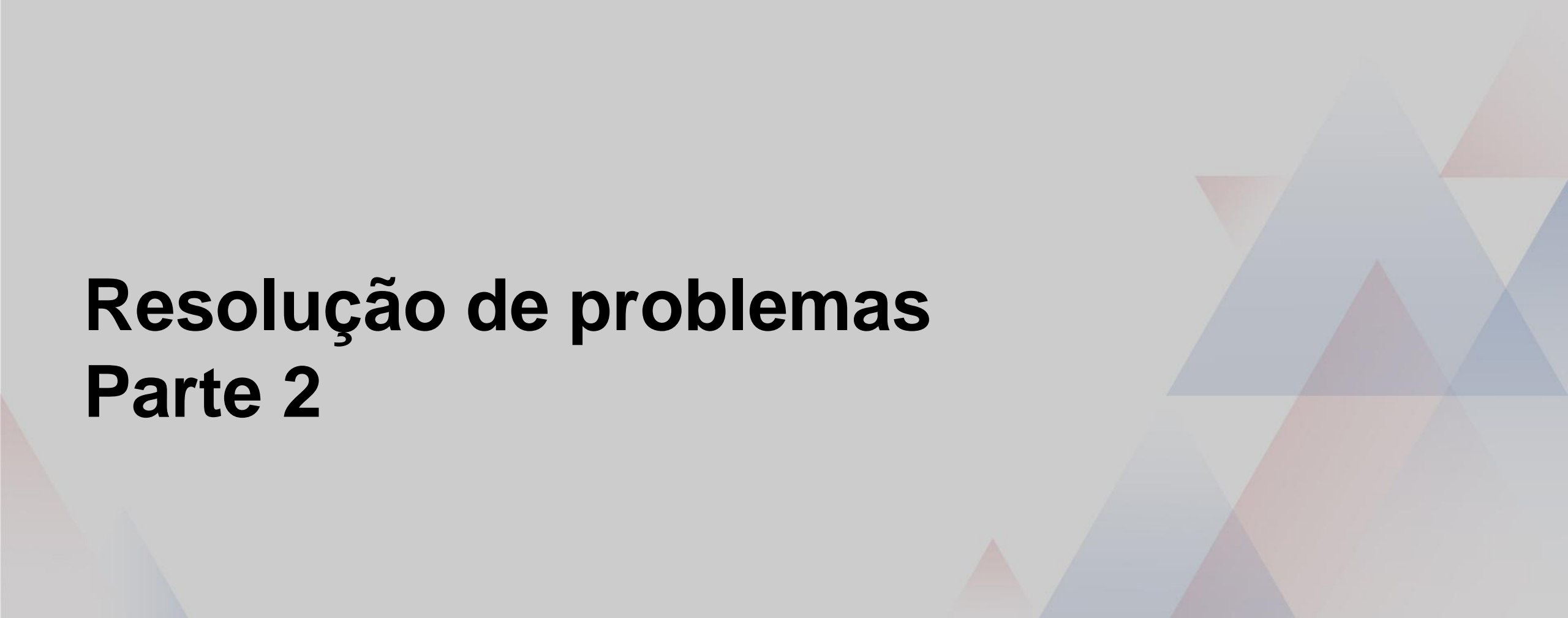


PENSAMENTO COMPUTACIONAL

Resolução de problemas

Parte 2

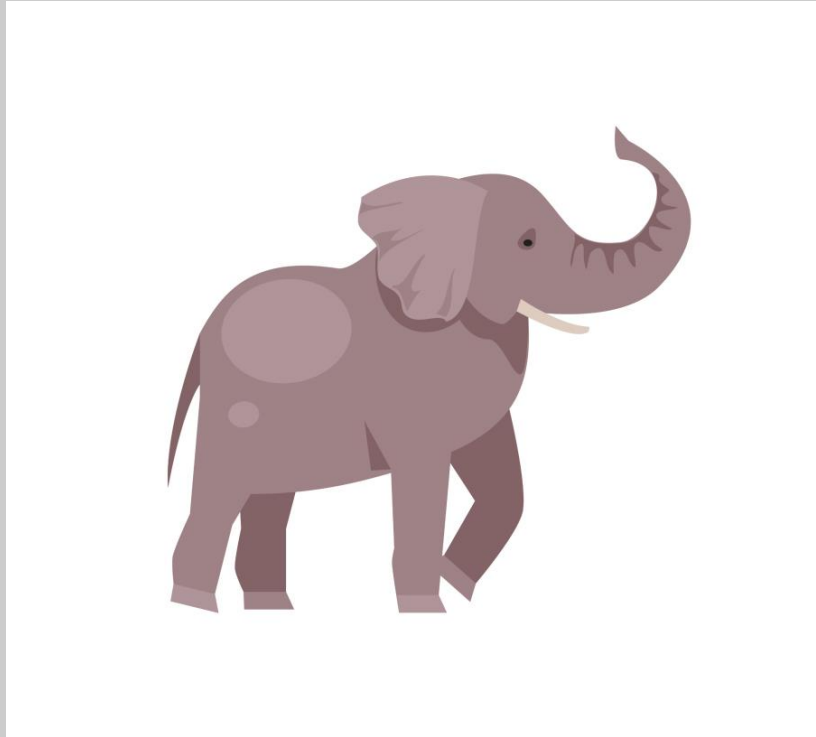


Técnicas para Construir Algoritmos

- **Técnicas que permitem desenvolver uma solução para o problema com mais eficiência e facilidade:**
 - **Decomposição:** consiste em decompor o problema em problemas menores
 - Refinamento, reuso e Recursão
 - **Generalização:** consiste em construir uma solução (algoritmo) mais genérico a partir de outro, permitindo que este novo algoritmo seja utilizado em outros contextos
 - Reconhecimento de Padrões e Reuso
 - **Transformação:** consiste em utilizar a solução de um problema para solucionar outro, através de transformação
 - Reuso

