UFRR

UNIVERSIDADE FEDERAL DE RORAIMA

Análise Recursiva

Aula Baseada nos slides do Profº. Eduardo Nakamura- UFAM

Prof. Dr. Herbert Oliveira Rocha
herberthb12@gmail.com

Somatórios

$$\sum_{l=1}^{10000} \sum_{i=1}^{n-5} \sum_{j=i+1}^{n/2} \sum_{k=1}^{n} =$$

Somatórios

$$\sum_{l=1}^{10000} \sum_{i=1}^{n-5} \sum_{j=i+1}^{n/2} \sum_{k=1}^{n} =$$

$$\sum_{l=1}^{10000} \sum_{i=1}^{n-5} \sum_{j=i+1}^{n/2} n =$$

$$\sum_{l=1}^{10000} \sum_{i=1}^{n-5} \left[\frac{n}{2} - (i+1) + 1\right] n =$$

$$\sum_{l=1}^{10000} \sum_{i=1}^{n-5} \left[\frac{n^2}{2} - in\right] =$$

$$\sum_{l=1}^{10000} \left[\sum_{i=1}^{n-5} \frac{n^2}{2} - n \sum_{i-1}^{n-5} i\right] =$$

Somatórios

$$\sum_{l=1}^{10000}\left[\sum_{i=1}^{n-5}\frac{n^2}{2} - n\sum_{i-1}^{n-5} i\right] =$$

$$\sum_{l=1}^{10000}\left[\frac{n^2\,(n-5)}{2} - \frac{n(n-5)(n-4)}{2}\right] =$$

$$\sum_{l=1}^{10000}\left[\frac{n^3 - 5n^2}{2} - \frac{n^3 - 9n^2 + 20n}{2}\right] =$$

$$\sum_{l=1}^{10000}\left[\frac{4n^2 - 20n}{2}\right] =$$

$$\left[\frac{4n^2 - 20n}{2}\right]\sum_{l=1}^{10000} 1 =$$

Somatórios

$$\left[\frac{4n^2 - 20n}{2}\right]\sum_{l=1}^{10000} 1 =$$

$$\left[\frac{4n^2 - 20n}{2}\right]10000 =$$

$$\frac{40000n^2 - 200000\mathrm{n}}{2} =$$

$$20000n^2 - 100000n$$

Qual é o custo do Bubble-sort ?

```
BUBBLE-SORT(A, n)
for i ← 1 to n do
   for j ← i + 1 to n do
      if A[j] < A[i] then
         troca(A[i], A[j]);
      end if
   end for
end for
```

Qual é o custo do Bubble-sort ?

```
BUBBLE-SORT(A, n)
for i ← 1 to n do
   for j ← i + 1 to n do
      if A[j] < A[i] then
         troca(A[i], A[j]);
      end if
   end for
end for
```

Avaliar o número de comparações

Análise de Algoritmos

Qual é o custo do Bubble-sort ?

Qual o custos das operações?

```
BUBBLE-SORT(A, n)
for i ← 1 to n do
   for j ← i + 1 to n do
      if A[j] < A[i] then
         troca(A[i], A[j]);
      end if
   end for
end for
```

$$T(n) = \quad 1$$

Análise de Algoritmos

Qual é o custo do Bubble-sort ?

Qual o custos das operações?

```
BUBBLE-SORT(A, n)
  for i ← 1 to n do
    for j ← i + 1 to n do
      if A[j] < A[i] then
        troca(A[i], A[j]);
      end if
    end for
  end for
```

$$T(n) = \left(\sum_{j=i+1}^{n} 1 \right)$$

Análise de Algoritmos

Qual é o custo do Bubble-sort ?

Qual o custos das operações?

```
BUBBLE-SORT(A, n)
for i ← 1 to n do
  for j ← i + 1 to n do
    if A[j] < A[i] then
      troca(A[i], A[j]);
    end if
  end for
end for
```

$$T(n) = \sum_{i=1}^{n} \left(\sum_{j=i+1}^{n} 1 \right)$$

Bubble-sort - Resolvendo os somatórios temos

$$T(n) = \sum_{i=1}^{n} \left(\sum_{j=i+1}^{n} 1 \right)$$

$$T(n) = \sum_{i=1}^{n} \left(n - (i+1) + 1 \right)$$

Análise de Algoritmos

Bubble-sort - Resolvendo os somatórios temos

$$T(n) = \sum_{i=1}^{n}\left(\sum_{j=i+1}^{n} 1\right)$$

$$T(n) = \sum_{i=1}^{n}\left(n-(i+1)+1\right)$$

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

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

$$T(n) = n^2 - \frac{n^2}{2} - \frac{n}{2} = \frac{1}{2}\left(n^2 - n\right)$$

Recorrência

Uma equação que descreve uma função em termos de seu valor em entradas menores

Exemplo

$$T(n) = \begin{cases} 1 & , \text{ se } n = 1 \\ 2T(n/2) + n & , \text{ se } n > 1 \end{cases}$$

cuja solução é

$$T(n) = n \log n + n = \Theta(n \lg n)$$

Analise o algoritmo abaixo que calcula o n-ésimo elemento de uma PG de razão 2

```
NUM-PG2(n)
  if n = 0 then
     return 1;
  else
     return NUM-PG2(n-1) + NUM-PG2(n-1);
  end if
```

$$T(n) = \begin{cases} 1 & , \quad \text{se } n = 0 \\ 2T(n-1)+1 & , \quad \text{se } n > 0 \end{cases}$$

