

LABORATÓRIO DE ALGORITMOS E ESTRUTURAS DE DADOS I

Prof. Caio César de Freitas Dantas

Complexidade de Algoritmos

Do que depende o tempo de execução de um algoritmo?

- O tempo para um computador executar as linhas de código do algoritmo,
- Da velocidade do computador,
- Da linguagem de programação,
- Do compilador que traduziu o programa,
- Outros fatores.

Complexidade de Algoritmos

Vamos pensar no tempo de execução de algoritmo como:

- Uma função do tamanho da sua entrada;
- Quão rápido essa função cresce dado o tamanho da entrada;

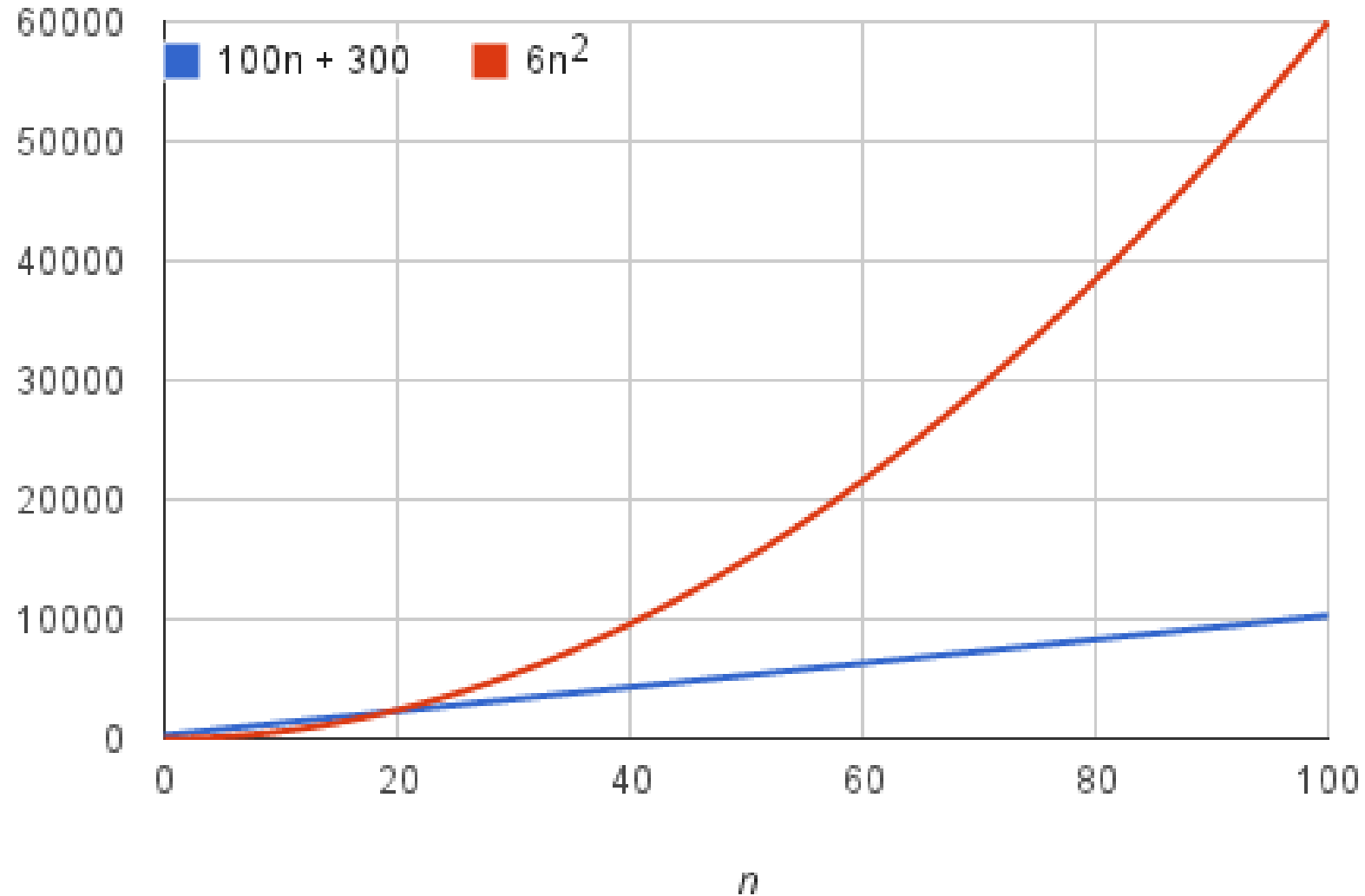
Taxa de Crescimento do tempo de execução

