

MAC0122 Princípios de Desenvolvimento de Algoritmos

Edição 2013

Administração

Página da disciplina: aulas, cadastro, eps ...

<http://paca.ime.usp.br/>

Chave de inscrição: grauna-2013

Exercício programa 1: disponível na página

MAC0122

MAC0122 Princípios de Desenvolvimento de Algoritmos é uma disciplina introdutória em:

- ▶ projeto, correção e eficiência de algoritmos e
- ▶ estruturas de dados

AULA 1

Livros

Nossa referência básica é o livro
PF = Paulo Feofiloff,
Algoritmos em linguagem C,



Este livro é baseado no material do sítio
Projeto de Algoritmos em C.

Outros livros são

S = Robert Sedgewick,
Algorithms in C, vol. 1

CLRS = Cormen-Leiserson-Rivest-Stein,
Introductions to Algorithms

MAC0122

MAC0122 combina técnicas de

- ▶ programação
- ▶ correção de algoritmos (relações invariantes)
- ▶ análise da eficiência de algoritmos e
- ▶ estruturas de dados elementares

que nasceram de aplicações cotidianas em ciência da computação.

Pré-requisitos

O pré-requisito oficial de **MAC0122** é

- ▶ **MAC0110** Introdução à Computação.

Costuma ser conveniente ter cursado

- ▶ **MAT0138** Álgebra I para Computação

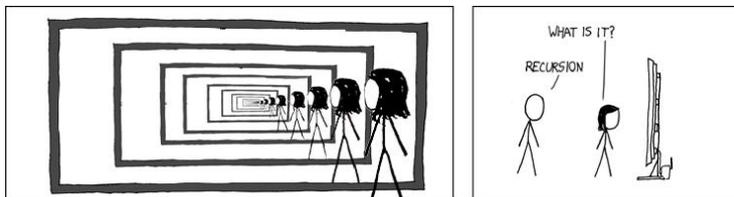
Localização

MAC0122 é um primeiro passo para

- ▶ **MAC0323** Estruturas de Dados
- ▶ **MAC0328** Algoritmos de Grafos
- ▶ **MAC0338** Análise de Algoritmos

Entretanto, várias outras disciplina se apoiam em **MAC0122**.

Recursão



Fonte: <http://xkcdsw.com/1105>

PF 2.1, 2.2, 2.3 S 5.1

<http://www.ime.usp.br/~pf/algoritmos/aulas/recu.html>

Principais tópicos

Alguns dos tópicos de **MAC0122** são:

- ▶ recursão;
- ▶ busca em um vetor;
- ▶ busca (binária) em vetor ordenado;
- ▶ listas encadeadas;
- ▶ listas lineares: filas e pilhas;
- ▶ algoritmos de enumeração;
- ▶ busca de palavras em um texto;
- ▶ algoritmos de ordenação: bubblesort, heapsort, mergesort,...; e

Tudo isso regado a muita **análise de eficiência de algoritmos e invariantes**.

Pausa para nossos comerciais

- ▶ **Encontro do BCC**: de 16 a 22 de agosto

Dojo de Love2D, Iniciação científica, Intercâmbio, Desenvolvimento de Software no mundo real, Como o Google oferece infraestrutura para manter seus serviços, Computação Natural, ...

<http://www.ime.usp.br/~encontrobcc/2013/>

- ▶ **XVII Maratona de Programação**: 17 de agosto

<http://www.ime.usp.br/~cef/XVIImaratona/>

- ▶ **Página do BCC**:

<http://bcc.ime.usp.br/>

Recursão

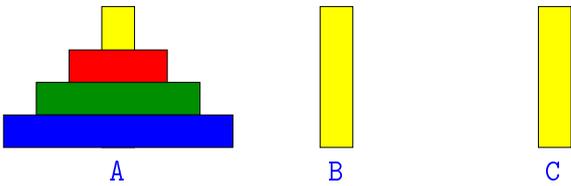
“To understand recursion, we must first understand recursion.”

–folclore

“Para fazer uma função recursivo é preciso ter fé.”

–Siang Wu Song

Torres de Hanoi

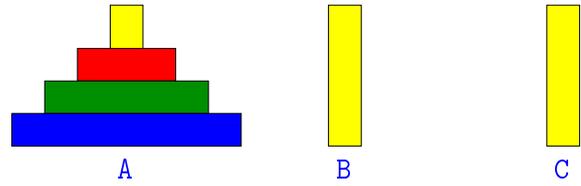


Desejamos transferir n discos do pino A para o pino C usando o pino B como auxiliar e repetindo as regras:

- ▶ podemos mover apenas um disco por vez;
- ▶ nunca um disco de diâmetro maior poderá ser colocado sobre um disco de diâmetro menor.

