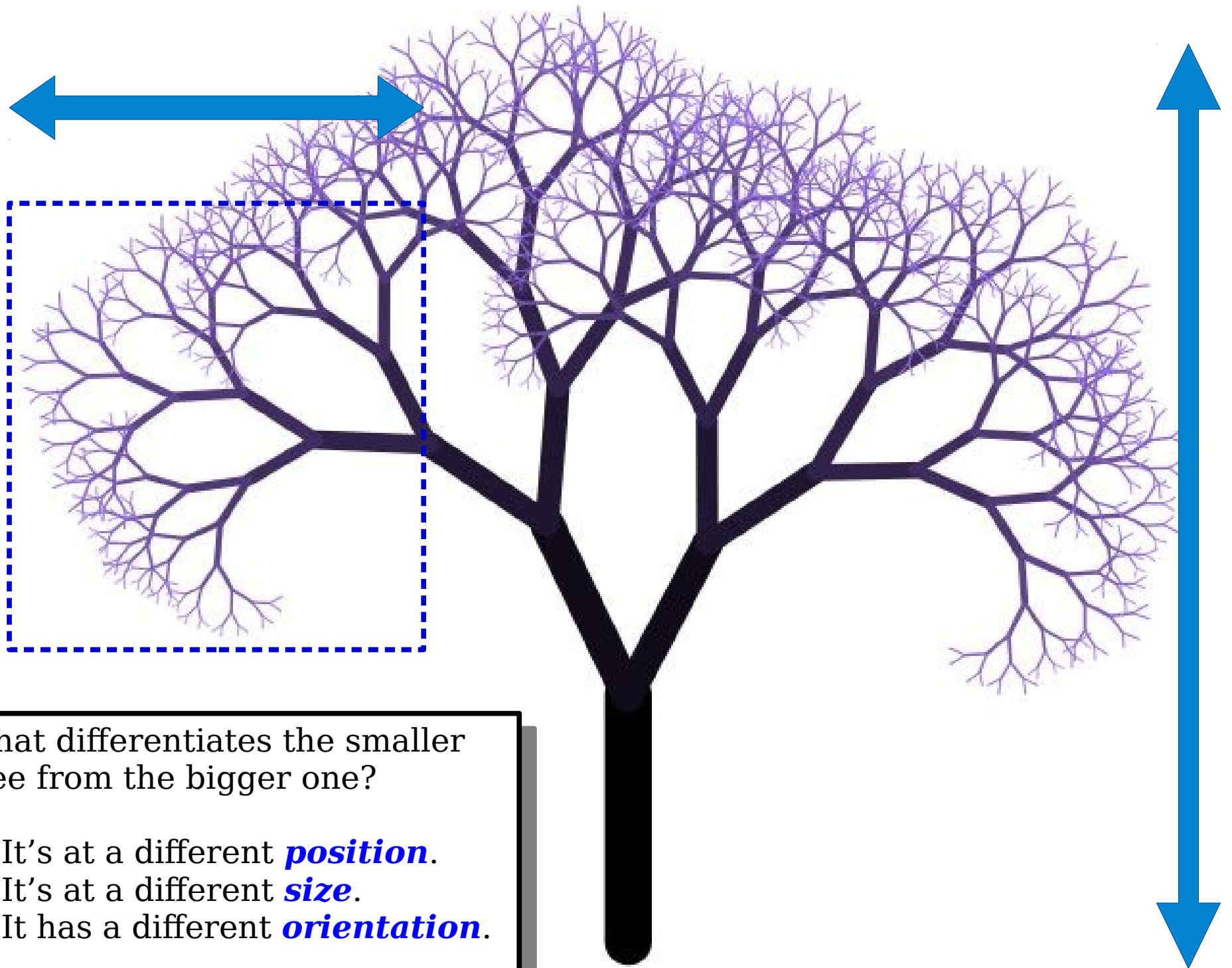
What differentiates the smaller tree from the bigger one?

1. It's at a different *position*.



What differentiates the smaller tree from the bigger one?

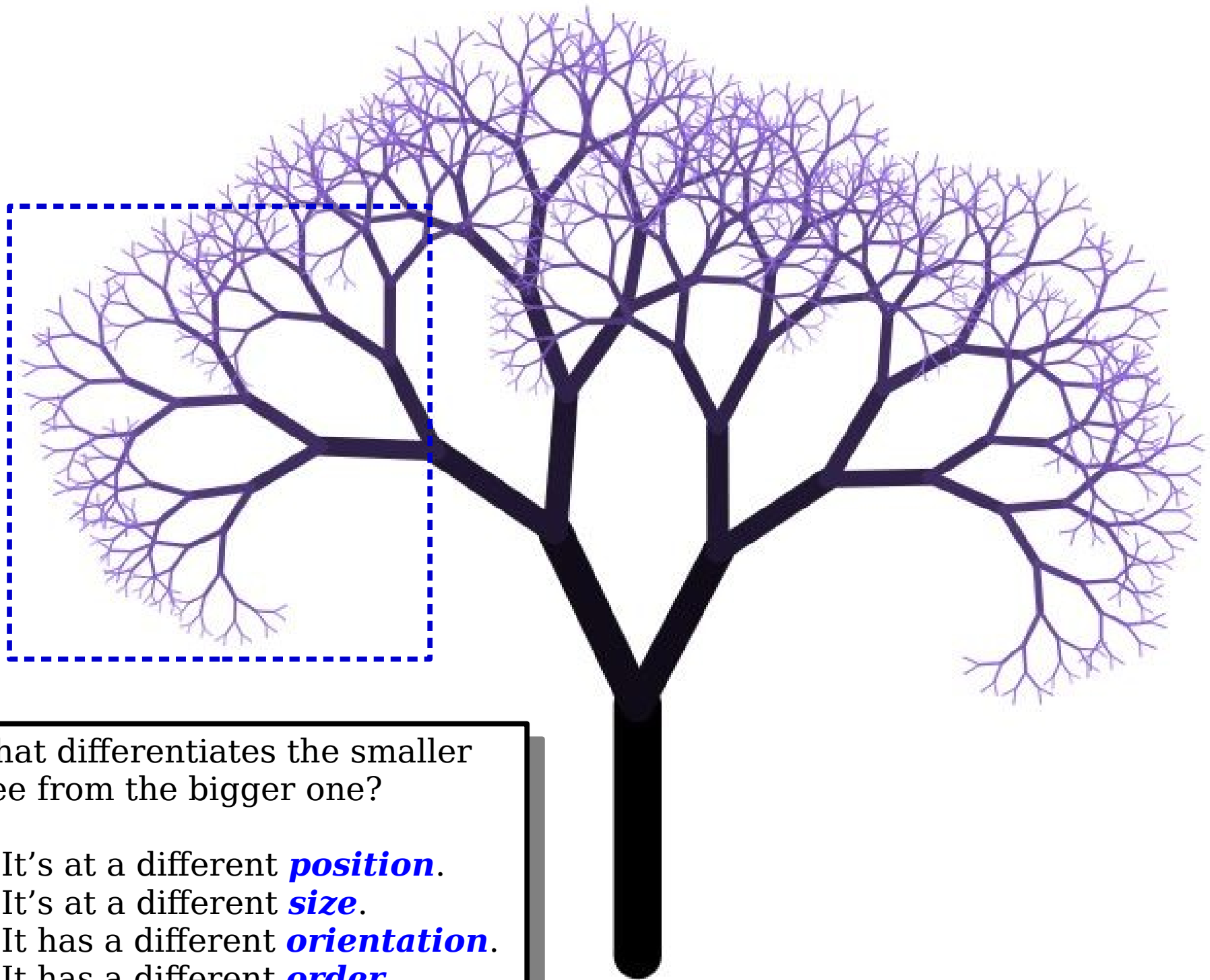
1. It's at a different ***position***.
2. It's at a different ***size***.



