Recursão
(IED-001)

Prof. Dr. Silvio do Lago Pereira

Departamento de Tecnologia da Informação

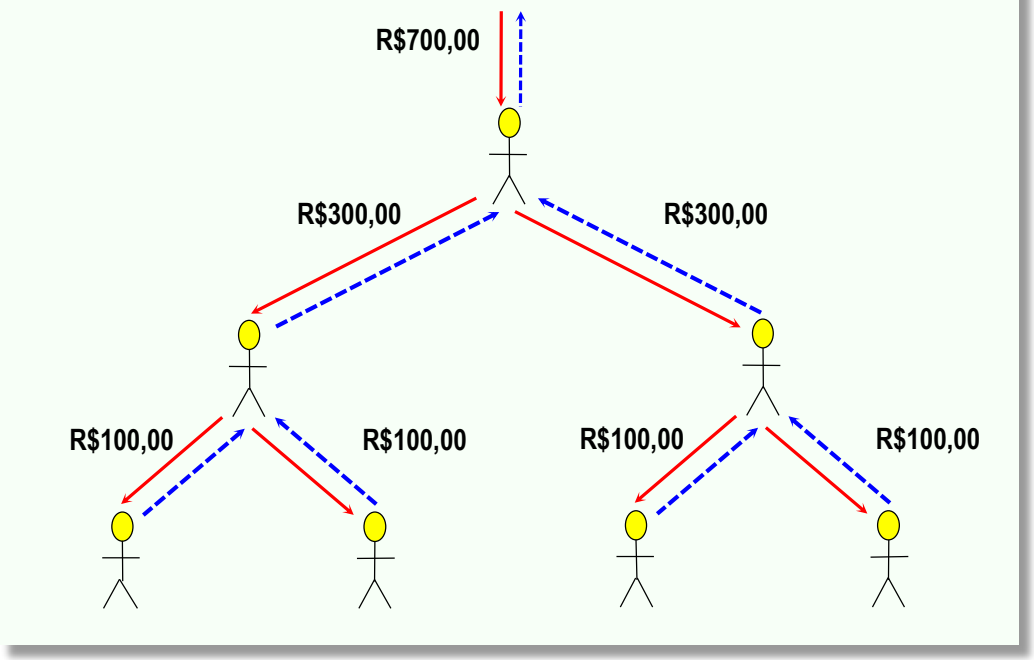
Faculdade de Tecnologia de São Paulo

Recursão

Recursão

é um método que permite obter a solução de uma instância de um problema P, a partir das soluções de instâncias mais simples de P.

Exemplo 1. Problema da arrecadação



Fases de uma solução recursiva:

- **De mposição**: decompõe a instância ini s mai s a
- **C** ções ples, até obter a solu da instância inicial.

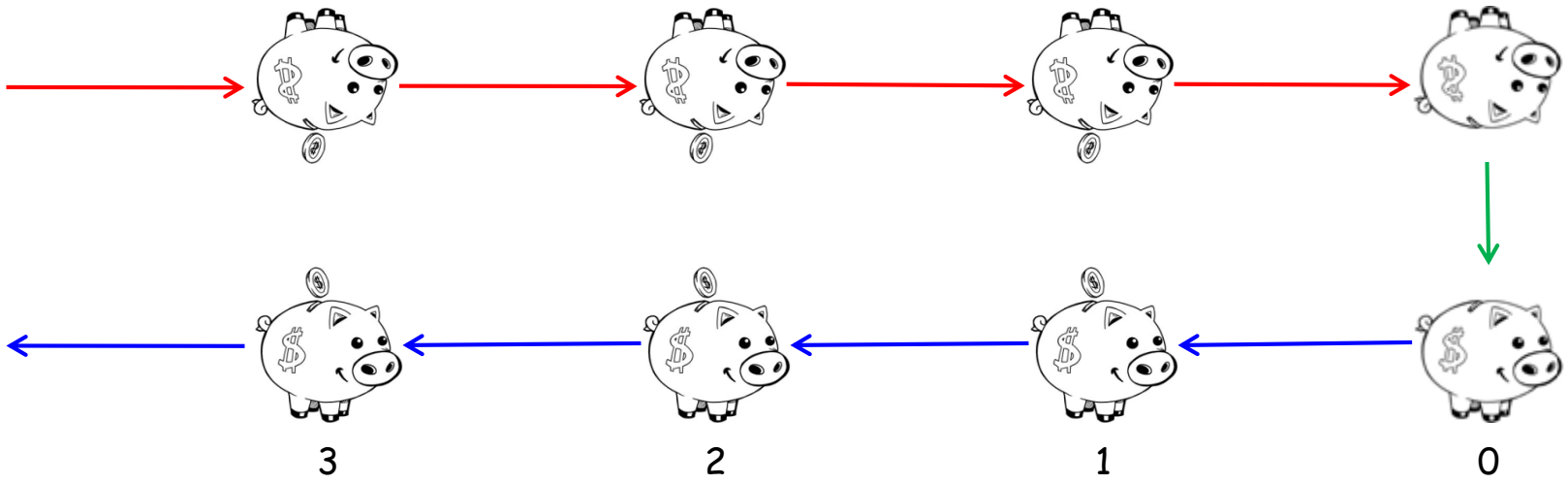
Arrecadar R$ 700,00 para doação, supondo que cada pessoa pode doar no máximo R$ 100,00.

