

RECURSÃO - PARTE II

Prof. Viviane Bonadia dos Santos

RELEMBRANDO...

“Recursão é uma técnica de resolução de problemas muito poderosa.”

“A solução de um problema é dita **recursiva** quando ela é escrita em função de si própria para instâncias menores do problema.”

EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

Queremos um algoritmo recursivo para calcular potências na base 2:

$$\text{potencia2}(n) = 2^n$$

para $n \geq 0$.

$$2^0 = 1$$

$$2^1 = 2$$

$$2^2 = 4$$

$$2^3 = 8$$

$$2^4 = 16$$

...

EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

Queremos um algoritmo recursivo para calcular potências na base 2:

$$\text{potencia2}(n) = 2^n$$

para $n \geq 0$.

Sabemos que:

1. $\text{potencia2}(0) = 1$
2. $\text{potencia2}(n) = 2 \times \text{potencia2}(n-1)$

EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

Queremos um algoritmo recursivo para calcular potências na base 2:

$$\text{potencia2}(n) = 2^n$$

para $n \geq 0$.

Sabemos que:

1. $\text{potencia2}(0) = 1$ (CASE BASE!)
2. $\text{potencia2}(n) = 2 \times \text{potencia2}(n-1)$

EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

```
/* função recursiva para calcular  
potencias (nao negativas) base 2 */
```

```
int potencia2(int n) {  
    int p;  
    if (n==0) {  
        p = 1;  
    }  
    else (p = 2*potencia2(n-1));  
    return p;  
}
```

EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

```
/* função recursiva para calcular  
potencias (nao negativas) base 2 */
```

```
int potencia2(int n) {  
    int p;  
    if (n==0) {  
        p = 1;  
    }  
    else (p = 2*potencia2(n-1));  
    return p;  
}
```

Memória

n = 2
p

Pilha de execução

potencia2(2)

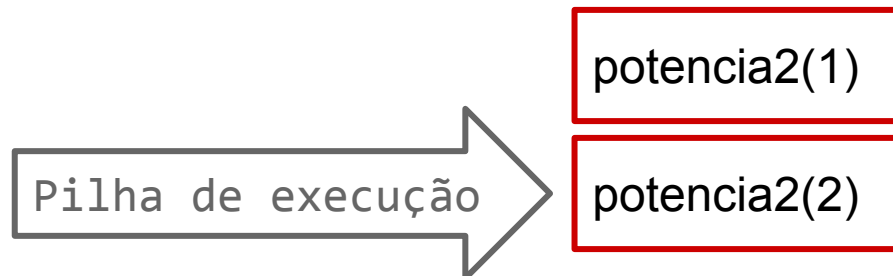
EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

```
/* função recursiva para calcular  
potencias (nao negativas) base 2 */
```

```
int potencia2(int n) {  
    int p;  
    if (n==0) {  
        p = 1;  
    }  
    else (p = 2*potencia2(n-1));  
    return p;  
}
```

Memória

n = 2
p
n = 1
p



EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

```
/* função recursiva para calcular  
potencias (nao negativas) base 2 */
```

```
int potencia2(int n) {  
    int p;  
    if (n==0) {  
        p = 1;  
    }  
    else (p = 2*potencia2(n-1));  
    return p;  
}
```



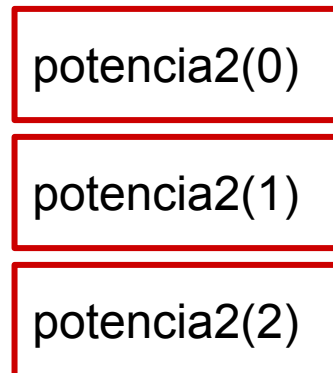
Memória

n = 2
p
n = 1
p
n = 0
p = 1

EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

```
/* função recursiva para calcular  
potencias (nao negativas) base 2 */
```

```
int potencia2(int n) {  
    int p;  
    if (n==0) {  
        p = 1;  
    }  
    else (p = 2*potencia2(n-1));  
    return p;  
}
```