- Para isso usamos uma função matemática

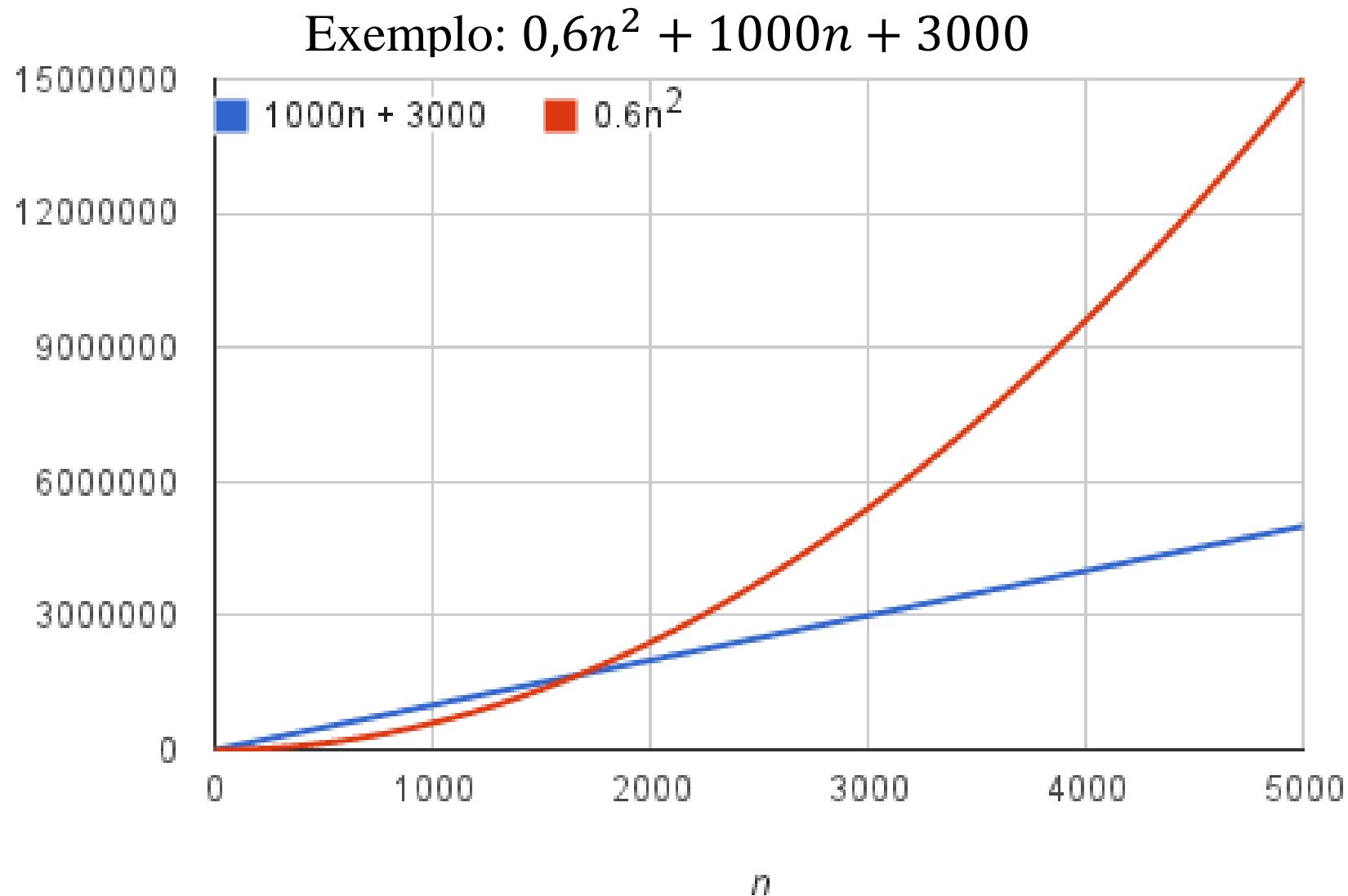
$$\text{Exemplo: } 6n^2 + 100n + 300$$

Complexidade de Algoritmos

Exemplo: $6n^2 + 100n + 300$



Complexidade de Algoritmos



Complexidade de Algoritmos

- Informações mais relevantes que outras;

Notação Assintótica

- É uma notação que nos permite descartar os termos constantes e menos significativos;
- Foco no que realmente é relevante;

Complexidade de Algoritmos

Vamos pensar no tempo de execução de algoritmo como:

- Uma função do tamanho da sua entrada;
- Quão rápido essa função cresce dado o tamanho da entrada;

Taxa de Crescimento do tempo de execução

- Para isso usamos uma função matemática;
- Para determinar o tempo de execução (processamento) de um algoritmo é necessário conhecer o numero de instruções que ele realiza.

Complexidade de Algoritmos

- Quão rápido essa função cresce dado o tamanho da entrada;
- Para determinar o tempo de execução (processamento) de um algoritmo é necessário conhecer o numero de instruções que ele realiza.

Contar numero
de instruções

Crescimento em função do
tamanho da entrada

Complexidade de Algoritmos

Instruções simples x complexas

Simple

- São aquelas que podem ser executadas em linguagem de máquina;
- Dizemos que medem 1;

Complexas

- Combinação de instruções simples;
- Construída através de instruções de controle de fluxo;
- Dizemos que medem a soma de suas instruções simples.

Complexidade de Algoritmos

Exemplos

1. Atribuições de valor de forma geral;
2. Incremento de valores;
