Processamento da Informação

Fabrício Olivetti de França

02 de Fevereiro de 2019

UFABC

CMCC
Centro de Matemática, Computação e Cognição
Center for Mathematics, Computation and Cognition

Topics

1. Recursão

Recursão

Indução Matemática

Em bases matemáticas vocês aprenderam sobre indução matemática:

- Provamos um caso base, trivial.
- Provamos que se para um caso k é verdade, para $k + 1$ também é.

Indução Matemática

Prove que:

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$$

Indução Matemática

Caso base: $n = 1$

$$1 = \frac{1(1+1)}{2} = 1$$

Dado que é valido para $n = k$, prove que também é válido para $n = k + 1$:

$$1 + 2 + \ldots + k = \frac{k(k+1)}{2}$$

$$\frac{k(k+1)}{2} + (k+1) = \frac{k(k+1) + 2k + 2}{2} = \frac{k^2 + 3k + 2}{2} = \frac{(k+1)(k+2)}{2}$$

Recursão

A recursão é uma abstração que pode ser usada na programação para simplificar a criação de algoritmos. Já vimos isso no algoritmo de Euclides!!

Euclides

```python
def mdc(x, y):
    if x==0: return y  # caso base 1
    if y==0: return x  # caso base 2
    return mdc(y, x%y) # inducao
```

Recursão

A recursão na computação se caracteriza por uma função chamar ela mesma para computar um problema menor.

Fatorial

Vamos considerar a definição de fatorial:

$$n! = n \cdot (n-1)!$$

Qual o caso base do fatorial?

Fatorial

Vamos considerar a definição de fatorial:

$$n! = n \cdot (n-1)!$$

Qual o caso base do fatorial?

$$0! = 1! = 1$$

Exercício

Implemente o algoritmo para calcular o fatorial recursivamente.

Somatória

Podemos pensar em uma somatória como uma função recursiva:

$$s(0) = 0$$

$$s(1) = 1$$

$$s(n) = n + s(n-1)$$

Somatória

Defina a função recursiva para fazer a somatória dos números de 1 até n.

Laços de Repetição vs Recursão

Todo laço de repetição pode ser transformado em uma recursão, considere o seguinte algoritmo:

```
def dec2bin(x):
    b = "0"
    while x != 0:
        r = x%2
        b = str(r) + b
        x = x//2
    return b
```

Laços de Repetição vs Recursão

O caso base geralmente é descrito na condição de parada do laço!

Tentem transformar esse algoritmo em recursivo!

Laços de Repetição vs Recursão

```
def dec2bin(x):
    if x==0:
        return ""
    r = x%2
    return dec2bin(x//2) + str(r)
```

Fibonacci

Lembrando da sequência de Fibonacci, qual seria o caso base?

Fibonacci

Lembrando da sequência de Fibonacci, qual seria o caso base?

$F(0) = 0$

$F(1) = 1$

Fibonacci

E o caso geral?

Fibonacci

E o caso geral?

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

Fibonacci

Defina a função recursiva de Fibonacci.