What differentiates the smaller tree from the bigger one?

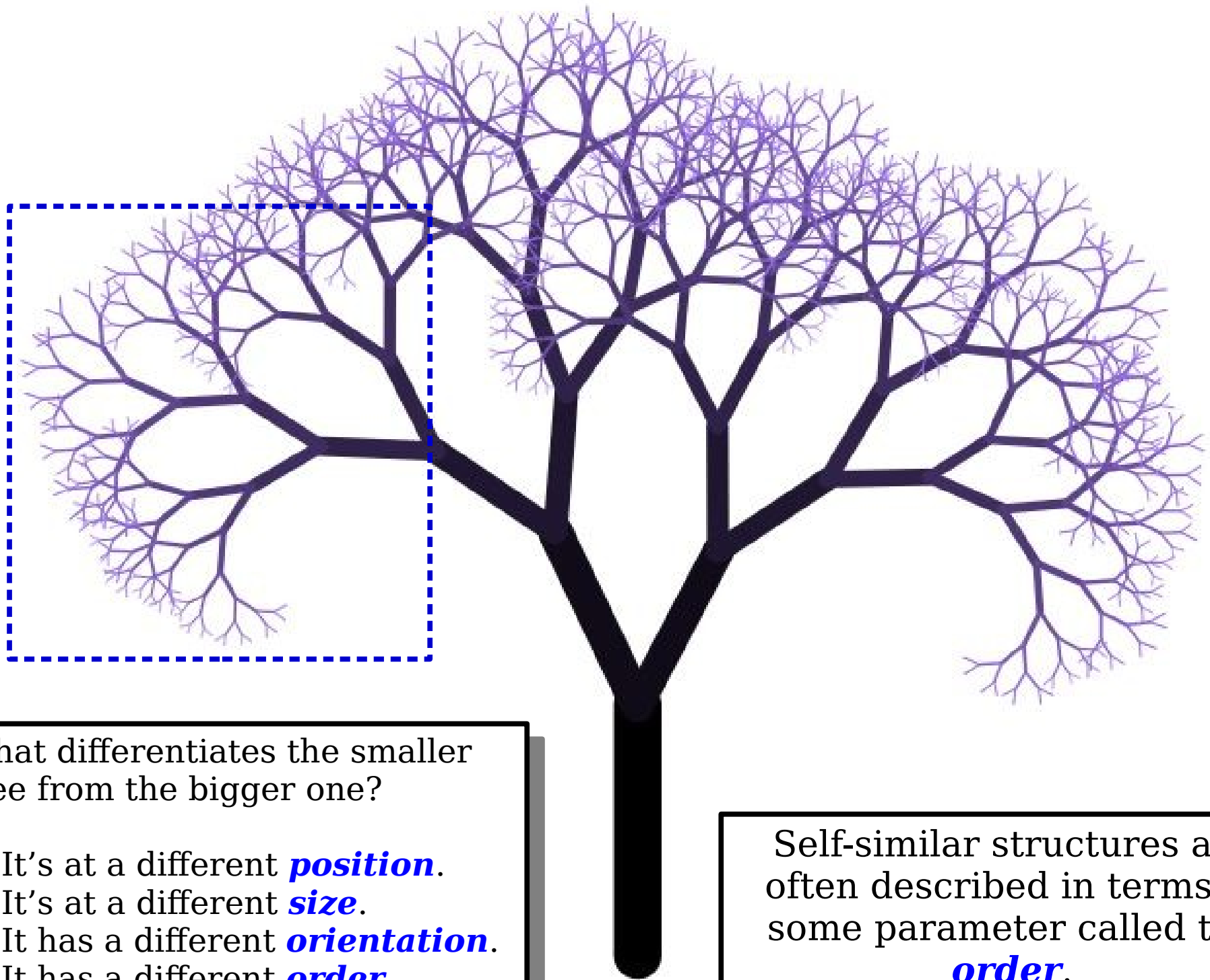
1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.





What differentiates the smaller tree from the bigger one?

1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.
4. It has a different ***order***.



What differentiates the smaller tree from the bigger one?

1. It's at a different **position**.
2. It's at a different **size**.
3. It has a different **orientation**.
4. It has a different **order**.

Self-similar structures are often described in terms of some parameter called the **order**.

# *An order-0 tree.*

What differentiates the smaller tree from the bigger one?

1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.
4. It has a different ***order***.

Self-similar structures are often described in terms of some parameter called the ***order***.

# *An order-1 tree.*

What differentiates the smaller tree from the bigger one?

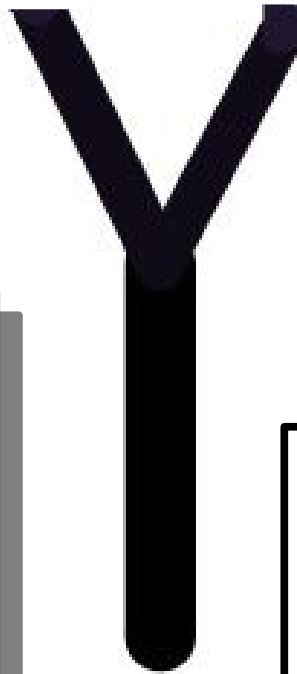
1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.
4. It has a different ***order***.

Self-similar structures are often described in terms of some parameter called the ***order***.

# *An order-2 tree.*

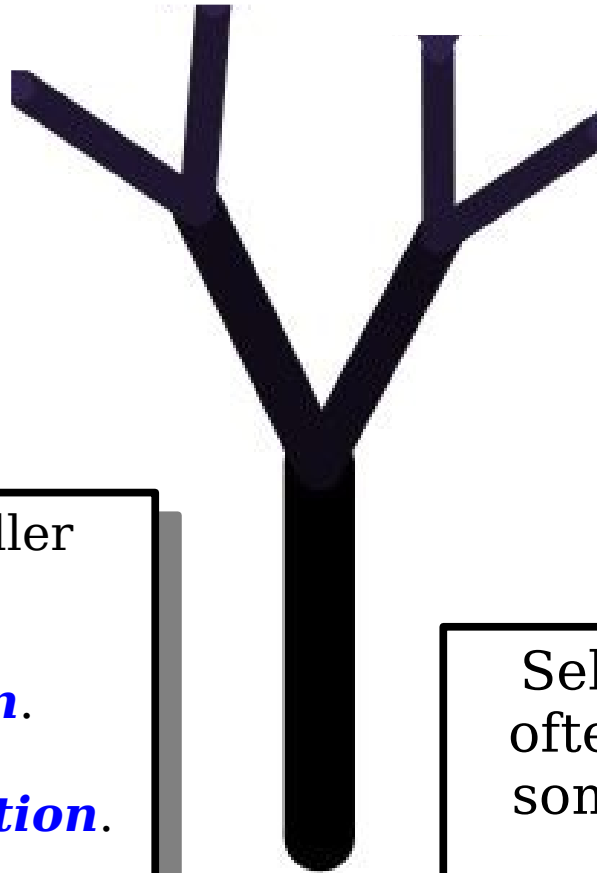
What differentiates the smaller tree from the bigger one?