3. Operações aritméticas;
4. Acesso ao valor de um elemento do vetor;
5. Expressões lógicas;
6. Operações de leitura e escrita;

Valem 1;

Complexidade de Algoritmos

Exemplos

1. Atribuições de valor de forma geral;

$a = 21;$

$b = a + 11;$

$x = x * x + (a-b);$

Algoritmo com 3 instruções;

Complexidade de Algoritmos

Exemplos

2. Incremento de valores;

`i++;`

3. Operações aritméticas;

`A+B`

4. Acesso ao valor de um elemento do vetor;

`a = vet[0];`

5. Expressões lógicas;

`(a==b) && (c!= 4)`

6. Operações de leitura e escrita;

`printf ("%d", c);`

Complexidade de Algoritmos

Contando Instruções: Exemplo 1




```
x = 0;
```

```
x = x + 1;
```

```
printf("%d",x);
```

Complexidade de Algoritmos

Contando Instruções: Exemplo 1

<code>x = 0;</code>		1
<code>x = x + 1;</code>		1
<code>printf("%d",x);</code>		1

Total 3 instruções

Complexidade de Algoritmos

Contando Instruções: Exemplo 2

```
scanf("%d", x);     —————→     1  
if (x % 2 == 0){    —————→     1  
x = x + 1;  
} else{  
x = x - 1;  
}  
printf("%d", x);    —————→     1
```

Total 3 instruções

Complexidade de Algoritmos

Contando Instruções: Exemplo 3

```
max = 0;
```

```
i = 0;
```

```
while (i < max) {
```

```
    i++;
```

```
}
```


Complexidade de Algoritmos

Contando Instruções: Exemplo 3

max = 0;	→	1
i = 0;	→	1
while (i < max) {	→	1
i++;	→	0
}		

Total 3 instruções

Complexidade de Algoritmos

Contando Instruções: Exemplo 4

```
max = 10;
```

```
i = 0;
```

```
while (i < max) {
```

```
    i++;
```

```
}
```

