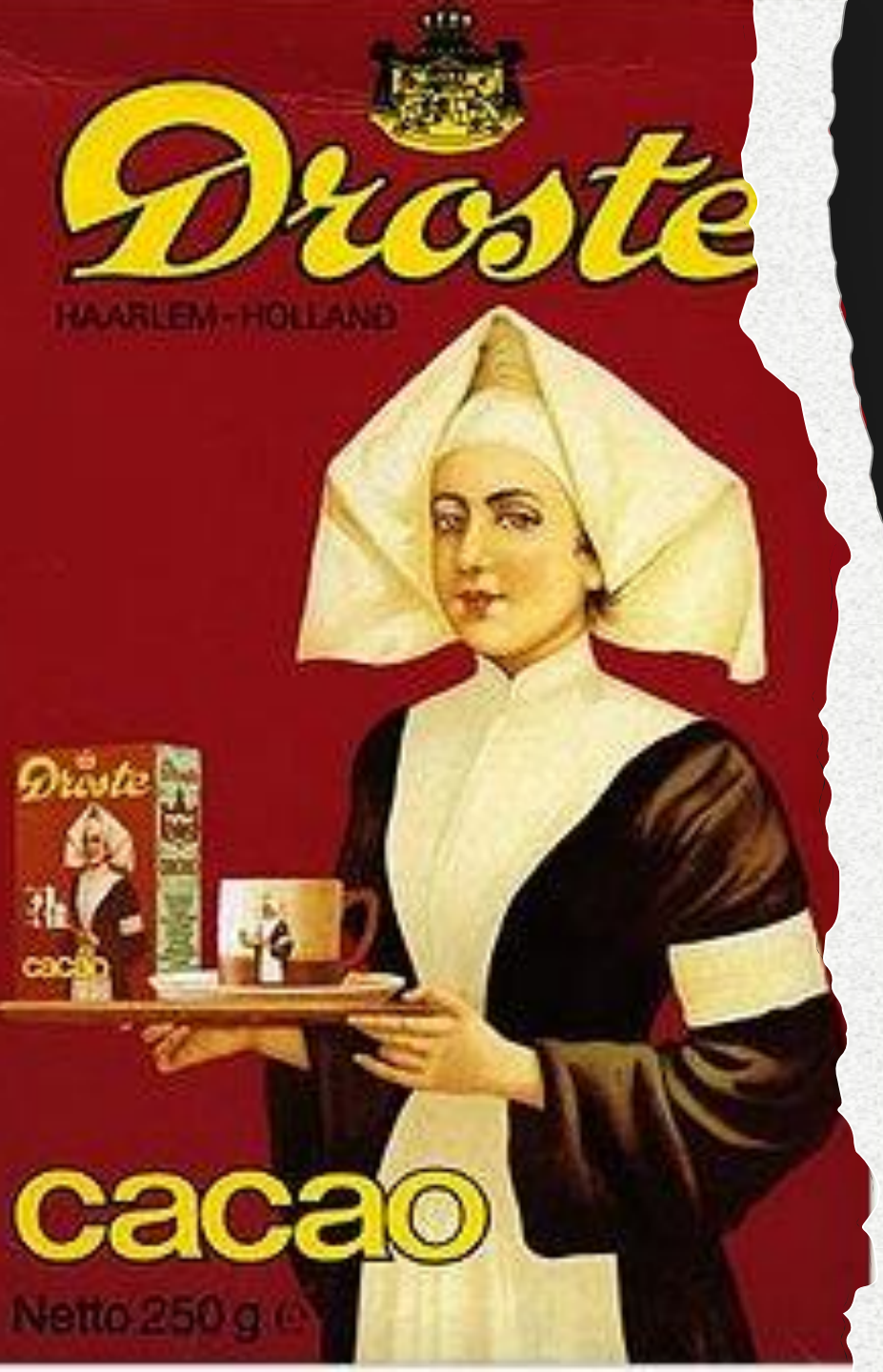
SÃO
PAULO
TECH
SCHOOL

Recursão



Recursão no nosso dia a dia

Imagem recursiva – efeito Droste (imagem dentro de outra)

