

# Projeto e Análise de Algoritmos I

*Design e Análise de Algoritmos por Indução*

Prof. Dr. Osvaldo Luiz de Oliveira

Estas anotações devem ser  
complementadas por  
apontamentos em aula.

## Verificação exaustiva de um teorema

Seja  $T$  um teorema que nós queremos verificar a validade para  $n$ , inteiro,  $1 \leq n \leq m$ . Uma verificação exaustiva consiste em verificar a validade do teorema para:

- $T(1)$ ;
- $T(2)$ ;
- ...
- $T(m)$ .

Quais a limitações desta abordagem?

## Princípio da indução matemática

Seja  $T$  um teorema que nós queremos provar a validade para  $n$ , inteiro,  $n \geq 1$ . Para provar que  $T$  vale para todo  $n$ :

1. Nós verificamos se  $T(1)$  é verdadeiro;

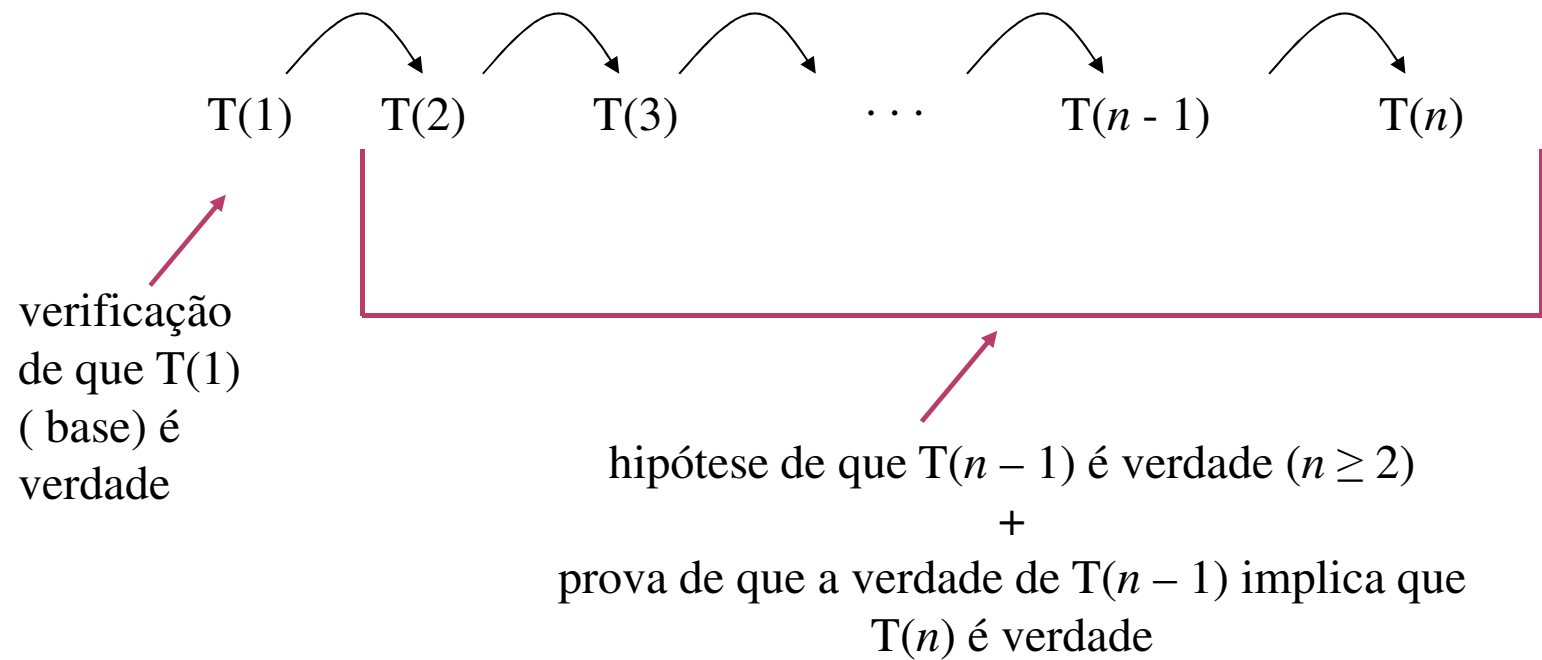
base

hipótese de indução

2. Admitindo, por hipótese, que  $T(n - 1)$  é verdadeiro ( $n \geq 2$ ), nós provamos que a validade de  $T(n - 1)$  implica que  $T(n)$  é verdadeiro.

Prova do “motor de indução” que permite admitir que o fato do teorema valer para  $n - 1$  implica que ele vale para  $n$ .

# Efeito dominó



# Soma dos $n$ primeiros naturais

Exemplo, prova do seguinte:

## **Teorema**

A soma  $1 + 2 + 3 + \dots + n$  ( $n \geq 1$ , inteiro) é

$$S(n) = \frac{n(n+1)}{2}.$$

# Solução

## i) Base

Para  $n = 1$  a soma é igual a 1.

Pela fórmula  $S(1) = 1 \cdot (1 + 1) / 2 = 1$ .

Logo, o Teorema é verdade para  $n = 1$ .

## ii) Hipótese de indução

O Teorema afirma que  $S(n) = n(n + 1) / 2$ .

Admitimos, por hipótese, que

$S(n - 1) = (n - 1)n / 2$  é verdade.

