MAC5711

Análise de Algoritmos

2008/1

Agradeço ao Prof. Feofilof por permitir o uso de seu material.

Prof. Rudini Sampaio (DC-UFC)

Paulo Feofiloff

IME, Universidade de São Paulo

Aula 1: Introdução

CLRS: Cormen, Leiserson, Rivest, Stein

- "Having a solid base of algorithmic knowledge and technique is one characteristic that separates the truly skilled programmers from the novices. With modern computing technology, you can accomplish some tasks without knowing much about algorithms, but with a good background in algorithms, you can do much, much more."

- "Uma base sólida de conhecimento e técnica de algoritmos é uma das características que separa o programador experiente do aprendiz. Com a moderna tecnologia de computação, você pode realizar algumas tarefas sem saber muito sobre algoritmos, mas com um boa base em algoritmos você pode fazer muito, muito mais."

O que é AA? Um exemplo

Problema: Rearranjar um vetor em ordem crescente

$A[1\,..\,n]$ é crescente se $A[1] \leq \cdots \leq A[n]$

22	33	33	33	44	55	11	99	22	55	77

11	22	22	33	33	33	44	55	55	77	99

Algoritmo: Rearranja $A[1\,..\,n]$ em ordem crescente

ORDENA-POR-INSERÇÃO (A, n)

```
1  para j ← 2 até n faça
2      chave ← A[j]
3      i ← j − 1
4      enquanto i ≥ 1 e A[i] > chave faça
5          A[i + 1] ← A[i]
6          i ← i − 1
7      A[i + 1] ← chave
```

INSERTION-SORT

Ordenação de cartas, recebendo uma por uma e pondo em ordem.

A[1] é a 1o carta. Não precisa ser ordenada. Por isso, j começa em 2.
A[j] é a j-ésima carta recebida.

"chave" é a carta analisada no momento atual

1						j				n
22	33	33	33	44	55	11	99	22	55	77

Note a documentação

Análise da correção: O algoritmo faz o que prometeu?

- Invariante: no início de cada iteração, $A[1\,..\,j-1]$ é crescente
- Se vale na última iteração, o algoritmo está correto!

ORDENA-POR-INSERÇÃO (A, n)

```
1  para j ← 2 até (*) n faça
2      chave ← A[j]
3      i ← j − 1
4      enquanto i ≥ 1 e A[i] > chave faça
5          A[i + 1] ← A[i]
6          i ← i − 1
7      A[i + 1] ← chave
```

1						j				n
22	33	33	33	44	55	11	99	22	55	77

- vale na primeira iteração
- se vale em uma iteração, vale na seguinte

Análise do desempenho: Quanto tempo consome?

- Regra de três
- Regra de três não funciona!
- Suponha 1 unidade de tempo por linha

linha	total de unidades de tempo		
1	$= n$		
2	$= n-1$		
3	$= n-1$		
4	$\leq 2+3+\cdots+n$	$=$	$(n-1)(n+2)/2$
5	$\leq 1+2+\cdots+(n-1)$	$=$	$n(n-1)/2$
6	$\leq 1+2+\cdots+(n-1)$	$=$	$n(n-1)/2$
7	$= n-1$		
total	$\leq \frac{3}{2}n^2+\frac{7}{2}n-4$	unidades de tempo	

- Algoritmo consome $\leq \frac{3}{2}n^2 + \frac{7}{2}n - 4$ unidades de tempo
- $\frac{3}{2}$ é pra valer?

  Não, pois depende do computador
- n^2 é pra valer?

  Sim, pois só depende do algoritmo
- Queremos um resultado que só dependa do algoritmo

Outra tática:

- número de comparações "$A[i] > chave$"

  $\leq\ 2+3+\cdots+n\ =\ \frac{1}{2}(n-1)(n+2)\ \leq\ \frac{1}{2}n^2+\frac{1}{2}n-1$
- $\frac{1}{2}$ é pra valer?

  Não, pois depende do computador
- n^2 é pra valer?