Exemplo 2. Problema do cofrinho

Suponha que você precisa saber **quantas moedas** há em um cofrinho. Suponha também que:

- você pode **chacoalhar** o cofrinho, para verificar se ele está vazio;
- e, caso ele não esteja, **retirar uma moeda** dele.

Usando essas operações, como você poderia resolver esse problema recursivamente?



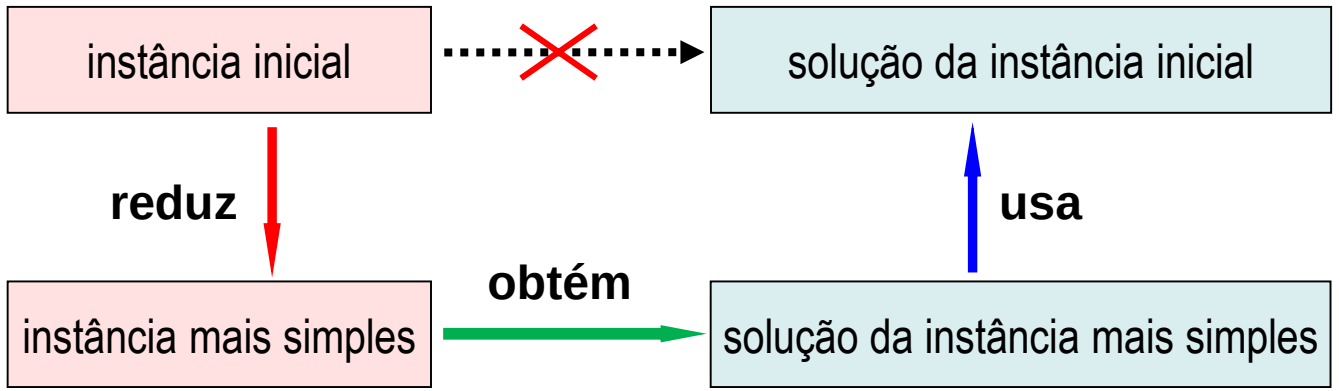
O que acontece se ninguém retirar uma moeda antes de pedir ajuda a outra pessoa?

Princípio de recursão

Teste a instância a ser resolvida:

- Se ela for **trivial**, resolva-a **diretamente**;
- Senão, resolva-a **recursivamente** (usando o princípio de recursão).

Princípio de recursão:



Função recursiva

Uma função recursiva deve ter:

Base: que resolve diretamente um **caso trivial** do problema;

Passo: que resolve recursivamente um **caso geral** do problema.

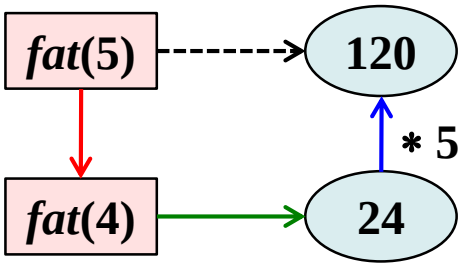
Função recursiva

Exemplo 3. Problema do fatorial

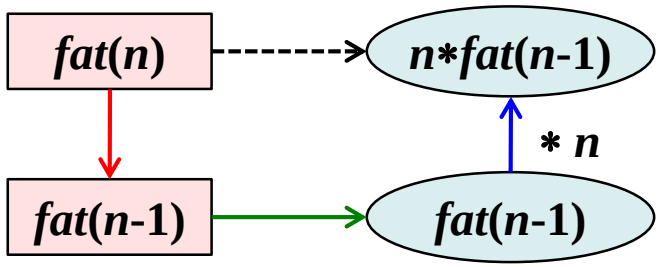
Crie a função recursiva ***fat(n)***, que calcula o fatorial de um número natural ***n***.