## Solução

### iii) Relação entre $S(n)$ e $S(n - 1)$

$$S(n) = 1 + 2 + \dots + n - 1 + n.$$

$$S(n - 1) = 1 + 2 + \dots + n - 1.$$

$$\text{Logo, } S(n) = S(n - 1) + n.$$

### iv) Prova de que $S(n - 1)$ implica em $S(n)$

$$S(n) = S(n - 1) + n.$$

$$\text{Usando a hipótese de indução, } S(n) = (n - 1) n / 2 + n.$$

Hipótese de indução

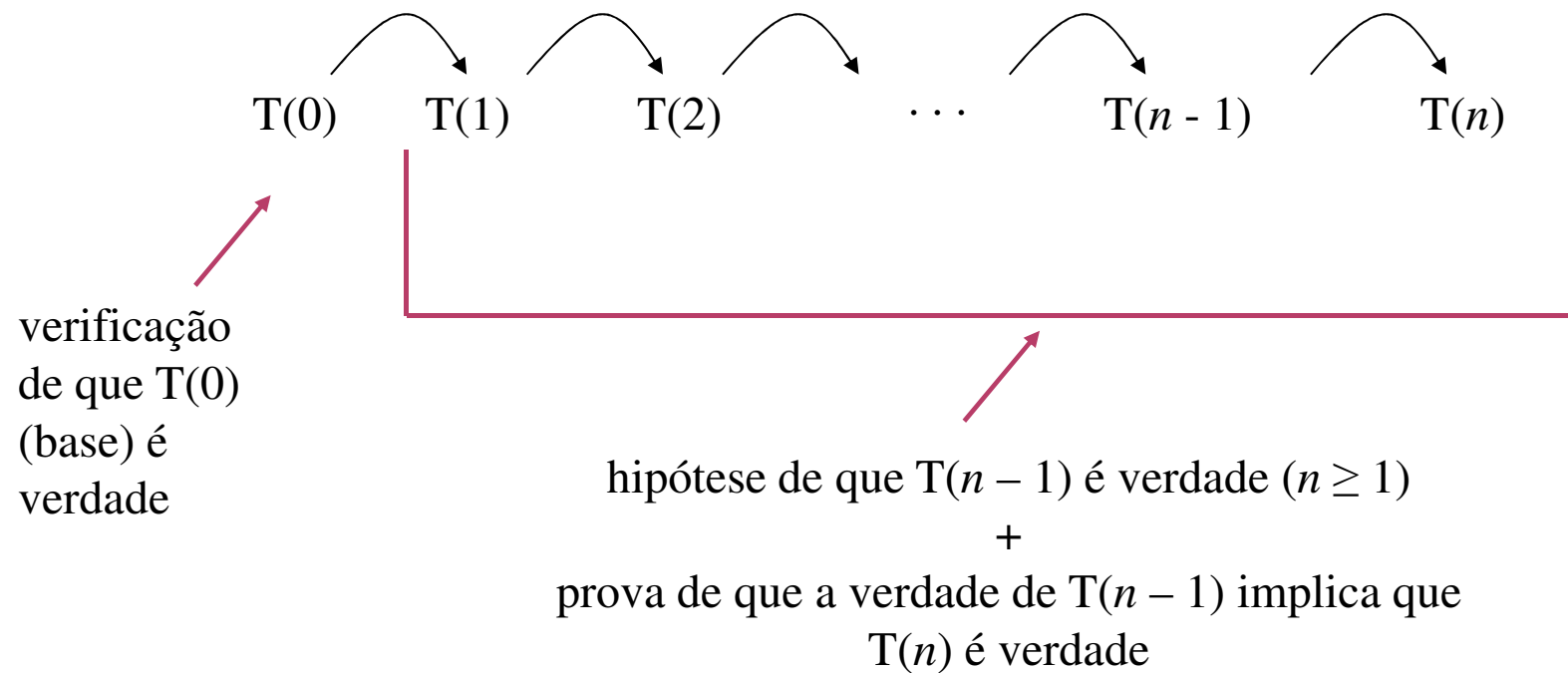
$$S(n - 1) = \boxed{(n - 1) n / 2}$$

$$\text{Logo, } S(n) = (n^2 - n + 2n) / 2 = (n^2 + n) / 2$$

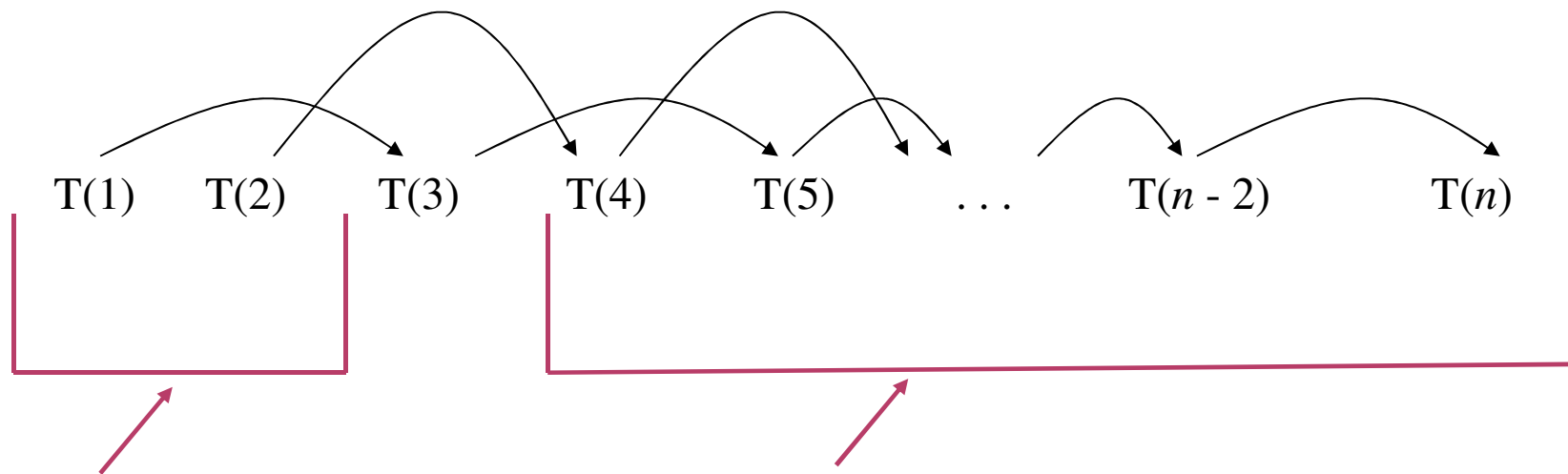
$$\text{Ou seja, } S(n) = n(n + 1) / 2. \quad \blacksquare$$

# Variantes do princípio de indução descrito

# Base não necessariamente para $n = 1$



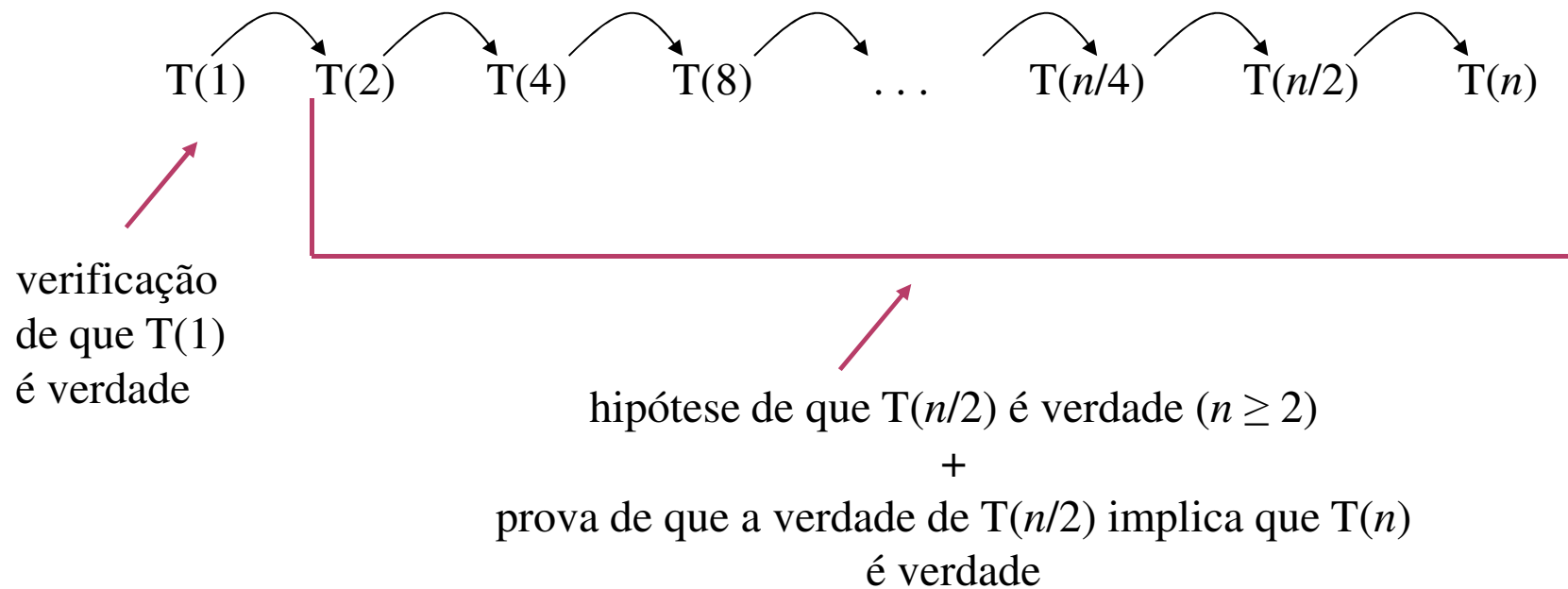
## Redução subtrativa não necessariamente de 1 em 1



