

Aula 05: Análise matemática de algoritmos não recursivos

David Déharbe

Programa de Pós-graduação em Sistemas e Computação

Universidade Federal do Rio Grande do Norte

Centro de Ciências Exatas e da Terra

Departamento de Informática e Matemática Aplicada

Introdução

Um exemplo introdutório

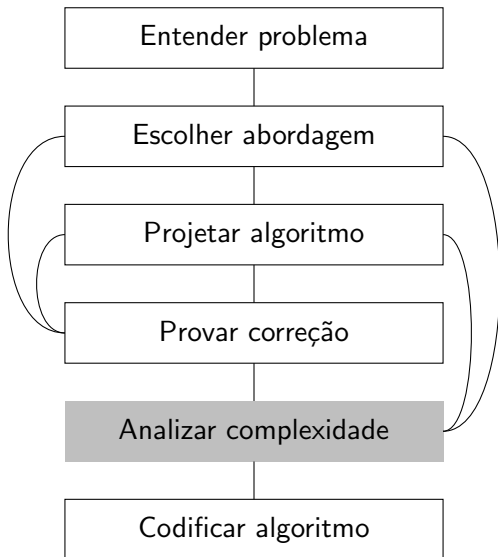
Estratégia de análise

Um segundo exemplo

Um terceiro exemplo

Um exemplo não tão simples

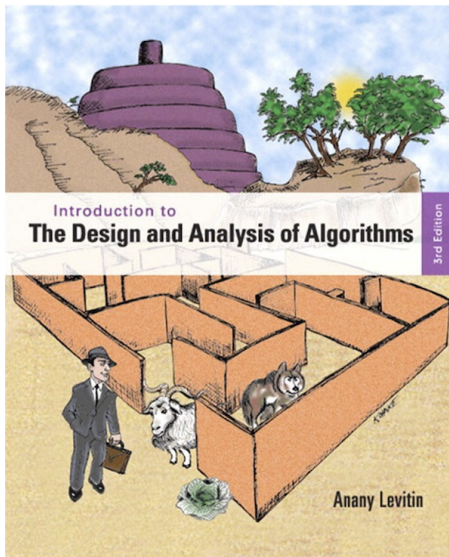
Prática



Estrutura da apresentação

1. arcabouço teórico;
2. melhor caso, pior caso, caso médio;
3. notações assintóticas; O , Ω , Θ ;
4. análise de algoritmos não recursivos;
5. análise de algoritmos recursivos.

Bibliografia usada



(seção 2.3)

Exemplo (Busca do maior elemento em uma sequência)

MAX-ITEM(A)

```
1   $max = A[1]$   
2  for  $i = 2$  to  $length[A]$   
3      if  $A[i] > max$   
4           $max = A[i]$   
5  return  $max$ 
```

Exemplo

MAX-ITEM(A)

```
1   $max = A[1]$ 
2  for  $i = 2$  to  $length[A]$ 
3      if  $A[i] > max$ 
4           $max = A[i]$ 
5  return  $max$ 
```

- ▶ Tamanho da entrada: $length[A]$, digamos n .
- ▶ Operações mais executadas: corpo do **for**.
 - ▶ A comparação é executada mais vezes que a atribuição
 - ▶ A comparação $A[i] > max$ é a *operação básica*
 - ▶ O número de comparações executadas é o mesmo para qualquer entrada de tamanho n
 - ⇒ Não precisa estudar pior caso, caso médio, nem melhor caso.

Exemplo

MAX-ITEM(A)

```
1   $max = A[1]$ 
2  for  $i = 2$  to  $length[A]$ 
3      if  $A[i] > max$ 
4           $max = A[i]$ 
5  return  $max$ 
```

Seja $C(n)$ o número de comparações realizadas.

