Ciência da Computação

Análise Assintótica de Algoritmos Iterativos

André Luiz Brun

Introdução

- Para descobrirmos a complexidade assintótica de um algoritmo é necessário analisar seu código (ou pseudocódigo) e construir sua função de complexidade

- Essa função determinará então, a complexidade assintótica do algoritmo

- O processo de análise pode ser iterativo ou recursivo. nessa aula trabalharemos com o primeiro.

Pior, Melhor e Caso Médio

- Exemplo: busca por um elemento em um conjunto

	0	1	2	3	4	5	6	7
Vetor	27	18	17	3	6	1	10	5

Pior, Melhor e Caso Médio

- Exemplo: busca por um elemento em um conjunto

	0	1	2	3	4	5	6	7
Vetor	27	18	17	3	6	1	10	5

- Melhor caso: encontrar o elemento já no primeiro teste (posição 0)
- nº Testes: 1
- Complexidade seria O(1)

Pior, Melhor e Caso Médio

- Exemplo: busca por um elemento em um conjunto

	0	1	2	3	4	5	6	7
Vetor	27	18	17	3	6	1	10	5

- Pior caso: encontrar o elemento no último teste (posição 7) ou não encontrar o elemento.
- Seria necessário executar a consulta n vezes
- nº Testes: 8 → n
- Complexidade seria O(n)

Pior, Melhor e Caso Médio

- Exemplo: busca por um elemento em um conjunto

	0	1	2	3	4	5	6	7
Vetor	27	18	17	3	6	1	10	5

- Caso médio: podemos pensar nele como o custo médio para todos os casos de busca

$$\frac{CustoPos0 + CustoPos1 + \ldots + CustoPos7}{N^{\underline{o}}\ de\ Elementos}$$

Pior, Melhor e Caso Médio

$$\frac{1 + 2 + 3 + \cdots + 8}{8} = \frac{36}{8} = 4{,}5$$

- O numerador pode ser interpretado como a soma dos termos de uma PA:

$$S = \frac{(a_1 + a_n) * n}{2}$$

$$S = \frac{(1 + 8) * 8}{2}$$

$$S = 36$$

Pior, Melhor e Caso Médio

$$S = \frac{(1+n)*n}{2}$$

$$S = \frac{n^2+n}{2}$$

$$\frac{Custo\ de\ Todos}{Nº\ de\ Posições} = \frac{\frac{n^2+n}{2}}{n}$$

Pior, Melhor e Caso Médio

$$\frac{Custo\ de\ Todos}{Nº\ de\ Posições} = \frac{n^2+n}{2} * \frac{1}{n} = \frac{n^2+n}{2n}$$

$$\frac{Custo\ de\ Todos}{Nº\ de\ Posições} = \frac{n+1}{2}$$

- nº Testes: 4,5 → $\frac{n+1}{2}$

Pior, Melhor e Caso Médio

$$\frac{Custo\ de\ Todos}{N^{o}\ de\ Posições} = \frac{n^2+n}{2} * \frac{1}{n} = \frac{n^2+n}{2n}$$

$$\frac{Custo\ de\ Todos}{N^{o}\ de\ Posições} = \frac{n+1}{2}$$

- nº Testes: 4,5 → $\cancel{\frac{n+1}{2}}$
- Complexidade seria O(n)

Princípios da Análise de Algoritmos

1) O tempo de execução de operações básicas é considerado o mesmo:

- Atribuição
- Soma / Subtração / Multiplicação / Divisão
- Indexação
- Retorno
- Teste lógico
- Entrada
- Saída
- Chamadas

Princípios da Análise de Algoritmos

2) O tempo de execução de uma sequência de comandos é o maior tempo de execução de qualquer comando da sequência

Sequência 1 → $n * log_2 n$

Sequência 2 → n^2

Sequência 3 → n

Princípios da Análise de Algoritmos

2) O tempo de execução de uma sequência de comandos é o maior tempo de execução de qualquer comando da sequência

$$O(f(n)) + O(g(n)) = O(Max(O(f(n)), O(g(n))))$$

$$n * \log_2 n \ + \ n^2 \ + \ n \ = \ Max(n * \log_2 n, n^2, n)$$

Princípios da Análise de Algoritmos

2) O tempo de execução de uma sequência de comandos é o maior tempo de execução de qualquer comando da sequência