verificação  
de que  $T(1)$   
e  $T(2)$   
(bases) são  
verdade

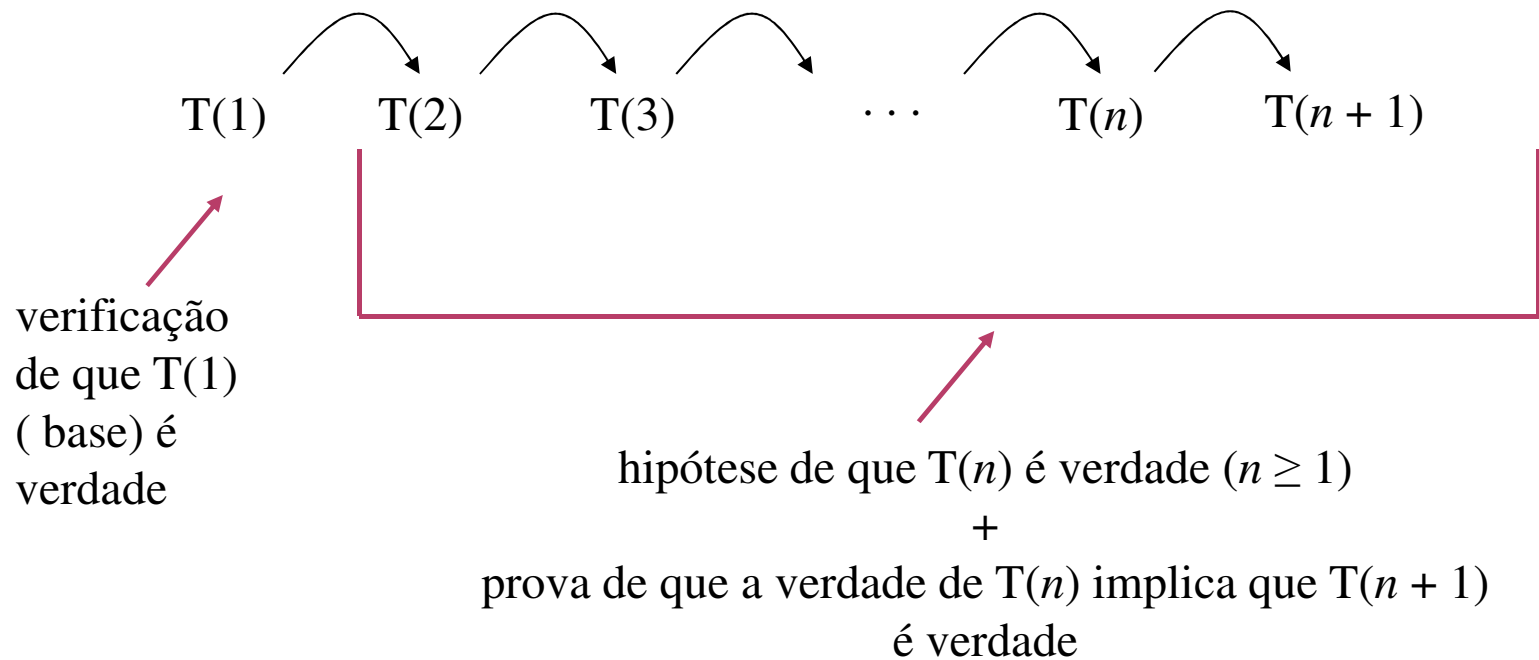
hipótese de que  $T(n-2)$  é verdade ( $n \geq 3$ )  
+  
prova de que a verdade de  $T(n-2)$  implica que  
 $T(n)$  é verdade

## Redução não necessariamente subtrativa



Obs.: nesta variante, a prova não valeria para todos os números naturais mas para 1, 2, 4, 8, ....

# Término não necessariamente em $T(n)$



# Partes de uma prova indutiva

Base ou bases

Hipótese de indução  $T(\alpha)$

Redução de  $T(\beta)$  para  $T(\alpha)$   
(ou ampliação  $T(\alpha)$  para  $T(\beta)$   
ou relação indutiva entre  $T(\alpha)$  e  $T(\beta)$  )

Prova de que  $T(\alpha)$  implica  
em  $T(\beta)$ .

# Não importa a ordem

As partes devem formar um todo consistente.

Redução de  $T(\beta)$  para  $T(\alpha)$

Hipótese de indução  $T(\alpha)$

Base ou bases

Prova de que  $T(\alpha)$  implica em  $T(\beta)$ .

# Estratégia de prova

1 – Proponha uma redução de  $T(\beta)$  para  $T(\alpha)$ .

2 – Como a redução proposta foi para  $T(\alpha)$ , escreva a hipótese de indução  $T(\alpha)$ .

3 – Prove que o fato: se  $T(\alpha)$  é verdadeiro então  $T(\beta)$  é verdadeiro.

4 – Observando a redução proposta, crie uma ou mais bases. Verifique se o teorema é verdadeiro para a(s) base(s).

## Nova prova do teorema da soma dos $n$ primeiros naturais

Use na prova a seguinte redução:

$$S(n) = S(n - 2) + 2n - 1.$$

# Outras provas

## Triângulo de Pascal

Ache uma expressão para a soma de uma linha do triângulo de Pascal e prove que sua expressão está correta.

$$\begin{array}{rcl} 1 & & = 1 \\ 1 & 1 & = 2 \\ 1 & 2 & 1 & = 4 \\ 1 & 3 & 3 & 1 & = 8 \\ 1 & 4 & 6 & 4 & 1 & = 16 \end{array}$$

# Árvore binária cheia

## Teorema

A quantidade de nós no nível  $i$  de uma árvore binária cheia não vazia é

$$Q(i) = 2^i, i \geq 0.$$

Obs.: estamos assumindo que a raiz da árvore binária cheia está no nível  $i = 0$ .

# Uma inequação

## **Teorema**

Se  $n$  é um número natural e  $1 + x > 0$  então

$$(1 + x)^n \geq 1 + nx.$$

# Quantidade de arestas e vértices de uma árvore

## **Teorema**

A quantidade de arestas de uma árvore com  $n \geq 1$  vértices é igual a

$$E(n) = n - 1.$$

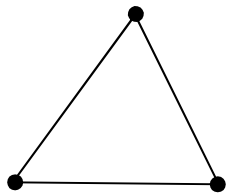
# Fórmula de Euler

Em um mapa planar conexo  $n + f = e + 2$ .

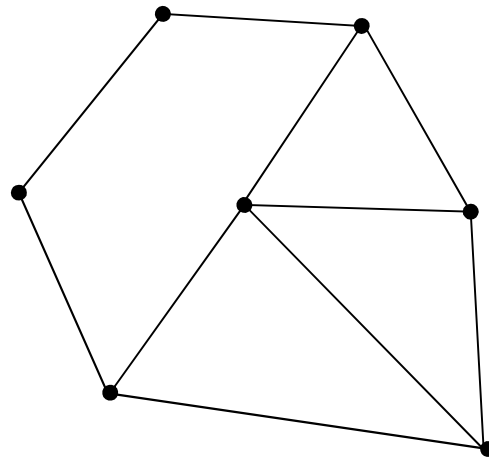
$n$ : quantidade de vértices

$f$ : quantidade de faces

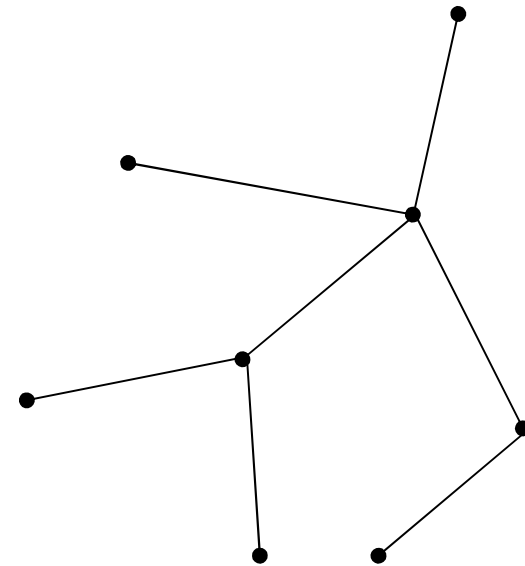
$e$ : quantidade de arestas



$$n = 3, f = 2, e = 3$$



$$n = 7, f = 5, e = 10$$

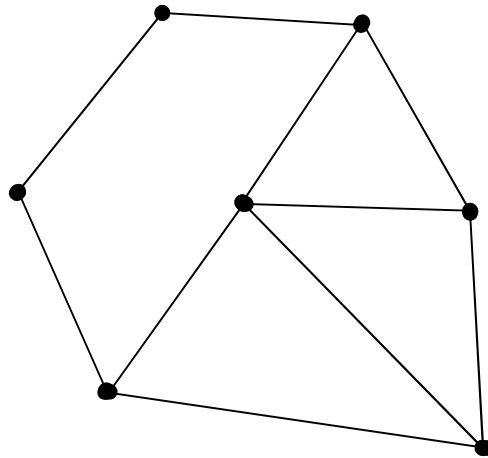


$$n = 8, f = 1, e = 7$$

# Problema e inspiração

**Problema:** três variáveis envolvidas e não duas como nos teoremas anteriores.

Inspiração no processo de redução (veja a animação):



# Indução dupla

- “Elimina” uma variável, fazendo  $f = 1$  para provar algo mais específico, i.e.,  $n + 1 = e + 2$ .

Lema 1: A quantidade de arestas de uma árvore com  $n \geq 1$  vértices é igual a  $E(n) = n - 1$ .

Isto já foi provado. Esta é a base da próxima indução.