◀ ▶ ◂ ◃ ▹ ▸ 🔍 ↺

Torres de Hanoi

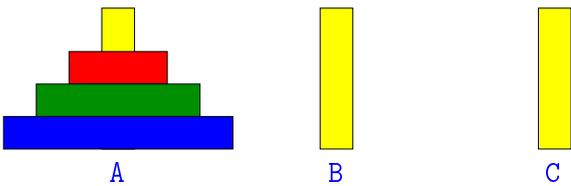


Denotaremos por $\text{Hanoi}(n,A,B,C)$ o problema de transferir n discos do pino A para o pino C usando o pino B como auxiliar

Como resolver $\text{Hanoi}(n,A,B,C)$?

◀ ▶ ◂ ◃ ▹ ▸ 🔍 ↺

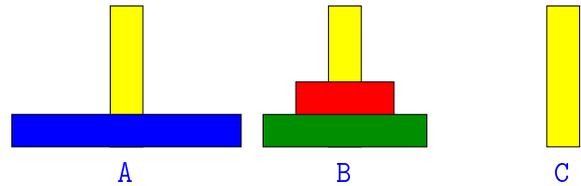
Idéia



Posso não saber qual o primeiro movimento, mas é fácil saber qual é o **movimento do meio**.

◀ ▶ ◂ ◃ ▹ ▸ 🔍 ↺

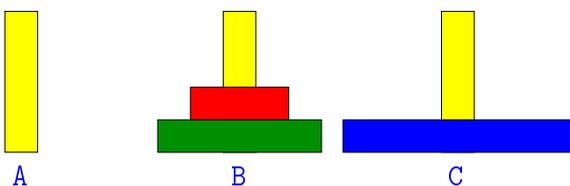
Idéia



Posso não saber qual o primeiro movimento, mas é fácil saber qual é o **movimento do meio**.

◀ ▶ ◂ ◃ ▹ ▸ 🔍 ↺

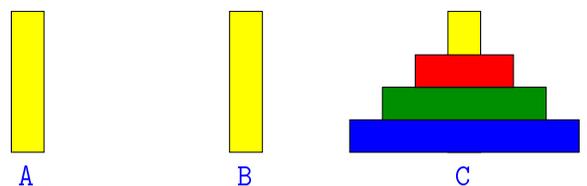
Idéia



Posso não saber qual o primeiro movimento, mas é fácil saber qual é o **movimento do meio**.

◀ ▶ ◂ ◃ ▹ ▸ 🔍 ↺

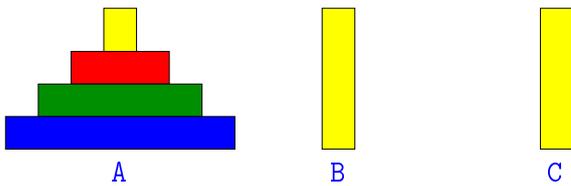
Idéia



Posso não saber qual o primeiro movimento, mas é fácil saber qual é o **movimento do meio**.

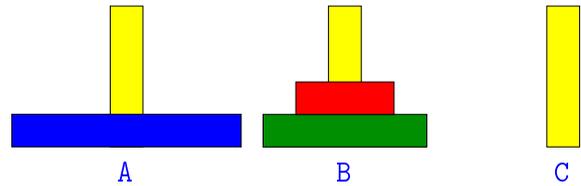
◀ ▶ ◂ ◃ ▹ ▸ 🔍 ↺

Solução



Para resolver $\text{Hanoi}(n, A, B, C)$ basta:

Solução



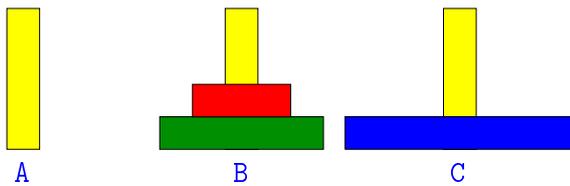
Para resolver $\text{Hanoi}(n, A, B, C)$ basta:

1. resolver $\text{Hanoi}(n-1, A, C, B)$

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ 🔍

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ 🔍

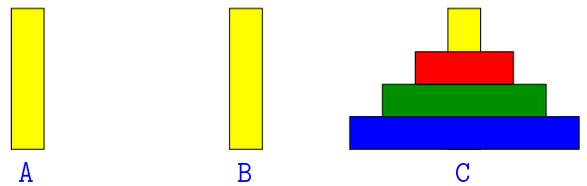
Solução



Para resolver $\text{Hanoi}(n, A, B, C)$ basta:

1. resolver $\text{Hanoi}(n-1, A, C, B)$
2. mover o disco n de A para C

Solução



Para resolver $\text{Hanoi}(n, A, B, C)$ basta:

1. resolver $\text{Hanoi}(n-1, A, C, B)$
2. mover o disco n de A para C
3. resolver $\text{Hanoi}(n-1, B, A, C)$

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ 🔍

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ 🔍

Solução

Para resolver $\text{Hanoi}(n, A, B, C)$ basta:

1. resolver $\text{Hanoi}(n-1, A, C, B)$
2. mover o disco n de A para C
3. resolver $\text{Hanoi}(n-1, B, A, C)$

E daí?

Solução

Para resolver $\text{Hanoi}(n, A, B, C)$ basta:

1. resolver $\text{Hanoi}(n-1, A, C, B)$
2. mover o disco n de A para C
3. resolver $\text{Hanoi}(n-1, B, A, C)$

E daí?

Reduzimos o problema com n discos para 2 problema com $n-1$ disco!

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ 🔍

◀ ▶ ⏪ ⏩ ⏴ ⏵ ⏶ ⏷ 🔍

Solução

Para resolver $\text{Hanoi}(n, A, B, C)$ basta:

1. resolver $\text{Hanoi}(n-1, A, C, B)$
2. mover o disco n de A para C
3. resolver $\text{Hanoi}(n-1, B, A, C)$

E daí?

Reduzimos o problema com n discos para 2 problema com $n-1$ disco!

Paramos de reduzir quando soubermos resolver o problema. Por exemplo, sabemos resolver

$\text{Hanoi}(0, \dots, \dots, \dots)$

$\text{hanoi}(3, 'A', 'B', 'C')$

- 1: mova o disco 1 do pino A para o pino C.
- 2: mova o disco 2 do pino A para o pino B.
- 3: mova o disco 1 do pino C para o pino B.
- 4: mova o disco 3 do pino A para o pino C.
- 5: mova o disco 1 do pino B para o pino A.
- 6: mova o disco 2 do pino B para o pino C.
- 7: mova o disco 1 do pino A para o pino C.

$\text{hanoi}(7, 'A', 'B', 'C')$

- 1: mova o disco 1 do pino A para o pino C.
- 2: mova o disco 2 do pino A para o pino B.
- 3: mova o disco 1 do pino C para o pino B.
- 4: mova o disco 3 do pino A para o pino C.
- 5: mova o disco 1 do pino B para o pino A.
- 6: mova o disco 2 do pino B para o pino C.
- 7: mova o disco 1 do pino A para o pino C.
- 8: mova o disco 4 do pino A para o pino C.
- 9: mova o disco 1 do pino B para o pino C.
- 10: mova o disco 2 do pino B para o pino A.
- 11: mova o disco 1 do pino C para o pino A.
- 12: mova o disco 3 do pino B para o pino C.
- 13: mova o disco 1 do pino A para o pino B.
- 14: mova o disco 2 do pino A para o pino C.
- 15: mova o disco 1 do pino B para o pino C.
- 16: mova o disco 5 do pino A para o pino C.
- 17: mova o disco 1 do pino B para o pino A.
- 18: mova o disco 2 do pino B para o pino C.
- 19: mova o disco 1 do pino A para o pino C.
- 20: mova o disco 3 do pino B para o pino C.
- 21: mova o disco 1 do pino C para o pino B.
- 22: mova o disco 2 do pino C para o pino A.
- 23: mova o disco 1 do pino B para o pino C.
- 24: mova o disco 4 do pino B para o pino C.
- 25: mova o disco 1 do pino A para o pino B.
- 26: mova o disco 2 do pino A para o pino C.
- 27: mova o disco 1 do pino C para o pino A.
- 28: mova o disco 3 do pino A para o pino C.
- 29: mova o disco 1 do pino B para o pino C.
- 30: mova o disco 2 do pino B para o pino A.
- 31: mova o disco 1 do pino A para o pino C.
- 32: mova o disco 6 do pino A para o pino C.
- 33: mova o disco 1 do pino C para o pino B.
- 34: mova o disco 2 do pino C para o pino A.
- 35: mova o disco 1 do pino B para o pino C.

