

Projeto e Análise de Algoritmos

Recorrências

Prof. Humberto Brandão
humberto@dcc.ufmg.br

*Universidade Federal de Alfenas
Laboratório de Pesquisa e Desenvolvimento (LP&D)
Instituto de Ciências Exatas (ICEx)
versão da aula: 0.3*

Últimas aulas...

▫ Notações

▫ O

▫ Ω

▫ Θ

▫ ω

▫ o

Curiosidade...

Função de custo	Tamanho n					
	10	20	30	40	50	60
n	0,00001 s	0,00002 s	0,00003 s	0,00004 s	0,00005 s	0,00006 s

supondo que uma operação gasta em média 10^{-6} segundos...

Curiosidade...

Função de custo	Tamanho n					
	10	20	30	40	50	60
n	0,00001 s	0,00002 s	0,00003 s	0,00004 s	0,00005 s	0,00006 s
n^2	0,0001 s	0,0004 s	0,0009 s	0,0016 s	0,0.35 s	0,0036 s

Curiosidade...

Função de custo	Tamanho n					
	10	20	30	40	50	60
n	0,00001 s	0,00002 s	0,00003 s	0,00004 s	0,00005 s	0,00006 s
n^2	0,0001 s	0,0004 s	0,0009 s	0,0016 s	0,0.35 s	0,0036 s
n^3	0,001 s	0,008 s	0,027 s	0,64 s	0,125 s	0.316 s

Curiosidade...

Função de custo	Tamanho n					
	10	20	30	40	50	60
n	0,00001 s	0,00002 s	0,00003 s	0,00004 s	0,00005 s	0,00006 s
n^2	0,0001 s	0,0004 s	0,0009 s	0,0016 s	0,0.35 s	0,0036 s
n^3	0,001 s	0,008 s	0,027 s	0,64 s	0,125 s	0.316 s
n^5	0,1 s	3,2 s	24,3 s	1,7 min	5,2 min	13 min

Curiosidade...

Função de custo	Tamanho n					
	10	20	30	40	50	60
n	0,00001 s	0,00002 s	0,00003 s	0,00004 s	0,00005 s	0,00006 s
n^2	0,0001 s	0,0004 s	0,0009 s	0,0016 s	0,0.35 s	0,0036 s
n^3	0,001 s	0,008 s	0,027 s	0,64 s	0,125 s	0.316 s
n^5	0,1 s	3,2 s	24,3 s	1,7 min	5,2 min	13 min
2^n	0,001 s	1 s	17,9 min	12,7 dias	35,7 anos	366 séc.

Curiosidade...

Função de custo	Tamanho n					
	10	20	30	40	50	60
n	0,00001 s	0,00002 s	0,00003 s	0,00004 s	0,00005 s	0,00006 s
n^2	0,0001 s	0,0004 s	0,0009 s	0,0016 s	0,0.35 s	0,0036 s
n^3	0,001 s	0,008 s	0,027 s	0,64 s	0,125 s	0.316 s
n^5	0,1 s	3,2 s	24,3 s	1,7 min	5,2 min	13 min
2^n	0,001 s	1 s	17,9 min	12,7 dias	35,7 anos	366 séc.
3^n	0,059 s	58 min	6,5 anos	3855 séc.	10^8 séc.	10^{13} séc.

Recorrência

- Quando um algoritmo contém uma chamadas recursivas, seu tempo de execução pode frequentemente ser descrito por uma recorrência;

Recorrência

- Quando um algoritmo contém uma chamadas recursivas, seu tempo de execução pode freqüentemente ser descrito por uma recorrência;
- Com o ferramental aprendido até o momento, não somos capazes de analisar a complexidade de algoritmos recursivos;

Recorrência

- Quando um algoritmo contém uma chamadas recursivas, seu tempo de execução pode freqüentemente ser descrito por uma recorrência;
- Com o ferramental aprendido até o momento, não somos capazes de analisar a complexidade de algoritmos recursivos;
- Para os algoritmos recursivos, **a ferramenta principal desta análise não é uma somatória**, mas um tipo especial de equação chamada relação de recorrência.