$$fat(n) = \begin{cases} 1, & \text{se } n = 0 \\ 1 \times 2 \times 3 \times \ldots \times n, & \text{se } n > 0 \end{cases}$$

Raciocínio recursivo:



(a) caso particular



(b) caso geral

```
int fat(int n) {
    if( n==0 ) return 1;       // base
    else return n*fat(n-1);    // passo
}
```

Recursão

Simulação por substituição

mostra o funcionamento de uma função recursiva, de um ponto de vista "matemático".

```
int fat(int n) {
   if( n==0 ) return 1;      // base
   else return n*fat(n-1);   // passo
}
```

Exemplo 4. Simulação por substituição para `fat(5)`

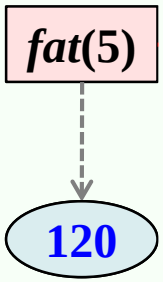
```
fat(5)
= 5*fat(4)
= 5*4*fat(3)
= 5*4*3*fat(2)
= 5*4*3*2*fat(1)
= 5*4*3*2*1*fat(0)
= 5*4*3*2*1*1 = 120
```

Simulação por fluxo de execução

mostra o funcionamento de uma função recursiva, de um ponto de vista "computacional".

```
int fat(int n) {
    if( n==0 ) return 1;       // base
    else return n*fat(n-1);    // passo
}
```

Exemplo 5. Simulação por fluxo de execução para `fat(5)`



As operações "pendentes" ficam aguardando numa **pilha**!

Recursão

Exercício 1. Cálculo de fatorial

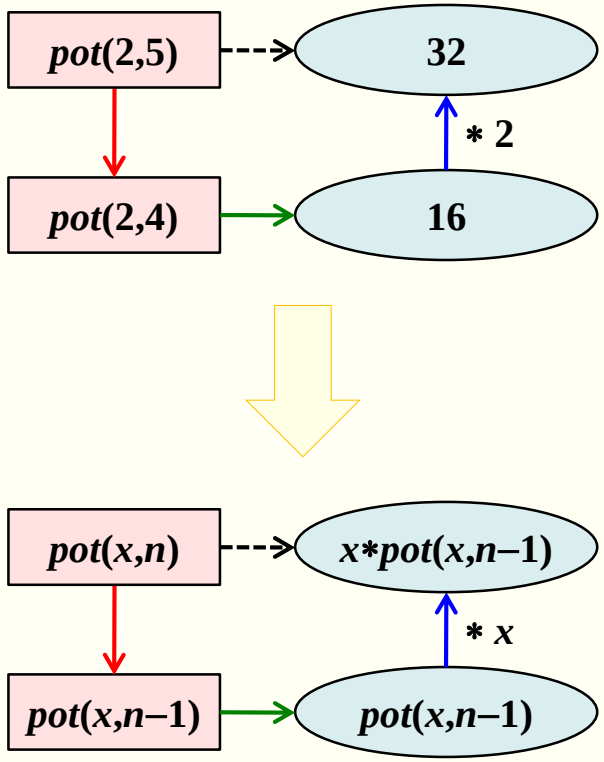
Crie e execute o programa a seguir.

```
#include <stdio.h>

int fat(int n) {
    if( n==0 ) return 1;
    return n*fat(n-1);
}

int main(void) {
    int n;
    printf("Num? ");
    scanf("%d",&n);
    printf("Fat = %d\n",fat(n));
    return 0;
}
```

Exercício 2. Cálculo de potência

Crie a função recursiva ***pot(x,n)***, que calcula a potência de um número real $x \neq 0$ elevado a um número natural n.

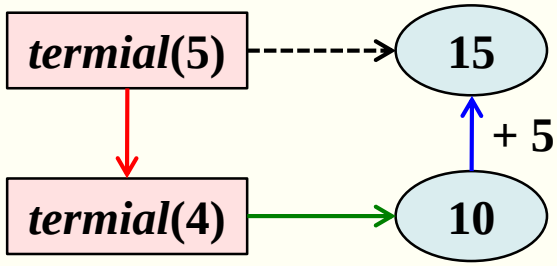
```
#include <stdio.h>

float pot(float x, int n) {
    if( n==0 ) return 1;
    return x*pot(x,n-1);
}

int main(void) {
    float x;
    int n;
    printf("Base e expoente? ");
    scanf("%f %d",&x,&n);
    printf("Pot = %.1f\n",pot(x,n));
    return 0;
}
```



Recursão

Exercício 3. Cálculo de termial

Crie a função recursiva ***termial*(*n*)**, que calcula a soma dos ***n*** primeiros números naturais (***n*** ≥ **0**).