1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.
4. It has a different ***order***.



Self-similar structures are often described in terms of some parameter called the ***order***.

# *An order-3 tree.*



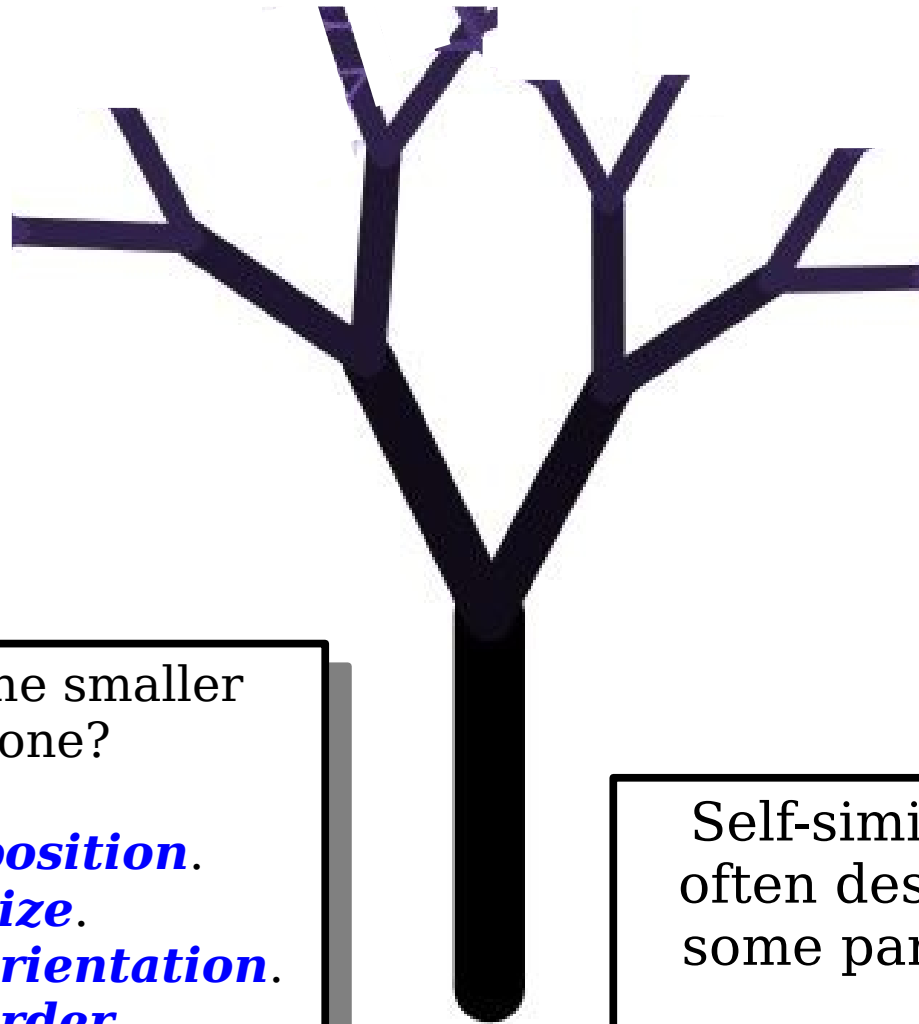
What differentiates the smaller tree from the bigger one?

1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.
4. It has a different ***order***.

Self-similar structures are often described in terms of some parameter called the ***order***.



# *An order-4 tree.*

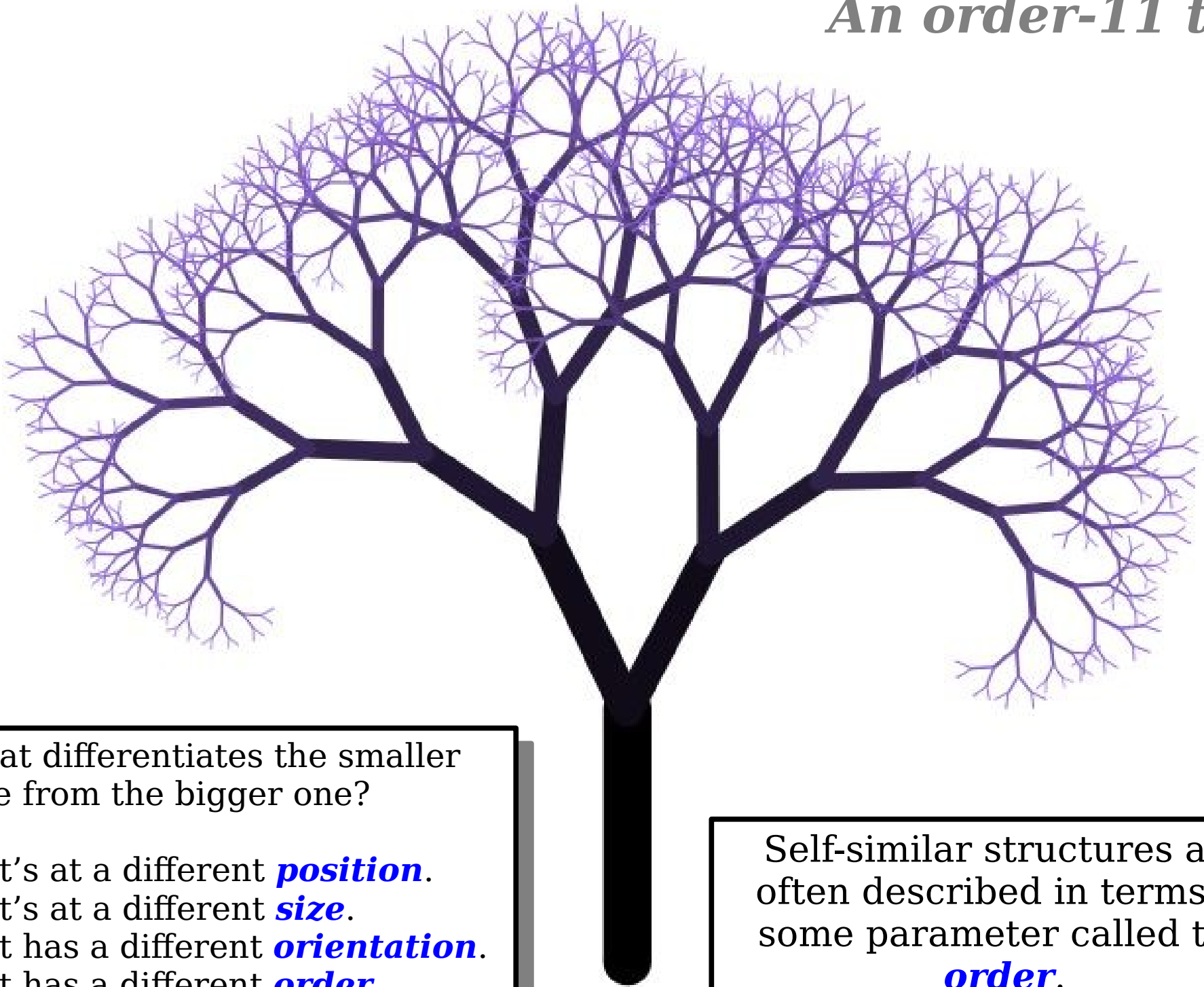


What differentiates the smaller tree from the bigger one?