Recorrência

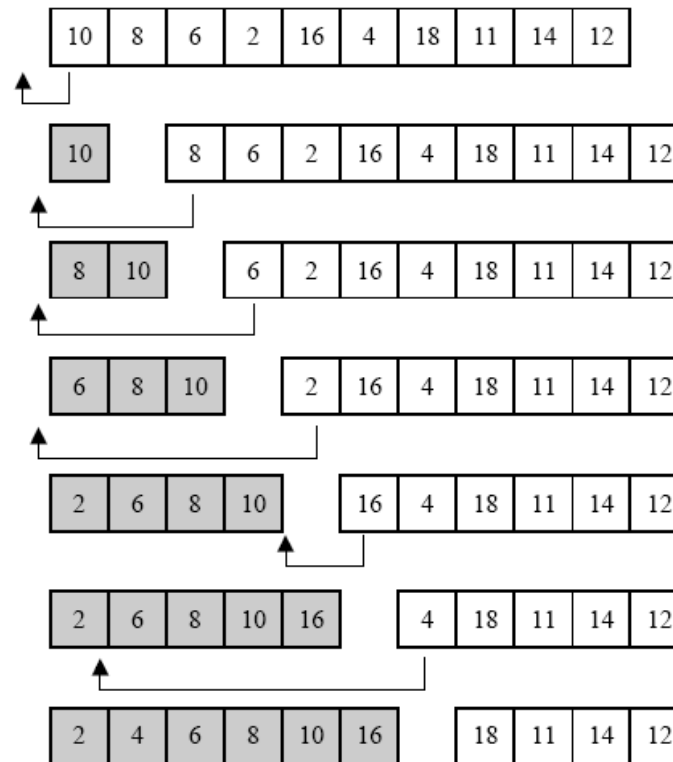
- Quando um algoritmo contém uma chamadas recursivas, seu tempo de execução pode freqüentemente ser descrito por uma recorrência;
- Com o ferramental aprendido até o momento, não somos capazes de analisar a complexidade de algoritmos recursivos;
- Para os algoritmos recursivos, a ferramenta principal desta análise não é uma somatória, mas um tipo especial de equação chamada relação de recorrência.
- Uma recorrência é uma equação ou desigualdade que **descreve uma função em termos de seu valor em entradas menores**;

Recorrência

- Para cada procedimento recursivo é associada uma função de complexidade $T(n)$ desconhecida, onde n mede o tamanho dos argumentos para o procedimento.
- **Equação de recorrência:** maneira de definir uma função por uma expressão envolvendo a mesma função.

Recorrência

- Vamos considerar o algoritmo de ordenação por Inserção.



Recorrência

- Considerando o pior caso:
 - Na primeira vez, apenas uma operação é necessária...
 - Da segunda, 2...
 - Da terceira, 3... E isso é executado... N vezes...
 - Ou seja, todos os elementos serão inseridos na última posição verificada;

Recorrência

- Considerando o pior caso:
 - Na primeira vez, apenas uma operação é necessária...
 - Da segunda, 2...
 - Da terceira, 3... E isso é executado... N vezes...

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

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

$$T(n-2) = T(n-3) + 3$$

$$T(n-3) = T(n-4) + 4$$

...

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

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

Recorrência

- O caso anterior ao caso base é a ordenação de um vetor de duas posições, onde é efetuada apenas uma comparação.
- Então...

$$T(n) = 1 + 2 + 3 + 4 + \dots + (n - 2) + (n - 1)$$

Recorrência

- O caso anterior ao caso base é a ordenação de um vetor de duas posições, onde é efetuada apenas uma comparação.
- Então...

$$T(n) = 1 + 2 + 3 + 4 + \dots + (n-2) + (n-1)$$

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

Recorrência

- O caso anterior ao caso base é a ordenação de um vetor de duas posições, onde é efetuada apenas uma comparação.
- Então...

$$T(n) = 1 + 2 + 3 + 4 + \dots + (n-2) + (n-1)$$

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

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

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

Recorrência

- O caso anterior ao caso base é a ordenação de um vetor de duas posições, onde é efetuada apenas uma comparação.
- Então...