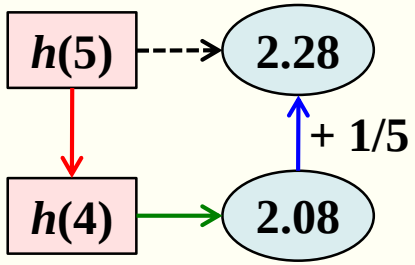



```c
#include <stdio.h>
int termial(int n) {
    if( n==0 ) return 0;
    return termial(n-1) + n;
}

int main(void) {
    int n;
    printf("Num? ");
    scanf("%d",&n);
    printf("Termial = %d\n",termial(n));
    return 0;
}
```

Recursão

Exercício 4. Cálculo da série harmônica

Crie a função recursiva $h(n)$, que calcula a soma dos $n \geq 1$ primeiros termos da série harmônica $(1 + 1/2 + 1/3 + \ldots + 1/n)$.

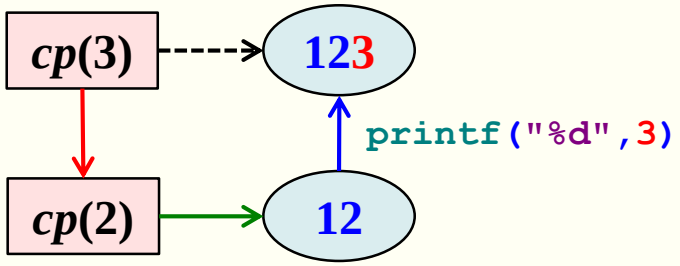



```
#include <stdio.h>
float h(int n) {
    if( n==1 ) return 1;
    return h(n-1) + 1.0/n;
}

int main(void) {
    int n;
    printf("Num? ");
    scanf("%d",&n);
    printf("Harmonica = %.2f\n",h(n));
    return 0;
}
```

Recursão

Exercício 5. Contagem progressiva

Crie a função recursiva ***cp(n)***, que exibe uma contagem progressiva de **1** até ***n*≥0**.



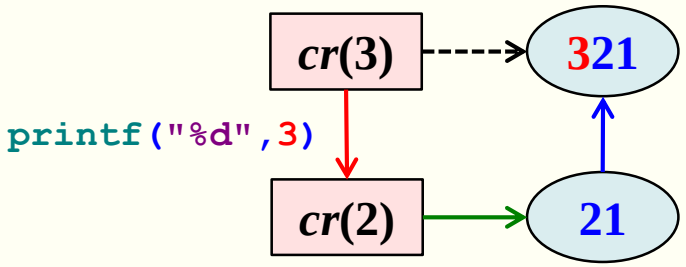

```
#include <stdio.h>
void cp(int n) {
    if( n==0 ) return;
    cp(n-1);
    printf("%d\n",n);
}
int main(void) {
    int n;
    printf("Num? ");
    scanf("%d",&n);
    cp(n);
    return 0;
}
```

Comandos depois da chamada recursiva são empilhados e executados em ordem inversa!

Recursão

Exercício 6. Contagem regressiva

Crie a função recursiva ***cr(n)***, que exibe uma contagem progressiva de $n \geq 0$ até **1**.



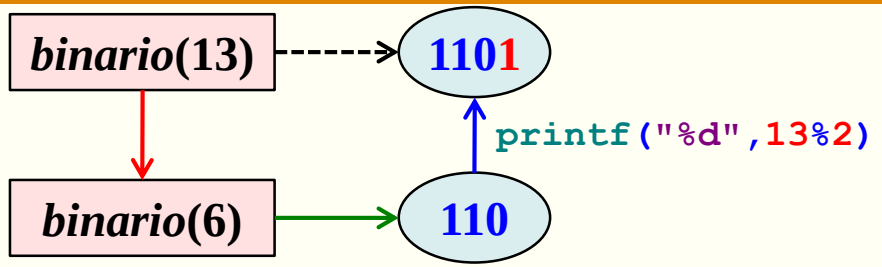

```
#include <stdio.h>
void cr(int n) {
    if( n==0 ) return;
    printf("%d\n",n);
    cr(n-1);
}
int main(void) {
    int n;
    printf("Num? ");
    scanf("%d",&n);
    cr(n);
    return 0;
}
```

Comandos antes da chamada recursiva são executados em ordem direta!