1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.
4. It has a different ***order***.

Self-similar structures are often described in terms of some parameter called the ***order***.

## *An order-11 tree.*

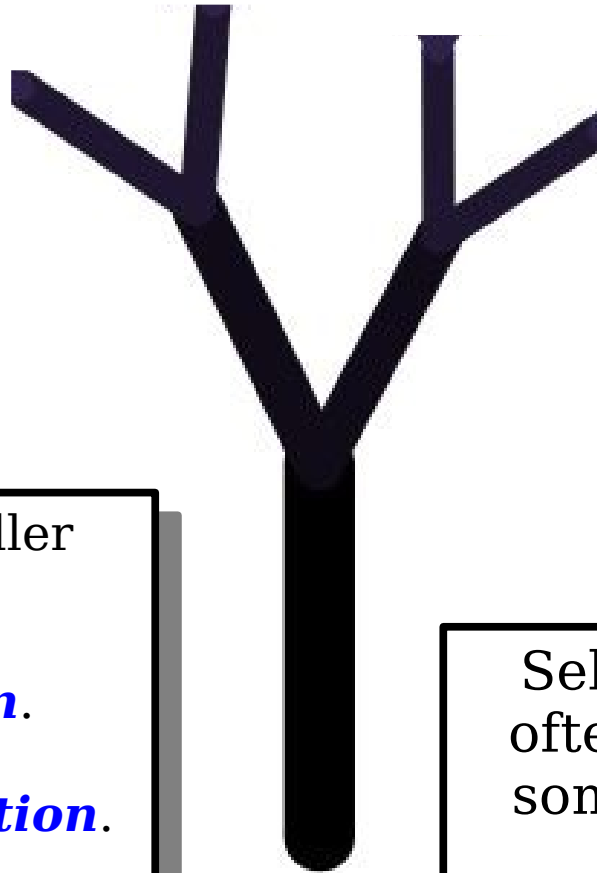


What differentiates the smaller tree from the bigger one?

1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.
4. It has a different ***order***.

Self-similar structures are often described in terms of some parameter called the ***order***.

# *An order-3 tree.*



What differentiates the smaller tree from the bigger one?

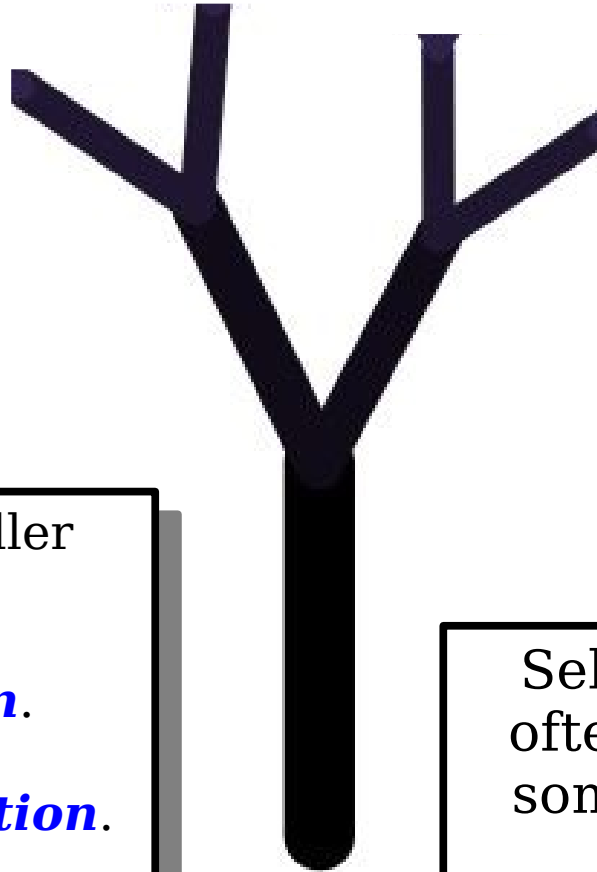
1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.
4. It has a different ***order***.

Self-similar structures are often described in terms of some parameter called the ***order***.

## *An order-3 tree.*

An order-0 tree is nothing at all.

An order- $n$  tree is a line with two smaller order- $(n-1)$  trees starting at the end of that line.



What differentiates the smaller tree from the bigger one?

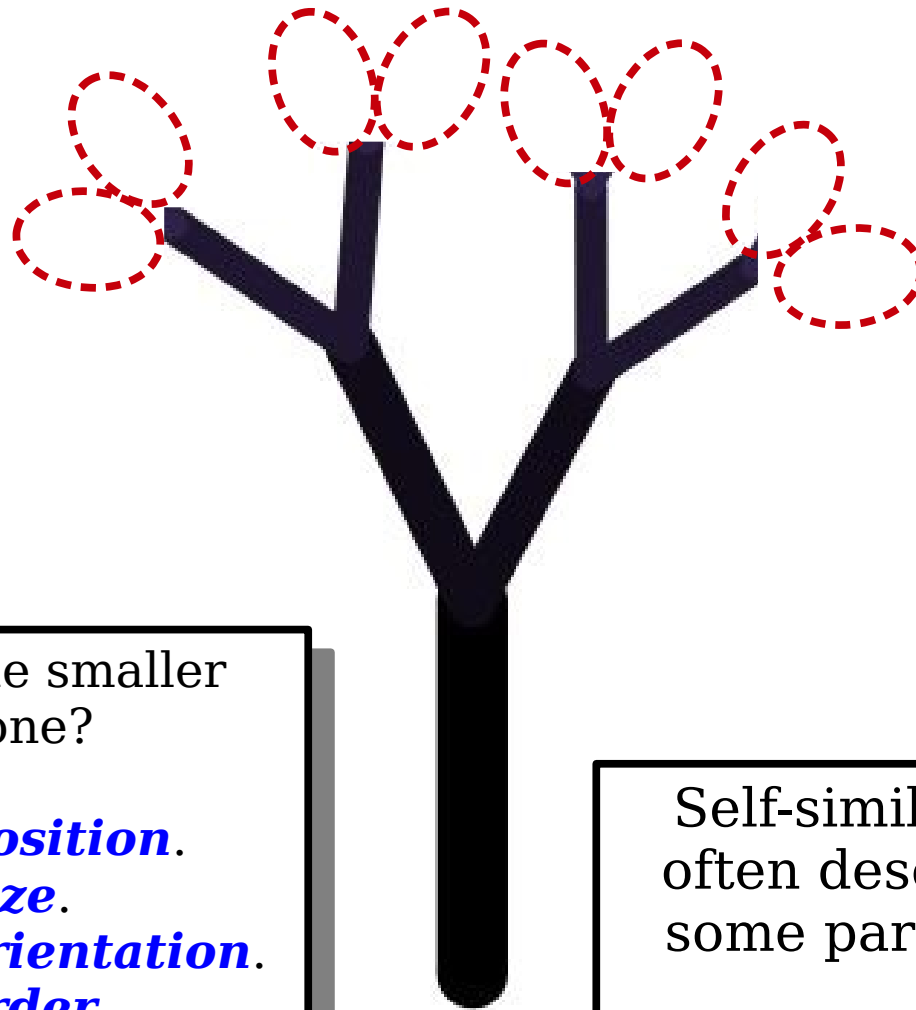
1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.
4. It has a different ***order***.