- ▶ A cada iteração, é realizada uma (1) comparação.
- ▶ $C(n) = \sum_{i=2}^n 1 = n - 2 + 1 = n - 1 \in \Theta(n)$.
- ▶ (resultado esperado)

Estratégia para analisar complexidade de algoritmos não recursivos

1. Escolher um parâmetro (ou vários) para indicar o tamanho da entrada
2. Identificar a operação básica do algoritmo
3. Verificar se o número de vezes que a operação básica é executada depende apenas do tamanho da entrada
⇒ Caso contrário, o pior caso, o caso médio e o melhor caso devem ser aferidos individualmente
4. Escreva um somatório que expressa quantas vezes a operação básica é executada
5. Usando as leis da aritmética, encontre uma formulação que só depende de n , ou pelo menos o seu crescimento assintótico.

Propiedades importantes

$$\sum_{i=L}^U (c \times a_i) = c \times \left(\sum_{i=L}^U a_i \right)$$

$$\sum_{i=L}^U (a_i \pm b_i) = \left(\sum_{i=L}^U a_i \right) \pm \left(\sum_{i=L}^U b_i \right)$$

$$\sum_{i=L}^U 1 = U - L + 1$$

$$\sum_{i=1}^n i = \frac{n \times (n+1)}{2} \approx \frac{1}{2} \times n^2 \in \Theta(n^2)$$

Um segundo exemplo

Exemplo (Teste de unicidade dos elementos)

UNICITY-ELEMS(A)

```
1  for  $i = 1$  to  $\text{length}[A] - 1$   
2      for  $j = i + 1$  to  $\text{length}[A]$   
3          if  $A[i] = A[j]$   
4              return FALSE  
5  return TRUE
```

Aplicação da estratégia

UNICITY-ELEMS(A)

```
1  for  $i = 1$  to  $\text{length}[A] - 1$ 
2      for  $j = i + 1$  to  $\text{length}[A]$ 
3          if  $A[i] = A[j]$ 
4              return FALSE
5  return TRUE
```

1. O tamanho da entrada é o número de elementos de A , digamos n .
2. O laço mais interno contém um teste de igualdade. A operação básica é o *teste de igualdade*.
3. O número de testes efetuados não depende só de n , mas também do conteúdo de A . Mas precisamente se tem elementos iguais, e quais posições eles ocupam.

Vamos nos interessar à complexidade **no pior caso**: $C_{\text{pior}}(n)$.



Aplicação da estratégia

UNICITY-ELEMS(A)

```
1  for  $i = 1$  to  $\text{length}[A] - 1$ 
2      for  $j = i + 1$  to  $\text{length}[A]$ 
3          if  $A[i] = A[j]$ 
4              return FALSE
5  return TRUE
```

O pior caso é quando o número de comparações é o maior possível.

1. ou todos os elementos são distintos dois a dois;
2. ou apenas os elementos nas últimas duas posições são iguais.

$$C_{\text{pior}}(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n 1$$

Aplicação da estratégia

UNICITY-ELEMS(A)

```
1  for  $i = 1$  to  $\text{length}[A] - 1$ 
2      for  $j = i + 1$  to  $\text{length}[A]$ 
3          if  $A[i] = A[j]$ 
4              return FALSE
5  return TRUE
```

$$\begin{aligned}C_{\text{pior}}(n) &= \sum_{i=1}^{n-1} \sum_{j=i+1}^n 1 = \sum_{i=1}^{n-1} n - (i + 1) + 1 = \sum_{i=1}^{n-1} n - i \\&= \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i = (n - 1) \times n - \frac{(n - 1) \times n}{2} \\&= \frac{(n - 1) \times n}{2} \approx \frac{1}{2} \times n^2 \in \Theta(n^2)\end{aligned}$$

Um terceiro exemplo

Exemplo

Multiplicação de duas matrizes quadradas

MULT-MATRIX(A, B, C)

```
1  for  $i = 1$  to  $NbLines[A]$ 
2      for  $j = 1$  to  $NbCols[B]$ 
3           $C[i, j] = 0$ 
4          for  $k = 1$  to  $NbCols[A]$ 
5               $C[i, j] = C[i, j] + A[i, k] \times B[k, j]$ 
```