Memória

n = 2
p
n = 1
p
n = 0
p = 1

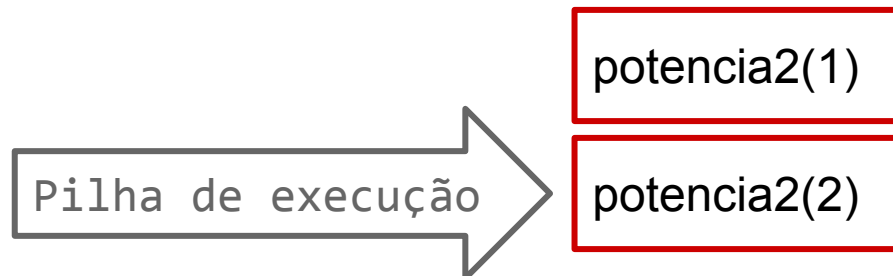
EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

```
/* função recursiva para calcular  
potencias (nao negativas) base 2 */
```

```
int potencia2(int n) {  
    int p;  
    if (n==0) {  
        p = 1;  
    }  
    else (p = 2*potencia2(n-1));  
    return p;  
}
```

Memória

n = 2
p
n = 1
p = 2 * 1



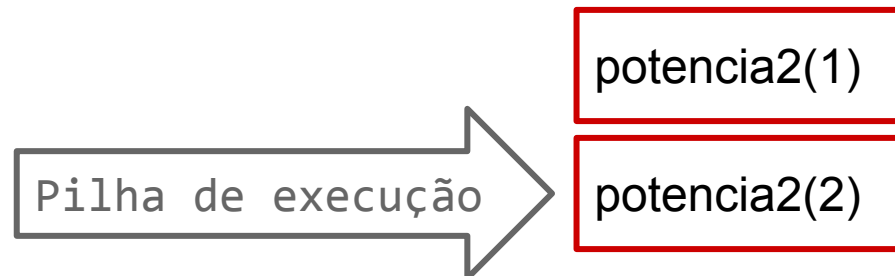
EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

```
/* função recursiva para calcular  
potencias (nao negativas) base 2 */
```

```
int potencia2(int n) {  
    int p;  
    if (n==0) {  
        p = 1;  
    }  
    else (p = 2*potencia2(n-1));  
    return p;  
}
```

Memória

n = 2
p
n = 1
p = 2



EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

```
/* função recursiva para calcular  
potencias (nao negativas) base 2 */
```

```
int potencia2(int n) {  
    int p;  
    if (n==0) {  
        p = 1;  
    }  
    else (p = 2*potencia2(n-1));  
    return p;  
}
```

Memória

n = 2
p = 2 * 2

Pilha de execução

potencia2(2)

EXEMPLO: CALCULAR POTÊNCIAS NA BASE 2

```
/* função recursiva para calcular  
potencias (nao negativas) base 2 */
```

```
int potencia2(int n) {  
    int p;  
    if (n==0) {  
        p = 1;  
    }  
    else (p = 2*potencia2(n-1));  
    return p;  
}
```

Memória

n = 2
p = 4

Pilha de execução

potencia2(2)

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

```
/* versão iterativa */
```

```
int soma(int m, int n) {  
    int soma = 0 , i;  
    if (i = m; i <= n; i++) {  
        soma += i;  
    }  
  
    return soma;  
}
```

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

/* versão recursiva #1

[1...5]

1 + 2 + 3 + 4 + 5

*/

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

/* versão recursiva #1

[1...5]

1 + 2 + 3 + 4 + 5

1 + (1+1) + ((1+1)+1) + (((1+1)+1)+1) + ((((1+1)+1)+1)+1)

*/

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

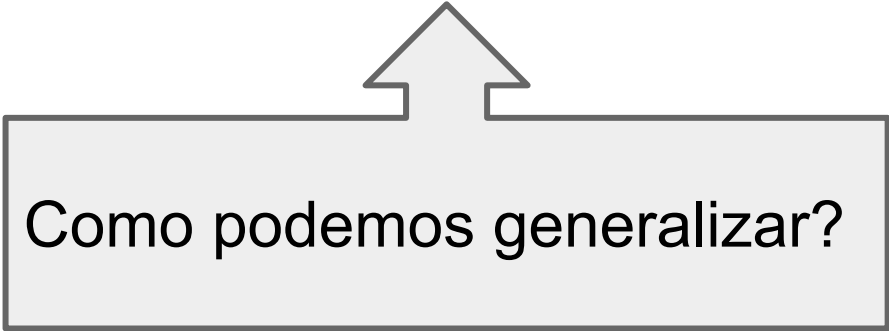
/* versão recursiva #1

[1...5]

1 + 2 + 3 + 4 + 5

1 + (1+1) + ((1+1)+1) + (((1+1)+1)+1) + ((((1+1)+1)+1)+1)

*/



Como podemos generalizar?