Self-similar structures are often described in terms of some parameter called the ***order***.

# *An order-3 tree.*

An order-0 tree is nothing at all.

An order- $n$  tree is a line with two smaller order- $(n-1)$  trees starting at the end of that line.



What differentiates the smaller tree from the bigger one?

1. It's at a different ***position***.
2. It's at a different ***size***.
3. It has a different ***orientation***.
4. It has a different ***order***.

Self-similar structures are often described in terms of some parameter called the ***order***.

# *An order-3 tree.*

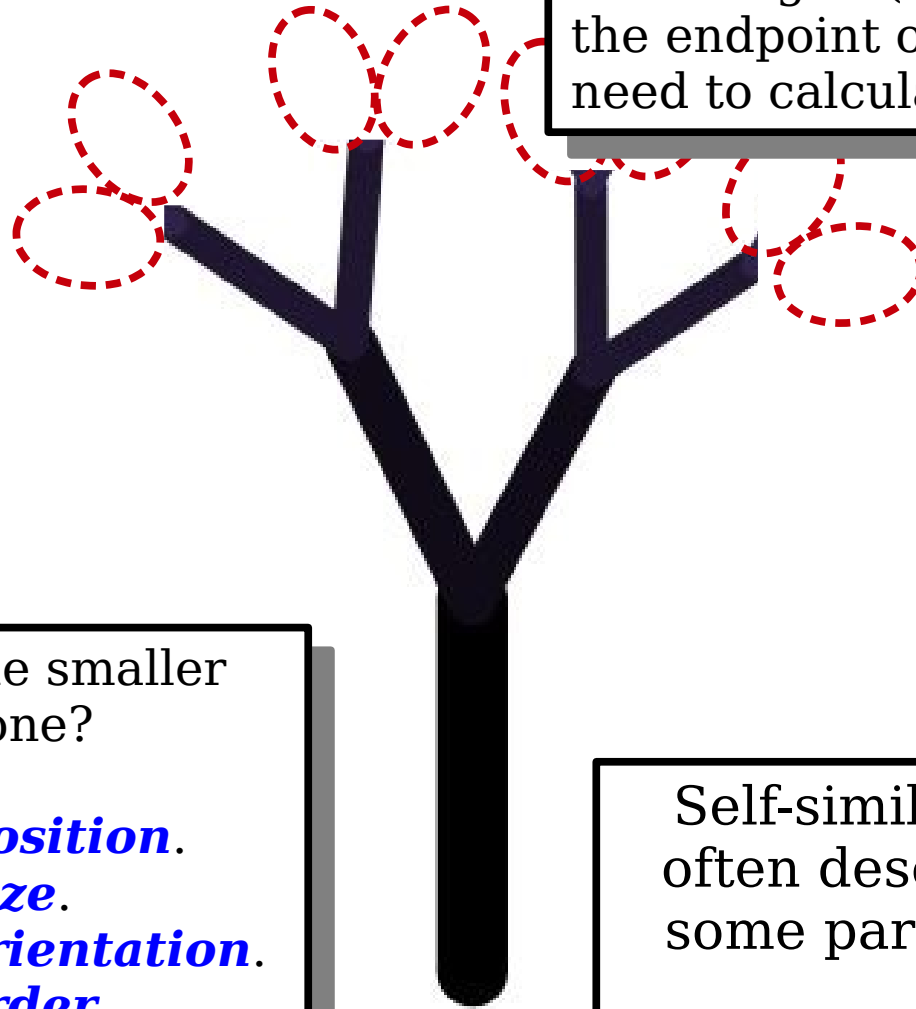
An order-0 tree is nothing at all.

An order- $n$  tree is a line with two smaller order- $(n-1)$  trees starting at the end of that line.

We can call the function

```
drawPolarLine(window, x, y, r,  $\theta$ )
```

to draw a line of radius  $r$  and angle  $\theta$  starting at  $(x, y)$ . It then returns the endpoint of the line so we don't need to calculate it ourselves!



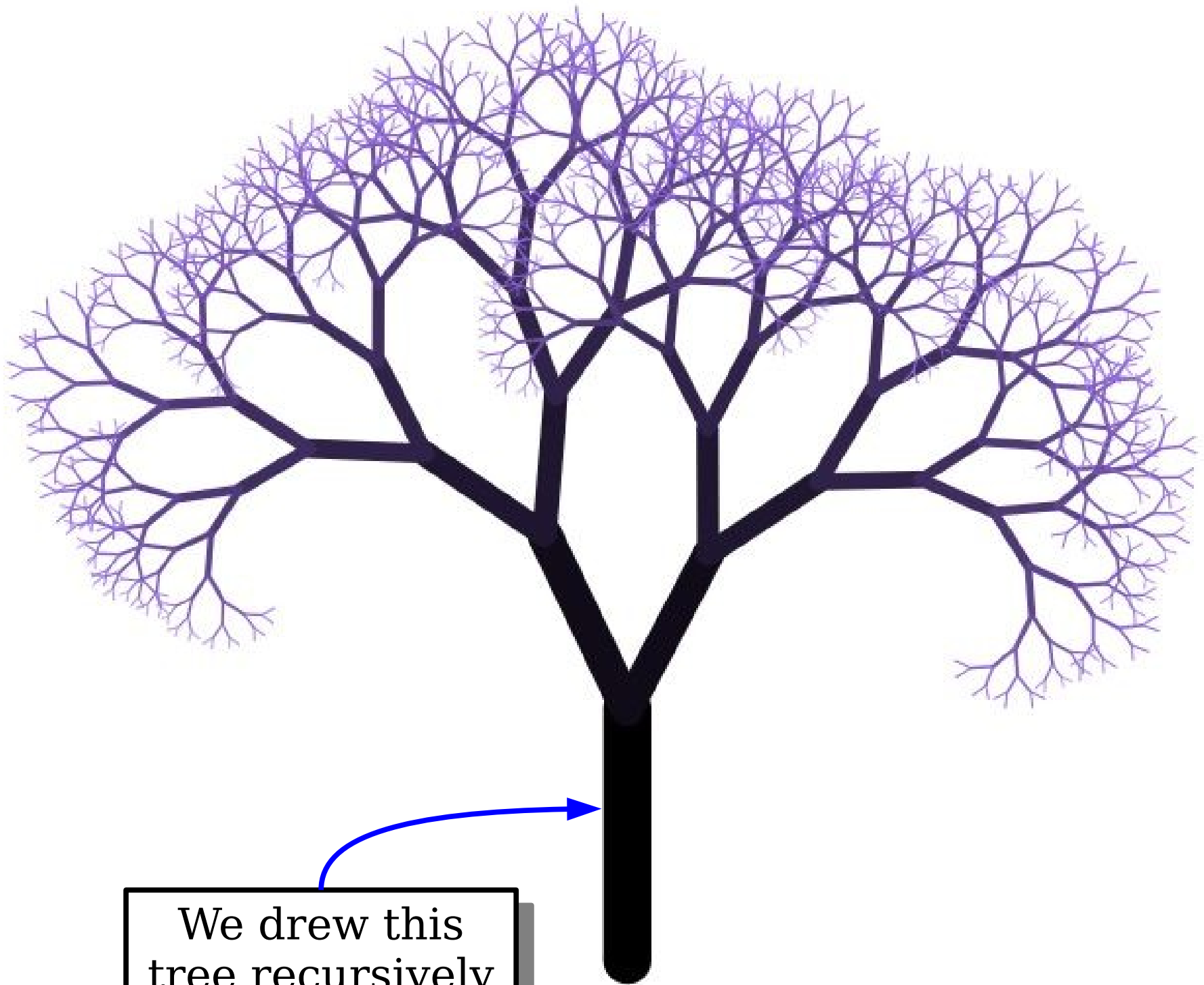
What differentiates the smaller tree from the bigger one?