- Fixa a variável  $n$ , provando que em um mapa planar com  $n$  vértices ( $n$  constante),  $n + f = e + 2$ .

Lema 2: A quantidade de faces de um mapa planar conexo de  $n \geq 2$  vértices ( $n$  constante) e  $e \geq 1$  arestas é igual a  $F(e) = e + 2 - n$ .

# Quantidade de regiões do plano

## Teorema

A quantidade de regiões do plano formada por  $n \geq 1$  retas em posição geral é

$$Q(n) = \frac{n(n+1)}{2} + 1.$$

Obs.: um conjunto de retas está em posição geral no plano se não existem no conjunto linhas paralelas e três ou mais linhas não têm um ponto de intersecção em comum.

# Prova da correção de algoritmos

**Algoritmo** Produto ( $m, n$ )

**Entrada:**  $m$  e  $n$ , inteiros,  $n \geq 0$  e  $m \geq 0$ .

**Saída:** retorna  $m \cdot n$ .

```
{  
     $i := 0; p := 0;$   
    enquanto ( $i < n$ )  
    {  
         $i := i + 1;$   
         $p := p + m$   
    }  
    retornar  $p$   
}
```

## Como fazer?

1. Proponha uma *invariante de laço* (IL), *i.e.*, uma relação entre as variáveis envolvidas no laço e cuja validade permanece durante toda a sua execução.
  
2. Faça:
  - I – BASE: mostre que a IL é verdadeira imediatamente antes do início do laço;
  - II – Hipótese de indução: admita que a IL é verdadeira no passo  $i - 1$ ;
  - III – Prove que a verdade da IL no passo  $i - 1$  do laço implica na verdade da IL no passo  $i$ .
  
- 3 – Mostre que, quando a laço termina, a IL implica que o algoritmo está correto.

# Design de algoritmos por indução

## Relação entre prova indutiva e algoritmo recursivo

### i) Base

Caracterizada por  $n = 1$ .

Para  $n = 1$  a soma é igual a 1.

Pela fórmula  $S(1) = 1 \cdot (1 + 1) / 2 = 1$ .

### ii) Hipótese de indução

$$S(n - 1) = (n - 1) n / 2$$

### iii) Relação entre $S(n)$ e $S(n - 1)$

$$S(n) = 1 + 2 + \dots + n - 1 + n$$

$$S(n - 1) = 1 + 2 + \dots + n - 1$$

Logo,

$$S(n) = S(n - 1) + n.$$

### iv) Prova de que $S(n - 1)$ implica em $S(n)$

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

Usando a hipótese de indução,

$$S(n) = (n - 1) n / 2 + n.$$

Logo,

$$S(n) = n (n + 1) / 2,$$

como queríamos provar.

### Algoritmo $S(n)$

**Entrada:**  $n$ , inteiro,  $n \geq 1$ .

**Saída:** a soma  $1 + 2 + 3 + \dots + n$ .

```
{
  se (  $n = 1$  )
    retornar 1
  senão
    retornar  $S(n - 1) + n$ 
}
```

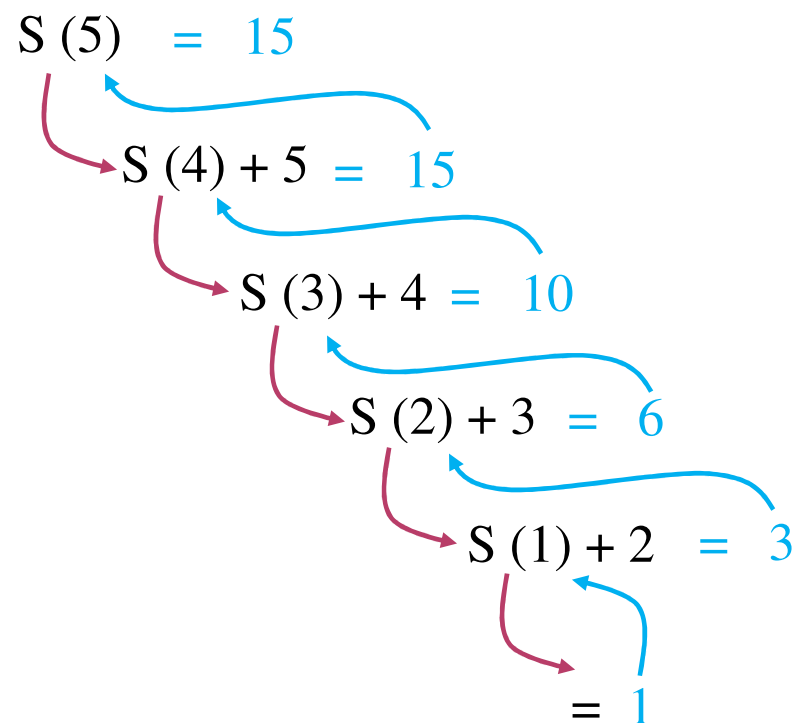
## O cálculo de $S(5)$

**Algoritmo**  $S(n)$

**Entrada:**  $n$ , inteiro,  $n \geq 1$ .

**Saída:** a soma  $1 + 2 + 3 + \dots + n$ .

```
{  
  se (  $n = 1$  )  
    retornar 1  
  senão  
    retornar  $S(n - 1) + n$   
}
```



## Relação entre prova indutiva e algoritmo recursivo

### i) Bases

- Para  $n = 1$  a soma é igual a  $1$  e pela fórmula  $S(1) = 1 \cdot (1 + 1) / 2 = 1$ .

- Para  $n = 2$  a soma é igual a  $3$  e pela fórmula  $S(2) = 2 \cdot (2 + 1) / 2 = 3$ .

### ii) Hipótese de indução

$$S(n - 2) = (n - 2)(n - 1) / 2$$

### iii) Relação entre $S(n)$ e $S(n - 2)$

$$S(n) = 1 + 2 + \dots + n - 2 + n - 1 + n$$

$$S(n - 2) = 1 + 2 + \dots + n - 2$$

Logo,

$$S(n) = S(n - 2) + 2n - 1.$$

### iv) Prova de que $S(n - 2)$ implica em $S(n)$

$$S(n) = S(n - 2) + 2n - 1.$$

Usando a hipótese de indução,

$$S(n) = (n - 2)(n - 1) / 2 + 2n - 1.$$

Logo,

$$S(n) = n(n + 1) / 2,$$

como queríamos provar.

### Algoritmo $S(n)$

**Entrada:**  $n$ , inteiro,  $n \geq 1$ .

**Saída:** a soma  $1 + 2 + 3 + \dots + n$ .

```
{
    se ( n = 1 )
        retornar 1
    senão
        se ( n = 2 )
            retornar 3
        senão
            retornar S(n - 2) + 2*n - 1
}
```

## O cálculo de $S(5)$

**Algoritmo**  $S(n)$

**Entrada:**  $n$ , inteiro,  $n \geq 1$ .

**Saída:** a soma  $1 + 2 + 3 + \dots + n$ .

```
{
  se (  $n = 1$  )
    retornar 1
  senão
    se (  $n = 2$  )
      retornar 3
    senão
      retornar  $S(n - 2) + 2*n - 1$ 
}
```

$$\begin{array}{l}
 S(5) = 15 \\
 \quad \swarrow \quad \searrow \\
 S(3) + 2*5 - 1 = 15 \\
 \quad \swarrow \quad \searrow \\
 S(1) + 2*3 - 1 = 6 \\
 \quad \swarrow \quad \searrow \\
 \quad \quad \quad = 1
 \end{array}$$

## Uma técnica de design de algoritmos por indução

I – Proponha uma **interface** para o algoritmo.

II – Descreva o **significado** da interface proposta.

A sentença que descreve o significado deve especificar os parâmetros de entrada e saída.

III – Desenvolva uma **redução** de tamanho da solução do problema.

IV – Tendo como referência a dinâmica das reduções, proponha uma ou mais **bases**.

V – Escreva o **algoritmo**.

## O algoritmo para o problema da soma dos $n$ primeiros naturais obtido pela técnica descrita

I – Interface

$S(n)$

II – Significado

Retorna a soma dos  $n$  primeiros números naturais.

