Programming Refresher Workshop

Session 5

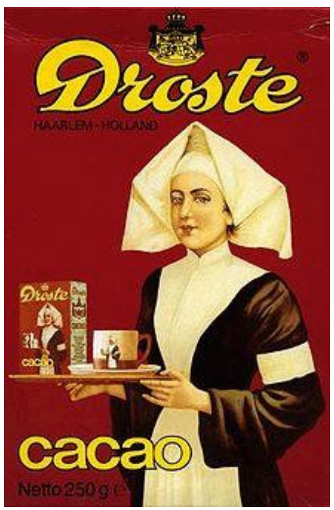
Dr Henry Chia

Contents

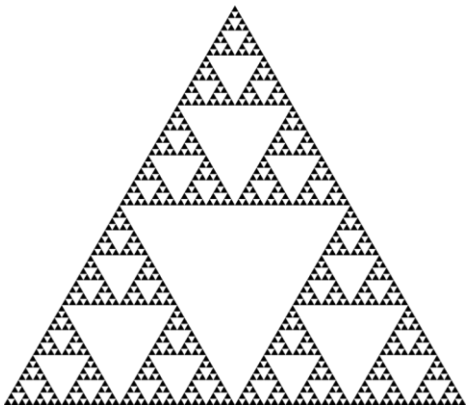
- What is recursion?
- Characteristics of recursion
- Tracing recursive calls
- Identifying the sub-problem
- General recursive problems
- Gist of recursion
- Testing a recursive function

What is Recursion? (1/2)

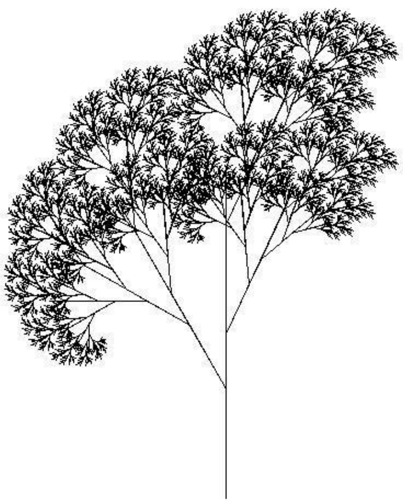
- Other forms of recursion



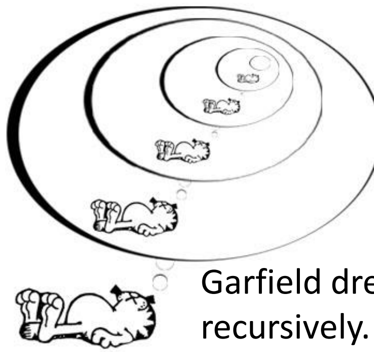
Droste effect



Sierpinksi triangle



Recursive tree



Garfield dreaming recursively.

What is Recursion? (2/2)

- A method of problem solving where the solution of a problem depends on solutions to smaller instances of the SAME problem.

Base/Degenerated case

$$Factorial(0) = 1$$

$$Factorial(1) = 1$$

$$Factorial(2) = 2 * 1 = 2$$

...

$$Factorial(n) = n * (n\text{-}1) * \ldots * 2 * 1$$

$$= n * Factorial(n\text{-}1)$$

Recursive case

Example

- Given two integers *a* and *b*, with *a* <= *b*, find the sum of the square of the numbers between *a* and *b*, both inclusive.

Base/Degenerated case

$sumSq(5,5) = 5^2$

$sumSq(5,6) = 5^2 + 6^2 = sumSq(5,5) + 6^2$

$sumSq(5,7) = 5^2 + 6^2 + 7^2 = sumSq(5,6) + 7^2$

...

$sumSq(5,15) = 5^2 + 6^2 + \ldots + 14^2 + 15^2$

$\quad = sumSq(5,14) + 15^2$

Recursive case

$sumSq(a,b) = sumSq(a, b\text{-}1) + b^2$

Characteristics of Recursion

- Does base cases exist?
- Are the recursive argument(s) getting "smaller"?
- Does the recursion ever reach the base case?

sumSq(a,b) =
  pre: a <= b
  If (a < b) then
    return **sumSq(a,b-1) + b*b**
  else
    return a*a

Tracing the Recursive calls

sumSq(a,b) =
pre: a <= b
If ($a < b$) then
return **sumSq(a,b-1) + b*b**
else
return a*a

sumSq(5,7)
→ sumSq(5,6) + 7^2
→ sumSq(5,5) + 6^2
← return $5^2 = 25$ from sumSq(5,5)
← return $25 + 6^2 = 61$ from sumSq(5,6)
← return $61 + 7^2 = 110$ from sumSq(5,7)

Identifying the sub-problem (1/2)

- Other ways to perform the sum of squares?
- sumSq(5,5) → 5^2
- sumSq(5,7) → $5^2 + 6^2 + 7^2$

  → sumSq(5,6) + 7^2 ?