1. It's at a different **position**.
2. It's at a different **size**.
3. It has a different **orientation**.
4. It has a different **order**.

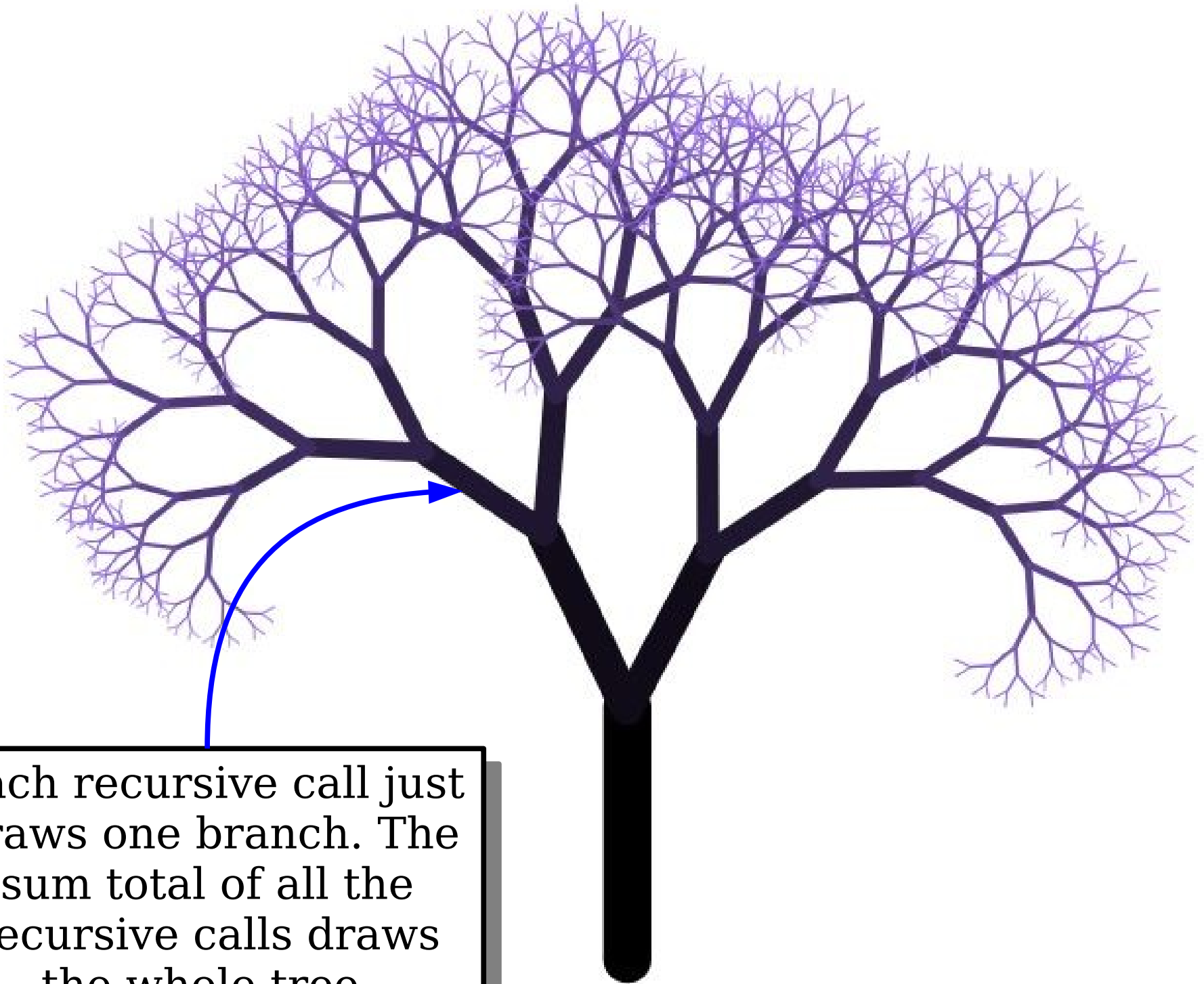
Self-similar structures are often described in terms of some parameter called the **order**.



To Summarize



We drew this tree recursively



Each recursive call just draws one branch. The sum total of all the recursive calls draws the whole tree.

# An Amazing Website

***<http://recursivedrawing.com/>***

**Time-Out for Announcements!**

# Assignment 2

- Assignment 2 is due on Friday.
  - If you're following our suggested timetable, you should be done with Rising Tides at this point and should be working on You Got Hufflepuff!
- Have questions?
  - Stop by the LaIR!
  - Email your section leader!
  - Ask on Piazza!
  - Visit Keith's or Katherine's office hours!

# Submitting Your Work

- Each assignment handout has a “Submission Instructions” section at the end with information about what files to submit.
- ***Please submit all the files listed there.*** Otherwise, we can't grade your work.
- Thanks!

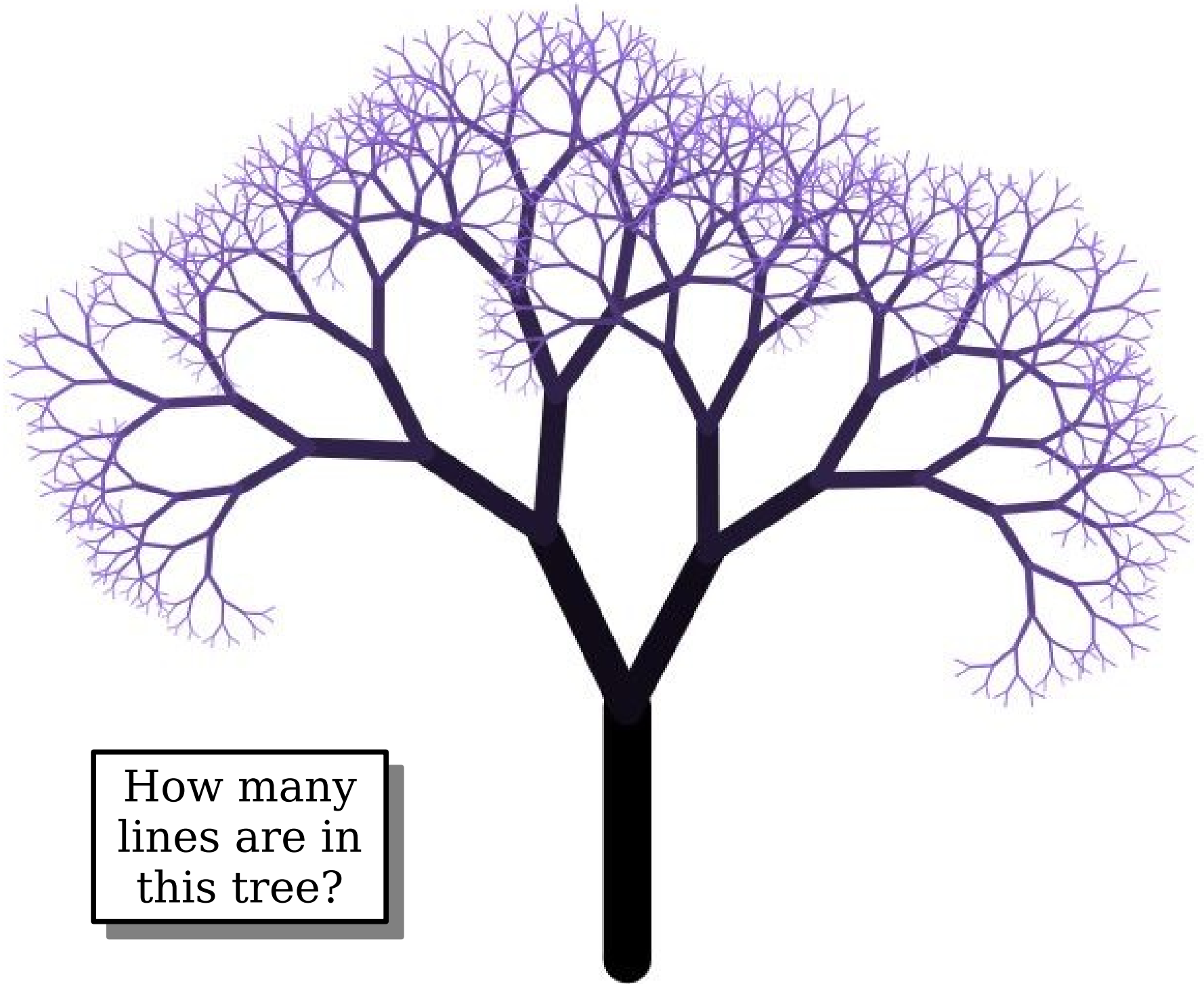
Onward and Forward!