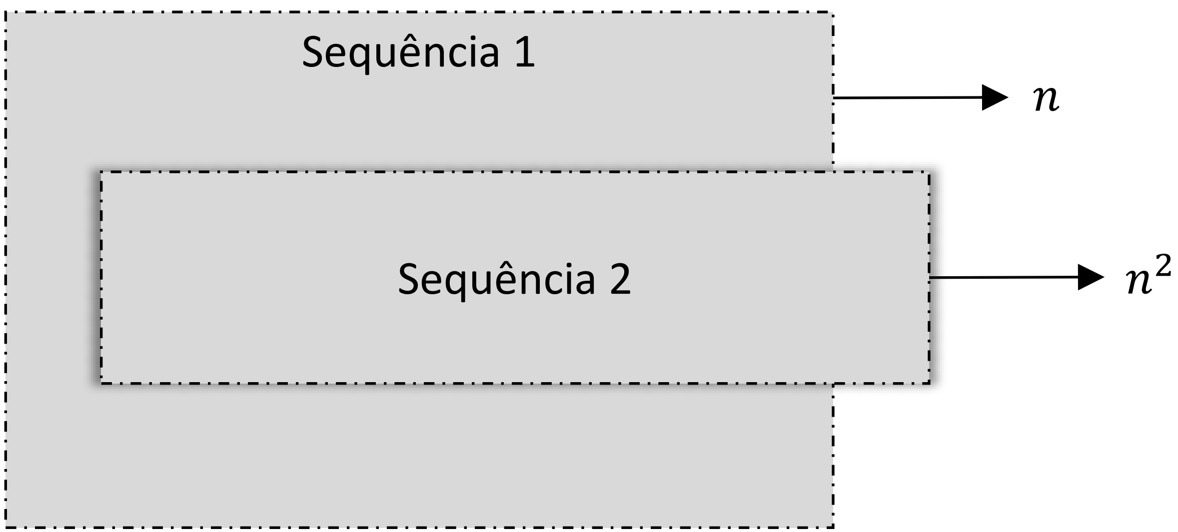
Sequência 1 $\longrightarrow n * log_2 n$

Sequência 2 $\longrightarrow n^2$ $\qquad O(n^2)$

Sequência 3 $\longrightarrow n$

Princípios da Análise de Algoritmos

3) O tempo de execução de uma sequência de comandos aninhada à outra é obtida pelo produto das complexidades



Princípios da Análise de Algoritmos

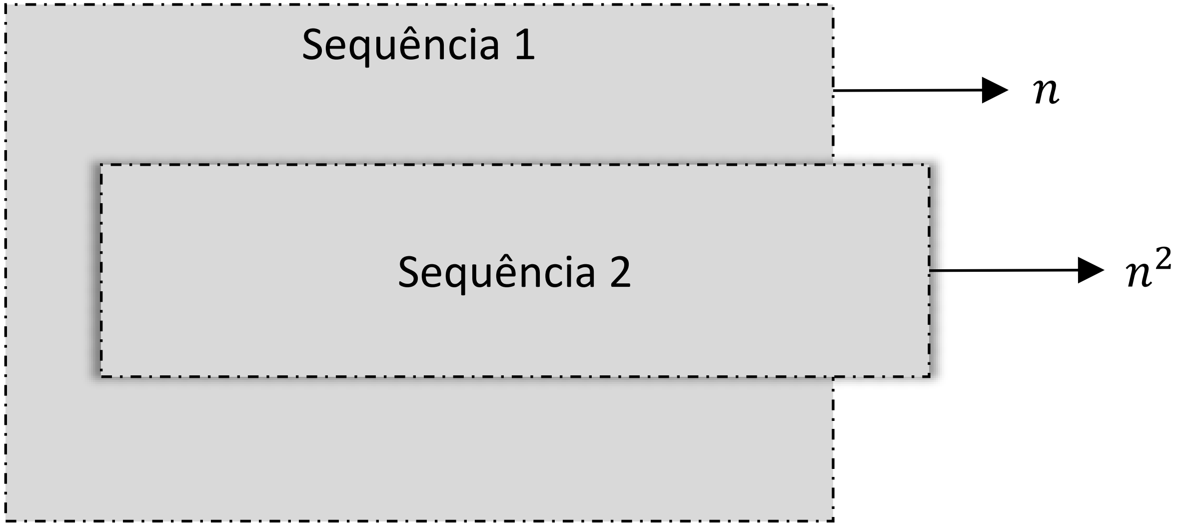
3) O tempo de execução de uma sequência de comandos aninhada à outra é obtida pelo produto das complexidades

$$O\big(f(n)\big) * O\big(g(n)\big) = O\big(f(n) * g(n)\big)$$

$$n * n^2 = n^3$$

Princípios da Análise de Algoritmos

3) O tempo de execução de uma sequência de comandos aninhada à outra é obtida pelo produto das complexidades