Recursão

Exercício 7. Conversão em binário

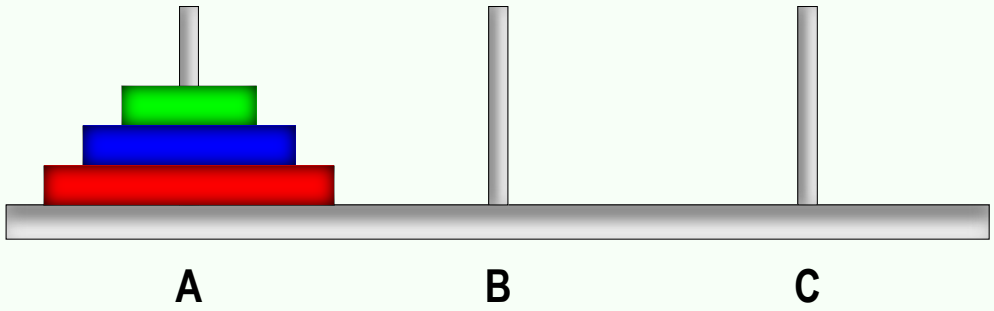
Crie a função recursiva ***binário*(*n*)**, que exibe um natural ***n*≥0** em binário.




```
#include <stdio.h>
void binario(int n) {
    if( n<2 ) printf("%d",n);
    else {
        binario(n/2);
        printf("%d",n%2);
    }
}
int main(void) {
    int n;
    printf("Num? ");
    scanf("%d",&n);
    binario(n);
    return 0;
}
```

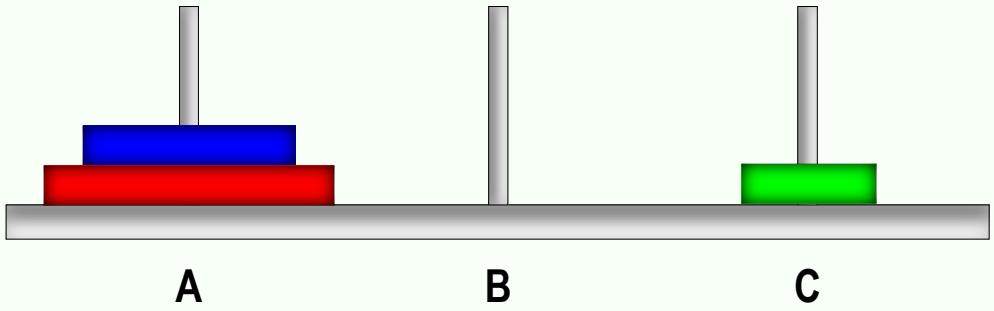
Exemplo 6. Torres de Hanói

- **Problema**:
  - Mover todos os discos da torre **A** para a torre **C**, usando a torre **B** como espaço de manobra.
- **Restrições**:
  - Mover um disco de cada vez.
  - Não colocar um disco sobre outro menor.
  - Transferir os discos de uma torre para outra, imediatamente.



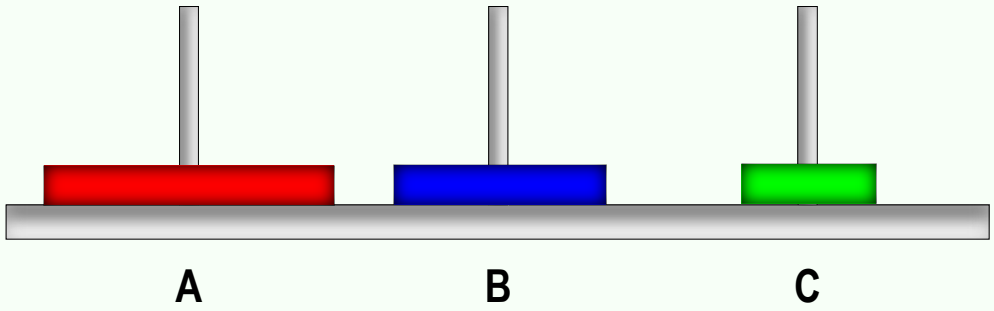
Exemplo 6. Torres de Hanói

- **Problema**:
  - Mover todos os discos da torre **A** para a torre **C**, usando a torre **B** como espaço de manobra.
- **Restrições**:
  - Mover um disco de cada vez.
  - Não colocar um disco sobre outro menor.
  - Transferir os discos de uma torre para outra, imediatamente.