$$T(n) = 1 + 2 + 3 + 4 + \dots + (n-2) + (n-1)$$

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

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

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

$$T(n) = \frac{n^2 - n}{2} \in \Theta(n^2)$$

Recorrência

- Outro exemplo de recorrência:
 - Considere o algoritmo “pouco formal” abaixo:
 - O algoritmo inspeciona n elementos de um conjunto qualquer;
 - De alguma forma, isso permite descartar 2/3 dos elementos e fazer uma chamada recursiva sobre um terço do conjunto original.

```
Algoritmo Pesquisa(vetor)
  if vetor.size() ≤ 1 then
    inspecione elemento;
  else
    inspecione cada elemento recebido (vetor);
    Pesquisa(vetor.subLista(1, (vetor.size() / 3) );
  end if
end.
```

Recorrência

- Montando a equação de recorrência:

```
L1: Algoritmo Pesquisa(vetor)
L2:   if vetor.size() ≤ 1 then
L3:     inspecione elemento;
L4:   else
L5:     inspecione cada elemento recebido (vetor);
L6:     Pesquisa(vetor.subLista(1, (vetor.size() / 3) );
L7:   end if
L8: end.
```

- Caso base da recursão:
 - O custo da linha 2 é $\theta(1)$;
 - O custo da linha 3 é $\theta(1)$;

Recorrência

- Montando a equação de recorrência:

	L1:	Algoritmo Pesquisa(vetor)
$\Theta(1)$	L2:	if <i>vetor.size()</i> ≤ 1 then
$\Theta(1)$	L3:	<i>inspecione elemento</i> ;
	L4:	else
	L5:	inspecione cada elemento recebido (vetor);
	L6:	Pesquisa(<i>vetor.subLista</i> (1, (<i>vetor.size()</i> / 3));
	L7:	end if
	L8:	end.

- Caso geral da recursão:
 - O custo da linha 5 é $\theta(n)$;
 - A linha 6 é onde a própria função Pesquisa é chamada em um conjunto reduzido.

Recorrência

- Montando a equação de recorrência:

	L1:	Algoritmo Pesquisa(vetor)
$\Theta(1)$	L2:	if <i>vetor.size()</i> ≤ 1 then
$\Theta(1)$	L3:	<i>inspecione elemento</i> ;
	L4:	else
$\Theta(n)$	L5:	inspecione cada elemento recebido (vetor);
Chamada recursiva	L6:	Pesquisa(<i>vetor.subLista</i> (1, (<i>vetor.size()</i> / 3)));
	L7:	end if
	L8:	end.

- Monte a equação de recorrência...

Recorrência

$$T(n) = \begin{cases} 1, & \text{se } n \leq 1 \\ T(n/3) + n, & \text{caso contrário} \end{cases}$$

	L1:	Algoritmo Pesquisa(vetor)
$\Theta(1)$	L2:	if <i>vetor.size()</i> ≤ 1 then
$\Theta(1)$	L3:	<i>inspecione elemento</i> ;
	L4:	else
$\Theta(n)$	L5:	inspecione cada elemento recebido (vetor);
Chamada recursiva	L6:	Pesquisa(<i>vetor.subLista</i> (1, (<i>vetor.size()</i> / 3)));
	L7:	end if
	L8:	end.

- Resolva a equação de recorrência...

Recorrência

$$T(n) = \begin{cases} 1, & \text{se } n \leq 1 \\ T(n/3) + n, & \text{caso contrário} \end{cases}$$

$$T(n) = n + T(n/3)$$

$$T(n/3) = n/3 + T(n/3/3)$$

$$T(n/3/3) = n/3/3 + T(n/3/3/3)$$

$$T(n/3/3/3) = n/3/3/3 + T(n/3/3/3/3)$$

...

$$T(n/3/3.../3) = n/3/3/3.../3 + T(n/3/3/3.../3)$$

$$T(1) = 1$$

Recorrência

$$T(n) = \begin{cases} 1, & \text{se } n \leq 1 \\ T(n/3) + n, & \text{caso contrário} \end{cases}$$

$$T(n) = n + T(n/3)$$

$$T(n/3) = n/3 + T(n/9)$$

$$T(n/9) = n/9 + T(n/27)$$

$$T(n/27) = n/27 + T(n/81)$$

...