Recursão no nosso dia a dia

Definição de número natural:

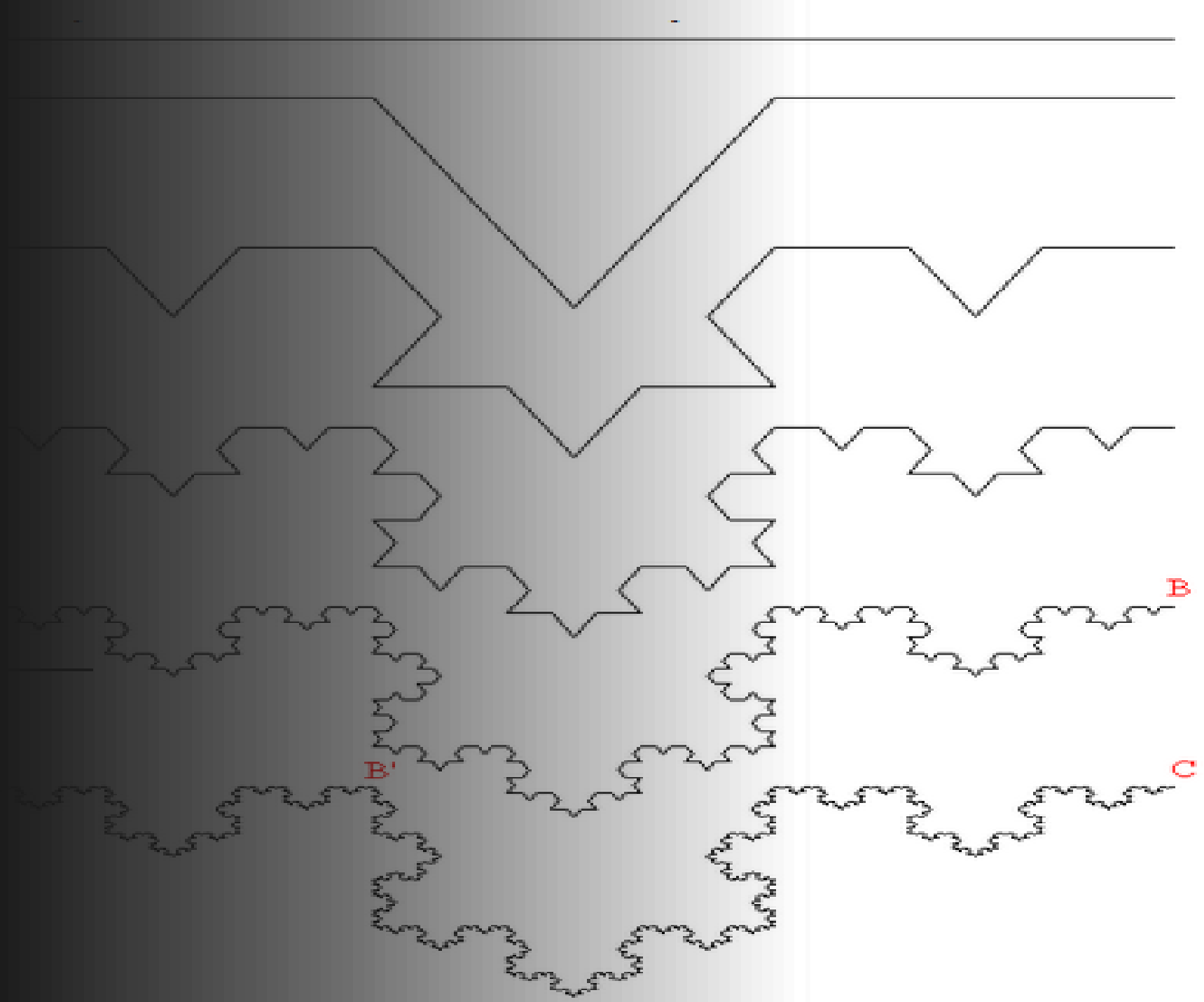
- 0 é um número natural.
- Se n é um número natural, então n+1 (sucessor de um número natural) também é um número natural.