Complexidade de Algoritmos

Contando Instruções: Exemplo 4

max = 10;	→	1
i = 0;	→	1
while (i < max) {	→	1 + 10
i++;	→	10
}		

Total 23 instruções

Complexidade de Algoritmos

Contando Instruções: Exemplo 5

```
max = 10;
```

```
i = 0;
```

```
while (i < max) {  
    printf("%d", max);  
    i++;  
}
```

Complexidade de Algoritmos

Contando Instruções: Exemplo 5

max = 10;	→	1
i = 0;	→	1
while (i < max) {	→	1 + 10
printf("%d", max);	→	10
i++;	→	10
}		

Total 33 instruções

Complexidade de Algoritmos

Contando Instruções: Exemplo 6 = for

```
max = 0;  
for (i = 0; i < max; i++){  
    printf("%d", i);  
}
```

Complexidade de Algoritmos

Contando Instruções: Exemplo 6 = for

max = 0;	→	1
for (i = 0; i < max; i++){	→	2
printf("%d", i);	→	0
}		

Total 3 instruções

Complexidade de Algoritmos

Contando Instruções: Exemplo 6 = for

max = 0;	→	1
for (i = 0; i < max; i++){	→	2
printf("%d", i);	→	0
}		

//Inicialização do i = 1 instrução

//Comparação com o max = 1 instrução, como max é 0 só ocorre 1x

//Incremento do i = 0 instruções, pois nunca ocorrerá se max é 0

//Impressão = 0 instruções, pois nunca entrar no laço

Total 3 instruções



Complexidade de Algoritmos

Contando Instruções: Exemplo 7 = for

```
max = 10;  
for (i = 0; i < max; i++){  
  
}
```

Complexidade de Algoritmos

Contando Instruções: Exemplo 7 = for

```
max = 10;            1  
for (i = 0; i < max; i++){  1 + 1+10 + 10  
  
}
```

//Inicialização do i = 1 instrução

//Comparação com o max = 1 + 10 instruções

//Incremento do i = 10 instruções

Total 23 instruções




Complexidade de Algoritmos

Contando Instruções: Exemplo 8 = for

```
max = 10;  
for (i = 0; i < max; i++){  
    printf("%d", i);  
}
```

Complexidade de Algoritmos

Contando Instruções: Exemplo 8 = for

<code>max = 10;</code>		1
<code>for (i = 0; i < max; i++){</code>		1 + 1 + 10 + 10
<code>printf("%d", i);</code>		10
<code>}</code>		

//Inicialização do i = 1 instrução

//Comparação com o max = 1 + 10 instruções

//Incremento do i = 10 instruções

//impressão = 10 instruções

Total 33 instruções

Complexidade de Algoritmos

Contando Instruções: Exemplo 9 = for

```
max = 100;  
for (i = 0; i < max; i++){  
    printf("%d", i);  
}
```

Complexidade de Algoritmos

Contando Instruções: Exemplo 9 = for

<code>max = 100;</code>		1
<code>for (i = 0; i < max; i++){</code>		1 + 1+100 + 100
<code>printf("%d", i);</code>		100
<code>}</code>		

//Inicialização do i = 1 instrução

//Comparação com o max = 1 + 100 instruções

//Incremento do i = 100 instruções

//impressão = 100 instruções

Total 303 instruções

Complexidade de Algoritmos

Contando Instruções: Exemplo 10 = for

```
for (i = 0; i < n; i++){  
    printf("%d", i);  
}
```

Complexidade de Algoritmos

Contando Instruções: Exemplo 10 = for

for (i = 0; i < n; i++){	→	1 + 1 + n + n
printf("%d", i);	→	n
}		

//Inicialização do i = 1 instrução

//Comparação com o max = 1 + n instruções

//Incremento do i = n instruções

//impressão = n instruções

Total $3n + 2$

Complexidade de Algoritmos

Contando Instruções: Exemplo 11 = for

```
for(i = 0; i < n; i++){  
    for(j = 0; j < n; j++){  
        printf ("%d", i);  
    }  
}
```