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

/* versão recursiva #1

$[1 \dots 5]$

$1 + 2 + 3 + 4 + 5$


$1 + (1+1) + ((1+1)+1) + (((1+1)+1)+1) + ((((1+1)+1)+1)+1)$

$m + (m+1) \dots$

*/

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

/* versão recursiva #1 */

```
int soma(int m, int n) {  
    if  {  
    }  
    else  
        return (m + soma(m+1, n));  
}
```

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

```
/* versão recursiva #1 */
```

```
int soma(int m, int n) {  
    if (m == n) {  
        return n;  
    }  
    else  
        return (m + soma(m+1, n));  
}
```

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

/* versão recursiva #2

$[1 \dots 5]$

$1 + 2 + 3 + 4 + 5$

*/

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

/* versão recursiva #2

[1...5]

“Se conhecemos a soma de m até $n-1$, basta somar n .”

Se $1+2+3+4 = x$, então $1+2+3+4+5 = x + 5$

*/

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

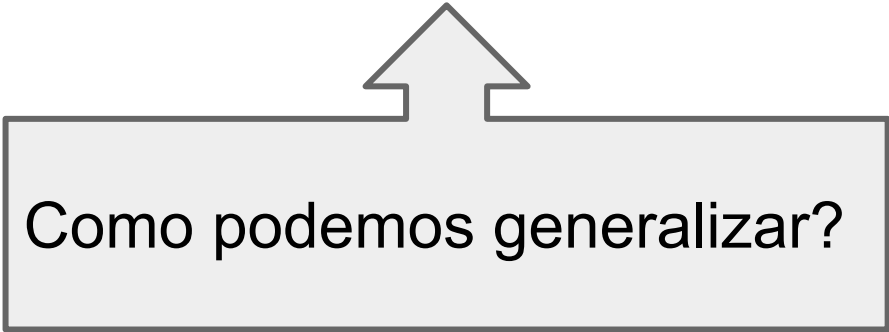
/* versão recursiva #2

[1...5]

“Se conhecemos a soma de m até $n-1$, basta somar n .”

Se $1+2+3+4 = x$, então $1+2+3+4+5 = x + 5$

*/



Como podemos generalizar?

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

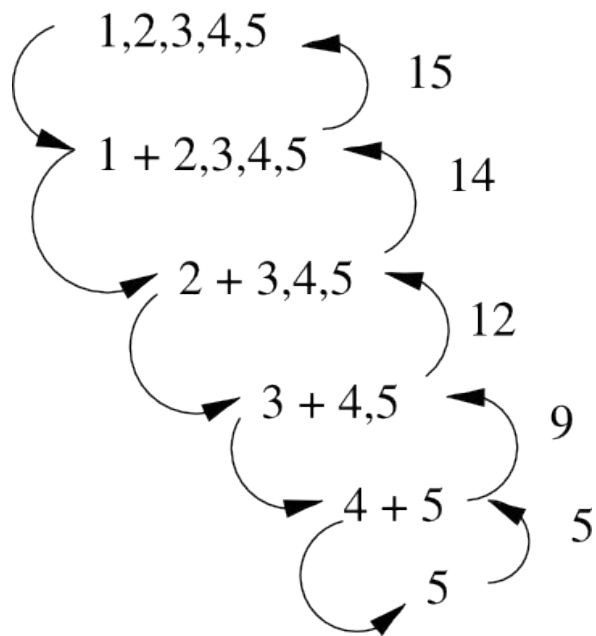
```
/* versão recursiva #2 */
```

```
int soma(int m, int n) {  
    if (m == n) {  
        return m;  
    }  
    else  
        return (soma(m, n-1)+n);  
}
```