Recursão no nosso dia a dia

Acrônimos recursivos:

- **GNU** – **G**nu is **N**ot **U**nix
- **PHP** – **P**HP: **H**ypertext **P**reprocessor
  - originalmente: Personal Home Page
- **BING** – **B**ing **I**s **N**ot **G**oogle

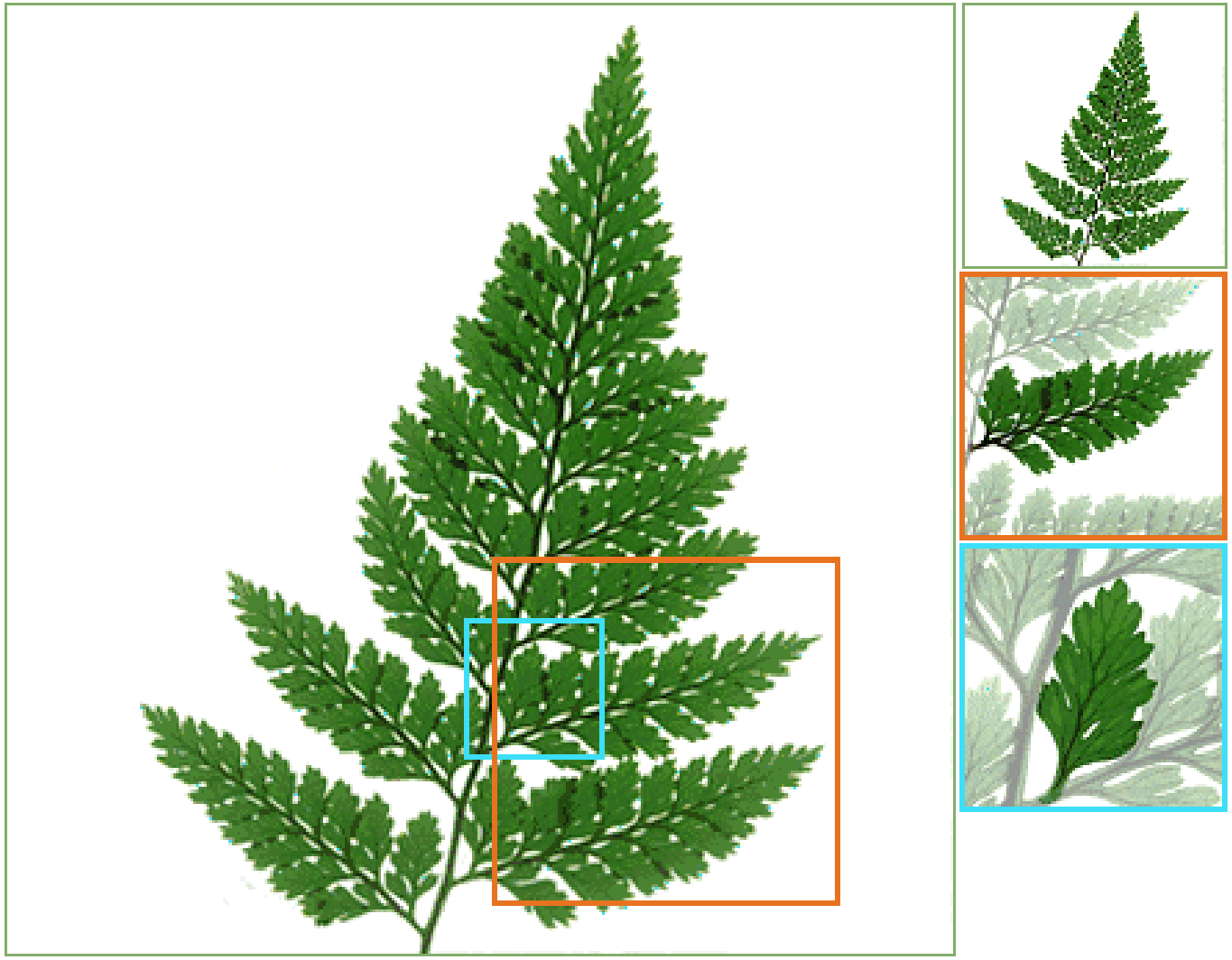