Soluções de Recorrência

Método de Substituição

- ✓ Define um limite hipotético + Aplicação de indução matemática

Método de Interação

- ✓ Expandir a recorrência
- ✓ Usar propriedade algébricas para determinar um padrão

Método de Árvore de Recursão

- ✓ Converte a recorrência em uma árvore cujos os nós representam os custos envolvidos em diferentes níveis da recursão

Método Mestre

- ✓ Fornece limites para a recorrência em um dado formato

Soluções de Recorrência

Método de Substituição

- ✓ Define um limite hipotético + Aplicação de indução matemática

Método de Interação

- ✓ Expandir a recorrência
- ✓ Usar propriedade algébricas para determinar um padrão

Método de Árvore de Recursão

- ✓ Converte a recorrência em uma árvore cujos os nós representam os custos envolvidos em diferentes níveis da recursão

Método Mestre

- ✓ Fornece limites para a recorrência em um dado formato

Método Mestre

Sejam $a \geq 1$ e $b > 1$ constantes, seja $T(n)$ uma função definida sobre os inteiros não negativos pela recorrência

$$T(n) = \begin{cases} 1 & , \ \text{se } n = 1 \\ aT(n/b) + f(n) & , \ \text{se } n > 1 \end{cases}$$

Método Mestre

Sejam $a \geq 1$ e $b > 1$ constantes, seja $T(n)$ uma função definida sobre os inteiros não negativos pela recorrência

$$T(n) = \begin{cases} 1 & , \text{ se } n = 1 \\ aT(n/b) + f(n) & , \text{ se } n > 1 \end{cases}$$

então $T(n)$ pode ser limitado assintoticamente como a seguir:

1. Se $f(n) = O\left(n^{\log_b a - \epsilon}\right)$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta\left(n^{\log_b a}\right)$
2. Se $f(n) = \Theta\left(n^{\log_b a}\right)$, então $T(n) = \Theta\left(n^{\log_b a} \log n\right)$
3. Se $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ para alguma constante $\epsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$ e n grande, então $T(n) = \Theta(f(n))$

Método Mestre

Sejam $a \geq 1$ e $b > 1$ constantes, seja $T(n)$ uma função definida sobre os inteiros não negativos pela recorrência

$$T(n) = \begin{cases} 1 & , \text{ se } n = 1 \\ aT(n/b) + f(n) & , \text{ se } n > 1 \end{cases}$$

Caso 1: a função $n^{\log_b a}$ for a maior, então a solução será

$$T(n) = \Theta(n^{\log_b a})$$

Caso 2: as duas funções tiverem o mesmo tamanho, teremos

$$T(n) = \Theta\left(n^{\log_b a} \lg n\right) = \Theta(f(n) \lg n)$$

Caso 3: a função $f(n)$ for a maior, então teremos

$$T(n) = \Theta(f(n))$$

Método Mestre

Exemplo #01

$$T(n) = \begin{cases} 1 & , \text{ se } n = 1 \\ T(n/2) + 1 & , \text{ se } n > 1 \end{cases}$$

$a = 1 \qquad f(n) = 1 = \Theta(1)$

$b = 2 \qquad \log_b a = \log_2 1 = 0$

$$f(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^0\right) = \Theta(1)$$

Qual caso?

Método Mestre

Exemplo #01

$$T(n) = \begin{cases} 1 & , \text{ se } n = 1 \\ T(n/2) + 1 & , \text{ se } n > 1 \end{cases}$$

$a = 1$ $\quad$ $f(n) = 1 = \Theta(1)$

$b = 2$ $\quad$ $\log_b a = \log_2 1 = 0$

$$f(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^0\right) = \Theta(1)$$

Caso 2

$$T(n) = \Theta\left(n^{\log_b a} \log n\right) = \Theta\left(n^0 \log n\right) = \Theta(\log n)$$

$$T(n) = \Theta(\log n)$$

Vamos Praticar!

Método Mestre

$$T(n) = 9T\left(\frac{n}{3}\right) + n$$

$$T(n) = T\left(\frac{2n}{3}\right) + 1$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n\log_2 n$$

Vamos Praticar!

