19. Recursividade		1 Recursão
DIM0321		2 Exemplos clássicos
2015.1		③ Torre de Hanoi
DIM0321 19. Recursividade	2015.1 1 / 53	DIM0321 19. Recursividade 2015.1 2 / 53  Matemática
		TVI de l'indered
<ol> <li>Recursão</li> <li>Exemplos clássicos</li> <li>Torre de Hanoi</li> </ol>		Definir um problem <b>recursivamente</b> = resolver o problema a partir da solução de uma instância menor do problema. $\begin{cases} 1 & \text{se } n = 0 \end{cases}$
		$n! = egin{cases} 1 &  ext{se } n = 0 \ n*(n-1)! &  ext{sen} ilde{ ext{so}} \end{cases}$
DIM0321 19. Recursividade	2015.1 3 / 53	DIM0321 19. Recursividade 2015.1 4 / 53

Outline

## Em programação

Uma forma de especificar uma função (sub-rotina) que é definida através de sua própria definição.

$$fib(n) = \begin{cases} x & x \leq 1 \\ fib(n-1) + fib(n-2) & n \geq 1 \end{cases}$$

A recursão pode ser usada também como procedimento ou mecanismo de alteração de dados.

- Recursão
- 2 Exemplos clássicos
- 3 Torre de Hanoi

DIM0321 19. Recursividade 2015.1 5 / 53 DIM0321 19. Recursividade 2015.1 6 / 5

### Fatorial

```
1  #include <stdio.h>
2  #include <assert.h>
3  unsigned fact(unsigned n)
4  {
5     if (n == 0) return 1;
6     return n * fact(n - 1);
7  }
8     int main(void)
11  {
12     unsigned n;
13     scanf("%u", &n);
14     printf("fact(%u) = %u\n", n, fact(n));
15     return 0;
16  }
```

## Sequência de Fibonacci

```
1 | #include <stdio.h>
 2 #include <assert.h>
 3 unsigned fib(unsigned n)
        if (n == 0) return 0;
        else if (n == 1) return 1;
        else return fib(n - 1) + fib(n - 2);
11 int main(void)
12 {
13
        unsigned n;
14
        scanf("%u", &n);
15
16
        printf("fib(\frac{u}{u}) = \frac{u}{n}, n, fib(n));
        return 0;
17 }
```

M0321 19. Recursividade 2015.1 7 / 53 DIM0321 19. Recursividade 2015.1 8 / 5:

## Ordenação

#### Problema

Dado um conjunto  $A = \{a_1, a_2, a_3, \cdots, a_n\}$ , ordenar A tal que para  $1 \le i < n, a_i < a_{i+1}$ 

#### Ideia

- Se  $\{a_2,\cdots,a_n\}$  estiver ordenado basta inserir  $a_1$  na sua posição ordenada em  $\{a_2,\cdots,a_n\}$
- Se  $\{a_2, \dots, a_n\}$  não estiver ordenado, aplica o mesmo princípio (mas o problema está menor)
- Um conjunto com apenas 1 elemento está obrigatoriamente ordenado

#### Parada

- A execução de uma função recursiva provoca a chamada a própria função
- Quando parar?

#### A base!

Definir e não esquecer o caso base

- ① O corpo da função deve incluir ao menos um caso onde o valor de retorno não depende de uma chamada recursiva. Este é o **caso base**.
- ② Se o cálculo de f(a) chama recursivamente f(b), então b tem que estar mais próximo do caso-base do que a

Recursão mútua

```
1 #include <stdbool.h>
 2 #include <assert.h>
 3 #include <stdio.h>
 5 bool even(unsigned n);
 6 bool odd(unsigned n);
 8 bool even(unsigned n)
        printf("even %d\n", n);
        if (n == 0) return true;
12
        else return odd(n - 1);
13 }
14
16 bool odd(unsigned n)
17
18
        printf("odd %d\n", n);
19
        if (n == 0) return false;
20
        else return even(n - 1);
21 }
22
23
   int main(void)
24
        int n:
        scanf("%d", &n);
        if (n < 0) {
             n = -n;
```