Complexidade de Algoritmos

Contando Instruções: Exemplo 11 = for

for(i = 0; i < n; i++){	→	1 + 1 + n + n
for(j = 0; j < n; j++){	→	n * (1 + 1 + n + n)
printf ("%d", i);	→	n
}		
}		

// laço externo:

// - inicialização do i: 1 instrução

// - comparações do i com o n: n + 1 instruções

// - incrementos do i no laço externo: n instruções

// - laço interno que executa n vezes:

// --- inicialização do j: 1 instrução

// --- comparações do j com o n: n + 1 instruções

// --- incrementos do j: n instruções

// --- impressão(): executa n vezes

Complexidade de Algoritmos

Contando Instruções: Exemplo 11 = for

for(i = 0; i < n; i++){	→	1 + 1 + n + n
for(j = 0; j < n; j++){	→	n * (1 + 1 + n + n)
printf ("%d", i);	→	n
}		
}		

// laço externo:





- // - inicialização do i: 1 instrução
- // - comparações do i com o n: n + 1 instruções
- // - incrementos do i no laço externo: n instruções
- // - laço interno que executa n vezes:
- // --- inicialização do j: 1 instrução
- // --- comparações do j com o n: n + 1 instruções
- // --- incrementos do j: n instruções
- // --- impressão(): executa n vezes

$$= 2 + 2n + n * (2 + 3n)$$
$$= 2 + 2n + 2n + 3n^2$$

Total = $2 + 4n + 3n^2$





Complexidade de Algoritmos

Contando Instruções: Exemplo 12

<code>vet = [1,2,3,4,5,6,7,8..];</code>		1
<code>for (i = 0; i < Tam.Vet; i++){</code>		1 + 1 + n + n
<code>if (vet[i] % 2 == 0) {</code>		?
<code>printf("%d", vet[i]);</code>		?
<code>}</code>		
<code>}</code>		





Complexidade de Algoritmos

Contando Instruções: Exemplo 12

<code>vet = [1,3,5,7,9..];</code>		1
<code>for (i = 0; i < Tam.Vet; i++){</code>		1 + 1 + n + n
<code>if (vet[i] % 2 == 0) {</code>		?
<code>printf("%d", vet[i]);</code>		?
<code>}</code>		
<code>}</code>		

Complexidade de Algoritmos

Contando Instruções: Exemplo 12





<code>vet = [2,4,6,8,10..];</code>		1
<code>for (i = 0; i < Tam.Vet; i++){</code>		1 + 1 + n + n
<code>if (vet[i] % 2 == 0) {</code>		?
<code>printf("%d", vet[i]);</code>		?
<code>}</code>		
<code>}</code>		

Dependendo da entrada de dados dos valores:

- Ele pode executar menos vezes ou mais vezes;
- Pode executar nenhuma vez ou todas as vezes.

Complexidade de Algoritmos

Contando Instruções: Exemplo 12





<code>vet = [2,4,6,8,10..];</code>		<code>1</code>
<code>for (i = 0; i < Tam.Vet; i++){</code>		<code>1 + 1 + n + n</code>
<code>if (vet[i] % 2 == 0) {</code>		<code>n</code>
<code>printf("%d", vet[i]);</code>		<code>n</code>
<code>}</code>		
<code>}</code>		

Dependendo da entrada de dados dos valores:

- Ele pode executar menos vezes ou mais vezes;
- Pode executar nenhuma vez ou todas as vezes.

Complexidade de Algoritmos

Contando Instruções: Exemplo 12

<code>vet = [2,4,6,8,10..];</code>		<code>1</code>
<code>for (i = 0; i < Tam.Vet; i++){</code>		<code>1 + 1 + n + n</code>
<code>if (vet[i] % 2 == 0) {</code>		<code>n</code>
<code>printf("%d", vet[i]);</code>		<code>n</code>
<code>}</code>		
<code>}</code>		





$$T = 4n + 3$$

Dependendo da entrada de dados dos valores:

- Ele pode executar menos vezes ou mais vezes;
- Pode executar nenhuma vez ou todas as vezes.