Método Mestre

$$T(n) = 9T\left(\frac{n}{3}\right) + n$$

$$T(n) = T\left(\frac{2n}{3}\right) + 1$$

$$\boldsymbol{CASO}\ 1: \boldsymbol{T(n) = \Theta(n^2)}$$

$$\boldsymbol{CASO}\ 2: \boldsymbol{T(n) = \Theta(\log n)}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n\log_2 n$$

$\boldsymbol{NÃO\ SE\ APLICA, POIS\ n^{\log_b a} = n^{\epsilon}}$ é **polinomialmente** $\boldsymbol{maior}$ **que f(n)**

Método de Interação

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1, \\ 2T(n/2) + \Theta(n) & \text{se } n > 1. \end{cases}$$

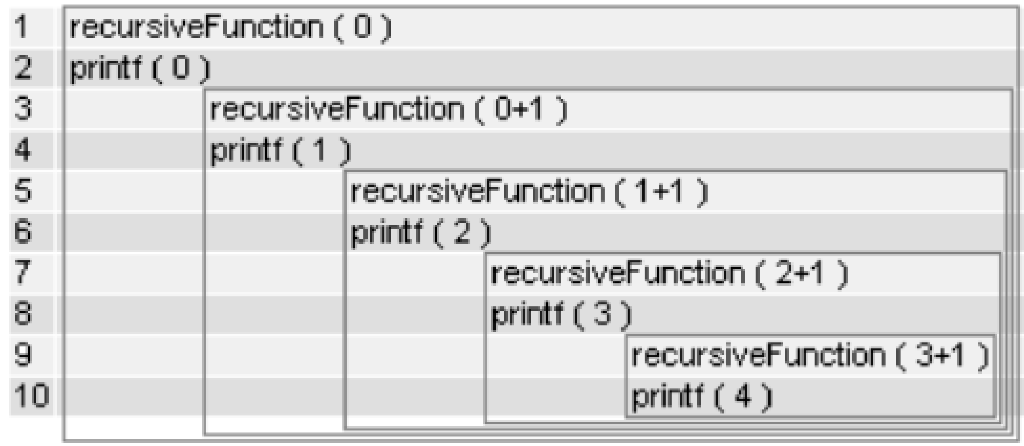
Recursividade

$$fatorial(n) = \begin{cases} 1 & \text{se } n = 0 \\ n \times fatorial(n-1) & \text{se } n > 0 \end{cases}$$

$$\begin{aligned} fatorial(3) &= 3 * fatorial(3-1) \\ &= 3 * fatorial(2) \\ &= 3 * 2 * fatorial(2-1) \\ &= 3 * 2 * fatorial(1) \\ &= 3 * 2 * 1 * fatorial(1-1) \\ &= 3 * 2 * 1 * fatorial(0) \\ &= 3 * 2 * 1 * 1 \\ &= 6 \end{aligned}$$

Recursividade

```c
void recursiveFunction(int num)
{
    if (num < 5)
    {
        printf("%d\n", num);
        recursiveFunction(num + 1);
    }
}
```



Recursividade

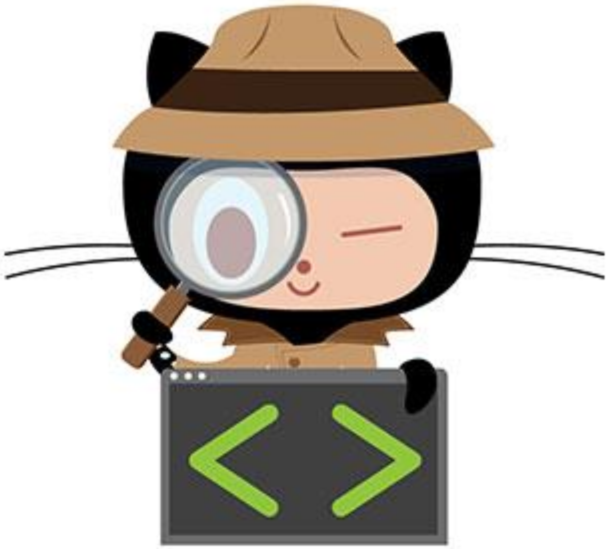
```c
void recursiveFunction(int num)
{
    if (num < 5)
    {
        printf("%d\n", num);
        recursiveFunction(num + 1);
    }
}
```

1	recursiveFunction (0)				
2	printf (0)				
3		recursiveFunction (0+1)			
4		printf (1)			
5			recursiveFunction (1+1)		
6			printf (2)		
7				recursiveFunction (2+1)	
8				printf (3)	
9					recursiveFunction (3+1)
10					printf (4)

```c
void recursiveFunction(int num)
{
    if (num < 5)
    {
        recursiveFunction(num + 1);
        printf("%d\n", num);
    }
}
```

1	recursiveFunction (0)				
2		recursiveFunction (0+1)			
3			recursiveFunction (1+1)		
4				recursiveFunction (2+1)	
5					recursiveFunction (3+1)
6					printf (4)
7				printf (3)	
8			printf (2)		
9		printf (1)			
10	printf (0)				

See you



Perguntas?