III – Redução

$$S(n) = 1 + 2 + \dots + n - 1 + n$$

$$S(n - 1) = 1 + 2 + \dots + n - 1$$

$$\text{Logo, } S(n) = S(n - 1) + n. \longrightarrow \text{retornar } S(n - 1) + n$$

Comandos (tendo como referência o significado):

IV – Base

A redução ocorre de forma subtrativa de 1 em

1, i.e.,  $S(n) \rightarrow S(n - 1) \rightarrow \dots \rightarrow S(2) \rightarrow S(1)$ .

Propomos a base para  $n = 1$ . Para  $n = 1$  o

resultado da soma é igual a 1.  $\longrightarrow$  **retornar 1**

Comandos (tendo como referência o significado):

## O algoritmo para o problema da soma dos $n$ primeiros naturais obtido pela técnica descrita

V – Algoritmo

**Algoritmo S ( $n$ )**

**Entrada:**  $n$ , inteiro,  $n \geq 1$ .

**Saída:** a soma  $1 + 2 + 3 + \dots + n$ .

```
{  
  se (  $n = 1$  )  
    retornar 1  
  senão  
    retornar  $S(n - 1) + n$   
}
```

# Cálculo do maior elemento de um conjunto

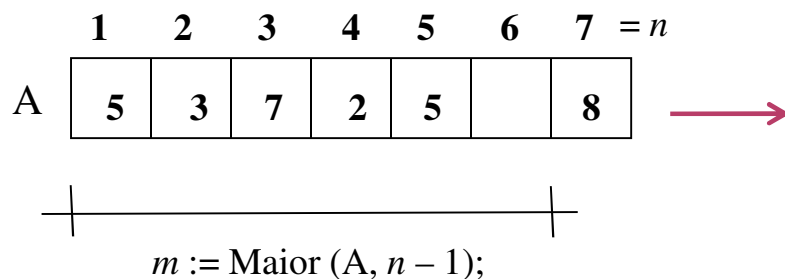
I – Interface

Maior ( $A, n$ )

II – Significado

Retorna o índice do maior elemento do vetor  $A$  que possui  $n$  elementos.

III – Redução



Comandos:

$m := \text{Maior}(A, n-1);$   
**se** ( $A[n] > A[m]$ ) **retornar**  $n$   
**senão** *retornar*  $m$

$m$  recebe o índice do maior elemento entre os  $n-1$  primeiros

IV – Base

A redução ocorre de forma subtrativa de 1 em 1, assim propomos base para  $n = 1$ .

Um vetor com um único elemento tem o maior elemento na posição 1.

Comandos:

**retornar** 1

## Cálculo do maior elemento de um conjunto

V – Algoritmo

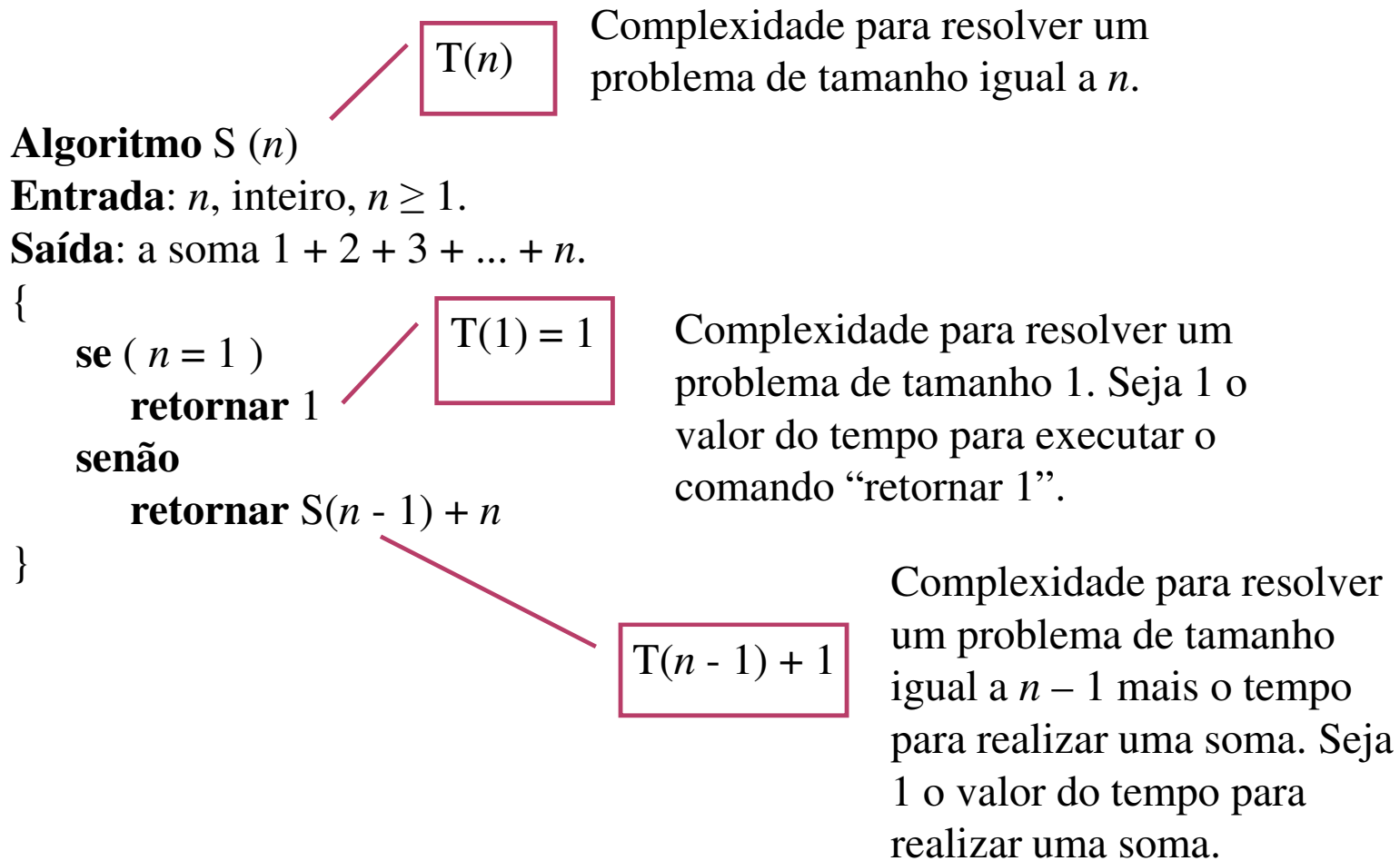
**Algoritmo** Maior ( $A, n$ )

**Entrada:**  $A$ , vetor, de  $n$  elementos,  $n \geq 1$ .

**Saída:** o índice do maior elemento do vetor  $A$ .

```
{
  se (  $n = 1$  )
    retornar 1
  senão
    {
       $m :=$  Maior ( $A, n - 1$ );
      se (  $A[n] > A[m]$  ) retornar  $n$ 
      senão retornar  $m$ 
    }
}
```

# Complexidade de algoritmos recursivos



# Resolução de relações de recorrência

$$T(1) = 1$$

$$T(n) = T(n - 1) + 1, n \geq 2.$$

# Resolução de relações de recorrência

- Método iterativo progressivo;
- Método “dividir para conquistar”;
- Método da substituição.

Obs.: existem outros métodos.

## Exercícios sobre método iterativo progressivo

Resolver as seguintes relações de recorrência:

1)  $T(1) = 1$

$$T(n) = T(n - 1) + 1, n \geq 2.$$

2)  $T(1) = 1$

$$T(n) = T(n - 1) + n, n \geq 2.$$

# Complexidade do algoritmo Maior

$T(n)$

Complexidade para resolver um problema de tamanho igual a  $n$ .

**Algoritmo** Maior ( $A, n$ )

**Entrada:**  $A$ , vetor, de  $n$  elementos,  $n \geq 1$ .

**Saída:** o índice do maior elemento do vetor  $A$ .

{

se ( $n = 1$ )

retornar 1

$T(1) = 1$

Complexidade para resolver um problema de tamanho 1. Seja 1 o valor do tempo (constante) para executar o comando “retornar 1”.

senão

{

$m := \text{Maior}(A, n - 1);$

se ( $A[n] > A[m]$ )

retornar  $n$

senão retornar  $m$

}

}

$T(n - 1) +$

Tempo para executar uma Instância de tamanho igual a  $n - 1$ .

1

Seja 1 o valor do tempo (constante) para executar estes comandos.