$O(n^3)$

Princípios da Análise de Algoritmos

4) Como obter o custo de cada trecho de código? Contando o número de operação básicas executadas em cada um

Exemplo:

Sequência 1

```
1    A = 30
2    B = 167
3    C = (A + B)
4    Vet[A] = C*A
```

Princípios da Análise de Algoritmos

4) Como obter o custo de cada trecho de código? Contando o número de operação básicas executadas em cada um

Exemplo:

Sequência 1

```
1       A = 30
2       B = 167
3       C = (A + B)
4       Vet[A] = C*A
```

$\rightarrow$ 1 atrib = 1

Princípios da Análise de Algoritmos

4) Como obter o custo de cada trecho de código? Contando o número de operação básicas executadas em cada um

Exemplo:

Sequência 1

1	A = 30	→	1 atrib = 1
2	B = 167	→	1 atrib = 1
3	C = (A + B)		
4	Vet[A] = C*A		

Princípios da Análise de Algoritmos

4) Como obter o custo de cada trecho de código? Contando o número de operação básicas executadas em cada um

Exemplo:

Sequência 1

1	A = 30	→	1 atrib = 1
2	B = 167	→	1 atrib = 1
3	C = (A + B)	→	1 atrib + 1 soma = 2
4	Vet[A] = C*A		

Princípios da Análise de Algoritmos

4) Como obter o custo de cada trecho de código? Contando o número de operação básicas executadas em cada um

Exemplo:

Sequência 1

1	A = 30	→	1 atrib = 1
2	B = 167	→	1 atrib = 1
3	C = (A + B)	→	1 atrib + 1 soma = 2
4	Vet[A] = C*A	→	1 multi + 1 atrib + 1 index = 3

Princípios da Análise de Algoritmos

4) Como obter o custo de cada trecho de código? Contando o número de operação básicas executadas em cada um

Exemplo:

Sequência 1

A = 30	1 atrib = 1
B = 167	1 atrib = 1
C = (A + B)	1 atrib + 1 soma = 2
Vet[A] = C*A	1 multi + 1 atrib + 1 index = 3

Complexidade constante
O(7)

Princípios da Análise de Algoritmos

Exemplo com condicionais

Sequência 2

```
A = 30
B = 167
if (A ≥ B)
      C = A
else
      C = (A + B)
Vet[A] = C*A
```

1 atrib = 1

Princípios da Análise de Algoritmos

Exemplo com condicionais

Sequência 2

```
A = 30
B = 167
if (A ≥ B)
     C = A
else
     C = (A + B)
Vet[A] = C*A
```

1 atrib = 1
1 atrib = 1

Princípios da Análise de Algoritmos

Exemplo com condicionais

Sequência 2

```
A = 30
B = 167
if (A ≥ B)
      C = A
else
      C = (A + B)
Vet[A] = C*A
```

1 atrib = 1

1 atrib = 1

1 teste lógico = 1

Princípios da Análise de Algoritmos

Exemplo com condicionais

Sequência 2	
A = 30	1 atrib = 1
B = 167	1 atrib = 1
if (A ≥ B)	1 teste lógico = 1
C = A	1 atrib = 1
else	
C = (A + B)	1 atrib + 1 soma = 2
Vet[A] = C*A	

Quando houver uma estrutura condicional, deve-se "pagar" o teste lógico e considerar o trecho que levar ao maior custo

Neste caso, o comando do else tem mais operações, logo, ele seria o pior caso

Princípios da Análise de Algoritmos

Exemplo com condicionais

Sequência 2	
A = 30	1 atrib = 1
B = 167	1 atrib = 1
if (A ≥ B)	1 teste lógico = 1
C = A	
else	
C = (A + B)	1 atrib + 1 soma = 2
Vet[A] = C*A	1 multi + 1 atrib + 1 index = 3