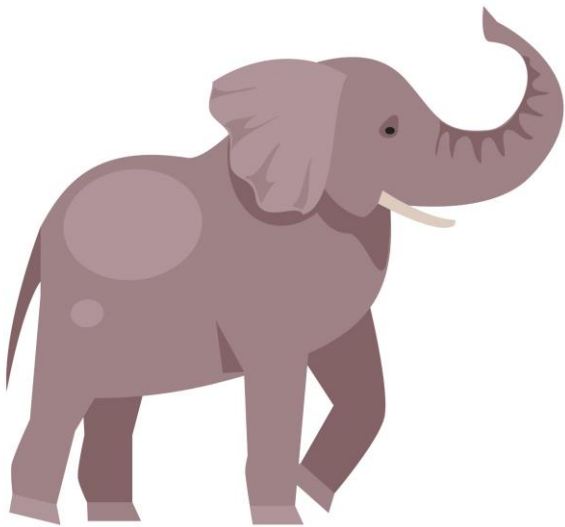
Problema do elefante

- Como você pode comer um elefante no almoço?



Problema do elefante

- Como você pode comer um elefante no almoço?



**Dando uma
mordida de cada
vez!**

Problema do elefante

- Problemas grandes e complexos são compostos por subproblemas ou tarefas menores e mais fáceis de resolver
- O processo / estratégia de identificar logicamente esses problemas menores e determinar como usar as soluções combinadas para resolver o problema maior é chamado de **Decomposição**
- Entretanto todos podem ser quebrados, divididos, decompostos em pequenas partes
- O exercício de decompor um problema é um **Método analítico**, dividir algo maior em pequenas partes
- O processo de juntar cada pequena parte novamente é um **Processo de síntese**

Tarefa do dia a dia: escovar os dentes!

- **Passos / instruções:**

1. Preparar a escova com creme dental
2. Escovar os dentes
3. Lavar a boca para extrair os resíduos da escovação
4. Lavar a escova
5. Guardar a escova e o creme dental

- **Um problema simples do cotidiano pensado em cinco partes**

- **Resolvendo cada uma delas, resolvemos o problema maior**



Problemas mais complexos

- Construir uma rodovia?
- Lançamento de um foguete?
- Cirurgia no cérebro?
- Gerir a linha de produção de um automóvel?
- Investigar a cena de um crime?
- Criar um aplicativo?

Decomposição: podem ser divididos, decompostos em pequenas partes, e cada parte pode sucessivamente ser dividida para deixar o problema mais simples.



Refinamentos Sucessivos

- É uma técnica que orienta o processo de decomposição de problemas
 1. Divida o problema em suas partes principais
 2. Analise a divisão obtida para garantir coerência
 3. Se alguma parte ainda estiver complexa, volta para 1
 4. Analise o resultado para garantir entendimento e coerência

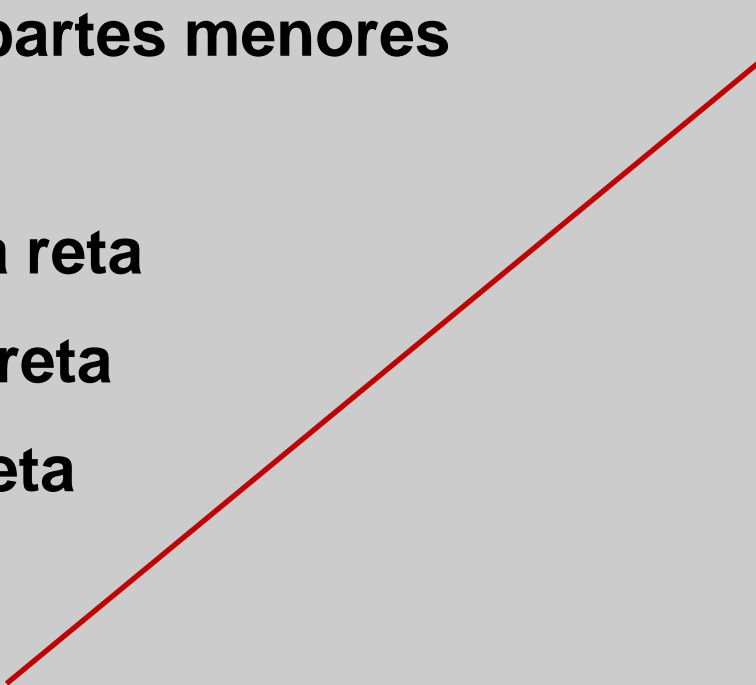
Essa técnica é comumente chamada de Top-Down

Desenhar um Quadrado

- **Dividir o problema em partes menores**
 - **Desenhar uma reta**
 - **Desenhar a segunda reta**
 - **Desenhar a terceira reta**
 - **Desenhar a quarta reta**
- **Algoritmo**
 - **Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**



Desenhar um Quadrado

- **Dividir o problema em partes menores**
 - **Desenhar uma reta**
 - **Desenhar a segunda reta**
 - **Desenhar a terceira reta**
 - **Desenhar a quarta reta**
 - **Algoritmo**
 - **Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
- 

Desenhar um Quadrado

- **Dividir o problema em partes menores**
 - **Desenhar uma reta**
 - **Desenhar a segunda reta**
 - **Desenhar a terceira reta**
 - **Desenhar a quarta reta**
- **Algoritmo**
 - **Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**



Desenhar um Quadrado

- **Dividir o problema em partes menores**
 - **Desenhar uma reta**
 - **Desenhar a segunda reta**
 - **Desenhar a terceira reta**
 - **Desenhar a quarta reta**
- **Algoritmo**
 - **Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**



Desenhar um Quadrado

- **Dividir o problema em partes menores**
 - **Desenhar uma reta**
 - **Desenhar a segunda reta**
 - **Desenhar a terceira reta**
 - **Desenhar a quarta reta**
- **Algoritmo**
 - **Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**
 - **Gire 90° sentido horário e Desenhar uma reta**