## Complexidade do algoritmo Maior

$$T(1) = 1$$

$$T(n) = T(n - 1) + 1, n \geq 2.$$

Resolvendo esta relação de recorrência,  
teremos  $T(n) = n = O(n)$ .

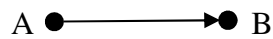
# Outros problemas

## Exercício

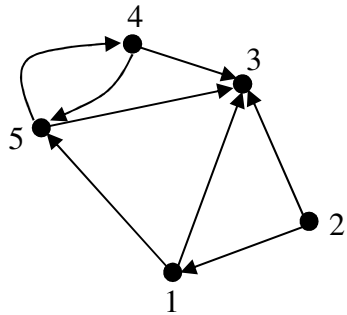
Escreva um algoritmo recursivo para buscar um elemento em um vetor (busca linear).

# Celebridade

- **Definição:** entre  $n$  pessoas uma **celebridade** é uma pessoa que é conhecida por todas as outras mas que não conhece ninguém.



Pessoa “A” conhece pessoa ‘B’”



	1	2	3	4	5
1			1		1
2	1		1		
3					
4			1		1
5			1	1	

C

$C[v, w] = 1: v$  CONHECE  $w$

Celebridade: 3.

# Solução 1

## I – Interface

Celebridade ( $C, n$ )

## II – Significado

Retorna o vértice celebridade  $v$  ( $1 \leq v \leq n$ ) da matriz de conhecimentos  $C$  de tamanho  $n$ . Retorna  $v = 0$ , caso não haja um vértice celebridade.

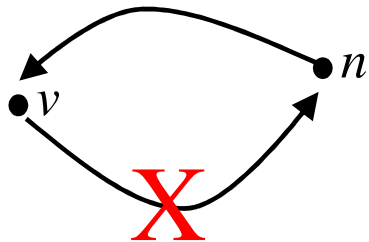
# Solução 1

## III – Redução

$v := \text{Celebridade}(C, n - 1);$

CASO 1:  $1 \leq v \leq n - 1$ , i.e., existe celebridade entre os vértices de 1 a  $n - 1$ .

CASO 1.1:



**retornar**  $v$ ; //  $v$  é celebridade.

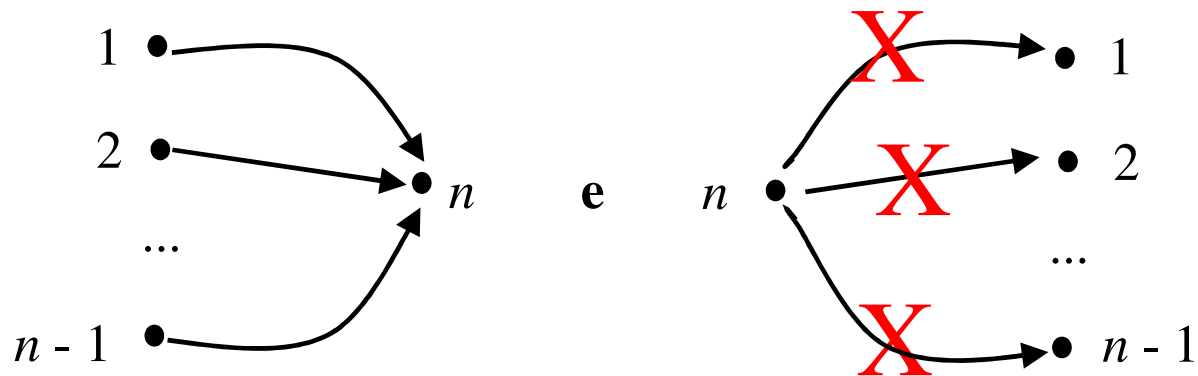
CASO 1.2: alguma das condições indicadas na figura acima falham.

**retornar** 0; // não há celebridade.

# Solução 1

CASO 2:  $v = 0$ , i.e., não existe celebridade entre os vértices de 1 a  $n - 1$ .

CASO 2.1:



**retornar  $n$ ; //  $n$  é celebridade.**

CASO 2.2: alguma das condições indicadas nas figuras acima falham.

**retornar 0; // não há celebridade.**

# Solução 1

III – Base

Caracterizada por  $n = 1$ .

- 1

**retornar** 1; // 1 é (trivialmente) *celebridade*.

# Complexidade da solução 1

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

$$T(1) = 0$$

Resolvendo:  $T(n) = n(n - 1) = O(n^2)$ .

Esta solução tem a complexidade igual ao tamanho  $n^2$  da entrada (uma matriz  $n \times n$ ).

## Solução 2

Idéia para outro tipo de redução:

Descobrir as **não celebridades**.

Sejam A e B duas pessoas.

Se A conhece B então A não é celebridade. Caso contrário, B não é celebridade.

Em qualquer caso, a reduz-se a quantidade de não celebridades em 1.

## Exercício

Escrever o algoritmo Celebridade tendo como motivação a ideia da solução 2.

## Solução 2 - conclusões

- Tamanho da entrada do problema:  $n^2$ .
- Complexidade da solução 2:  $O(n)$ .
- Conseguimos resolver o problema em uma quantidade de tempo menor que o tamanho da entrada.
- Não reduzimos simplesmente o tamanho da entrada de  $n$  para  $n - 1$ . Escolhemos uma redução específica de  $n$  para  $n - 1$  que nos promoveu o ganho de complexidade.

## Divisão e conquista

- Em vez de reduções subtrativas, usaremos reduções que dividem o tamanho do problema.
- Duas fases:
  - **Divisão:** por exemplo, resolver dois problemas de tamanho  $n/2$ ;
  - **Conquista:** combinar as soluções individuais para chegar na solução do problema de tamanho igual a  $n$ .

## Exemplo de Design por “divisão e conquista”

Merge Sort (ordenação por intercalação)

## *Desenvolvendo o Merge Sort*

### I) Interface

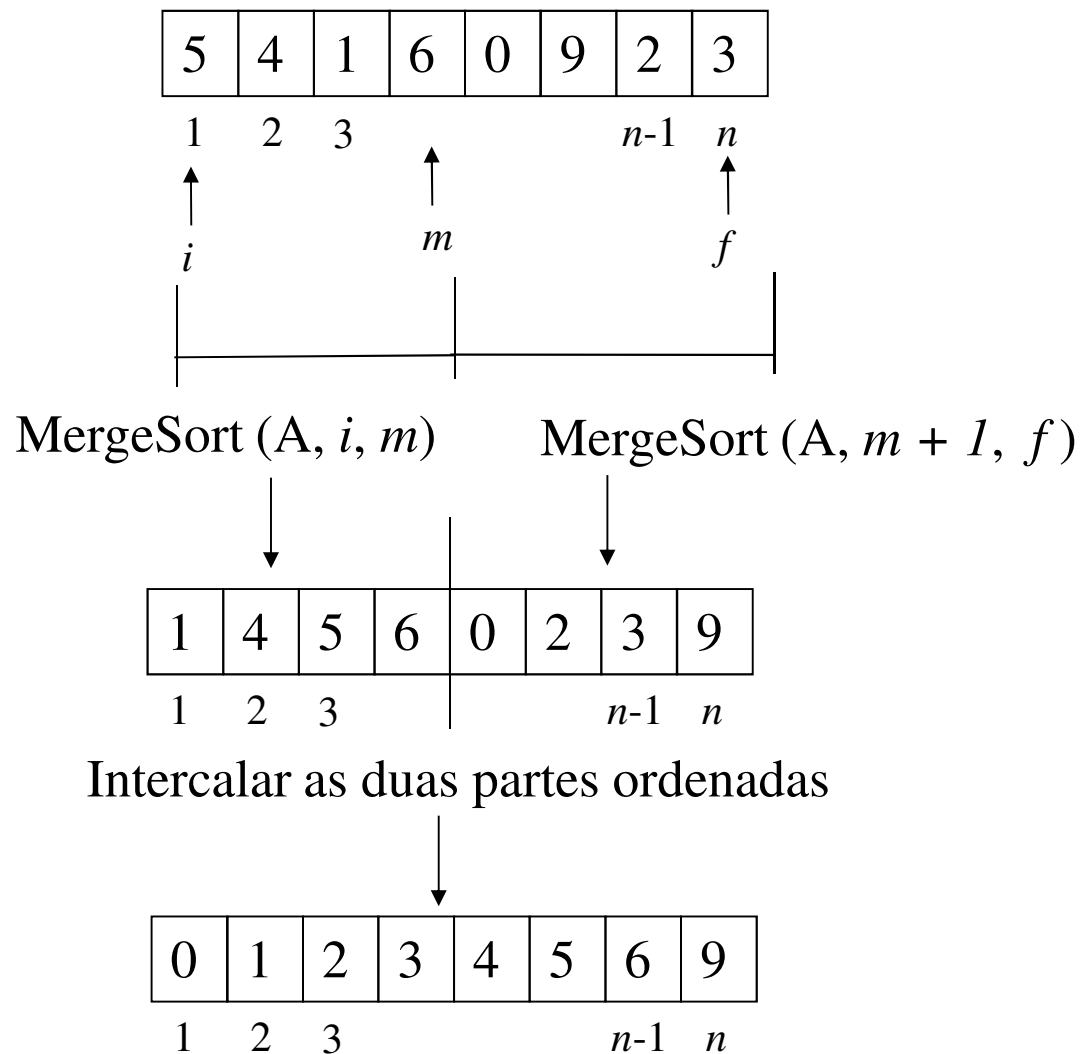
MergeSort ( $A, i, f$ )