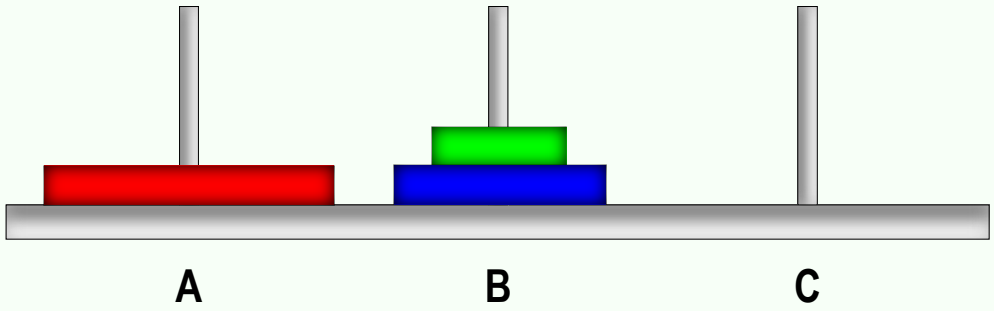
Exemplo 6. Torres de Hanói

- **Problema**:
  - Mover todos os discos da torre **A** para a torre **C**, usando a torre **B** como espaço de manobra.
- **Restrições**:
  - Mover um disco de cada vez.
  - Não colocar um disco sobre outro menor.
  - Transferir os discos de uma torre para outra, imediatamente.



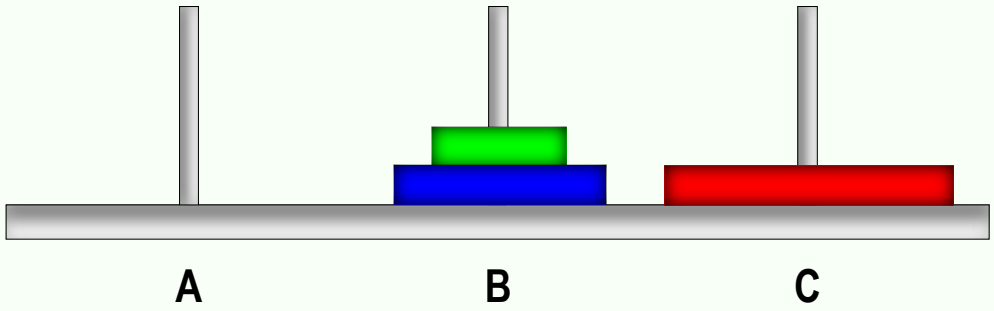
Exemplo 6. Torres de Hanói

- **Problema**:
  - Mover todos os discos da torre **A** para a torre **C**, usando a torre **B** como espaço de manobra.
- **Restrições**:
  - Mover um disco de cada vez.
  - Não colocar um disco sobre outro menor.
  - Transferir os discos de uma torre para outra, imediatamente.



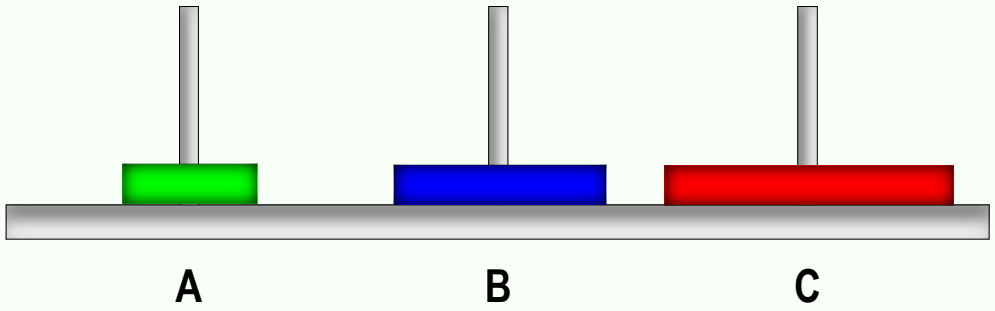
Exemplo 6. Torres de Hanói

- **Problema**:
  - Mover todos os discos da torre **A** para a torre **C**, usando a torre **B** como espaço de manobra.
- **Restrições**:
  - Mover um disco de cada vez.
  - Não colocar um disco sobre outro menor.
  - Transferir os discos de uma torre para outra, imediatamente.