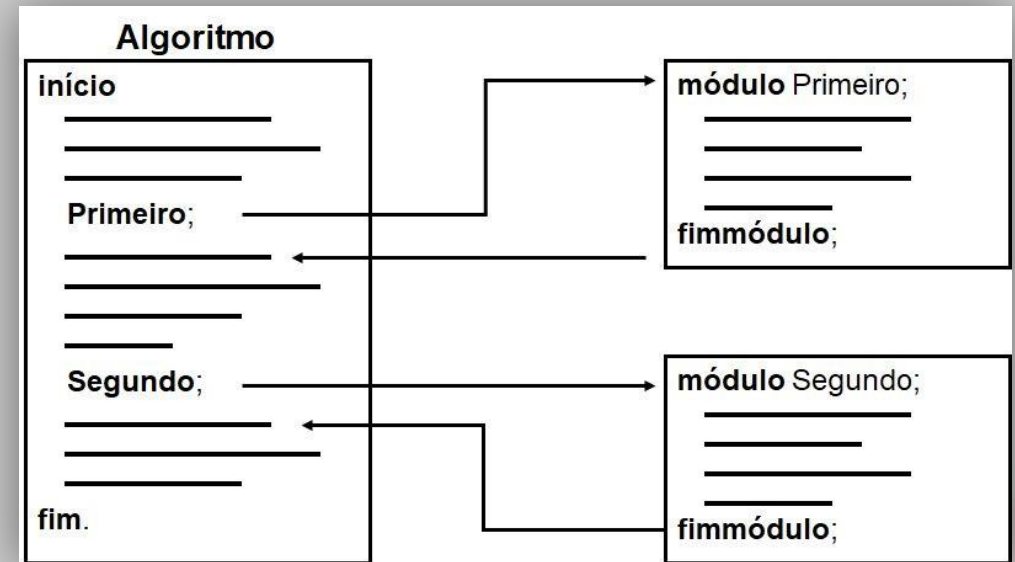
Modularização e Reuso

- Algoritmo
 - **Desenhar uma reta**
 - Gire 90º sentido horário e **Desenhar uma reta**
 - Gire 90º sentido horário e **Desenhar uma reta**
 - Gire 90º sentido horário e **Desenhar uma reta**
- Cada divisão / parte obtida com a técnica de Refinamentos Sucessivos (Decomposição) são conhecidas como Módulos, Subalgoritmos, Rotinas ou Componentes
- Vantagens: facilitar a resolução, desenvolvimento, gerenciamento, reaproveitamento (reuso) e paralelização

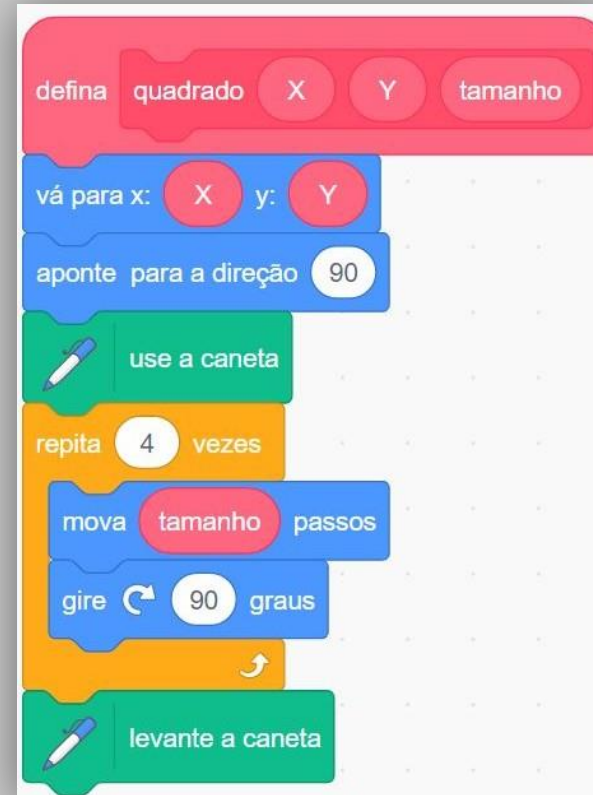
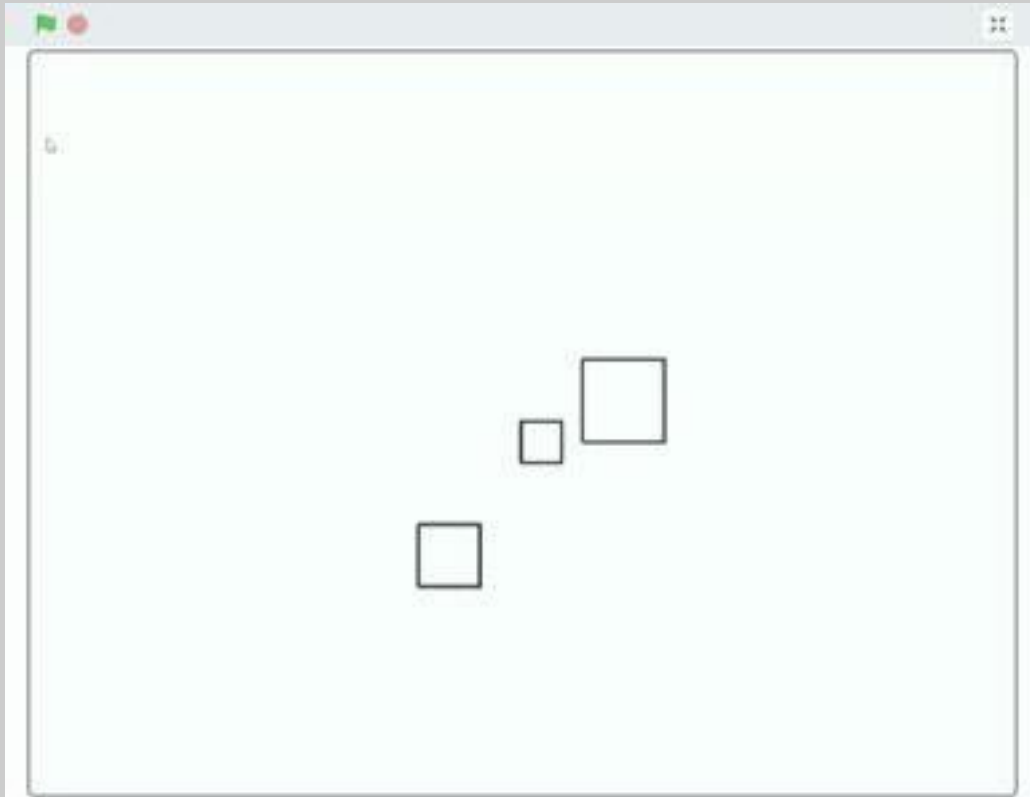