Princípios da Análise de Algoritmos

Exemplo com condicionais

Complexidade constante

O(8)

Sequência 2

A = 30	1 atrib = 1
B = 167	1 atrib = 1
if (A ≥ B)	1 teste lógico = 1
C = A	
else	
C = (A + B)	1 atrib + 1 soma = 2
Vet[A] = C*A	1 multi + 1 atrib + 1 index = 3

Princípios da Análise de Algoritmos

- Funções puras
  - Complexidade fixa
  - Determinística
  - Exemplos
    - raiz(n)
    - potencia(base,expoente)
    - maior(A,B)

Princípios da Análise de Algoritmos

- Exemplo com comandos de repetição
  - Contabiliza todos os custos de execução de cada linha, tanto internas quanto externas às estruturas de repetição
  - Aquelas que estiverem aninhadas, serão executadas uma quantidade de vezes de acordo com o controle da estrutura de repetição
  - Importante destacar que mesmo as operações usadas no comando de repetição devem ser consideradas

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição

Sequência 3

```
A = 0
for(i=0;i<n;i++)
      A = A + Vetor[i]
Imprime("Media: ", A/n)
```

1 atrib = 1

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição

Sequência 3

```
A = 0
for(i=0;i<n;i++)
      A = A + Vetor[i]
Imprime("Media: ",A/n)
```

1 atrib = 1

1 index + 1 soma + 1 atrib = 3

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição

Sequência 3

`A = 0`	1 atrib = 1
`for(i=0;i<n;i++)`	
`    A = A + Vetor[i]`	1 index + 1 soma + 1 atrib = 3
Imprime(“Media: “,$\frac{A}{n}$)	1 divisão + 1 print = 2

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição

Último teste lógico

2 for(i=0;i<n;i++)

O for começa com zero e é executado n vezes (0..n-1)

Cada vez que ele é executado, deve-se fazer um incremento em direção ao limite e um teste lógico para saber se este limite já foi alcançado ou não

A operação de inicializar o valor do iterador (neste caso, i=0) é realizada apenas no início do laço de repetição

Ou seja, o comando i=0 é executado uma única vez
Os comandos i<n e i++ são executados n vezes
Quando execução deixa o laço, realizou-se um último teste lógico

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição

2   for(i=0;i<n;i++)

Assim, o custo de execução do for seria:

1 atri (n=1) + (1 teste lógico (1<n) + 1 incremento)*n +1 teste lógico = 2n + 2

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição

Sequência 3

`A = 0`	1 atrib = 1
`for(i=0;i<n;i++)`	1 atrib + n(1 tes log + 1 incr) + 1 tes log = 2n + 2
`    A = A + Vetor[i]`	1 index + 1 soma + 1 atrib = 3
Imprime("Media: ",$\frac{A}{n}$)	1 divisão + 1 print = 2

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição

Sequência 3

Código		Custo
1	A = 0	1 atrib = 1
2	for(i=0;i<n;i++)	1 atrib + n(1 tes log + 1 incr) + 1 tes log = 2n + 2
3	A = A + Vetor[i]	1 index + 1 soma + 1 atrib = 3n
4	Imprime(“Media: “,$\frac{A}{n}$)	1 divisão + 1 print = 2

A linha 3, no entanto, está aninhada à estrutura de repetição, ou seja, cada vez que houver uma repetição do laço, os comandos da linha 3 serão executados.

Como o for é executado n vezes, o custo da linha 3 será $3 * n$

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição

Assim, a função de custo da Sequência 3 seria:

$$f(n) = 1 + 2n + 2 + 3n + 2$$

$$f(n) = 5n + 5$$

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição

Assim, a função de custo da Sequência 3 seria:

$$f(n) = 1 + 2n + 2 + 3n + 2$$

$$f(n) = \cancel{5}n + 5$$

Complexidade linear

O(n)

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Sequência 4

```
A = 0
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        A = A + Vetor[i] + Vetor[j]
Imprime(A)
```

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Sequência 4

```
A = 0
for(i=0;i<n;i++)
      for(j=0;j<n;j++)
            A = A + Vetor[i] + Vetor[j]
Imprime(A)
```

1 atrib = 1

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Sequência 4

```
A = 0
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        A = A + Vetor[i] + Vetor[j]
Imprime(A)
```