- 1 Recursão
- 2 Exemplos clássico
- 3 Torre de Hanoi

10 Becureivi

2015 1 1

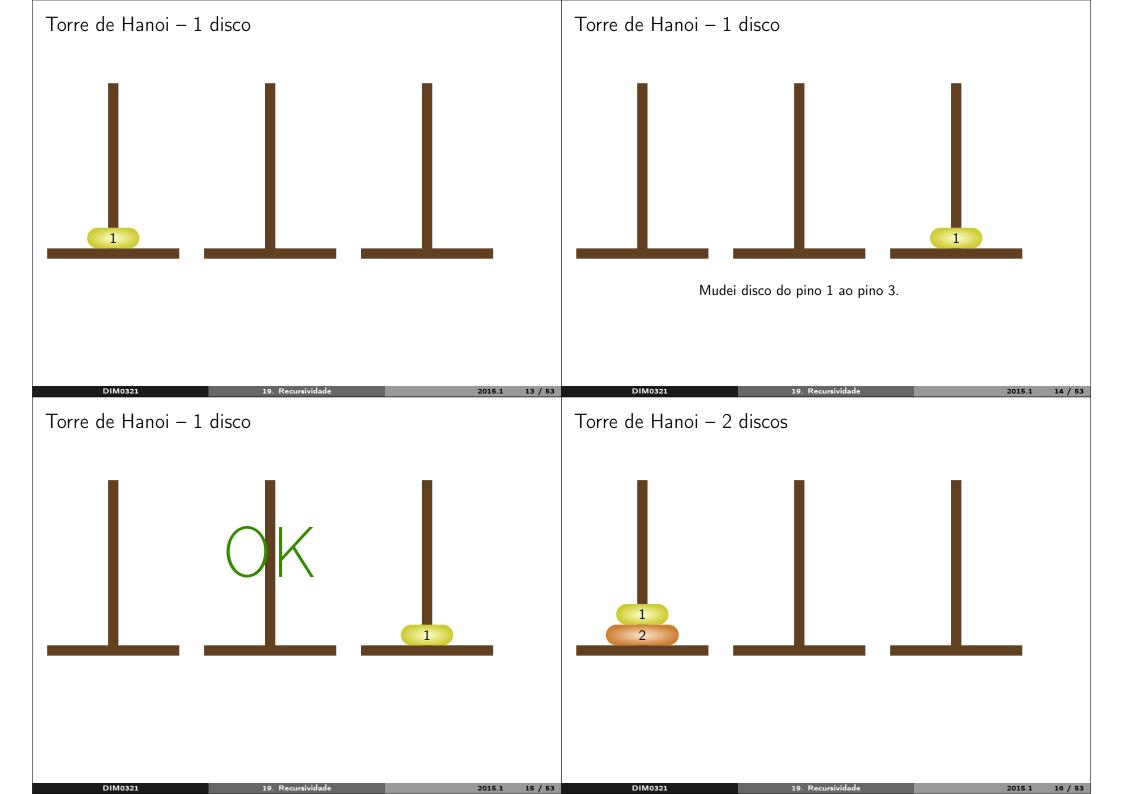
DIM0321

10 Pecursivida

2015.1

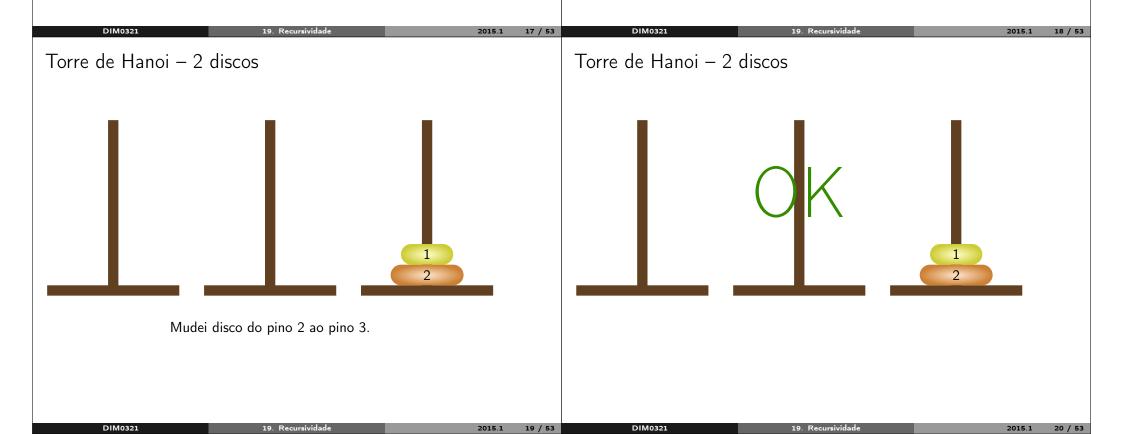
12 /

2 / 53

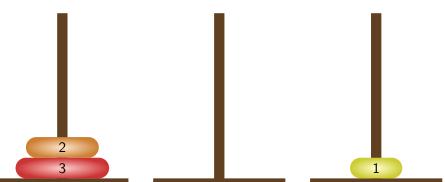


Torre de Hanoi – 2 discos

Mudei disco do pino 1 ao pino 2. Mudei disco do pino 1 ao pino 3.



Torre de Hanoi – 3 discos

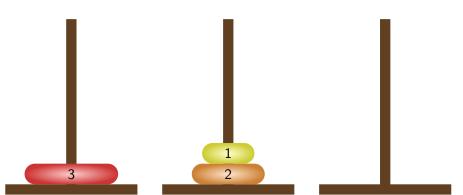


Mudei disco do pino 1 ao pino 3.