Modularização

- Um módulo ou procedimento pode ser acionado de qualquer ponto do algoritmo principal ou de outro módulo
- O acionamento de um módulo também é conhecido por **chamada** ou **ativação**
- Quando ocorre uma chamada, o fluxo de execução passa para o módulo chamado
- Quando se conclui a execução do módulo chamado o controle retorna para o módulo chamador



Modularização no Scratch



<https://scratch.mit.edu/projects/375153792>

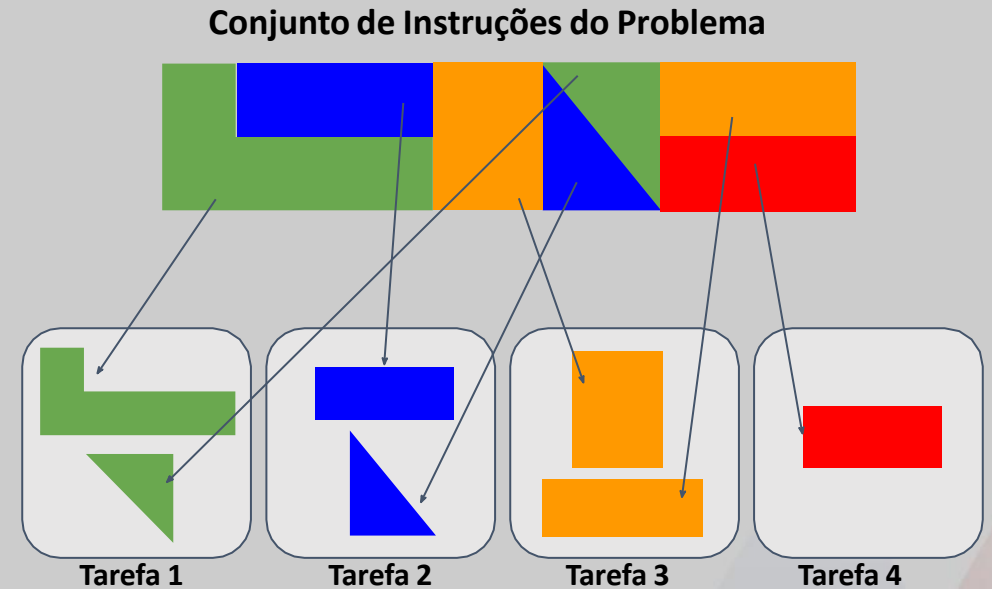
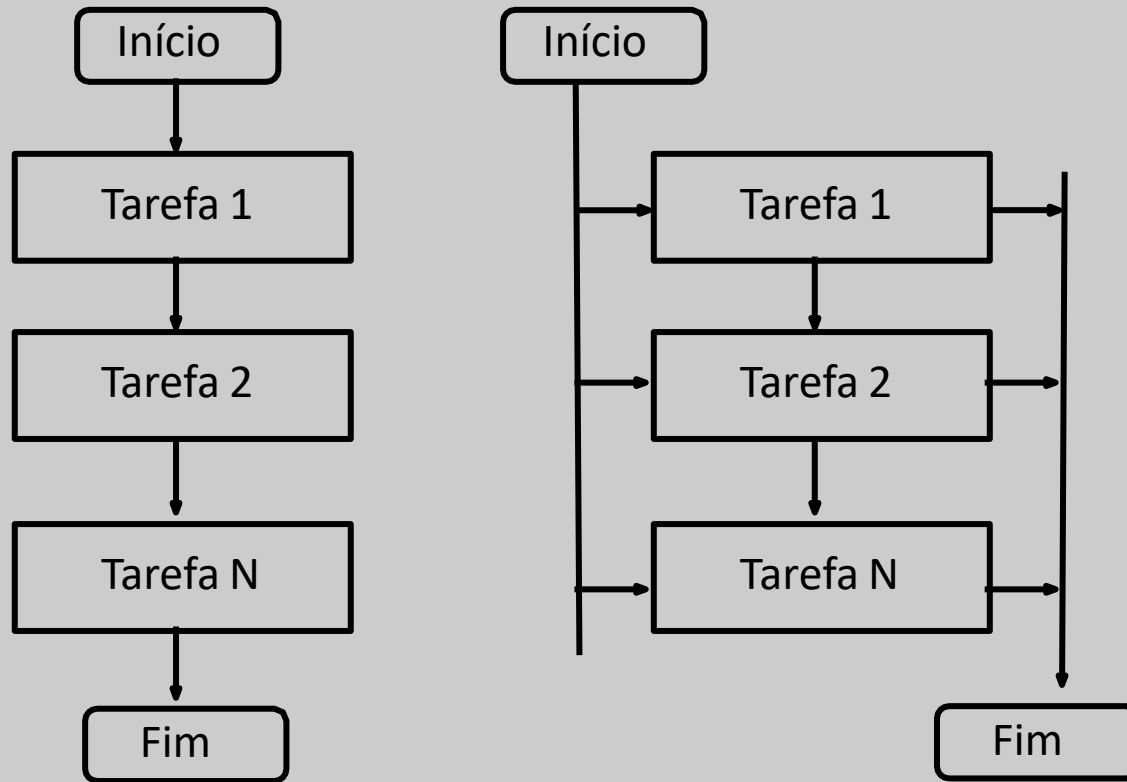
Paralelização/Paralelismo