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

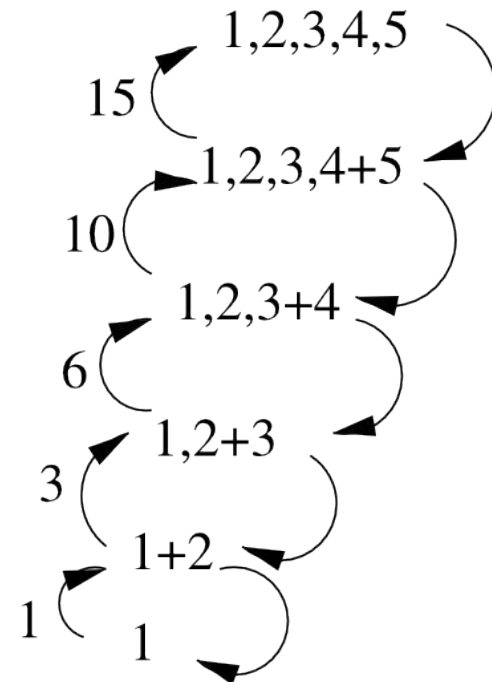
/ versão recursiva #1 */*

Recurso Crescente



/ versão recursiva #2 */*

Recurso Decrescente



EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

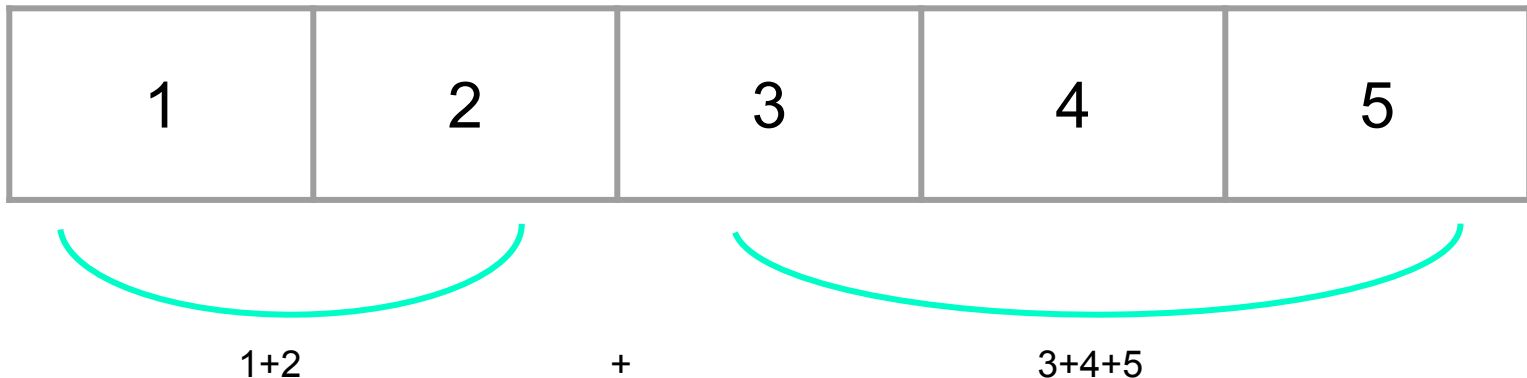
/* versão recursiva #3

Estratégia: divisão e conquista.

$\text{soma}(m, (m+n)/2) + \text{soma}(((m+n)/2)+1, n)$

Dividir o problema em subproblemas similares (divisão), resolver os subproblemas recursivamente e combinar as soluções para obter a solução do problema de tamanho maior (conquista).

*/



EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

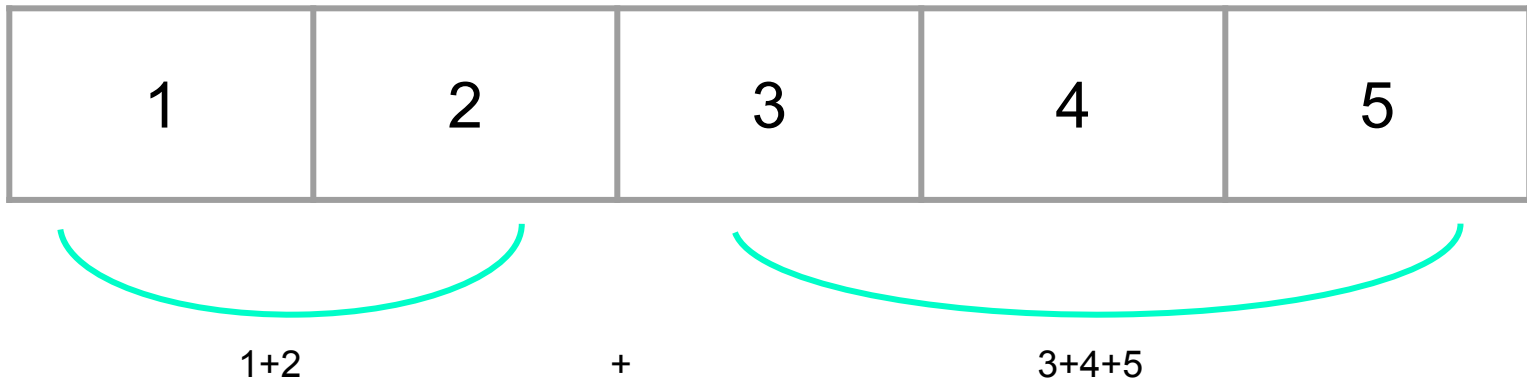
/* versão recursiva #3

Estratégia: divisão e conquista.