  Sim, pois só depende do algoritmo
- Queremos resultado que só dependa do algoritmo

- "n^2" é informação valiosa
- mostra evolução do tempo com n:

n	n^2
$2n$	$4n^2$
$10n$	$100n^2$

AA "in a nutshell"

- Problema
- Instâncias do problema
- Tamanho de uma instância
- Algoritmo
- Análise da correção do algoritmo
- Análise do desempenho (velocidades) do algoritmo
- Desempenho em função do tamanho das instâncias
- Pior caso
- Evolução do consumo de tempo em função de n
- Independente do computador e da implementação

Aula 2

Notação O

Notação O: comparação assintótica de funções

- Comparação assintótica grosseira de funções $f(n)$ e $g(n)$
- Qual das duas cresce mais rápido?
- Algo com sabor de $f(n)$ "$\leq$" $g(n)$
- Despreze valores pequenos de n: $n \to \infty$
- Despreze constantes multiplicativas: $100\,n^2$ "$\leq$" n^2
- Despreze termos de ordem inferior: $n^2 + 10n + 10$ "$\leq$" $2\,n^2$

$f(n)$ e $g(n)$ assintoticamente não-negativas

Definição: Dizemos que $f = O(g)$ se existem constantes $c > 0$ e $N > 0$ tais que $f(n) \leq c \cdot g(n)$ para todo $n \geq N$

Em outras palavras: para todo n suficientemente grande, $f(n)$ é dominado por um múltiplo de $g(n)$

Comentários:

- constante = não dependende de n
- "$f \in O(g)$" seria mais correto

Exemplo 1

$n^2 + 10n \;=\; O(n^2)$

Prova:

- se $n \geq 10$ então $n^2 + 10n \leq n^2 + n \cdot n = n^2 + n^2 = 2 \cdot n^2$
- resumo: $n^2 + 10n \leq 2\,n^2$ para todo $n \geq 10$

Como adivinhei que 2 e 10 são bons valores para c e N respectivamente?
Resposta: fiz o seguinte rascunho:

- quero $n^2 + 10n \leq c\,n^2$
- dividindo por n^2, quero $1 + 10/n \leq c$
- se $n \geq 10$ então $1 + 10/n \leq 2$
- parece que basta tomar $c \geq 2$ e $N \geq 10$

Exemplo 2

$9n + 9 = O(n^2)$

Prova:

- se $n \geq 9$ então $9n + 9 \leq n \cdot n + n^2 = 2 \cdot n^2$
- resumo: $9n + 9 \leq 2n^2$ para todo $n \geq 9$

Outra prova:

- se $n \geq 1$ então $9n + 9 \leq 9n \cdot n + 9 \cdot n^2 = 18\,n^2$

Exemplo 3

$n^2 \stackrel{?}{=} O(9n + 9)$

Não é verdade! Prova, por contradição:

- suponha que existem c e N tais que $n^2 \leq c \cdot (9n + 9)$ para todo $n \geq N$
- então $n^2 \leq c \cdot (9n + 9n) = 18\,c\,n$ para todo $n \geq N$
- então $n \leq 18\,c$ para todo $n \geq N$
- absurdo

Exemplo 4

$3n = O(2^n)$

Prova:

- vou mostrar que $3n \leq 2^n$ para $n \geq 4$
- prova por indução:
- se $n = 4$ então $3n = 3 \cdot 4 = 12 \leq 16 = 2^4 = 2^n$
- se $n > 4$ então

$$\begin{aligned} 3n &= 3(n-1+1) \\ &= 3(n-1) + 3 \\ &\leq 3(n-1) + 3(n-1) \\ &\leq 2^{n-1} + 2^{n-1} \quad \text{(hipótese de indução)} \\ &= 2^n \end{aligned}$$

Também poderia mostrar que $3n \leq 2 \cdot 2^n$ para $n \geq 1$

Exemplo 5

$\log_2 n = O(n)$

Prova:

- $n \leq 2^n$ quando $n \geq 1$ (como no exemplo anterior)
- $\log_2$ é crescente
- logo, $\log_2 n \leq n$ quando $n \geq 1$

$a \leq b$ se e somente se $2^a \leq 2^b$

logo, $m \leq n$ implica $\log_2 m \leq \log_2 n$

Bases de logaritmos

- $\log_2 n = \dfrac{\log_3 n}{\log_3 2} = \dfrac{1}{\log_3 2} \log_3 n$

- todos os log, qualquer que seja a base, estão na mesma order O

Reprise: algoritmo da ordenação por inserção

Desempenho de ORDENA-POR-INSERÇÃO:

- algoritmo consome $O(n^2)$ unidades de tempo
- não é verdade que consome $O(n)$ unidades de tempo
- existem instâncias que levam o algoritmo a consumir tempo proporcional a n^2

Aula 3

Problema da intercalação (merge)

Mergesort

Novo problema/algoritmo: intercalação

Problema: Rearranjar um vetor em ordem crescente sabendo que o lado e o lado direito são crescentes

Sei fazer em $O(n^2)$ unidades de tempo

Dá pra melhorar?

p				q						r
22	44	55	88	88	33	66	77	99	99	99

p				q						r
22	33	44	55	66	77	88	88	99	99	99

Possíveis valores de p, q e r?