- Organizar recursos para, simultaneamente, realizar tarefas para alcançar um objetivo comum
- O processo de paralelização de um problema, requer que este seja realizado ao mesmo tempo por diferentes unidades de processamento (CPU)
- Vantagens:
 - Reduzir o tempo necessário para solucionar um problema
 - Resolver problemas mais complexos e de maior dimensão



Execução Paralela

- Fluxograma - Algoritmo Sequencial X Fluxograma Algoritmo Paralelo



- Cientistas pesquisam métodos para dividir problemas de forma que estes possam ser resolvidos por computadores trabalhando em paralelo

Paralelização/Paralelismo

- Eventos “paralelos” são fundamentais em alguns programas, pois, como num jogo ou filme, muitas vezes é importante que várias coisas aconteçam ao mesmo tempo
- De maneira visual, basta entender que não precisamos adicionar os comandos em sequência, mas em blocos separados, com ações conectadas ou independentes
- Execução concorrente: o "paralelismo" de fato, não necessariamente existe no hardware e é simulado usando uma intercalação de processos

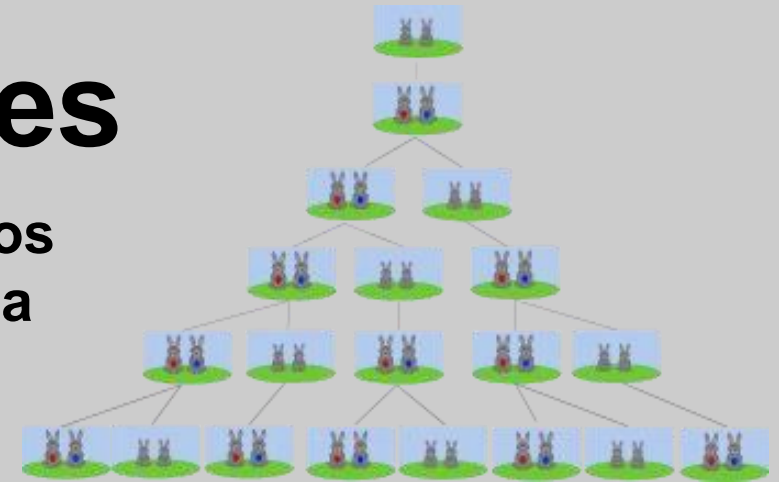


Generalização

- Associada à identificação de padrões, semelhanças e conexões, e à exploração desses recursos
- Resolver rapidamente novos problemas com base em soluções anteriores
- Reconhecimento de padrões nos dados usados e nos processos - estratégias que estão sendo usadas
- Algoritmos podem ser adaptados para resolver toda uma classe de problemas semelhantes
- Vantagens em encontrar padrões:
 - Economia: adaptar soluções, ou partes de soluções, para que se apliquem a toda uma classe de problemas
 - Criar modelos preemptivos, ou seja, modelos que são capazes de ajudar a prever ações futuras
 - Transferir ideias e soluções de uma área problemática para outra

Reconhecimento de Padrões

- A Sequência de Fibonacci é uma sucessão de números na qual, cada termo subsequente corresponde à soma dos dois anteriores
- Resultados das etapas anteriores são reutilizados
- Os cálculos podem ser representados através de uma fórmula que generaliza os demais (padrão)



Recursividade

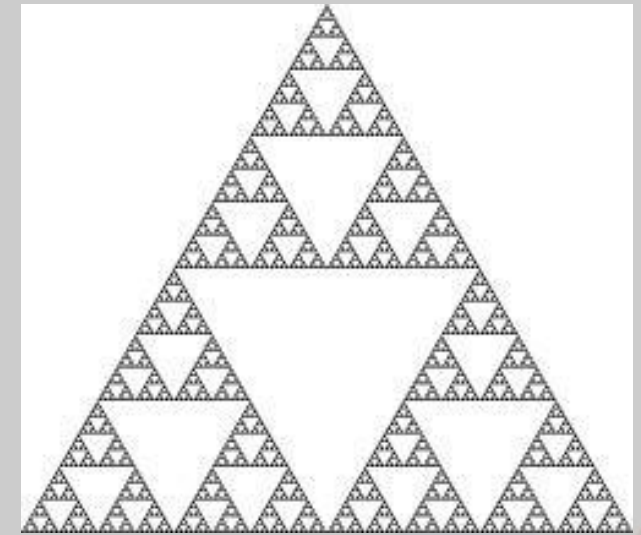
- A recursão é uma forma interessante de resolver problemas, pois o divide em problemas menores de **mesma natureza**
- Um objetivo é dito recursivo se pode ser definido em termos de si próprio
- Consiste em duas partes:
 - O caso trivial, cuja solução é conhecida (condição de parada)
 - Um método geral que reduz o problema a um ou mais problemas menores de mesma natureza



“Para fazer iogurte, você precisa de leite e de um pouco de iogurte!!”