$\text{soma}(m, (m+n)/2) + \text{soma}(((m+n)/2)+1, n)$

Dividir o problema em subproblemas similares (divisão), resolver os subproblemas recursivamente e combinar as soluções para obter a solução do problema de tamanho maior (conquista).

*/



EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

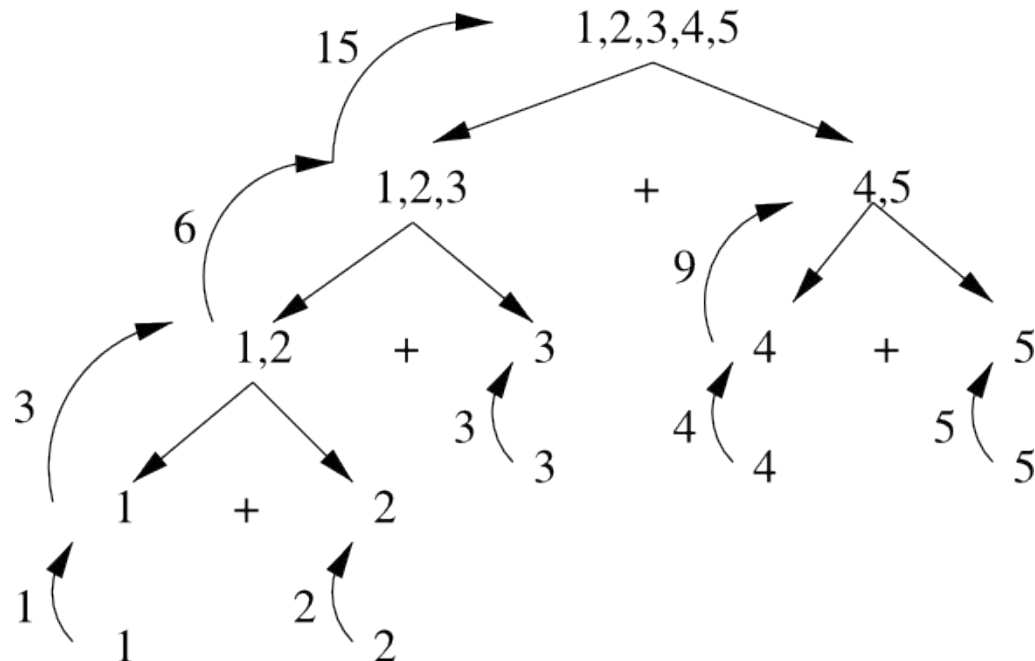
/* versão recursiva #3 */

```
int soma(int m, int n) {  
    if (m == n) {  
        return n;  
    }  
    else  
        return ( soma(m, (n+m)/2) + soma((n+m)/2+1, n) );  
}
```

EXEMPLO: CALCULAR A SOMA DOS NÚMEROS INTEIROS NO INTERVALO $[M, N]$ ($M \leq N$)

/* versão recursiva #3 */

Recursão por Divisão e Conquista



EXEMPLO: ORDENAÇÃO

```
int ordena(int *v, int n) {  
    int i, j;  
    //A condição de parada é não fazer nada. Caso contrário..  
    if(n > 1){  
        //chamada recursiva  
        ordena(v, n-1);  
  
        //insere o último elemento na posição correta  
        i = n-2; j = n-1;  
        while((i >= 0) && v[i] > v[j]){  
            troca(&v[i], &v[j]);  
            i--;  
            j--;  
        }  
    }  
}
```

ALGUMAS REFERÊNCIAS...

- Notas de aula do professor Alexandre Xavier Falcão
 - <http://www.ic.unicamp.br/~mc102/recursao.html>
- Material de estudo da OBI
 - <http://olimpiada.ic.unicamp.br/pratique/estude>