1 atrib = 1

2 index + 2 soma + 1 atrib = 5

1 print = 1

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Sequência 4

```
A = 0
for(i=0;i<n;i++)
    for(j=0;j<n;j++)
        A = A + Vetor[i] + Vetor[j]
Imprime(A)
```

1 atrib = 1

1 atrib + n(1 tlog + 1 incr) + 1 tlog = 2n + 2

2 index + 2 soma + 1 atrib = 5

1 print = 1

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Sequência 4

```
A = 0
for(i=0;i<n;i++)
      for(j=0;j<n;j++)
            A = A + Vetor[i] + Vetor[j]
Imprime(A)
```

- 1 atrib = 1
- 1 atrib + n(1 tlog + 1 incr) + 1 tlog = 2n + 2
- 1 atrib + n(1 tlog + 1 incr) + 1 tlog = 2n + 2
- 2 index + 2 soma + 1 atrib = 5
- 1 print = 1

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Sequência 4

Código	Custo
1 `A = 0`	1 atrib = 1
2 `for(i=0;i<n;i++)`	1 atrib + n(1 tlog + 1 incr) + 1 tlog = 2n + 2
3 `for(j=0;j<n;j++)`	1 atrib + n(1 tlog + 1 incr) + 1 tlog = 2n + 2
4 `A = A + Vetor[i] + Vetor[j]`	2 index + 2 soma + 1 atrib = 5n
5 `Imprime(A)`	1 print = 1

A linha 4 está aninhada à estrutura de repetição do j, ou seja, cada vez que houver uma repetição do laço, os comandos da linha 4 serão executados.

Como o for do j é executado n vezes, o custo da linha 4 será $5 * n$

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Sequência 4

1	`A = 0`	1 atrib = 1
2	`for(i=0;i<n;i++)`	1 atrib + n(1 tlog + 1 incr) + 1 tlog = 2n + 2
3	`    for(j=0;j<n;j++)`	1 atrib + n(1 tlog + 1 incr) + 1 tlog = 2n + 2
4	`        A = A + Vetor[i] + Vetor[j]`	2 index + 2 soma + 1 atrib = $5n^2$
5	`Imprime(A)`	1 print = 1

Além disso, a linha 4 está aninhada ao for do iterador i, ou seja, cada vez que o laço é executado a linha 4 também é. Como o laço é executado n vezes, o custo da linha será $5n * n$

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Sequência 4

```
1   A = 0
2   for(i=0;i<n;i++)
3       for(j=0;j<n;j++)
4           A = A + Vetor[i] + Vetor[j]
5   Imprime(A)
```

1 atrib = 1

1 atrib + n(1 tlog + 1 incr) + 1 tlog = 2n + 2

1 atrib + n(1 tlog + 1 incr) + 1 tlog = $2n^2 + 2n$

2 index + 2 soma + 1 atrib = $5n^2$

1 print = 1

A linha 3 está aninhada ao for do iterador i, ou seja, cada vez que o laço é executado a linha 3 também é. Como o custo da linha 3 já é 2n+1, seu custo final será $n * (2n + 2) \rightarrow 2n^2 + 2n$

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Assim, a função de custo da Sequência 4 seria:

$$f(n) = 1 + (2n + 2) + (2n^2 + 2n) + 5n^2 + 1$$

$$f(n) = 7n^2 + 4n + 4$$

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Assim, a função de custo da Sequência 4 seria:

$$f(n) = 1 + (2n + 2) + (2n^2 + 2n) + 5n^2 + 1$$

$$f(n) = \cancel{7}n^2 + \cancel{4}n + 4$$

$$f(n) = \max(n^2, n, 4)$$

Princípios da Análise de Algoritmos

Exemplo com comandos de repetição aninhados

Assim, a função de custo da Sequência 4 seria:

$$f(n) = 1 + (2n + 2) + (2n^2 + 2n) + 5n^2 + 1$$

$$f(n) = \cancel{7}n^2 + \cancel{4}n + 4$$

Complexidade quadrática

$O(n^2)$

Princípios da Análise de Algoritmos

Exemplo combinando elementos

Sequência 5

```
A = -1
for(i=0;i<n;i++)
      if (A < Vetor[i])
            A = Vetor[i]
Imprime("Maior",A)
```