Função que resolve o problema

```
void
hanoi(int n, char origem, char auxiliar, char destino)
{
1   if (n > 0)
    {
2       hanoi(n-1, origem, destino, auxiliar);
3       printf("mova disco %d de %c para %c.\n",
              n, origem, destino);
4       hanoi(n-1, auxiliar, origem, destino);
    }
}
```

Primeira chamada: $\text{hanoi}(n, 'A', 'B', 'C')$;

$\text{hanoi}(4, 'A', 'B', 'C')$

- 1: mova o disco 1 do pino A para o pino B.
- 2: mova o disco 2 do pino A para o pino C.
- 3: mova o disco 1 do pino B para o pino C.
- 4: mova o disco 3 do pino A para o pino B.
- 5: mova o disco 1 do pino C para o pino A.
- 6: mova o disco 2 do pino C para o pino B.
- 7: mova o disco 1 do pino A para o pino B.
- 8: mova o disco 4 do pino A para o pino C.
- 9: mova o disco 1 do pino B para o pino C.
- 10: mova o disco 2 do pino B para o pino A.
- 11: mova o disco 1 do pino C para o pino A.
- 12: mova o disco 3 do pino B para o pino C.
- 13: mova o disco 1 do pino A para o pino B.
- 14: mova o disco 2 do pino A para o pino C.
- 15: mova o disco 1 do pino B para o pino C.

Recursão

A resolução recursiva de um problema tem tipicamente a seguinte estrutura:

se a instância em questão é “pequena” resolva-a diretamente (use força bruta se necessário);

senao reduza-a a uma instância “menor” do mesmo problema, aplique o método à instância menor e volte à instância original.

Curiosidades

Veja "Debugging recursive code" em :
<http://devopsreactions.tumblr.com/>

Fatorial recursivo

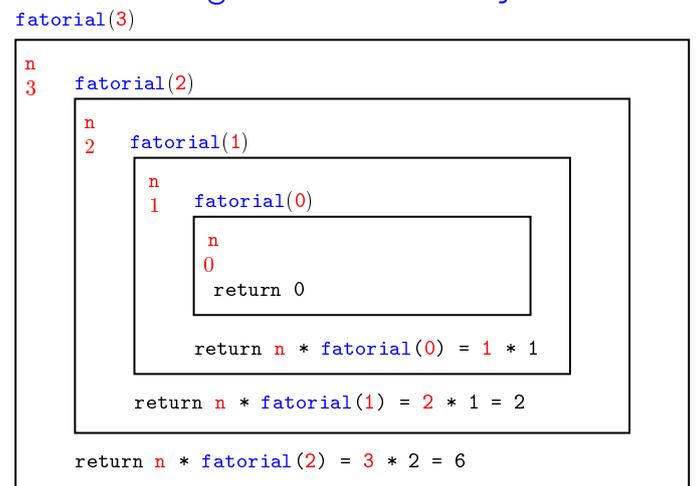
$$n! = \begin{cases} 1, & \text{quando } n = 0, \\ n \times (n - 1)!, & \text{quando } n > 0. \end{cases}$$

```
long
fatorial(long n)
{
1  if (n == 0) return 1;
2  return n * fatorial(n-1);
}
```

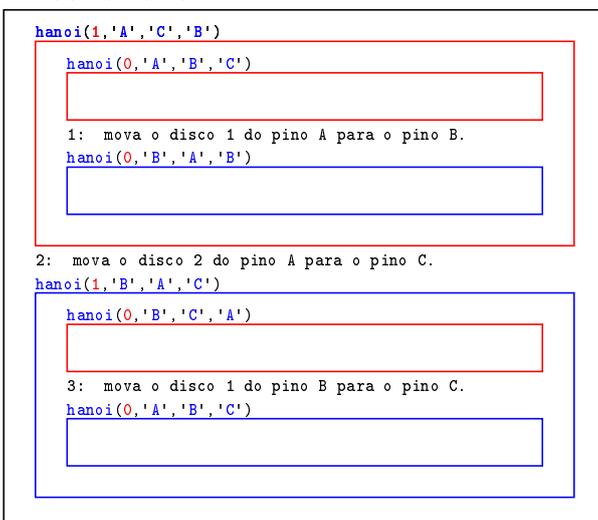
fatorial(10)

```
fatorial(10)
  fatorial(9)
    fatorial(8)
      fatorial(7)
        fatorial(6)
          fatorial(5)
            fatorial(4)
              fatorial(3)
                fatorial(2)
                  fatorial(1)
                    fatorial(0)
fatorial de 10 e' 3628800.
```

Diagramas de execução



```
hanoi(2,'A','B','C')
```



Fatorial iterativo

```
long
fatorial(long n)
{
  int i, ifat;
1  ifat = 1;
2  for(i = 1; /*1*/ i <= n; i++)
3    ifat *= i;
4  return ifat;
}
```

Em /*1*/ vale que `ifat == (i-1)!`