### II) Significado

Ordena “in-loco” o vetor  $A$  do índice  $i$  até o índice  $f$ .

# Desenvolvendo o Merge Sort

## III) Redução



## *Desenvolvendo o Merge Sort*

Comandos:

$m := \lfloor (i + f) / 2 \rfloor$ ;

MergeSort (A, i, m);

MergeSort (A, m + 1, f)

Intercalar (A, i, m, f)

### IV) Base

É caracterizada por  $i = f$  (veja reduções “dividir para conquistar”).

Comandos (para ordenar uma faixa de vetor com 1 elemento):

Nenhum comando é necessário.

## *Desenvolvendo o Merge Sort*

### V) O Algoritmo

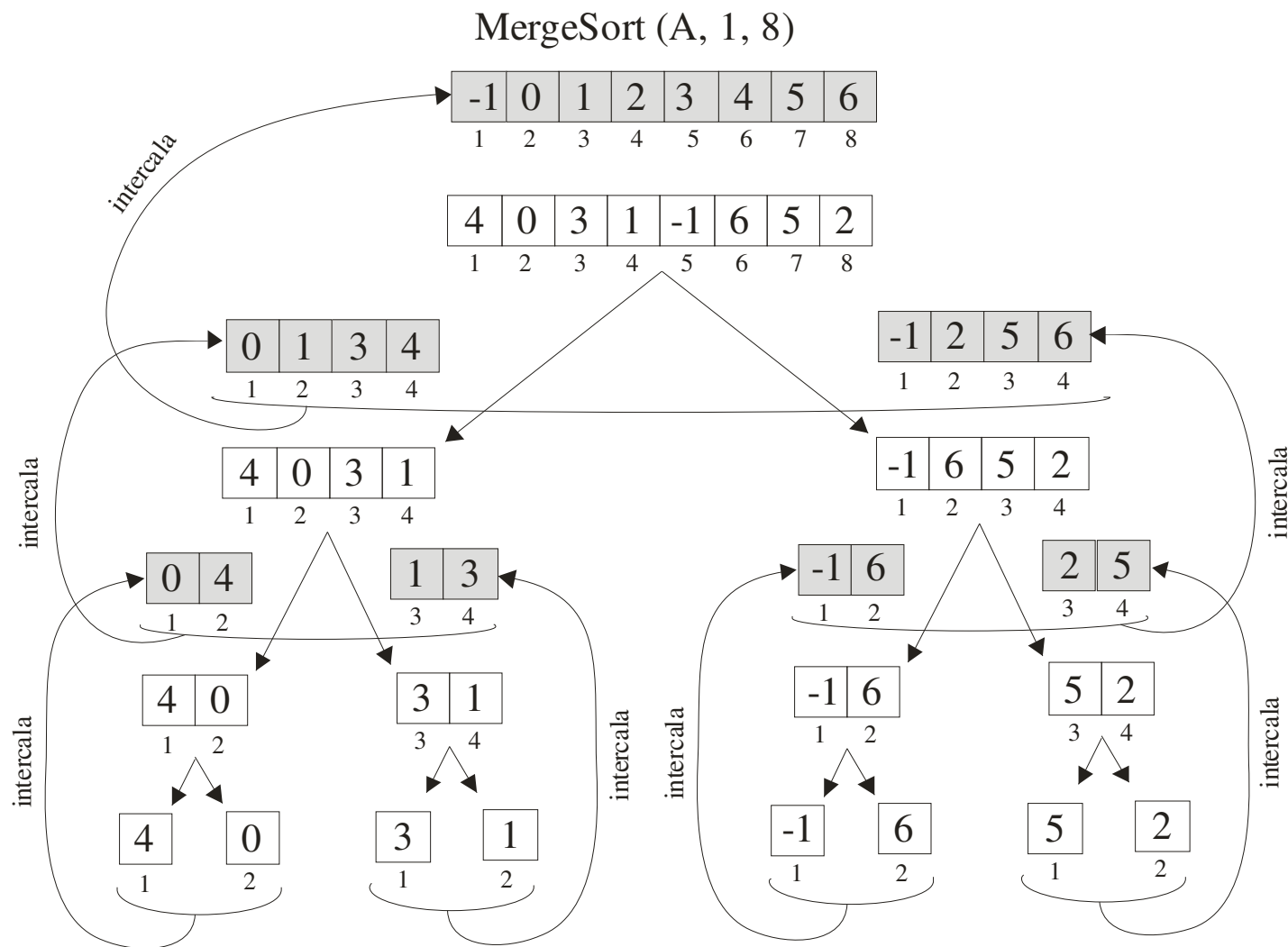
Algoritmo MergeSort (A, i, f )

Entrada: vetor 'A' e índices 'i' e 'f' do vetor ( $i \leq f$ ).

Saída: o vetor A, ordenado “in-loco” do índice 'i' até o índice 'f'.

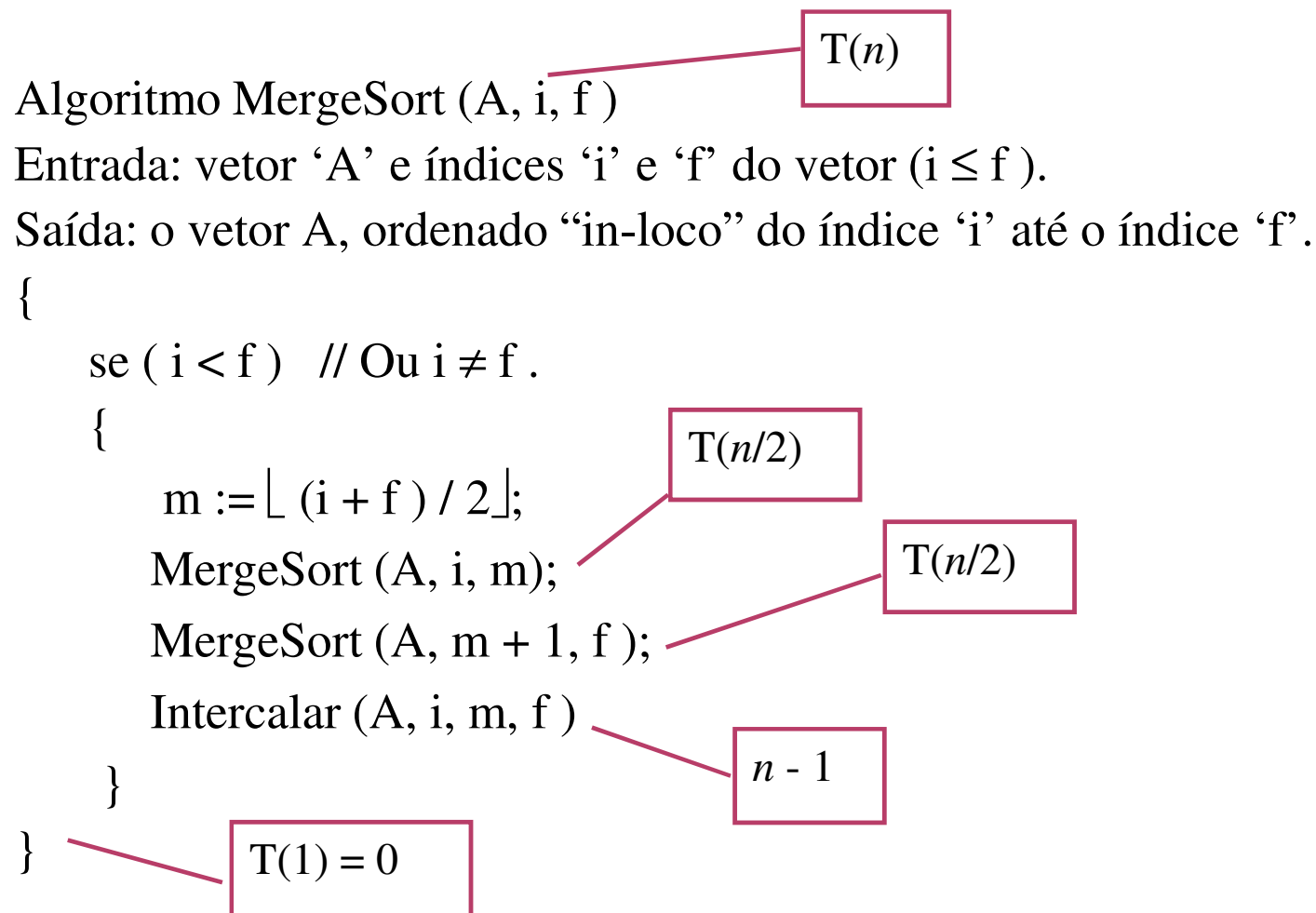
```
{
  se ( i < f ) // Ou i ≠ f .
  {
    m := ⌊ ( i + f ) / 2 ⌋;
    MergeSort (A, i, m);
    MergeSort (A, m + 1, f );
    Intercalar (A, i, m, f )
  }
}
```