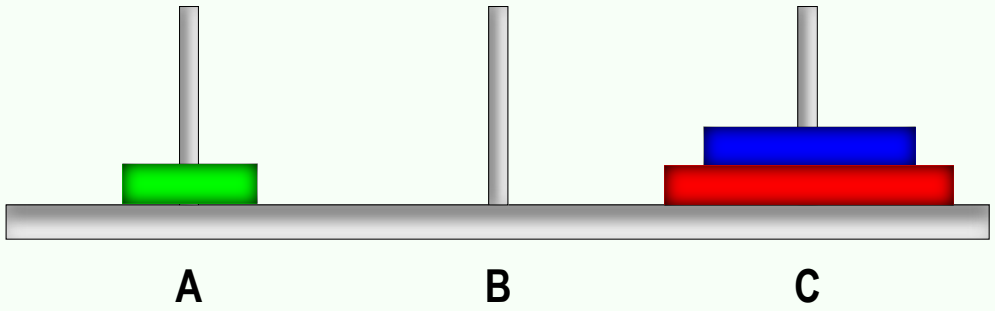
Exemplo 6. Torres de Hanói

- **Problema**:
  - Mover todos os discos da torre **A** para a torre **C**, usando a torre **B** como espaço de manobra.
- **Restrições**:
  - Mover um disco de cada vez.
  - Não colocar um disco sobre outro menor.
  - Transferir os discos de uma torre para outra, imediatamente.



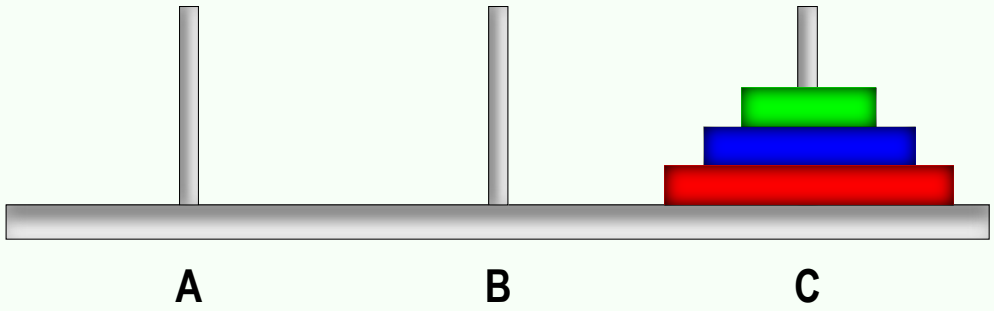
Exemplo 6. Torres de Hanói

- **Problema**:
  - Mover todos os discos da torre **A** para a torre **C**, usando a torre **B** como espaço de manobra.
- **Restrições**:
  - Mover um disco de cada vez.
  - Não colocar um disco sobre outro menor.
  - Transferir os discos de uma torre para outra, imediatamente.



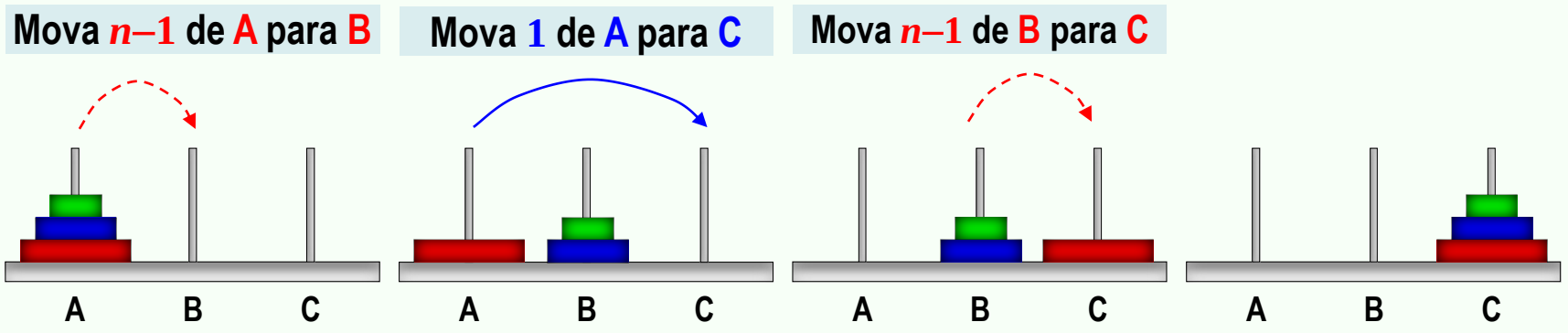
Exemplo 6. Torres de Hanói

- **Problema**:
  - Mover todos os discos da torre **A** para a torre **C**, usando a torre **B** como espaço de manobra.
- **Restrições**:
  - Mover um disco de cada vez.
  - Não colocar um disco sobre outro menor.
  - Transferir os discos de uma torre para outra, imediatamente.