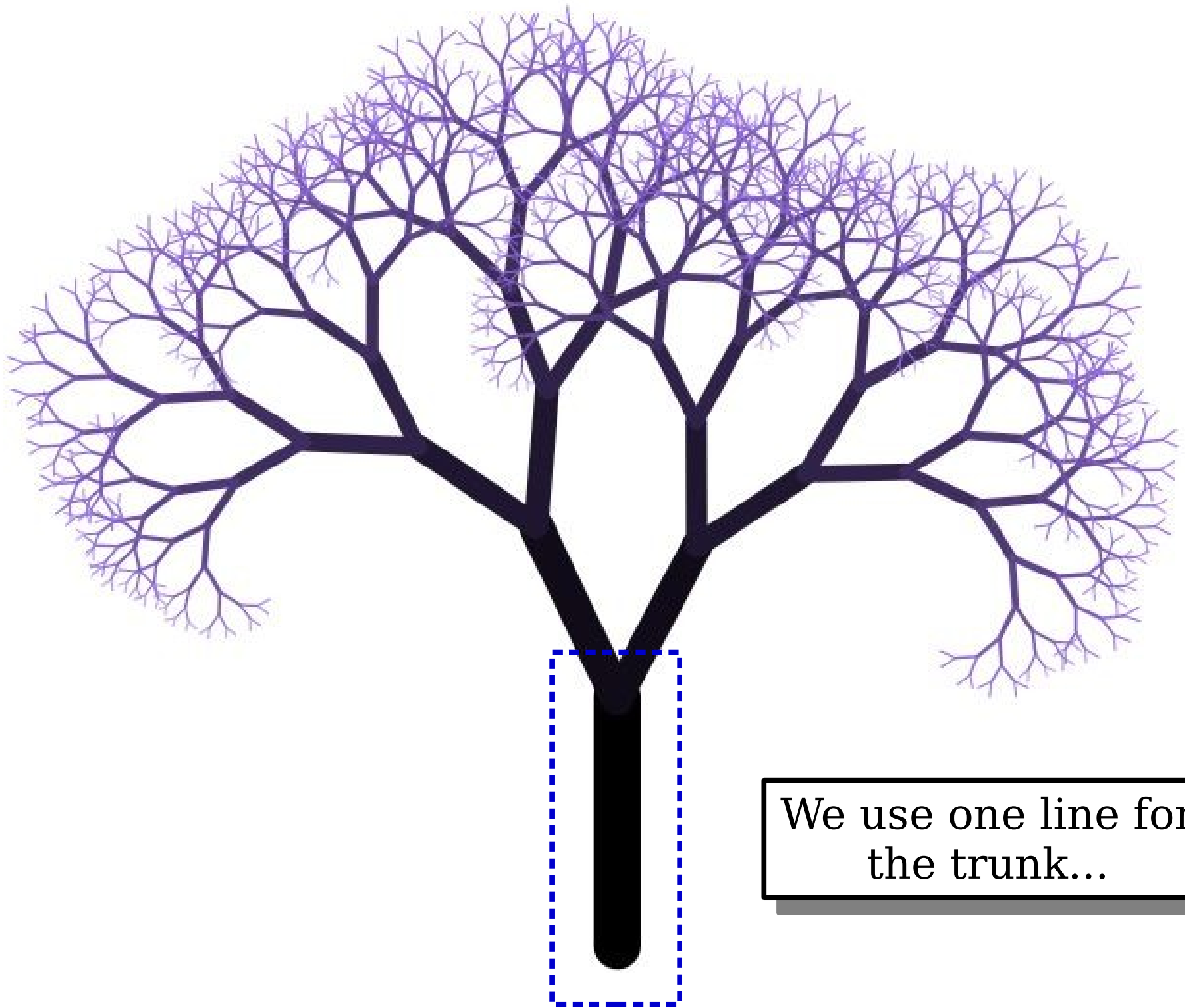
How many lines make up each tree?