Torre de Hanoi – 3 discos

Torre de Hanoi – 3 discos

Mudei disco do pino 1 ao pino 2.

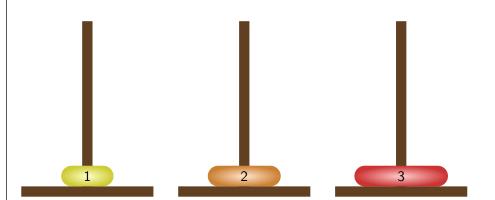


Mudei disco do pino 3 ao pino 2.

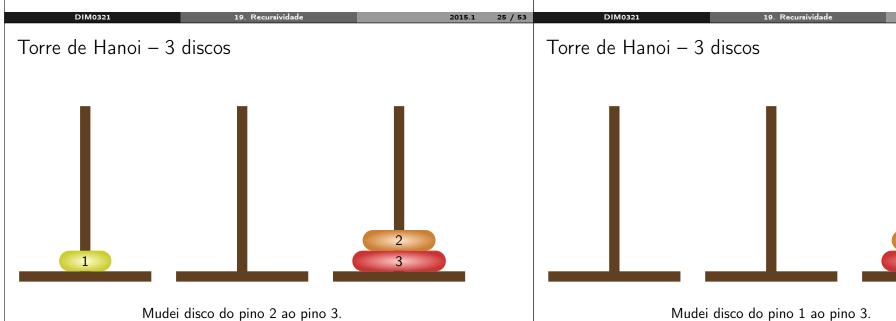
DIM0321 19. Recursividade 2015.1 23 / 53 DIM0321 19. Recursividade 2015.1 24 / 5

Mudei disco do pino 1 ao pino 3.

Torre de Hanoi – 3 discos

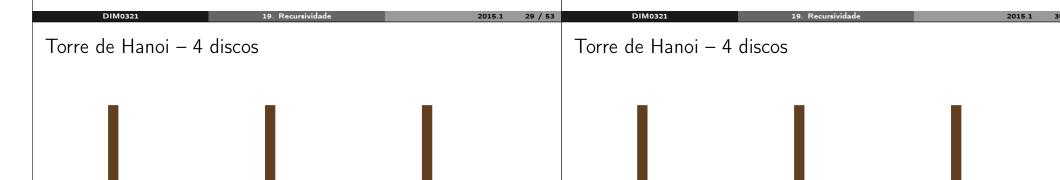


Mudei disco do pino 2 ao pino 1.