Algoritmo: Recebe $A[p\,..\,r]$ tal que $A[p\,..\,q]$ e $A[q{+}1\,..\,r]$ são crescentes. Rearranja o vetor todo em ordem crescente.

```
INTERCALA (A, p, q, r)
1   n1 ← q − p + 1
2   n2 ← r − q
3   crie vetores L[1 .. n1 + 1] e R[1 .. n2 + 1]
4   para i ← 1 até n1 faça L[i] ← A[p + i − 1]
5   para j ← 1 até n2 faça R[j] ← A[q + j]
6   L[n1 + 1] ← R[n2 + 1] ← ∞
⋮   ⋮
```



```
⋮   ⋮
7   i ← j ← 1
8   para k ← p até r faça
9     se L[i] ≤ R[j]
0       então A[k] ← L[i]
1             i ← i + 1
2       senão A[k] ← R[j]
3             j ← j + 1
```

Análise do desempenho:

- tamanho de instância: $n := r - p + 1$
- algoritmo consome $O(n)$ unidades de tempo
- isso é muito rápido!

Novo problema/algoritmo: Mergesort

Problema: Rearranjar um vetor em ordem crescente

Algoritmo: Rearranja $A[p\,..\,r]$ em ordem crescente supondo $p \le r$

```
MERGE-SORT (A, p, r)
1  se p < r
2    então q ← ⌊(p + r)/2⌋
3          MERGE-SORT (A, p, q)
4          MERGE-SORT (A, q + 1, r)
5          INTERCALA (A, p, q, r)
```

Método de divisão e conquista

Segredo da velocidade do MERGE-SORT: INTERCALA é rápido

Desempenho: Quanto tempo MERGE-SORT consome?

- Tamanho de instância: $n := r - p + 1$
- $T(n)$:= tempo para pior instância de tamanho n
- Quero cota superior: $T(n) = O(n \lg n)$
- Se $n > 1$ então $T(n) \le T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + f(n)$

  sendo $f(n)$ é consumo de tempo de INTERCALA: $f(n) = O(n)$
- Vamos provar que $T(n) = O(n \lg n)$

Notação: $\lg n := \log_2 n$ e $\ln n := \log_e n$

Aula 4

Análise do Mergesort

Solução de recorrências

Novo problema/algoritmo: Mergesort

Problema: Rearranjar um vetor em ordem crescente

Algoritmo: Rearranja $A[p\,..\,r]$ em ordem crescente, supondo $p \le r$

```
MERGE-SORT (A, p, r)
1  se p < r
2     então q ← ⌊(p + r)/2⌋
3           MERGE-SORT (A, p, q)
4           MERGE-SORT (A, q + 1, r)
5           INTERCALA (A, p, q, r)
```

Desempenho: Quanto tempo MERGE-SORT consome?

- Tamanho de instância: $n := r - p + 1$
- $T(n)$:= tempo de pior instância de tamanho n
- Vou mostrar que $T(n) = O(n \lg n)$


```
MERGE-SORT (A, p, r)
1  se p < r
2     então q ← ⌊(p + r)/2⌋
3           MERGE-SORT (A, p, q)
4           MERGE-SORT (A, q + 1, r)
5           INTERCALA (A, p, q, r)
```

$$T(n) \le \begin{cases} a & \text{se } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + f(n) & \text{se } n = 2, 3, 4, 5, \ldots \end{cases}$$

a é o consumo de tempo da linha 1

$f(n)$ é o consumo de INTERCALA mais o das linhas 1 e 2

Queremos uma "fórmula fechada" para $T(n)$

Exemplo: solução de recorrência

Recorrência semelhante à do Mergesort:

$$T(n) = \begin{cases} 3 & \text{se } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 10\,n & \text{se } n = 2, 3, 4, , \ldots \end{cases}$$

Recorrência define função T sobre inteiros positivos:

n		$T(n)$
1	3	3
2	$3 + 3 + 10 \cdot 2$	26
3	$3 + 26 + 10 \cdot 3$	59
4	$26 + 26 + 10 \cdot 4$	92
5	$26 + 59 + 10 \cdot 5$	135

Quero fórmula fechada $T(n) = \cdots$

Não sei dar uma fórmula fechada...