# Communicating Across Calls

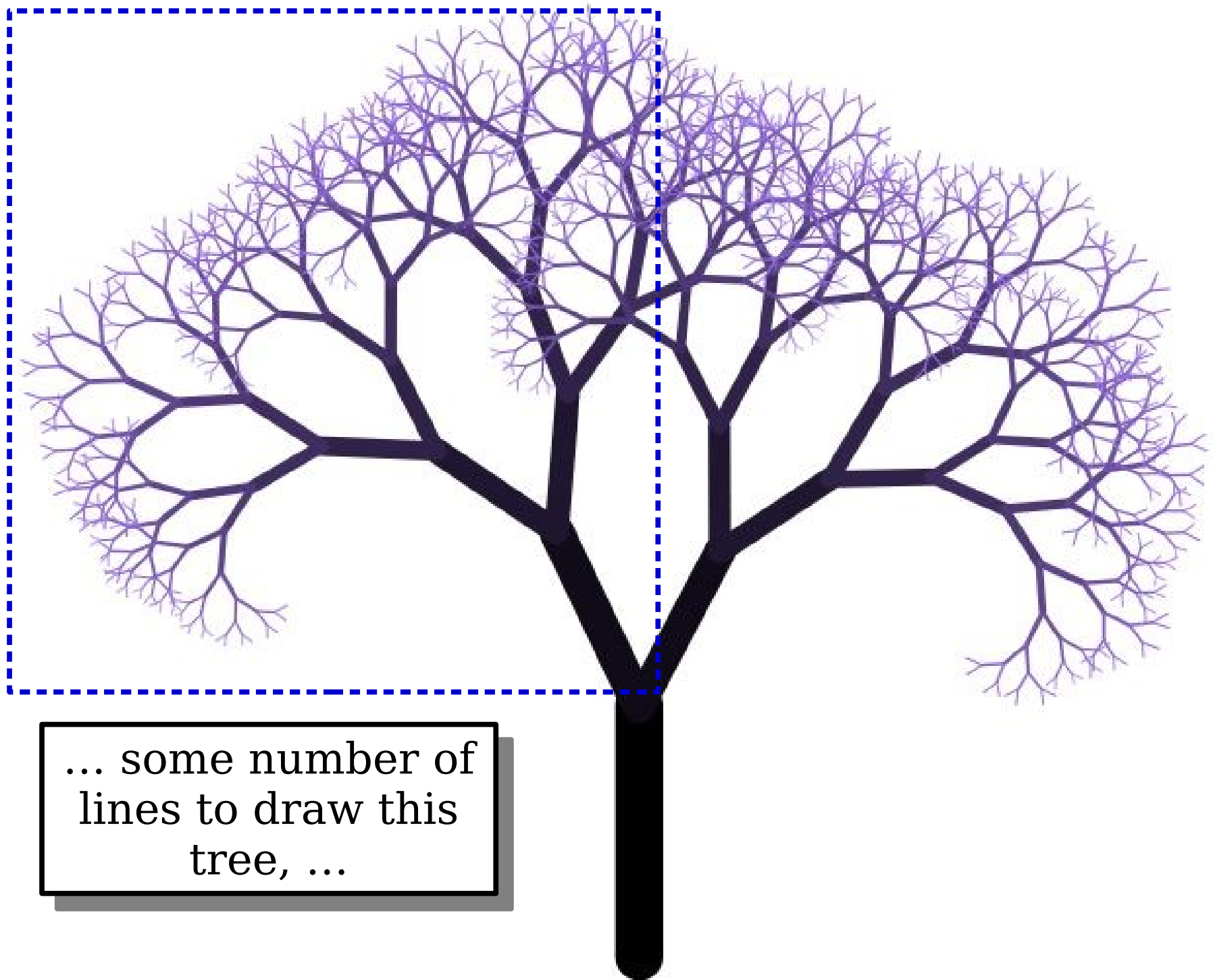
- Each copy of a recursive call gets its own copy of each local variable.
- Changing a local variable in one recursive call does not change other copies of those variables across calls.
- How do we aggregate information across multiple recursive calls?



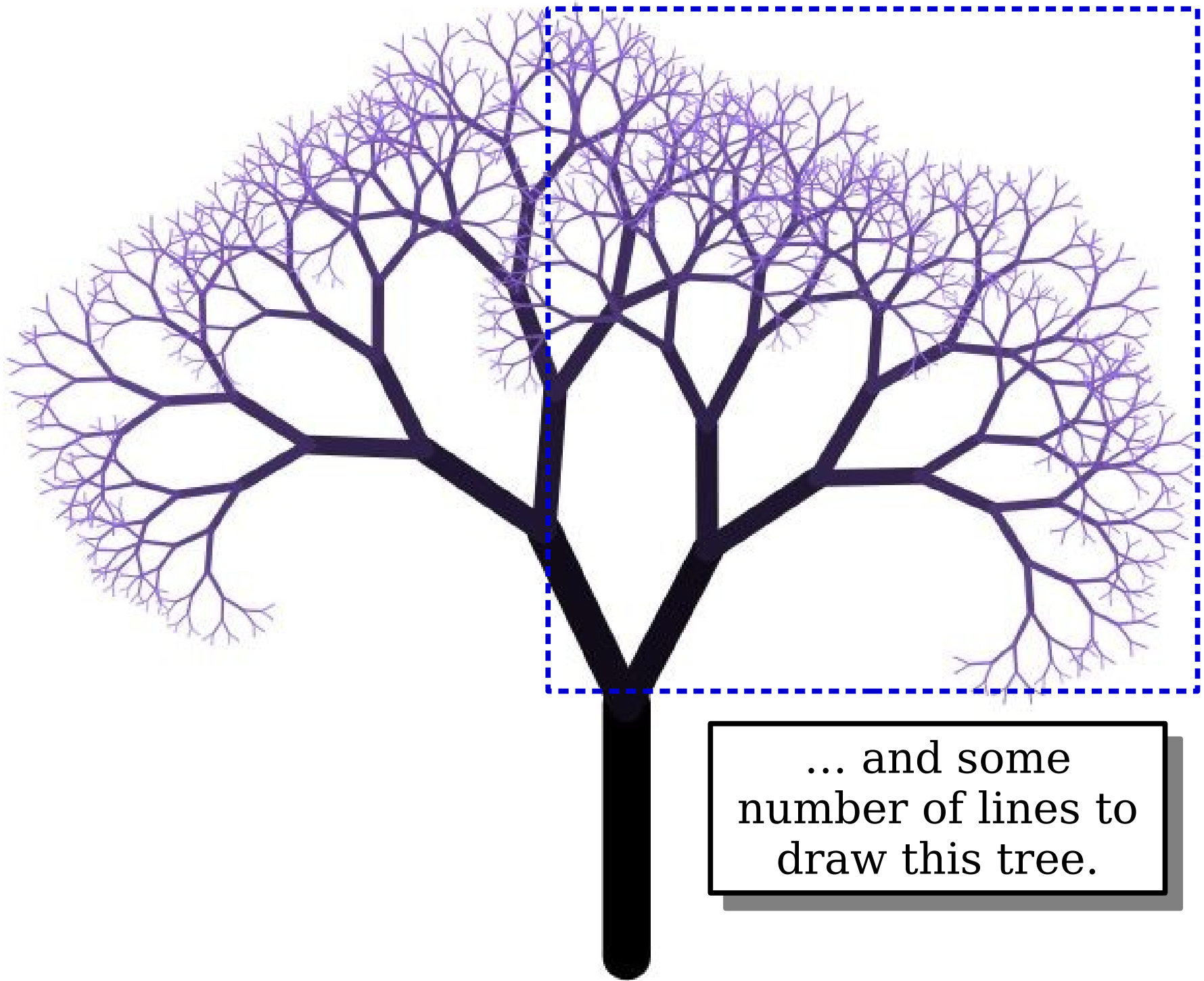
How many  
lines are in  
this tree?



We use one line for  
the trunk...




... some number of  
lines to draw this  
tree, ...



... and some  
number of lines to  
draw this tree.

# A Practical Application



A photograph of a dense forest. In the foreground, a large, dark tree trunk with thick moss curves from the right side towards the center. The background is filled with many tall, thin, vertical tree trunks, likely redwoods or sequoias, reaching towards a bright sky. The foliage is a vibrant green, and the overall scene is dappled with sunlight and shadow.

The beautiful thing is that the distribution of the sizes of individual trees in the forest appears to exactly match the distribution of the sizes of individual branches within a single tree [...]

[S]tudying a single tree will make it easier to predict how much carbon dioxide an entire forest can absorb.

- ***Hunting the Hidden Dimension***



# Your Action Items

- ***Read Chapter 8.***
  - There's a ton of goodies in there! It'll help you solidify your understanding.
- ***Finish Assignment 2.***
  - Best of luck! Reach out to us when you need help.

# Next Time

- ***Recursive Enumeration***
  - Finding all objects of a given type.
- ***Enumerating Subsets***
  - A classic combinatorial problem!