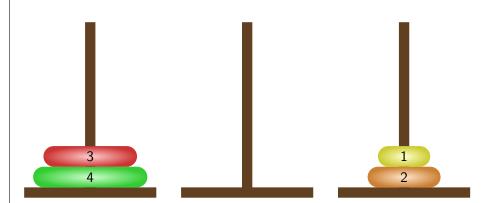




Mudei disco do pino 1 ao pino 2.

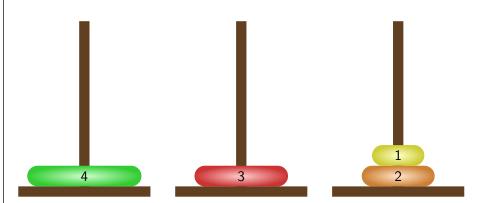
Mudei disco do pino 1 ao pino 3.

DIM0321 19. Recursividade 2015.1 31 / 53 DIM0321 19. Recursividade 2015.1 32 /



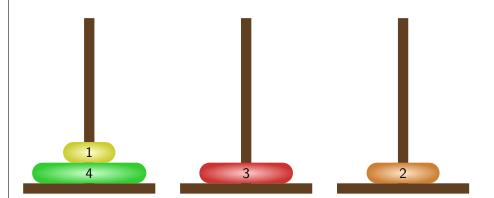
Mudei disco do pino 2 ao pino 3.

Torre de Hanoi – 4 discos

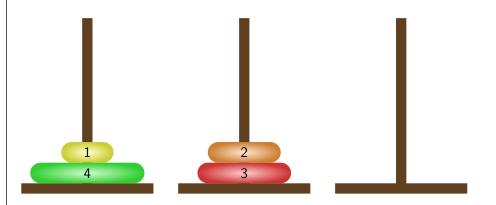


Mudei disco do pino 1 ao pino 2.



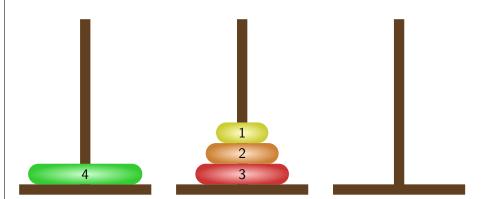


Mudei disco do pino 3 ao pino 1.



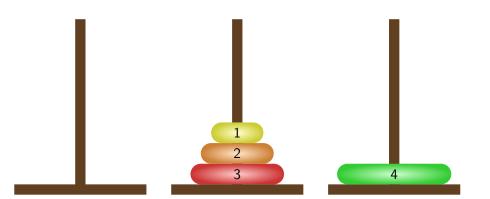
Mudei disco do pino 3 ao pino 2.

DIM0321 19. Recursividade 2015.1 35 / 53 DIM0321 19. Recursividade 2015.1 36 / 5:

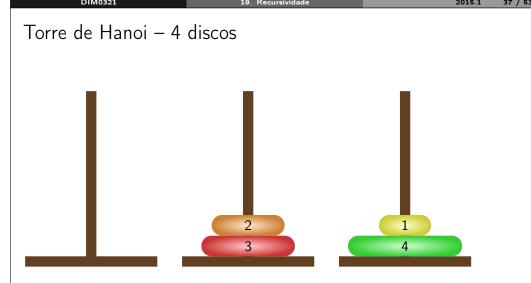


Mudei disco do pino 1 ao pino 2.

Torre de Hanoi – 4 discos

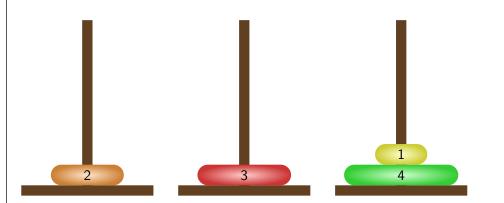


Mudei disco do pino 1 ao pino 3.



Mudei disco do pino 2 ao pino 3.