Complexidade de Algoritmos

Contando Instruções: Exemplo 12





<code>vet = [2,4,6,8,10..];</code>		<code>1</code>
<code>for (i = 0; i < Tam.Vet; i++){</code>		<code>1 + 1 + n + n</code>
<code>if (vet[i] % 2 == 0) {</code>		<code>n</code>
<code>printf("%d", vet[i]);</code>		<code>n</code>
<code>}</code>		
<code>}</code>		

$$T = 4n + 3$$

Por que analisamos o caso em que pegar todos os valores e não o outro caso que não aceita nenhum valor?

Complexidade de Algoritmos

Contando Instruções: Exemplo 12

<code>vet = [2,4,6,8,10..];</code>		<code>1</code>
<code>for (i = 0; i < Tam.Vet; i++){</code>		<code>1 + 1 + n + n</code>
<code>if (vet[i] % 2 == 0) {</code>		<code>n</code>
<code>printf("%d", vet[i]);</code>		<code>n</code>
<code>}</code>		
<code>}</code>		

$$T = 4n + 3$$

Por que analisamos o caso em que pegar todos os valores e não o outro caso que não aceita nenhum valor? PIOR CASO

Complexidade de Algoritmos

Pior caso

Custo dominante;

$$T = 4n + 3$$

Ao analisarmos um algoritmo devemos escolher o pior caso do algoritmo, ou seja:
O com o maior numero de instruções a ser executado.

Complexidade de Algoritmos

Função de complexidade

$$f(n) = 4n + 3$$

Será que todos esses valores são importantes?

Notação Assintótica

Permite destacar os termos mais importantes.

Complexidade de Algoritmos

Análise Assintótica

- Em análise de algoritmos, é um método de descrever o comportamento de limites.
- Queremos saber como a função de complexidade de tempo se comporta quando n tende ao infinito;
- Quando processa uma grande quantidade de dados a complexidade do tempo pode ser alterada.