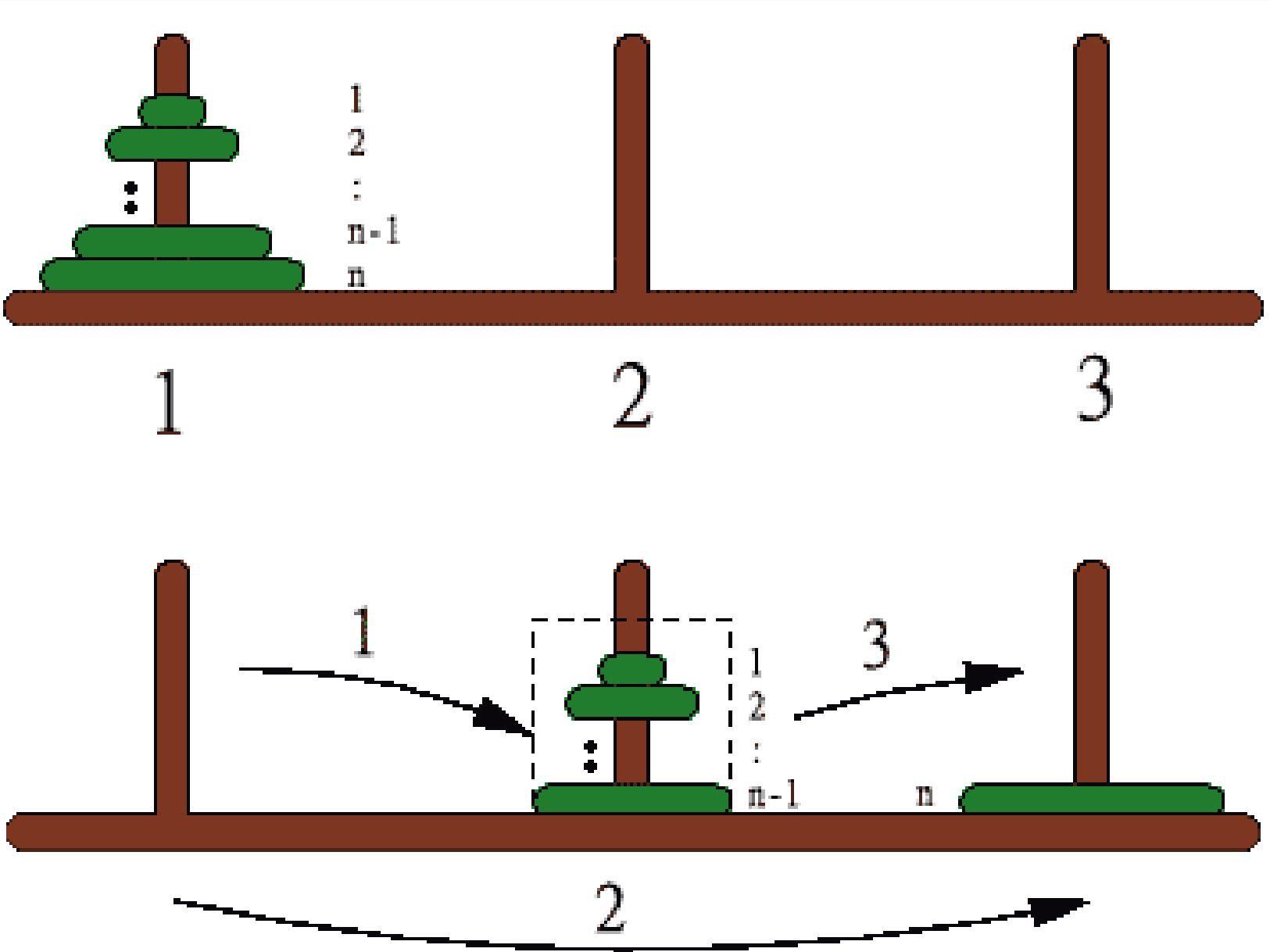
Torre de Hanói

A Torre de Hanói é um problema em que você possui três pinos denominados origem, transição e destino, e n discos empilhados no pino de origem. Cada disco com um tamanho diferente.

Torre de Hanói

O objetivo do jogo é transportar todos os discos para o pino de destino movendo um disco de cada vez e de tal forma que um disco menor nunca fica embaixo de um disco maior.

Torre de Hanói



Torre de Hanói

Qual o caso trivial para esse problema?

Torre de Hanói

Com $n = 1$ basta mover o disco da origem ao destino!

Torre de Hanói

Dado o seguinte estado:

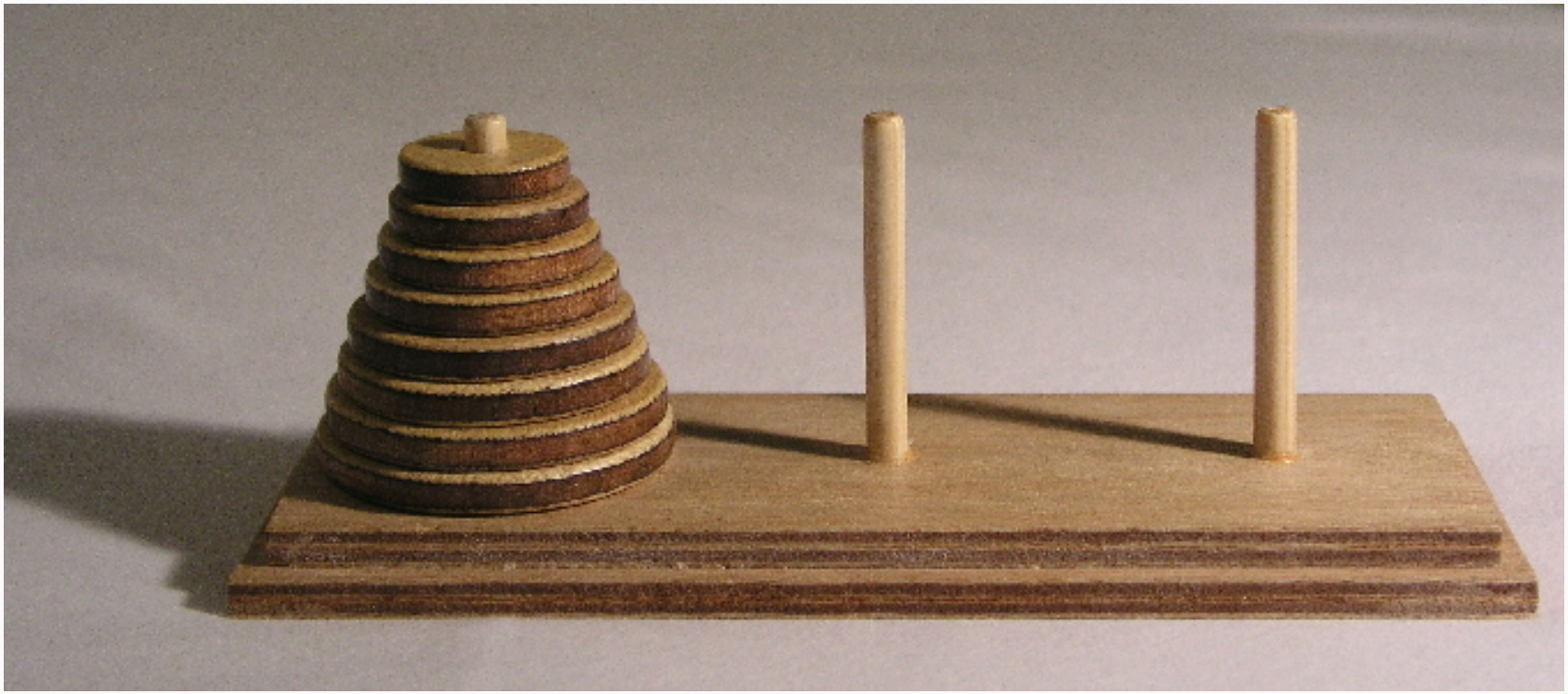


Figura 2: FONTE: desconhecida

O que deve ser feito para chegarmos no caso trivial?

Torre de Hanói

Transportar todos os $n - 1$ discos para o pino de transição! Se resolvermos isso, chegamos ao trivial!

Torre de Hanói

Vamos construir o nosso algoritmo:

```
def hanoi(n, orig, trans, dest):
    if n==1:
        print("mover de " + orig + " para " + dest + ".")
    else:
        hanoi(n-1, orig, dest, trans)
        print("mover de " + orig + " para " + dest + ".")
```

Torre de Hanói

É só isso?

```
hanoi(3, "A", "B", "C")
mover de A para C.
mover de A para B.
mover de A para C.
```

Torre de Hanói

Torre de Hanói Online

Torre de Hanói

Uma vez que resolvemos o problema para $n - 1$ e movemos o disco atual para o destino, devemos resolver o problema de levar os $n - 1$ do auxiliar para o destino!

Torre de Hanói

Vamos construir o nosso algoritmo:

```
def hanoi(n, orig, trans, dest):
    if n==1:
        print("mover de " + orig + " para " + dest + ".")
    else:
        hanoi(n-1, orig, dest, trans)
        print("mover de " + orig + " para " + dest + ".")
        hanoi(n-1, trans, orig, dest)
```

Torre de Hanói

```
hanoi(3, "A", "B", "C")
mover de A para C.
mover de A para B.
mover de C para B.
mover de A para C.
mover de B para A.
mover de B para C.
mover de A para C.
```

Torre de Hanói

Vamos tentar novamente:

Torre de Hanói Online

Torre de Hanói

Desafio: tente encontrar uma solução para o problema da Torre de Hanói sem usar recursão!

Não é impossível (porém muito difícil)

Recursão

A recursão é uma poderosa ferramenta de abstração para criação de algoritmos.

Existem algumas linguagens de programação (paradigma funcional) em que os laços de repetição são completamente substituídos pela recursão.