Recursividade

- **Vantagens:**
 - Existem problemas complexos em que a solução é inerente recursiva
 - Descreve os algoritmos de forma mais clara e concisa
- **Desvantagens:**
 - Reduz o desempenho de execução devido ao tempo para gerenciamento de chamadas
- **Exemplos:**
 - $\text{Fibonacci}(n) = \text{Fibonacci}(n-1) + \text{Fibonacci}(n-2)$
 - $0 \Rightarrow \text{se } n = 0$
 - $1 \Rightarrow \text{se } n = 1$
 - $\text{Fatorial}(n) = n * \text{Fatorial}(n - 1)$
 - $1 \Rightarrow \text{se } n = 1$



[Triângulo de Sierpinski](#)

Reticulada geométrica

Algoritmo iterativo (não recursivo)

- Soma N primeiros números inteiros = $N = 5$

$$\text{Soma (5)} = 1 + 2 + 3 + 4 + 5 = 15$$

Programa Soma

Var i,n,soma: inteiro

Início

ESCREVA "Informe o valor de N:"

leia n

i = 1

soma = 0

enquanto (i <= n) faça

soma = soma + i

i = i + 1

fim enquanto

imprima 'Soma dos N primeiros números inteiros...: ', soma

Fim

Algoritmo Recursivo

- Soma N primeiros números inteiros = 5

$$\text{Soma (5)} = 1 + 2 + 3 + 4 + 5 = 15$$

$$\rightarrow \text{Soma (5)} = 5 + \text{Soma (4)}$$

$$\text{Soma (4)} = 1 + 2 + 3 + 4 = 10$$

$$\rightarrow \text{Soma (4)} = 4 + \text{Soma (3)}$$

$$\text{Soma (3)} = 1 + 2 + 3 = 6$$

$$\rightarrow \text{Soma (3)} = 3 + \text{Soma (2)}$$

$$\text{Soma (2)} = 1 + 2 = 3$$

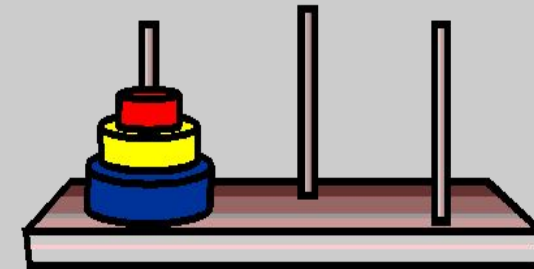
$$\rightarrow \text{Soma (2)} = 2 + \text{Soma (1)}$$

$$\text{Soma (1)} = 1 = 1$$

$$\rightarrow \text{Soma (1)} = 1 \text{ (solução trivial)}$$

```
funcao soma(n:inteiro):inteiro
início
    se n = 1 então
        retorne 1
    senão
        retorne n + soma(n -1)
fimse
fimfuncao
```

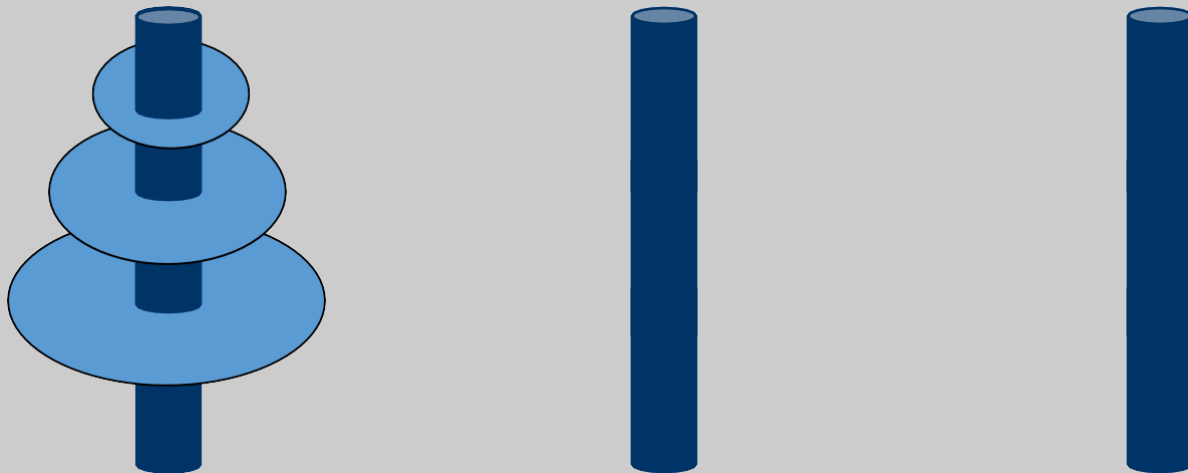
Desafio: Torre de Hanói



- É um quebra-cabeças, divulgado pelo matemático francês Édouard Lucas, em 1883
- A utilização do jogo contribui para o desenvolvimento do raciocínio lógico e para a capacidade de resolução de problemas
- Objetivo do jogo:
 - Mover todos os discos de uma haste para outra, utilizando o menor número possível de movimentos e respeitando as regras
- Regras:
 - Passar todos os discos para o último eixo com a ajuda do eixo central, de modo que no momento da transferência o pino de maior diâmetro nunca fique sob o de menor diâmetro

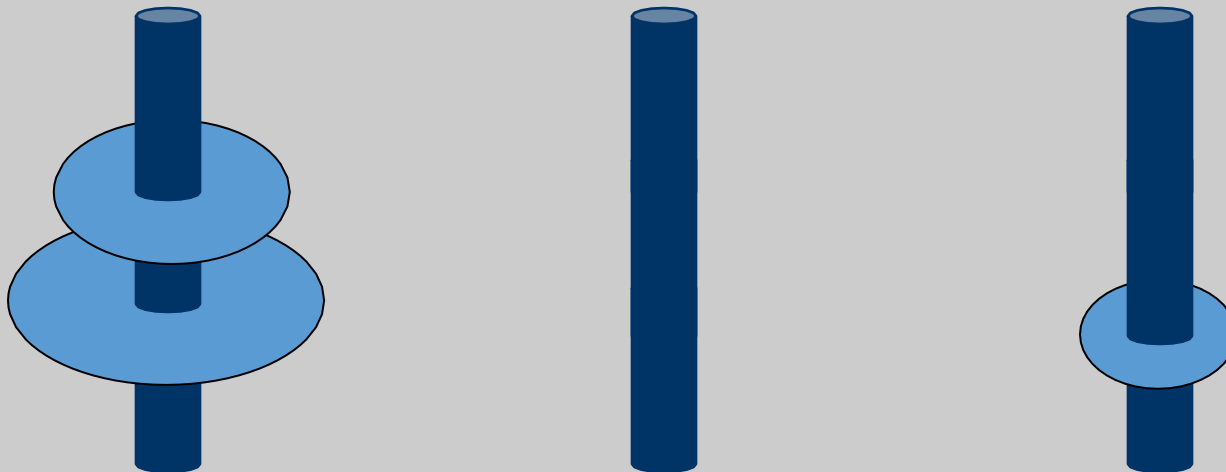
Desafio: Torre de Hanói

- O desafio é calcular o número de movimentos mínimos para resolver a Torre de Hanói com qualquer quantidade de discos
- Estado Inicial:



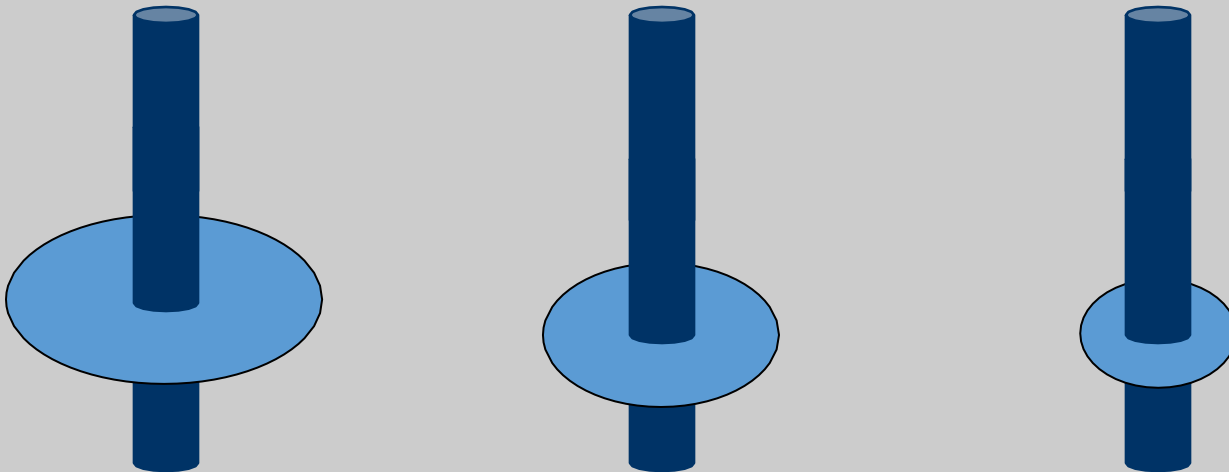
Algoritmo: Torre de Hanói

- **Movimento 1:**
mova disco menor para terceiro eixo



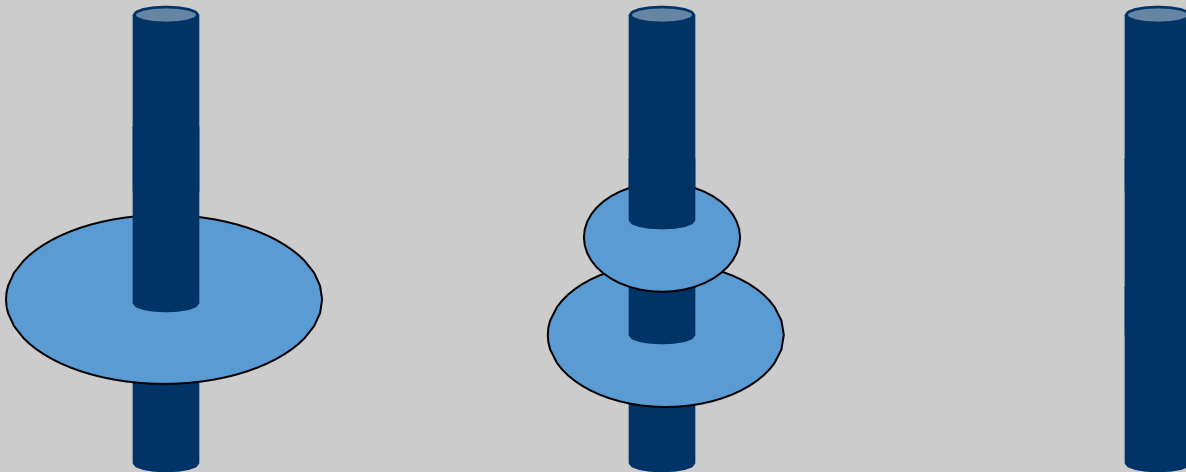
Algoritmo: Torre de Hanói

- **Movimento 2:**
mova disco médio para segundo eixo



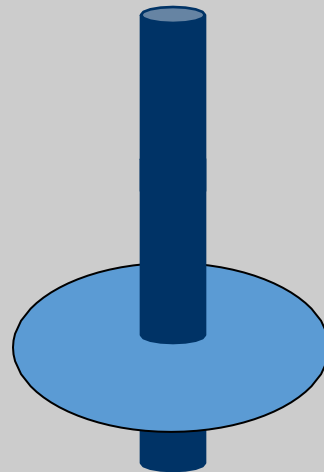
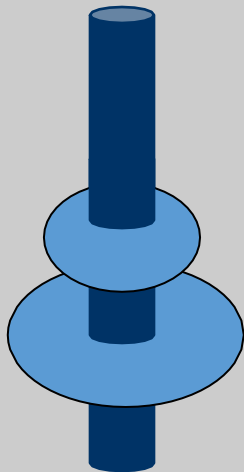
Algoritmo: Torre de Hanói