Aplicação da estratégia

MULT-MATRIX(A, B, C)

```
1  for  $i = 1$  to  $NbLines[A]$ 
2      for  $j = 1$  to  $NbCols[B]$ 
3           $C[i, j] = 0$ 
4          for  $k = 1$  to  $NbCols[A]$ 
5               $C[i, j] = C[i, j] + A[i, k] \times B[k, j]$ 
```

- ▶ Tamanho (digamos n): ordem das matrizes A, B, C .
Obs. número de elementos m das matrizes é tal que $m = n^2$.
- ▶ Operação básica (laço mais aninhado): 1 adição + 1 multiplicação.

$$M(n) = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n 1 = \sum_{i=1}^n \sum_{j=1}^n n = \sum_{i=1}^n n^2 = n^3.$$

Um exemplo não tão simples

Exemplo

Tamanho da representação binária de um inteiro

Aplicação da estratégia

- ▶ A entrada é um inteiro n . O tamanho da entrada é o próprio n .
- ▶ A operação executada mais vezes é a comparação $n > 1$. É igual ao número de vezes que o laço é executado mais um.
- ▶ n apenas recebe alguns valores da faixa formada pelos seus valores inicial e final.
- ▶ O valor de n é dividido a cada iteração.
- ▶ O número de vezes que a operação básica é executada é $\lfloor \log_2 n \rfloor + 1$.
- ▶ Utilizaremos uma **relação de recorrência** para mostrar isto.

Exercício (Cálculo de variação de amostra)

$$\begin{aligned} \mathcal{V}(x_1 \cdots x_n) &= \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1} \text{ onde } \bar{x} = \frac{\sum_{i=1}^n x_i}{n} \\ &= \frac{\sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2/n}{n-1}. \end{aligned}$$

Encontre e compare a quantidade de cada tipo de operação para calcular a variação de amostra com cada uma das fórmulas.

Análise de algoritmo 1

Exercício

MYSTERY-1(n)

```
1   $s = 0$   
2  for  $i = 1$  to  $n$   
3       $s = s + i \times i$   
4  return  $s$ 
```

1. *O que este algoritmo calcula?*
2. *Qual a sua operação básica? Quantas vezes é executada?*
3. *Qual a classe de complexidade do algoritmo?*
4. *Existe um algoritmo mais eficiente? Se existir, qual a classe de complexidade dele? Senão, porque?*

Análise de algoritmo 2

Exercício

MYSTERY-2(A)

```
1   $m_1 = A[1], m_2 = A[1]$ 
2  for  $i = 2$  to  $length[A]$ 
3      if  $A[i] < m_1$  then  $m_1 = A[i]$ 
4      if  $A[i] > m_2$  then  $m_2 = A[i]$ 
5  return  $m_2 - m_1$ 
```

1. *O que este algoritmo calcula?*
2. *Qual a sua operação básica? Quantas vezes é executada?*
3. *Qual a classe de complexidade do algoritmo?*
4. *Existe um algoritmo mais eficiente? Se existir, qual a classe de complexidade dele? Senão, porque?*



Análise de algoritmo 3

Exercício

MYSTERY-3(A)

```
1  for  $i = 1$  to  $NbLines[A] - 1$ 
2      for  $j = i + 1$  to  $NbLines[A]$ 
3          if  $A[i, j] \neq A[j, i]$  then return FALSE
4  return TRUE
```

1. *O que este algoritmo calcula?*
2. *Qual a sua operação básica? Quantas vezes é executada?*
3. *Qual a classe de complexidade do algoritmo?*
4. *Existe um algoritmo mais eficiente? Se existir, qual a classe de complexidade dele? Senão, porque?*