# Ilustrando o funcionamento



# Complexidade

Seja a quantidade de elementos  $n = f - i + 1$



Algoritmos “Dividir para Conquistar”  
dão origem a relações de recorrência  
“Dividir para Conquistar”

$$T(n) = 2T(n/2) + n - 1$$

$$T(1) = 0$$

# Solução geral de relações de recorrência do tipo “dividir para conquistar” ( Método da “Divisão e Conquista”)

## **Teorema:**

A solução da relação de recorrência  $T(n) = aT(n/b) + cn^k$ , onde  $a$  e  $b$  são constantes inteiras,  $a \geq 1$ ,  $b \geq 2$  e  $c$  e  $k$  constantes positivas, é

$$T(n) = \begin{cases} O(n^{\log_b a}) & \text{se } a > b^k \\ O(n^k \log n) & \text{se } a = b^k \\ O(n^k) & \text{se } a < b^k \end{cases}.$$

## Exercícios: resolva as seguintes relações de recorrência

1) Merge Sort

$$T(n) = 2T(n/2) + n \quad (\text{desprezamos o } -1)$$

$$T(1) = 0$$

2)  $T(n) = 4T(n/2) + n$

$$T(1) = 0$$

## Exercício

Aplique o Design “Divisão e Conquista” para buscar um elemento em um vetor ordenado.

## Método da substituição

- Algumas relações de recorrência não podem ser facilmente resolvidas pelo método da iteração progressiva e também não se enquadram no modelo que pode ser resolvido pelo método da “Divisão e Conquista”.
- O método da substituição, baseado em indução, pode ser útil nestes casos.

## Entendendo o método da substituição, aplicado a uma relação de recorrência simples

- Exercício: resolva a seguinte relação de recorrência pelo Método da Substituição.

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

$$T(1) = 1$$

## Aplicando o Método da Substituição a uma relação de recorrência de “história completa”

Exercício: Mostre, pelo Método da Substituição, que  $O(n \log(n))$  é solução da relação de recorrência:

$$T(n) = n - 1 + \frac{2}{n} \sum_{i=1}^{n-1} T(i)$$

$$T(1) = 0$$

## Solução da relação de recorrência

Pelo método da substituição.

Postulamos que  $T(n) = O(n \log n)$ , para  $n \geq 1$ .

Precisamos mostrar que  $\exists c > 0$  e  $N \mid T(n) \leq c n \log n, \forall n \geq N$ .

## Solução da relação de recorrência

i) Base

$$T(1) = 0.$$

Para satisfazer à base 1, precisamos  $T(1) = 0 \leq c \cdot 1 \cdot \log 1 = 0$ . Qualquer valor de  $c$  satisfaz.

ii) Redução

É dada pela expressão geral da relação de recorrência, para a qual:

$$T(n) = n - 1 + \frac{2}{n} \sum_{i=1}^{n-1} T(i) \cdot$$

iii) Hipóteses de indução

Como  $T(n)$  reduz para  $T(1)$ ,  $T(2)$ , ...,  $T(n - 1)$  então formulamos o conjunto de hipóteses:

$T(i) \leq c i \log i$ , para  $i \rightarrow 2, 3, 4, \dots, n - 1$ .

## Solução da relação de recorrência

iv) Prova de que o fato de  $T(i) \leq c i \log i$ ,  $i \rightarrow 2, 3, 4, \dots, n - 1$ , implica que  $T(n) \leq c n \log n$ .

$$T(n) = n - 1 + \frac{2}{n} \sum_{i=1}^{n-1} T(i)$$

$$T(n) \leq n - 1 + \frac{2}{n} \sum_{i=1}^{n-1} c i \log i \quad (\text{usamos a hipótese de indução}).$$

$$T(n) \leq n - 1 + \frac{2c}{n} \sum_{i=1}^{n-1} i \log i \quad (\text{tiramos a constante } c \text{ para fora do somatório}).$$

$$T(n) \leq n - 1 + \frac{2c}{n} \left( \frac{1}{2} n^2 \log n - \frac{1}{4} n^2 - 1 \right) \quad (\text{a expressão que substitui o somatório será mostrada depois}).$$

$$T(n) \leq n - 1 + c n \log n - \frac{1}{2} c n - \frac{2c}{n} \quad (\text{aplicamos a distributiva em relação a } 2c/n)$$

$$T(n) \leq n + c n \log n - \frac{1}{2} c n \quad (\text{eliminamos } -1 \text{ e } -2c/n)$$

## Solução da relação de recorrência

$$T(n) \leq cn \log n + \frac{2n - cn}{2} \quad (\text{rearranjamos a ordem dos termos e agrupamos } n \text{ e } -(1/2)cn).$$

Como  $2n - cn \leq 0$  para  $c \geq 2$  (restrição para o valor de  $c$ ),

$$T(n) \leq cn \log n \quad \text{para, digamos, } c = 2 \text{ e } N = 2.$$

Mostrando que  $\sum_{i=1}^{n-1} i \log i \leq \frac{1}{2} n^2 \log n - \frac{1}{4} n^2 - 1$

$$\sum_{i=1}^{n-1} i \log i \leq \int_2^n x \log x dx$$

Vamos resolver a integral pela regra da cadeia, onde  $\int u dv = uv - \int v du$  :

Sejam:

$$u = \log x \text{ e } dv = x dx.$$

Assim:

$$\frac{du}{dx} = \frac{1}{x} \quad \text{então,} \quad du = \frac{1}{x} dx ;$$

$$\int dv = \int x dx \quad \text{então,} \quad v = \frac{1}{2} x^2 ;$$

$$\int x \log x dx = \frac{1}{2} x^2 \log x - \int \frac{1}{2} x^2 \frac{1}{x} dx .$$

Mostrando que  $\sum_{i=1}^{n-1} i \log i \leq \frac{1}{2} n^2 \log n - \frac{1}{4} n^2 - 1$

$$\int x \log x dx = \frac{1}{2} x^2 \log x - \frac{1}{4} x^2.$$

Voltando à integral com limites definidos:

$$\int_2^n x \log x dx = \left[ \frac{1}{2} x^2 \log x - \frac{1}{4} x^2 \right]_2^n = \frac{1}{2} n^2 \log n - \frac{1}{4} n^2 - 2 \log 2 + 1 = \frac{1}{2} n^2 \log n - \frac{1}{4} n^2 - 1.$$

Logo,  $\sum_{i=1}^{n-1} i \log i \leq \frac{1}{2} n^2 \log n - \frac{1}{4} n^2 - 1$ .