Curva de Koch



Triângulo de Sierpinski

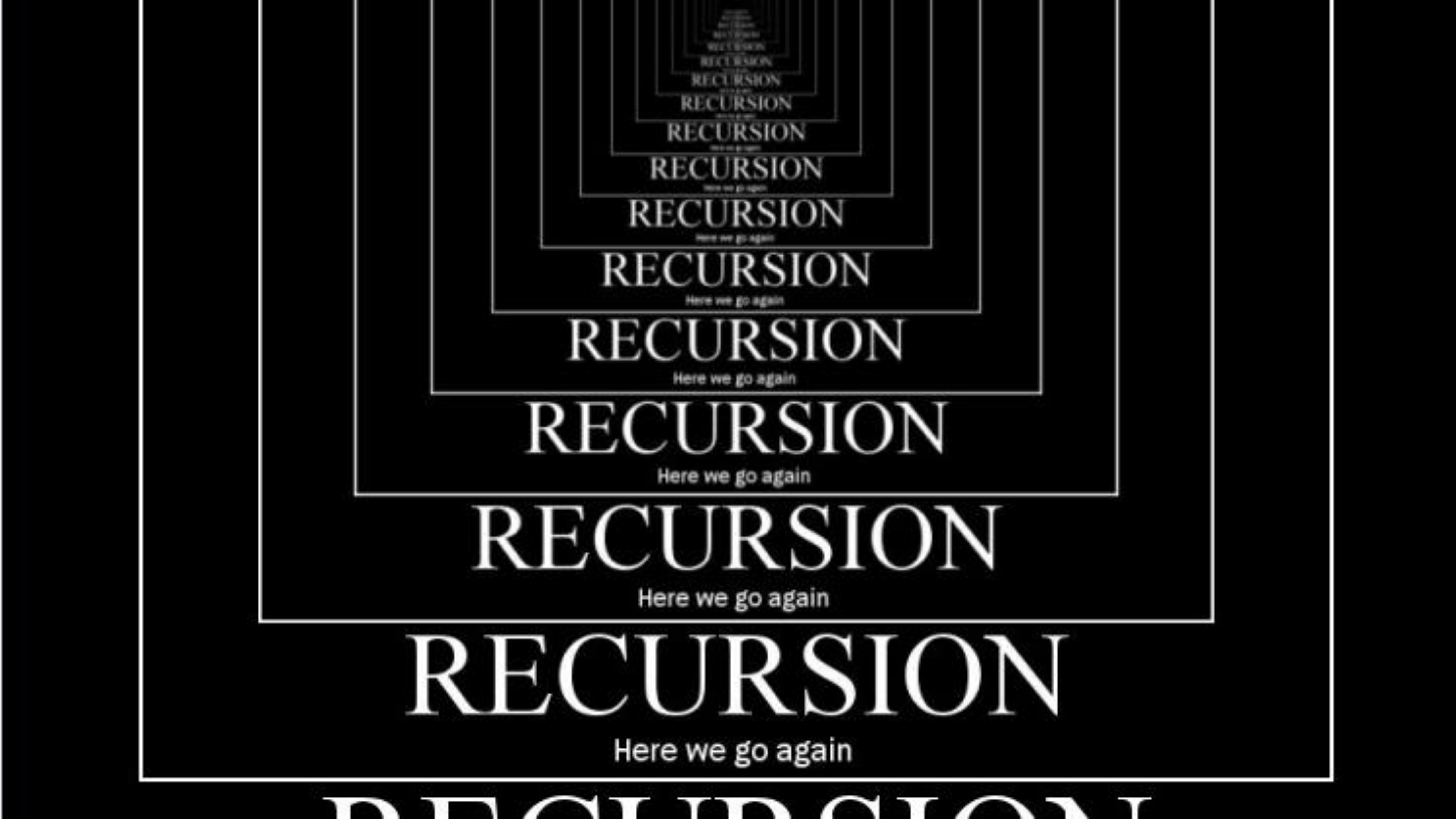

Couve-Flor 🥦

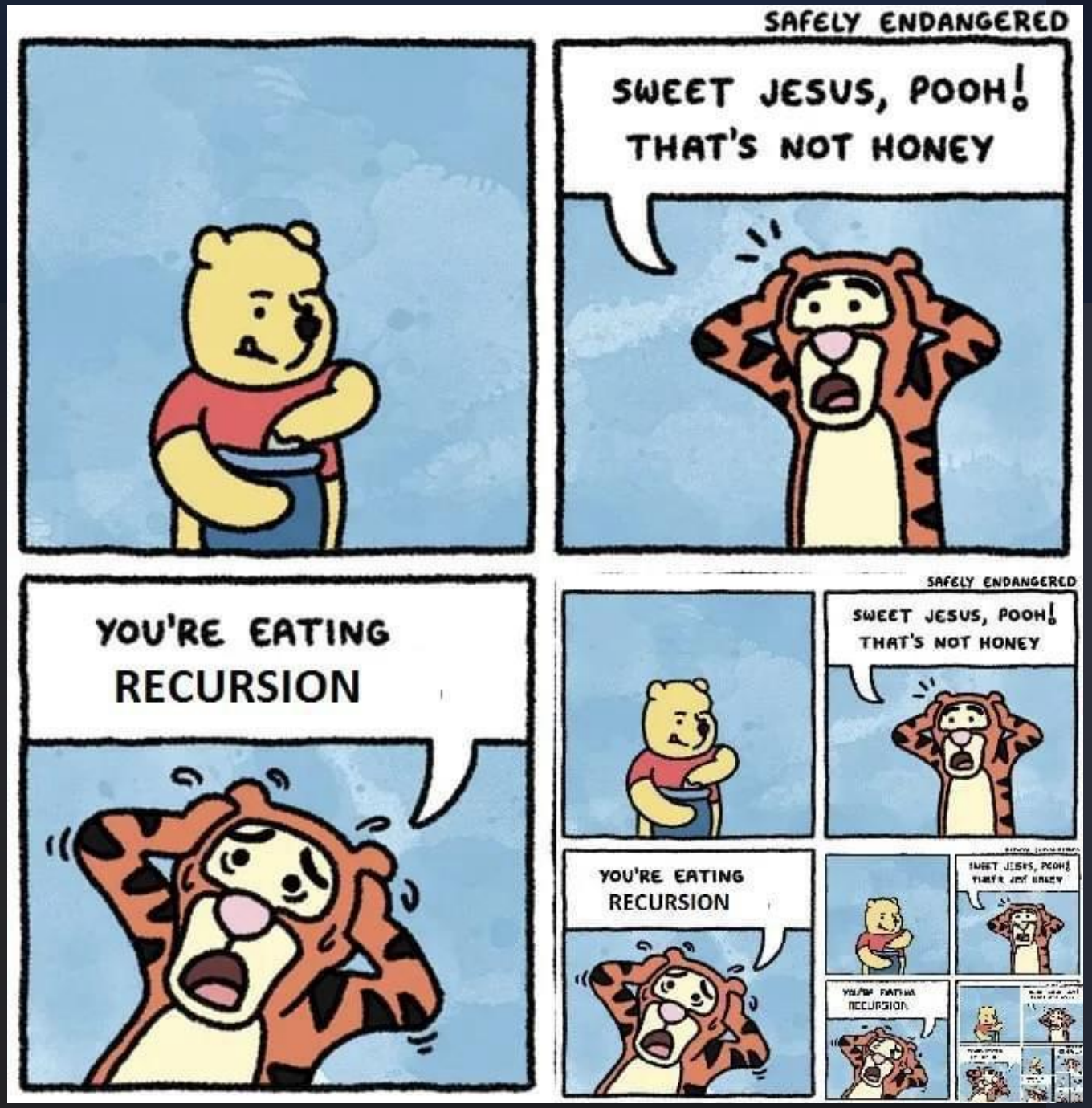
Samambaia



Recursão

- Técnica poderosa da matemática
- Permite definir um elemento em função de um caso “mais simples” dele mesmo
- Algoritmo recursivo
  - Algoritmo que chama a si mesmo, de forma direta ou indireta
  - Apropriado quando o problema a ser resolvido pode ser definido em termos recursivos
  - Exemplo: cálculo de potência ou fatorial