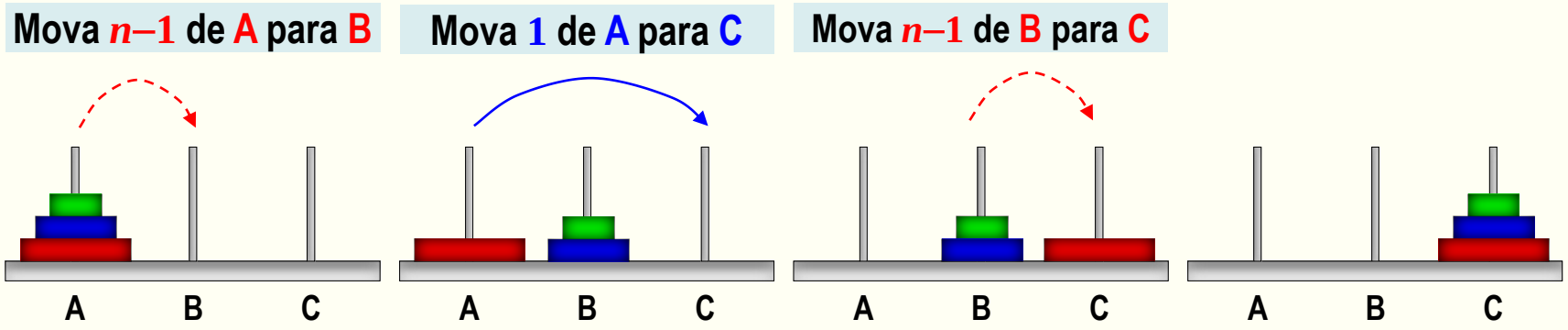
Exemplo 6. Torres de Hanói

- **Problema**:
  - Mover todos os discos da torre **A** para a torre **C**, usando a torre **B** como espaço de manobra.
- **Restrições**:
  - Mover um disco de cada vez.
  - Não colocar um disco sobre outro menor.
  - Transferir os discos de uma torre para outra, imediatamente.
- **Solução recursiva**:



Exercício 8. Função hanói

Crie a função recursiva ***hanoi(n,origem,auxiliar,destino)***, que resolve o problema das Torres de Hanói, movendo ***n*** discos da torre ***origem***, para a torre ***destino***, usando a torre ***auxiliar***. Por exemplo, a chamada ***hanoi*(3, 'A', 'B', 'C')** deve resolver o problema discutido no Exemplo 6.



Recursão

Exercício 9. Par

Crie a função recursiva **`par(n)`**, que determina se o natural **`n`** é par, usando apenas subtração.

Exercício 10. Produto

Crie a função recursiva **`prod(m,n)`**, que devolve o produto de dois números naturais **`m`** e **`n`**, usando apenas soma e subtração.

Exercício 11. Quociente

Crie a função recursiva **`quoc(m,n)`**, que devolve o quociente da divisão inteira do número natural **`m`** pelo número natural **`n`**$\neq$**`0`**, usando apenas soma e subtração.

Exercício 12. Resto

Crie a função recursiva **`resto(m,n)`**, que devolve o resto da divisão inteira do número natural **`m`** pelo número natural **`n`**$\neq$**`0`**, usando apenas subtração.

Exercício 13. Quadrado

O quadrado de um natural n é a soma dos n primeiros ímpares, i.e., $n^2 = 1 + 3 + 5 + \cdots + (2n-1)$.
Crie a função recursiva **`q(n)`**, que devolve o quadrado de **`n`**, como base nesta informação.

Recursão

Exercício 14. Soma de dígitos

Crie a função recursiva **`sd(n)`**, que devolve a soma dos dígitos do número natural **`n`**. Por exemplo, a chamada **`sd(7859)`** deve devolver **`29`** (pois **`7+8+5+9 = 29`**).

Exercício 15. Quantidade de dígitos

Crie a função recursiva **`qd(n)`**, que devolve a quantidade de dígitos binários para representar o natural **`n`**. Por exemplo, a chamada **`qd(13)`** deve devolver **`4`** (pois **`13`** em binário é **`1101`**).

Exercício 16. Torres de Hanói

Crie a função recursiva **`h(n)`**, que devolve o número mínimo de movimentos para resolver o problema das Torres de Hanói com **`n`** discos. Por exemplo, **`h(3)`** deve devolver **`7`**.

Exercício 17. Cadeia inversa

Crie a função recursiva **`inv(s,p,u)`**, que inverte a string **`s`**, cujo primeiro caractere está na posição **`p`** e cujo último caractere está na posição **`u`**. A função deve devolver **`s`** como resposta.

Exercício 18. Cadeia palíndroma

Crie a função recursiva **`pal(s,p,u)`**, que informa se a string **`s`**, cujo primeiro caractere está na posição **`p`** e cujo último caractere está na posição **`u`**, é palíndroma (ignorando brancos).

Fim