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

Sierpinski 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**
    - \( \text{Factorial}(0) = 1 \)
    - \( \text{Factorial}(1) = 1 \)
    - \( \text{Factorial}(2) = 2 \times 1 = 2 \)
    - \( \ldots \)
    - \( \text{Factorial}(n) = n \times (n-1) \times \ldots \times 2 \times 1 \)
    - \( = n \times \text{Factorial}(n-1) \)

  - **Recursive case**
Example

- Given two integers $a$ and $b$, with $a \leq b$, find the sum of the square of the numbers between $a$ and $b$, both inclusive.

\[
\text{sumSq}(5,5) = 5^2
\]

\[
\text{sumSq}(5,6) = 5^2 + 6^2 = \text{sumSq}(5,5) + 6^2
\]

\[
\text{sumSq}(5,7) = 5^2 + 6^2 + 7^2 = \text{sumSq}(5,6) + 7^2
\]

\[
\text{...}
\]

\[
\text{sumSq}(5,15) = 5^2 + 6^2 + \ldots + 14^2 + 15^2
\]

\[
= \text{sumSq}(5,14) + 15^2
\]

\[
\text{sumSq}(a,b) = \text{sumSq}(a, b-1) + b^2
\]
Characteristics of Recursion

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

\[
\text{sumSq}(a,b) =
\begin{align*}
\text{pre: } a & \leq b \\
\text{If } (a < b) \text{ then} & \\
\quad \text{return } \text{sumSq}(a,b-1) + b*b \\
\text{else} & \\
\quad \text{return } a*a
\end{align*}
\]
Tracing the Recursive calls

\[\text{sumSq}(a,b) = \]
\[\text{pre: } a \leq b\]
\[\text{If } (a < b) \text{ then}\]
\[\text{return } \text{sumSq}(a,b-1) + b^2\]
\[\text{else}\]
\[\text{return } a^2\]

\[\text{sumSq}(5,7)\]
\[\rightarrow \text{sumSq}(5,6) + 7^2\]
\[\rightarrow \text{sumSq}(5,5) + 6^2\]
\[\leftarrow \text{return } 5^2 = 25 \text{ from sumSq}(5,5)\]
\[\leftarrow \text{return } 25 + 6^2 = 61 \text{ from sumSq}(5,6)\]
\[\leftarrow \text{return } 61 + 7^2 = 110 \text{ from sumSq}(5,7)\]
Other ways to perform the sum of squares?

- \( \text{sumSq}(5,5) \to 5^2 \)
- \( \text{sumSq}(5,7) \to 5^2 + 6^2 + 7^2 \)
  - \( \to \text{sumSq}(5,6) + 7^2 \) ?
  - \( \to 5^2 + \text{sumSq}(6,7) \) ?
  - \( \to 5^2 + \text{sumSq}(6,6) + 7^2 \) ?
  - \( \to \text{sumSq}(5,6) + \text{sumSq}(7,7) \) ?
  - \( \to \ldots \)
Identifying the sub-problem (2/2)

- ‘Combining two half-solutions’ recursion:

\[
\text{sumSq}(a,b) =
\]

pre: \( a \leq b \)

If \( a < b \) then

\[
m = (a + b)/2
\]

return \( \text{sumSq}(a,m) + \text{sumSq}(m+1,b) \)

else

return \( a^2 \)
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-1]
        printArray(arr, n-1)
    return
```
Gist of Recursion (1/2)

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

Iterative thinker

I do f(3) all by myself…return 6 to my boss.

Recursive thinker

You, do f(2) for me. I’ll return 3 * your answer to my boss.
You, do f(1) for me. I’ll return 2 * your answer to my boss.
You, do f(0) for me. I’ll return 1 * your answer to my boss.
I will do f(0) all by myself, and return 1 to my boss.
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,

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

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