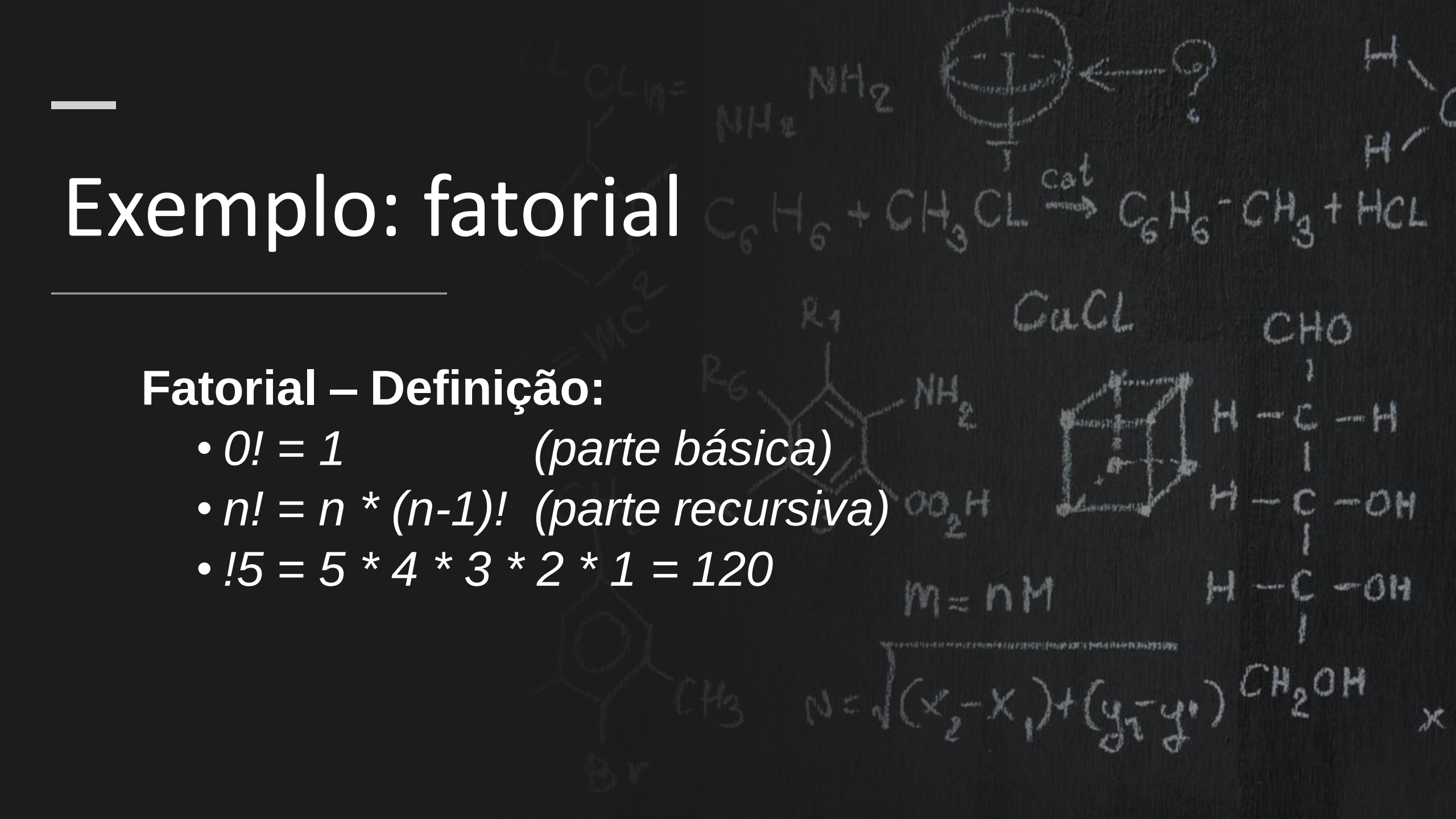

RECURSION
RECURSION
RECURSION
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again
RECURSION
Here we go again