- **Movimento 3:**
mova disco menor para segundo eixo



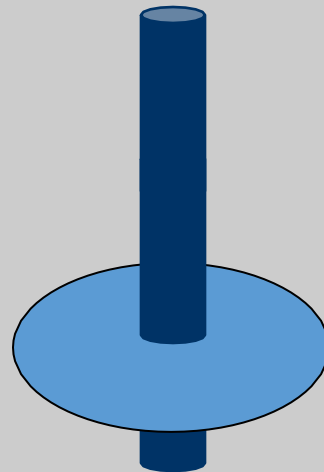
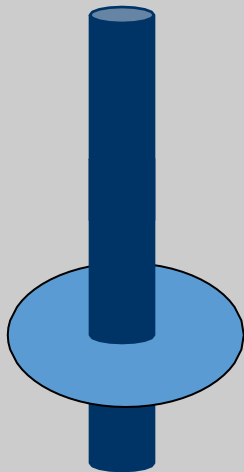
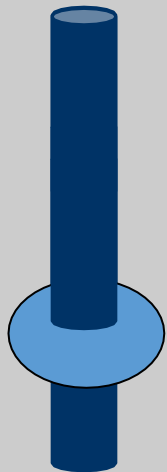
Algoritmo: Torre de Hanói

- **Movimento 4:**
mova disco maior para terceiro eixo



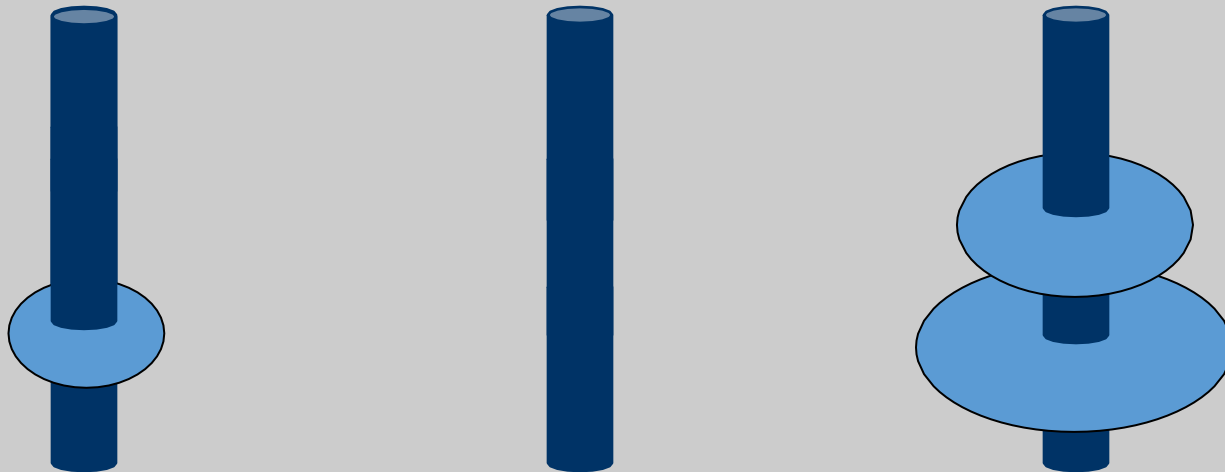
Algoritmo: Torre de Hanói

- **Movimento 5:**
mova disco menor para primeiro eixo



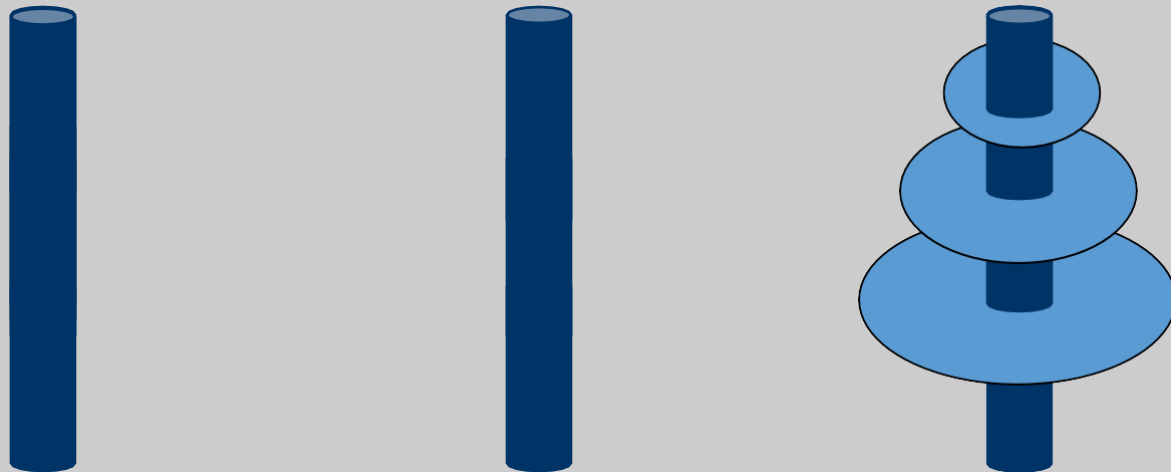
Algoritmo: Torre de Hanói

- **Movimento 6:**
mova disco médio para terceiro eixo



Algoritmo: Torre de Hanói

- **Movimento 7:**
mova disco menor para terceiro eixo



Algoritmo: Torre de Hanói

- **Sequência de Movimentos:**
 - **Passo 1: mova disco menor para terceiro eixo**
 - **Passo 2: mova disco médio para segundo eixo**
 - **Passo 3: mova disco menor para segundo eixo**
 - **Passo 4: mova disco maior para terceiro eixo**
 - **Passo 5: mova disco menor para primeiro eixo**
 - **Passo 6: mova disco médio para terceiro eixo**
 - **Passo 7: mova disco menor para terceiro eixo**

Considerações

- **Não existe um algoritmo para construir algoritmos**
- **A criação de um algoritmo é um exercício de criatividade e experiência**
- **A capacidade de solucionar problemas complexos é uma das habilidades mais importantes dos atuais e futuros profissionais**

PENSAMENTO COMPUTACIONAL

Resolução de problemas

Parte 2