Princípios da Análise de Algoritmos

Exemplo combinando elementos

Sequência 5

`A = -1`	1 atrib = 1
`for(i=0;i<n;i++)`	1 atrib + n(1 tes log + 1 incr) + 1 tlog= 2n + 2
`    if (A < Vetor[i])`	1 index + 1 teste log = 2
`        A = Vetor[i]`	1 index + 1 atrib = 2
`Imprime("Maior: ",A)`	1 print = 1

Princípios da Análise de Algoritmos

Exemplo combinando elementos

Sequência 5

`A = -1`	1 atrib = 1	
`for(i=0;i<n;i++)`	1 atrib + n(1 tes log + 1 incr) + 1 tlog= 2n + 2	
`    if (A < Vetor[i])`	1 index + 1 teste log = 2	Aninhados ao for
`        A = Vetor[i]`	1 index + 1 atrib = 2	
`Imprime("Maior",A)`	1 print = 1	

Princípios da Análise de Algoritmos

Exemplo combinando elementos

Sequência 5

`A = -1`	1 atrib = 1
`for(i=0;i<n;i++)`	1 atrib + n(1 tes log + 1 incr) + 1 tlog= 2n + 2
`    if (A < Vetor[i])`	1 index + 1 teste log = 2n
`        A = Vetor[i]`	1 index + 1 atrib = 2n
`Imprime(“Maior”,A)`	1 print = 1

Princípios da Análise de Algoritmos

Assim, a função de custo da Sequência 5 seria:

$$f(n) = 1 + (2n + 2) + 2n + 2n + 1$$

$$f(n) = 6n + 4$$

Complexidade linear
O(n)

Princípios da Análise de Algoritmos

Exemplo combinando elementos

Sequência 6

```
A = Vetor[0]
for(i=1;i<n;i++)
    if (A < Vetor[i])
        A = Vetor[i]
Imprime("Maior",A)
```

Princípios da Análise de Algoritmos

Exemplo combinando elementos

Sequência 6

```
A = Vetor[0]
for(i=1;i<n;i++)
      if (A < Vetor[i])
            A = Vetor[i]
Imprime("Maior",A)
```

1 index + 1 atrib = 2

1 atrib + (n-1)(1 tes log + 1 incr)+1 tlog = 2n

1 index + 1 teste log = 2

1 index + 1 atrib = 2

1 print = 1

Princípios da Análise de Algoritmos

Exemplo combinando elementos

Sequência 6

A = Vetor[0]	1 index + 1 atrib = 2	
for(i=1;i<n;i++)	1 atrib + (n-1)(1 tes log + 1 incr)+1 tlog = 2n	
if (A < Vetor[i])	1 index + 1 teste log = 2	Aninhados ao for
A = Vetor[i]	1 index + 1 atrib = 2	Aninhados ao for
Imprime(“Maior”,A)	1 print = 1	

Princípios da Análise de Algoritmos

Exemplo combinando elementos

Sequência 6

Código	Custo
`A = Vetor[0]`	1 index + 1 atrib = 2
`for(i=1;i<n;i++)`	1 atrib + (n-1)(1 tes log + 1 incr)+1 tlog = 2n
`    if (A < Vetor[i])`	1 index + 1 teste log = 2(n-1) = 2n - 2
`        A = Vetor[i]`	1 index + 1 atrib = 2(n-1) = 2n - 2
`Imprime(“Maior”,A)`	1 print = 1

Princípios da Análise de Algoritmos

Assim, a função de custo da Sequência 6 seria:

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

$$f(n) = 6n - 1$$

Complexidade linear
O(n)

Princípios da Análise de Algoritmos

Exemplo combinando laços

Sequência 7

```
for(i=1;i<n;i++)
    for(j=1;j<n;j++)
        A = VetA[i] + VetB[j]
Imprime("Soma: ",A)
```

Princípios da Análise de Algoritmos

Exemplo combinando laços

Sequencia 7

```
1    for(i=1;i<n;i++)
2         for(j=1;j<n;j++)
3              A = VetA[i] + VetB[j]
4    Imprime("Soma: ",A)
```

- 1 → 1 atrib + (n-1)(1 tlog + 1 incr) + 1 tlog = 2n
- 2 → 1 atrib + (n-1)(1 tlog + 1 incr) + 1 tlog = 2n
- 3 → 1 index + 1 soma+ 1 index + 1 atrib = 4
- 4 → 1 print = 1

Neste caso, temos 3 níveis de identação. Uma forma interessante de analisar é começar daquela mais interna e ir expandindo. Neste caso, a linha 3.