SAFELY ENDANGERED
SWEET JESUS, POOH! THAT'S NOT HONEY
YOU'RE EATING RECURSION
SAFELY ENDANGERED
SWEET JESUS, POOH! THAT'S NOT HONEY
YOU'RE EATING RECURSION

Exemplo: fatorial

Fatorial – Definição:

- *0! = 1 (parte básica)*
- *n! = n * (n-1)! (parte recursiva)*
- *!5 = 5 * 4 * 3 * 2 * 1 = 120*

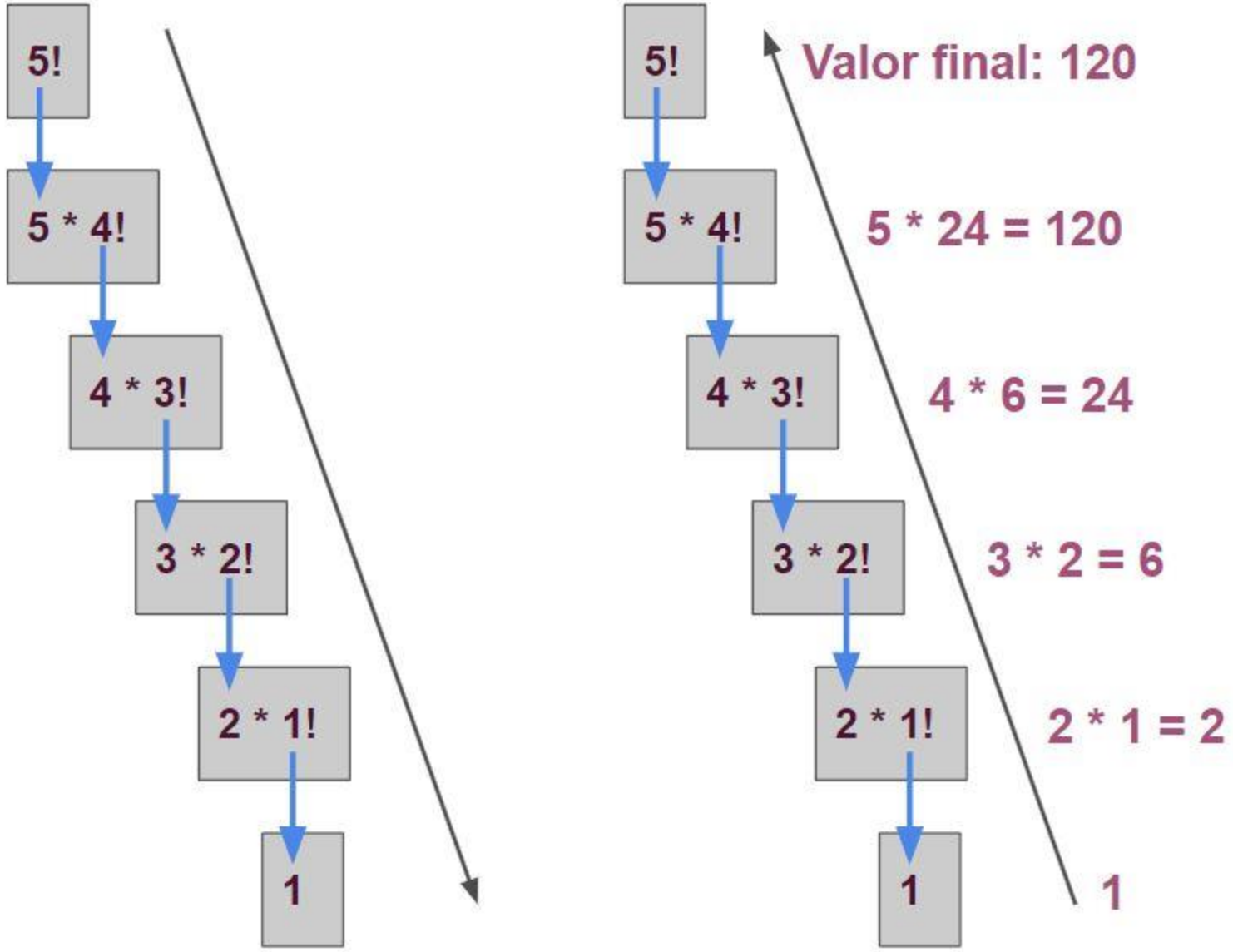


Algoritmo Fatorial

```java
public static int fatorial(int n){
    if (n == 0){
        return 1;
    }else{
        return n * fatorial(n-1);
    }
}

public static void main(String[] args) {
     System.out.println(fatorial(4));
}
```

Simulação de fatorial(5)



Algoritmo Fatorial

Os problemas resolvidos por algoritmos recursivos podem também ser resolvidos de forma iterativa (através de um loop ou laço)

```
static int fatorialIterativo(int n){
    int resultado = 1;
    while (n >= 1) {
        resultado = resultado * n;
        n--;
    }
    return resultado;
}
```

Recursão x Iteração

- Tanto iteração como recursão se baseiam em uma instrução de controle:
  - Iteração utiliza uma instrução de repetição (por exemplo: for, while ou do...while).
  - Recursão utiliza uma instrução de seleção (por exemplo: if, if...else ou switch)
- Ambas envolvem repetição:
  - A iteração utiliza explicitamente uma instrução de repetição.
  - A recursão alcança a repetição por meio de chamadas sucessivas do próprio método.

Recursão x Iteração

- Tanto uma como outra envolvem um teste de terminação:
  - A iteração termina quando a condição de continuação do loop falha.
  - A recursão termina quando um caso básico é alcançado.
- Loop infinito pode ocorrer em ambos:
  - Um loop infinito ocorre com iteração se o teste de continuação do loop nunca se tornar falso.
  - A recursão infinita ocorre se o passo de recursão não reduzir o problema sempre em uma maneira que convirja para o caso básico, ou se o caso básico não for testado.

Recursão x Iteração

- Eficiência
  - A versão iterativa normalmente é mais eficiente do que a recursiva
  - A recursão envolve várias chamadas consecutivas ao próprio algoritmo, acarretando num maior consumo de tempo e memória do que a versão iterativa
- Clareza
  - Muitas vezes, a versão recursiva apresenta uma maior clareza do que o correspondente iterativo

Exemplo: exibição de um vetor de forma recursiva

- No algoritmo exibeVetor anterior (o 1º), se invertermos as linhas 1 e 2, o algoritmo passará a exibir o vetor de forma invertida, pois:
  - primeiro chamará o método recursivamente para exibir o restante,
  - e depois exibe o elemento corrente.
- No 2º algoritmo também, se invertermos as linhas 3 e 4, o algoritmo deixará de exibir o vetor de forma invertida

Recursão indireta - Exemplo

- Recursão indireta
  - Quando um algoritmo A chama um B, que por sua vez chama novamente o algoritmo A
  - Exemplo:
    - algoritmo que verifica se um número é par e algoritmo que verifica se um número é ímpar
    - Definição tradicional de par e ímpar:
      - Número é par quando for divisível por 2
      - Número é ímpar quando não for divisível por 2
    - Definição recursiva de par e ímpar:
      - Número n>1 é par se n-1 for ímpar; 0 é par, 1 é ímpar
      - Número n>0 é ímpar se n-1 é par; 1 é ímpar, 0 é par

Agradeço
a sua atenção!

Célia Taniwaki

celia.taniwaki@sptech.school

SÃO PAULO TECH SCHOOL