Complexidade de Algoritmos

Analise Assintótica

$$f(n) = 4n + 3;$$

$$\text{se } n = 1; f(n) = 7;$$

$$\text{se } n = 10; f(n) = 43;$$

$$f(n) = 4n + 3;$$

Complexidade de Algoritmos

Analise Assintótica

$$f(n) = 4n + 3;$$

$$\text{se } n = 1; f(n) = 7;$$

$$\text{se } n = 10; f(n) = 43;$$

$$f(\mathbf{n}) = 4\mathbf{n} + 3;$$

Complexidade de Algoritmos

Analise Assintótica

$$f(n) = 4n + 3;$$

$$\text{se } n = 1; f(n) = 7;$$

$$\text{se } n = 10; f(n) = 43;$$

$$f(n) = 4n + 3;$$

Complexidade de Algoritmos

Analise Assintótica

$$f(n) = 4n + 3;$$

$$\text{se } n = 1; f(n) = 7;$$

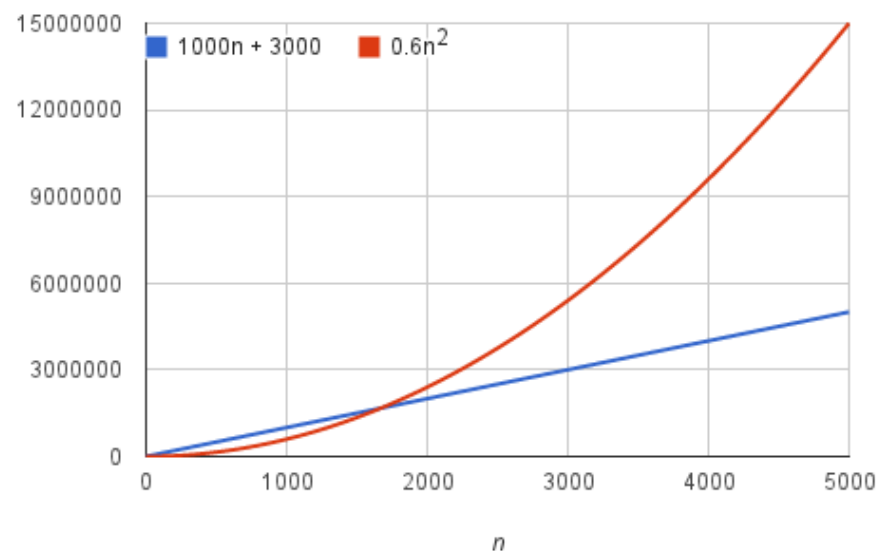
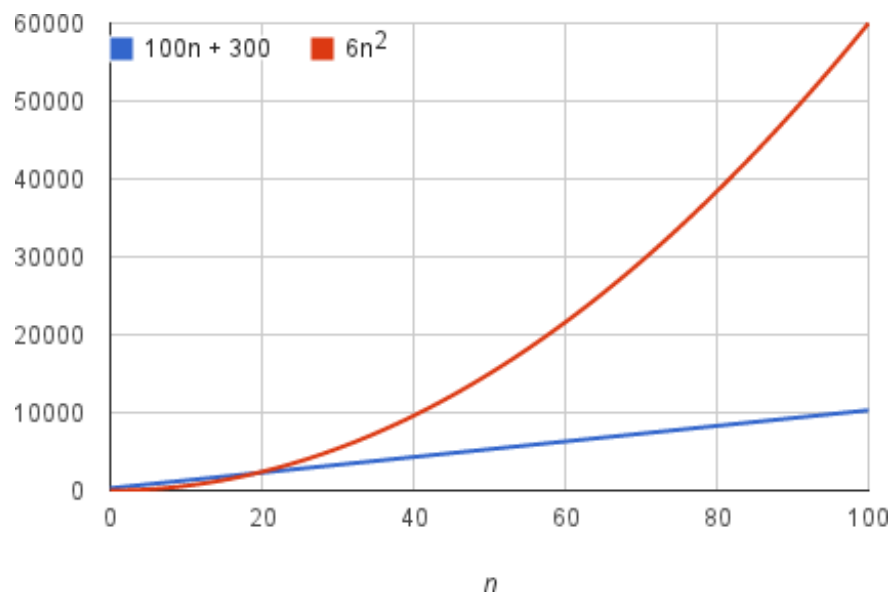
$$\text{se } n = 10; f(n) = 43;$$

$$f(n) = 4n + 3;$$

Complexidade de Algoritmos

Análise Assintótica

- Descartar os termos constantes e menos significativos;
- Manter apenas o que cresce mais rápido à medida que o n é maior;
- Algumas expressões fazem a função crescer mais rápido que outros;



Complexidade de Algoritmos

Análise Assintótica

- Descartar os termos constantes e menos significativos;
- Manter apenas o que cresce mais rápido à medida que o n é maior;
- Algumas expressões fazem a função crescer mais rápido que outros;

Redução:

$$f(n) = 4n + 3;$$

$$f(n) = 4n;$$

$$f(n) = n;$$

Complexidade de Algoritmos

Análise Assintótica

- Descartar os termos constantes e menos significativos;
- Manter apenas o que cresce mais rápido à medida que o n é maior;
- Algumas expressões fazem a função crescer mais rápido que outros;

Redução:

$$f(n) = 4n + 3;$$

$$f(n) = 4n;$$

$$f(n) = n;$$

Podemos Excluir os termos menos importantes das funções e considerar apenas os que possuem maior grau.

Complexidade de Algoritmos

Analise Assintótica

- Funções de Custo:

$$f(n) = 30$$

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

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

$$f(n) = n^3 + 500n + 1200$$

Complexidade de Algoritmos

Analise Assintótica

- Funções de Custo:

$$f(n) = 30$$

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

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

$$f(n) = n^3 + 500n + 1200$$

$$f(n) = 1$$

$$f(n) = n$$

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

$$f(n) = n^3$$

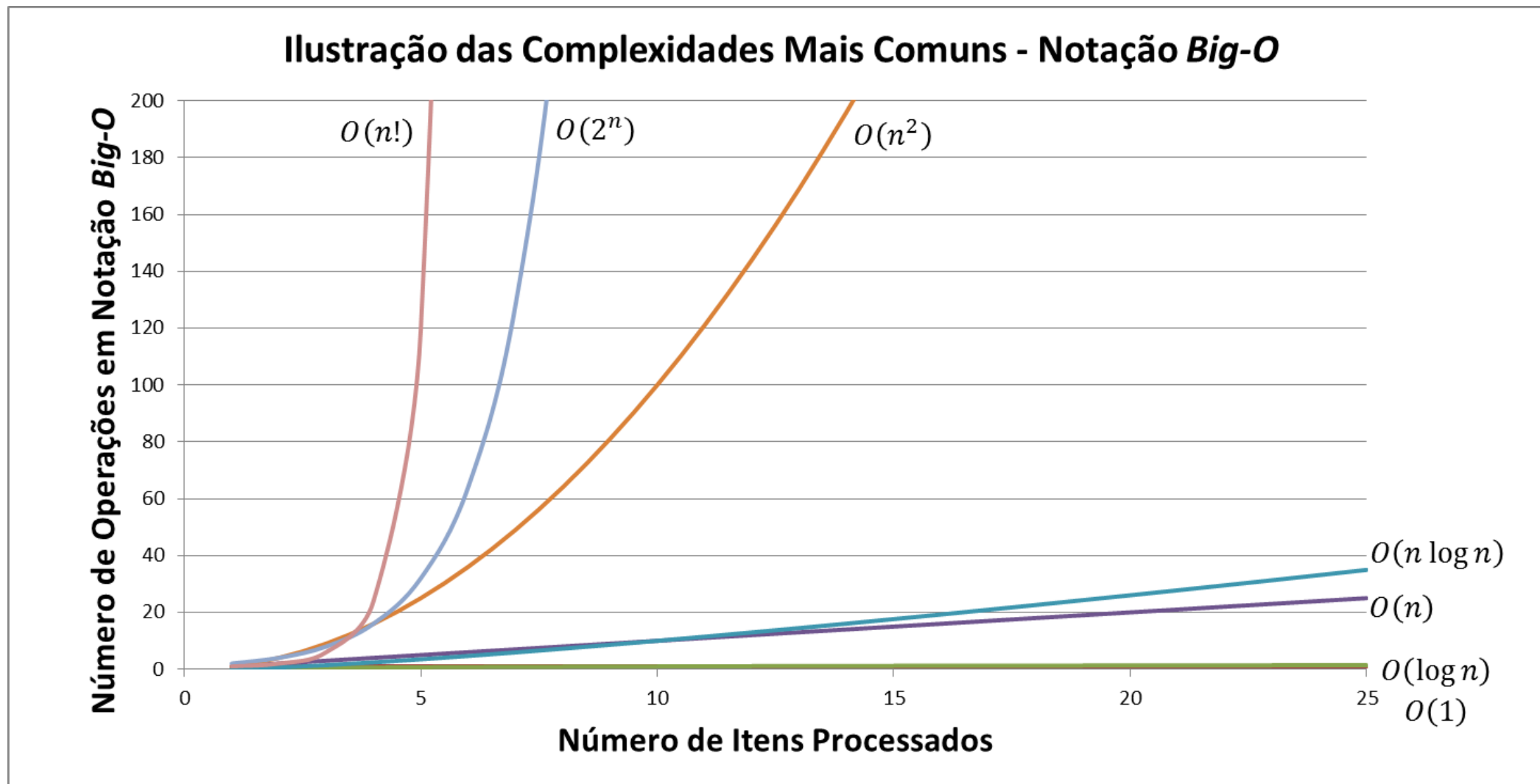
Complexidade de Algoritmos

Analise Assintótica

- Notação Big O – Pior Caso

$f(n) = 1$	$O(1)$
$f(n) = n$	$O(n)$
$f(n) = n^2$	$O(n^2)$
$f(n) = n^3$	$O(n^3)$

Complexidade de Algoritmos



FIM!

A thick horizontal green line spans the width of the slide, starting from the left edge. A second green line starts from the left edge and extends diagonally downwards towards the bottom-left corner.