Vou começar tentando algo mais simples: somente potências de 2

$$S(n) = \begin{cases} 3 & \text{se } n = 1 \\ 2\,S(\frac{n}{2}) + 10\,n & \text{se } n = 2^1, 2^2, 2^3, \ldots \end{cases}$$

Fórmula fechada: $S(n) = 10\,n \lg n + 3n$

Prova?

Lembrete: $\lg n := \log_2 n$

Teorema: $S(n) = 10\,n \lg n + 3n$ para $n = 2^0, 2^1, 2^2, 2^3, \ldots$

Prova, por indução:

- Base: se $n = 1$ então $S(n) = 3 = 10n \lg n + 3n$
- Passo: se $n > 1$ então

$$\begin{aligned} S(n) &= 2\,S(\tfrac{n}{2}) + 10\,n \\ &= 2\left(10\,\tfrac{n}{2} \lg \tfrac{n}{2} + 3\tfrac{n}{2}\right) + 10\,n \quad \text{(hipótese de indução)} \\ &= 10\,n(\lg n - 1) + 3n + 10\,n \\ &= 10\,n \lg n - 10n + 3n + 10\,n \\ &= 10\,n \lg n + 3n \end{aligned}$$

Resposta mais grosseira: $S(n) = O(n \lg n)$

Prova: ...

Como adivinhei a fórmula "$10\,n \lg n + 3n$"?

Desenrolei a recorrência:

$$\begin{aligned} S(n) &= 2\,S(n/2) + 10\,n \\ &= 4\,S(n/4) + 20\,n \\ &= 8\,S(n/8) + 30\,n \\ &\vdots \quad \vdots \\ &= n\,S(1) + 10\,\lg n\,n \\ &= 3\,n + 10\,n\,\lg n \end{aligned}$$

De volta ao desempenho do Mergesort

$$T(n) \;\le\; T\Big(\Big\lceil \frac{n}{2} \Big\rceil\Big) + T\Big(\Big\lfloor \frac{n}{2} \Big\rfloor\Big) \;+\; O(n)$$

Diferenças em relação ao exemplo anterior:

- "$\le$" no lugar de "$=$"
- $\lceil . \rceil$ e $\lfloor . \rfloor$
- "$1, 2, 3, 4, 5, \ldots$" no lugar de "$2^0, 2^1, 2^2, 2^3, \ldots$"
- "$O(n)$" no lugar de "$f(n)$"

Apesar dessas diferenças, CLRS (sec.4.3 p.73–75) garante que

$$T(n) = O(n \lg n)$$

Exercício

Resolva a recorrência

$$R(n) \;=\; \begin{cases} a & \text{se } n = 1 \\ 2\,R(\frac{n}{2}) + b\,n & \text{se } n = 2^1, 2^2, 2^3, \ldots \end{cases}$$

Solução: $R(n) = b\,n \lg n + a\,n$

Prova, por indução:

- Base: se $n = 1$ então $R(n) = a$ e $bn \lg n + an = a$
- Passo: se $n > 1$ então

$$\begin{aligned} R(n) &= 2\,R(\tfrac{n}{2}) + b\,n \\ &= 2\left(b\tfrac{n}{2} \lg \tfrac{n}{2} + a\tfrac{n}{2}\right) + b\,n \\ &= b\,n(\lg n - 1) + a\,n + b\,n \\ &= b\,n \lg n - b\,n + a\,n + b\,n \\ &= b\,n \lg n + a\,n \end{aligned}$$

Exercício

O algoritmo supõe $n \ge 1$ e devolve o valor de um elemento máximo de $A[1\,..\,n]$

```
MAX (A, n)
1  se n = 1
2      então devolva A[1]
3      senão x ← MAX (A, n − 1)
4            se x ≥ A[n]
5                então devolva x
6                senão devolva A[n]
```

Análise do desempenho:

- tamanho de instância: n
- $T(n)$: tempo de pior caso para instância de tamanho n

$$T(n) \;\le\; \begin{cases} a & \text{se } n = 1 \\ T(n-1) + b & \text{se } n = 2, 3, 4, 5, \ldots \end{cases}$$

sendo a e b constantes

Teorema: $T(n) \le a + b(n-1)$ para todo inteiro positivo n

Prova: . . .

Colorário: $T(n) = O(n)$

Prova: . . .