  → 5^2 + sumSq(6,7) ?

  → 5^2 + sumSq(6,6) + 7^2 ?

  → sumSq(5,6) + sumSq(7,7) ?

  → …

Identifying the sub-problem (2/2)

- 'Combining two half-solutions' recursion:

sumSq(a,b) =
  pre: a <= b
  If ($a < b$) then
    $m = (a + b)/2$
    return **sumSq(a,m) + sumSq(m+1,b)**
  else
    return a*a

General Recursive Problems

- Example: Define a recursive function to print the first *n* elements of an array *arr* in reverse
- Print the last element, then call the function recursively to print *arr* from the start till just before the last element.
- What is the base case?

printArray (arr,n) =
  If ($n > 0$) then
    print $arr[n\text{-}1]$
    printArray(arr,n-1)

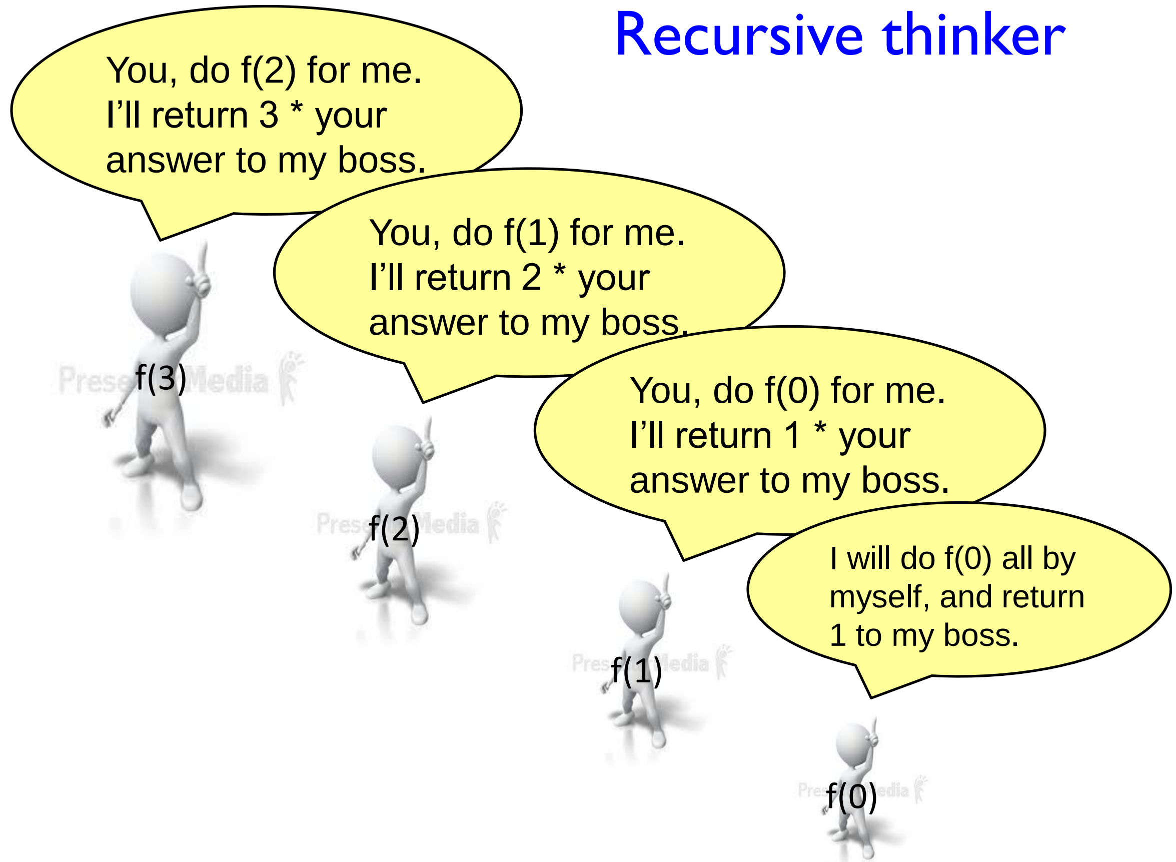
  return

Gist of Recursion (1/2)

Iteration vs Recursion: How to compute factorial(3)?

Iterative thinker

Recursive thinker



Gist of Recursion (2/2)

- **The One-Layer Thinking Maxim**

Don't try to think recursively about a recursive process

Illustration: ***Compute n^2 recursively.***

Moment of inspiration:

$$(n-1)^2 = n^2 - 2n + 1$$

Thus,

$$n^2 = \begin{cases} 0 & \textit{if} \quad n = 0 \\ (n-1)^2 + 2n - 1 & \textit{otherwise} \end{cases}$$

There is no need to think about how $(n-1)^2$ computes

Testing a Recursive Function/Method

- Check that it runs on **base cases**
- Check that it runs on slightly more complicated (than base) recursive cases
- Check the correctness of recursive cases via **tracing**

The End