Torre de Hanoi – 4 discos



Mudei disco do pino 2 ao pino 1.

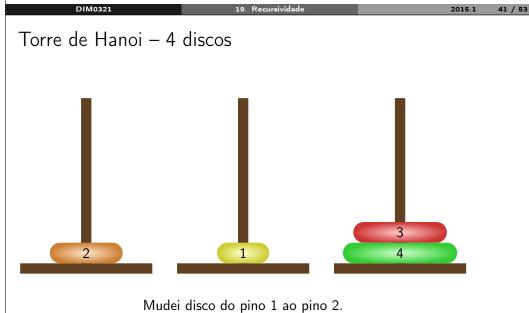
DIM0321 19. Recursividade 2015.1 39 / 53 DIM0321 19. Recursividade 2015.1 40 / 53

Mudei disco do pino 3 ao pino 1.

Torre de Hanoi – 4 discos



Mudei disco do pino 2 ao pino 3.



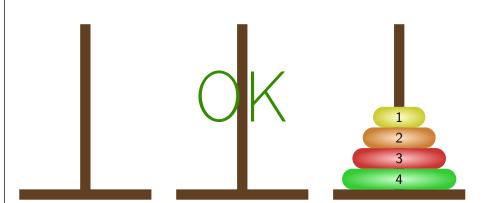
Torre de Hanoi – 4 discos

Mudei disco do pino 1 ao pino 3.

19 Recursividade 2015 1 43 / 53 DIM0321 19 Recursividade 2015 1 44 / 5

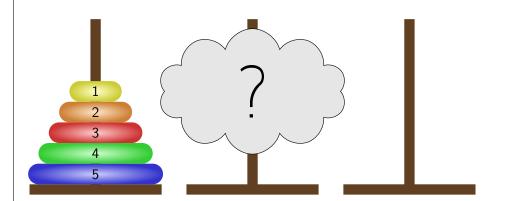
Mudei disco do pino 2 ao pino 3.

Torre de Hanoi – 4 discos



DIM0321 19. Recursividade 2015.1 45 / 53 DIM0321 19. Recursividade 2015.1 46 / 53

Tower of Hanoi – 5 Disc



### História

• Proposto pelo matemático Édouard Lucas em 1883.

#### Assunto

- Considerando uma torre de *n* discos de circunferência crescente, inicialmente empilhados em ordem crescente de circunferência sobre uma entre três varas.
- O objetivo é transferir a torre inteira para outra vara, movendo apenas um disco de cada vez, e nunca colocando um disco maior sobre um menor.

DIM0321 19. Recursividade 2015.1 47 / 53 DIM0321 19. Recursividade 2015.1 48 / 9

## Resolução programática

Definir uma sub-rotina que imprime as instruções para resolver o quebra-cabeça:

- Ter como parâmetro o número de discos
- Imprimir uma lista de movimentos: "mover X para Y" onde  $1 \le X, Y \le 3$ .

### Exemplo, para 2 discos:

- mover 1 para 2
- mover 1 para 3
- mover 2 para 3

Solução imperativa

Para mais tarde!

### Solução recursiva

```
1 #include <stdio.h>
 2 #include <assert.h>
   int nmoves; // Init = 0;
 6 void move_disc(char orig, char dest)
7 {
8 9
10 }
        printf("%d: moving %c -> %c\n", nmoves, orig, dest);
11
12 void move(int n, char orig, char dest, char aux)
13 {
14
15
        if (n == 1) move_disc(orig, dest);
16
             move(n - 1, orig, aux, dest);
17
             move_disc(orig, dest);
18
             move(n - 1, aux, dest, orig);
19
20
21 }
22
23 int main(void)
24 {
25
        int n:
26
        printf("Tower of Hanoi\n");
        printf("How many discs ( > 0)? ");
        scanf("%d", &n);
        assert(n > 0);
        nmoves = 0;
```

Referências

### Precisões

[?] 5.1 - 5.4

[?] 10

# Perguntas ?



http://dimap.ufrn.br/~richard/dim0321

DIM0321 19. Recursividade 2015.1 53 / 5