Princípios da Análise de Algoritmos

Exemplo combinando laços

Sequência 7

```
1   for(i=1;i<n;i++)
2        for(j=1;j<n;j++)
3            A = VetA[i] + VetB[j]
4   Imprime("Soma: ",A)
```

Linha	Custo
1	1 atrib + (n-1)(1 tlog + 1 incr) + 1 tlog = 2n
2	1 atrib + (n-1)(1 tlog + 1 incr) + 1 tlog = 2n
3	1 index + 1 soma+ 1 index + 1 atrib = 4n-4
4	1 print = 1

A linha 3 está aninhada ao for da linha 2, assim, cada vez que a linha dois for executada, a linha 3 também será. Uma vez que a linha 2 será executada $n - 1$ vezes, o custo da linha 3 será $4n - 1$

Princípios da Análise de Algoritmos

Exemplo combinando laços

Sequência 7

```
1    for(i=1;i<n;i++)
2         for(j=1;j<n;j++)
3              A = VetA[i] + VetB[j]
4    Imprime("Soma: ",A)
```

1 atrib + (n-1)(1 tlog + 1 incr) + 1 tlog = 2n

$(2n)*(n-1) = 2n^2 - 2n$

$(4n-4)*(n-1) = 4n^2 - 8n + 4$

1 print = 1

As linhas 2 e 3 estão aninhadas ao for da linha 1, assim, cada vez que ele for executado, as duas linhas também serão. Como o for é executado n-1 vezes...

Princípios da Análise de Algoritmos

Assim, a função de custo da Sequência 7 seria:

$$f(n) = (2n) + (2n^2 - 2n) + (4n^2 - 8n + 4) + 1$$

$$f(n) = 6n^2 - 8n + 5$$

Complexidade quadrática
$O(n^2)$

Princípios da Análise de Algoritmos

Exemplo combinando laços dependentes

Sequência 8

```
A = 0;
for(i=0;i<n;i++)
    for(j=i;j<n;j++)
        A++;
Imprime("Contagem: ",A)
```

Princípios da Análise de Algoritmos

Exemplo combinando laços dependentes

Sequência 8

```
A = 0;
for(i=0;i<n;i++)
    for(j=i;j<n;j++)
        A++;
Imprime(“Contagem: ”,A)
```

Nestes casos especiais, o for interno não rodará as n vezes, mas um número variado de vezes, dependendo do laço mais externo.

Uma forma de levantar esse custo é trabalhar com os dois laços de forma conjunta

Princípios da Análise de Algoritmos

Exemplo combinando laços dependentes

Considere que n tem valor 5

Cada vez que o laço externo é executado (e incrementado), o laço interno executa uma vez a menos:

- Quando i=0, laço interno executa 5x
- Quando i=1, laço interno executa 4x
- Quando i=2, laço interno executa 3x
- Quando i=3, laço interno executa 2x
- Quando i=4, laço interno executa 1x

Princípios da Análise de Algoritmos

Exemplo combinando laços dependentes

Considere que n tem valor 5

Cada vez que o laço externo é executado, o laço interno executa uma vez a menos:

- Quando i=0, laço interno executa 5x
- Quando i=1, laço interno executa 4x
- Quando i=2, laço interno executa 3x
- Quando i=3, laço interno executa 2x
- Quando i=4, laço interno executa 1x

Soma dos termos de uma PA de razão 1 com n termos (neste caso, 5 termos)

O primeiro elemento é 1
O último elemento é 5

Princípios da Análise de Algoritmos

$$S = 5 + 4 + 3 + 2 + 1 = 15$$

$$S = \frac{(a_1 + a_n) * n}{2}$$

$$S = \frac{(1 + n) * n}{2} \rightarrow S = \frac{n^2 + n}{2}$$

$$S = \frac{5^2 + 5}{2} = \frac{25 + 5}{2} = 15$$

Assim, o número de vezes que o trecho é executado será $\frac{n^2+n}{2}$

Princípios da Análise de Algoritmos

Exemplo combinando laços dependentes

Sequência 8

```
A = 0;
for(i=0;i<n;i++)
    for(j=i;j<n;j++)
        A++;
Imprime("Contagem: ",A)
```

1 atrib = 1

1 atrib + (n)(1 tes log + 1 incr)+1 tes log = 2n + 2

1 atrib + (1 tes log + 1 incr))+1 tes log = ?