$$T(n/3^k) = n/3^k + T(n/3^{k+1})$$

$$T(1) = 1$$

$$T(n) = n + n/3 + n/9 + \dots + n/3^k + 1$$

Recorrência

$$T(n) = \begin{cases} 1, & \text{se } n \leq 1 \\ T(n/3) + n, & \text{caso contrário} \end{cases}$$

$$T(n) = n + n/3 + n/3/3 + \dots + n/3/3/3\dots/3 + 1$$

- A formula representa a soma de uma série geométrica de razão $1/3$, multiplicada por n , e adicionada de $T(n/3/3/3/3 \dots /3)$, que é menor ou igual a 1.

$$T(n) = n \cdot \sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^i + 1$$

Recorrência

$$T(n) = \begin{cases} 1, & \text{se } n \leq 1 \\ T(n/3) + n, & \text{caso contrário} \end{cases}$$

$$T(n) = n + n/3 + n/3/3 + \dots + n/3/3/3\dots/3 + 1$$

$$T(n) = n \cdot \sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^i + 1 \rightarrow \text{usando: } \sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$$

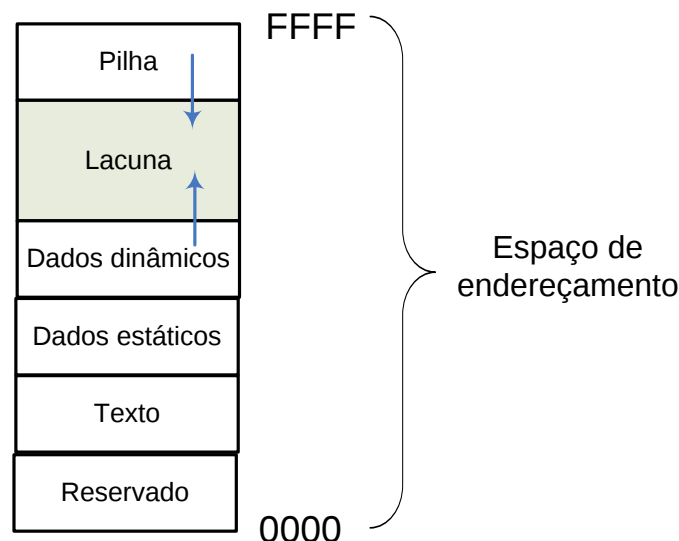
$$T(n) = n \left(\frac{1}{1-1/3} \right) + 1 = \frac{3n}{2} + 1$$

portanto

$$T(n) \in \Theta(n)$$

Recursividade

- Com as arquiteturas atuais, devemos evitar o uso de recursividade quando a solução iterativa é óbvia;
- Exemplos:
 - Fatorial;
 - Fibonacci;
- Lembrem-se do crescimento da pilha de execução nos algoritmos recursivos;



Exercício 1

- Faça a análise de recorrência do algoritmo que calcula o fatorial de um número

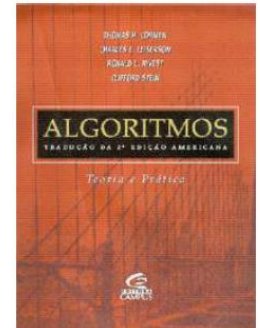
```
L1: Algoritmo Fatorial(n)
L2:   if  $n \leq 1$  then
L3:     retorne 1;
L4:   else
L5:     retorne  $n \cdot \text{fatorial}(n - 1)$ 
L6:   end if
L7: end.
```

Exercício 2

- Implemente os algoritmos de fatorial e fibonacci nas versões:
 - Iterativa;
 - Recursiva.
- Apresente suas curvas de crescimento em função do tamanho do problema (análise experimental).
- Faça uma análise sobre os resultados.

Leitura para a próxima aula

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; (2002). Algoritmos – Teoria e Prática. Tradução da 2ª edição americana. Rio de Janeiro. Editora Campus.
 - Seção 4.3 – O método mestre (teorema mestre)



Bibliografia

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; (2002).
Algoritmos – Teoria e Prática. Tradução da 2ª edição americana.
Rio de Janeiro. Editora Campus.
- TAMASSIA, ROBERTO; GOODRICH, MICHAEL T. (2004).
Projeto de Algoritmos - Fundamentos, Análise e Exemplos da
Internet.
- ZIVIANI, N. (2007). Projeto e Algoritmos com implementações
em Java e C++. São Paulo. Editora Thomson;