1 incr = ?

1 print = 1

Princípios da Análise de Algoritmos

Exemplo combinando laços dependentes

Sequência 8

```
1   A = 0;
2   for(i=0;i<n;i++)
3       for(j=i;j<n;j++)
4           A++;
5   Imprime("Contagem: ",A)
```

- 1 atrib = 1
- 1 atrib + (n)(1 tes log + 1 incr)+1 tes log = 2n + 2
- 1 atrib + (1 tes log + 1 incr)+1 tes log
- 1 incr = $1\ \left(\frac{n^2+n}{2}\right) = \frac{n^2+n}{2}$
- 1 print = 1

A linha 4 está aninhada aos laços das linhas 2 e 3 que são dependentes. Assim, ele será executado o número de vezes que os laços forem executados: $\frac{n^2+n}{2}$

Princípios da Análise de Algoritmos

Exemplo combinando laços dependentes

Sequência 8

```
A = 0;
for(i=0;i<n;i++)
    for(j=i;j<n;j++)
        A++;
Imprime("Contagem: ",A)
```

1 atrib = 1

1 atrib + (n)(1 tes log + 1 incr)+1 tes log = 2n + 2

1 atrib + (1 tes log + 1 incr)$(\frac{n^2+n}{2})$ +1 tes log

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

1 print = 1

Da mesma forma, os comandos de teste lógico e incremento do for interno estão condicionados ao número de execução dos laços dependentes: $\frac{n^2+n}{2}$

Princípios da Análise de Algoritmos

Exemplo combinando laços dependentes

Sequência 8

```
1   A = 0;
2   for(i=0;i<n;i++)
3       for(j=i;j<n;j++)
4           A++;
5   Imprime("Contagem: ",A)
```

→ 1 atrib = 1

→ 1 atrib + (n)(1 tes log + 1 incr)+1 tes log = 2n + 2

→ 1 atrib(n) + (1 tes log + 1 incr)$\left(\frac{n^2+n}{2}\right)$ +1 tes log(n)

→ 1 incr = 1 $\left(\frac{n^2+n}{2}\right) = \frac{n^2+n}{2}$

→ 1 print = 1

n^2+3n

No entanto, o comando de atribuição do laço da linha 3 só ocorre no início dos ciclos, ou seja, quando o valor de i é incrementado no laço da linha 2. O mesmo ocorre para o último teste lógico (aquele que confirma a saído do laço)

Princípios da Análise de Algoritmos

$$f(n) = 1 + (2\mathrm{n} + 2) + (\mathrm{n}^2 + 3\mathrm{n}) + \frac{n^2+n}{2} + 1$$

$$f(n) = \frac{3n^2}{2} + \frac{11n}{2} + 4$$

$$f(n) = \frac{\cancel{3}n^2}{\cancel{2}} + \frac{\cancel{1}1n}{\cancel{2}} + 4$$

$$f(n) = O(n^2)$$

Exemplo 1

Qual o custo (em número de operações) para executar o seguinte algoritmo?

```
for(i=1;i<n;i++)
    for(k=2;k<=n;k++)
        S[i][k] = S[i][k] – S[i][k] * S[i][i] / S[i][i];
```

Exemplo 2
Calcular O para pior e melhor caso

```
void OrdenaSort (int *Vet, int n)
{
    int i, j, aux;
    for (i=0;i<n;i++)
   {
        for (j=0;j<n-1;j++)
        {
            if (Vet[j] > Vet[j+1])
            {
                aux = Vet[j];
                Vet[j] = Vet[j+1];
                Vet[j+1] = aux;
            }
        }
    }
}
```

Exemplo 2
Calcular O para pior e melhor caso

Exemplo 3

- Suponha um algoritmo A e um algoritmo B, com funções de complexidade de tempo $a(n) = n^2 - n + 549$ e $b(n) = 49n + 49$, respectivamente. Determine quais valores de n pertencentes ao conjunto dos números naturais para os quais A leva menos tempo para executar do que B.

Exemplo 3

Exemplo 3

Exemplo 3