Algoritmo e Estrutura de Dados II
COM-112

Aula 3

Vanessa Souza

Universidade Federal de Itajubá – Instituto de Matemática e Computação

COMPLEXIDADE

Análise de Algoritmos

- Como medir?
  - Execução do programa em um computador real – medir o tempo diretamente.
    - Parar todos os demais serviços e aplicativos
    - Limpar a memória entre um teste e outro
    - Rodar o teste, no mínimo, 30 vezes
  - Medir a quantidade de "trabalho" necessário para sua execução, expressa em função das operações fundamentais, as quais variam de acordo com o algoritmo, e em função do volume de dados.
    - Calcular a complexidade assintótica do programa

Análise de Algoritmos

- Na aula passada aprendemos a calcular a função de complexidade f(n)

- Observações importantes:
  - Para valores pequenos de n, praticamente qualquer algoritmo custa pouco para ser executado.
  - Logo: a escolha do algoritmo tem pouquíssima influência em problemas de tamanho pequeno.
  - A análise de algoritmos deve ser realizada para valores grandes de n.

Complexidade Assintótica

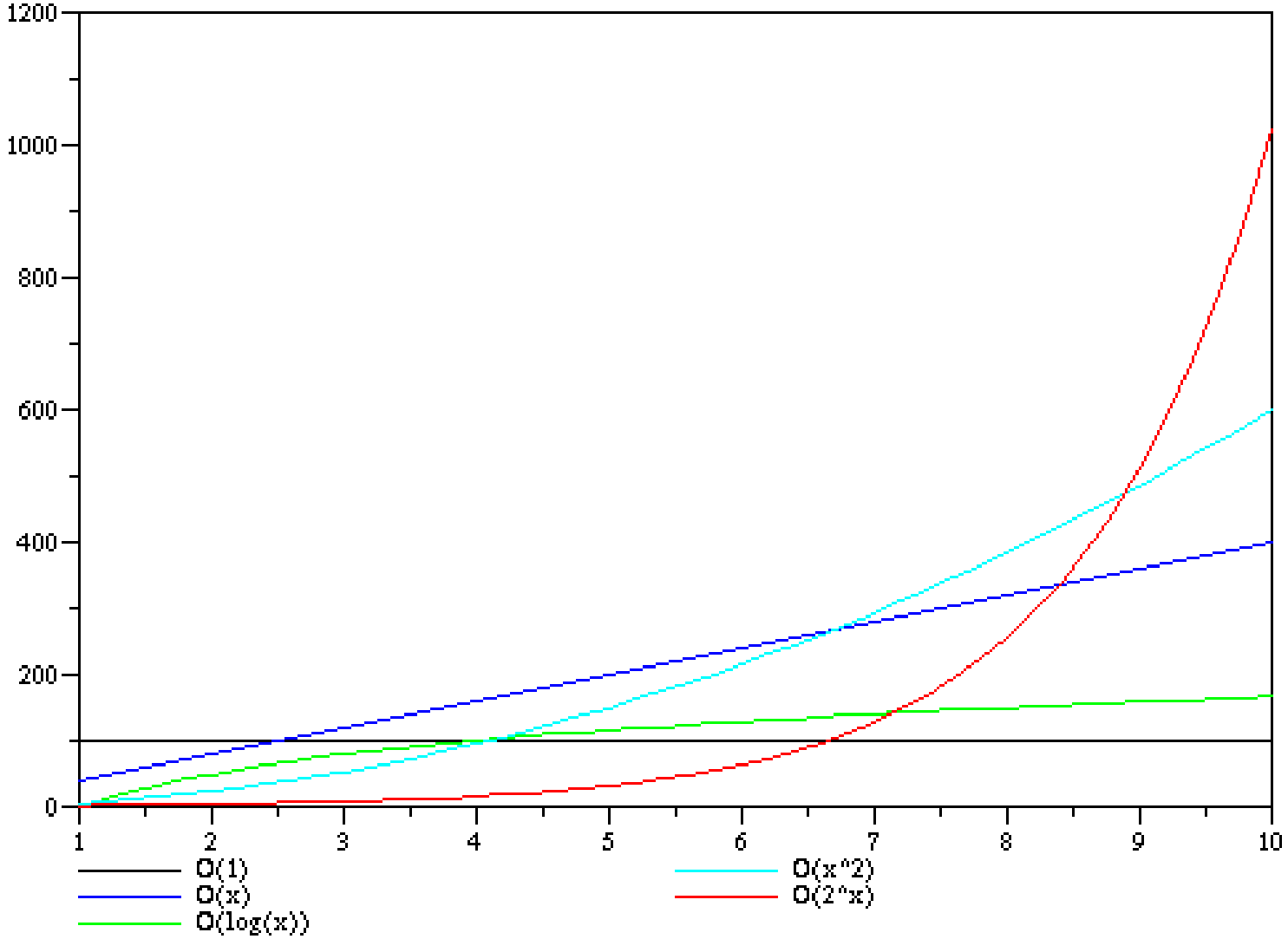
- Reflete o comportamento das funções para valores grandes de n.

- O comportamento assintótico de *f(n)* representa o limite do comportamento do custo quando *n* cresce.
  - A medida de custo, ou medida de complexidade, relata o crescimento assintótico da operação considerada.

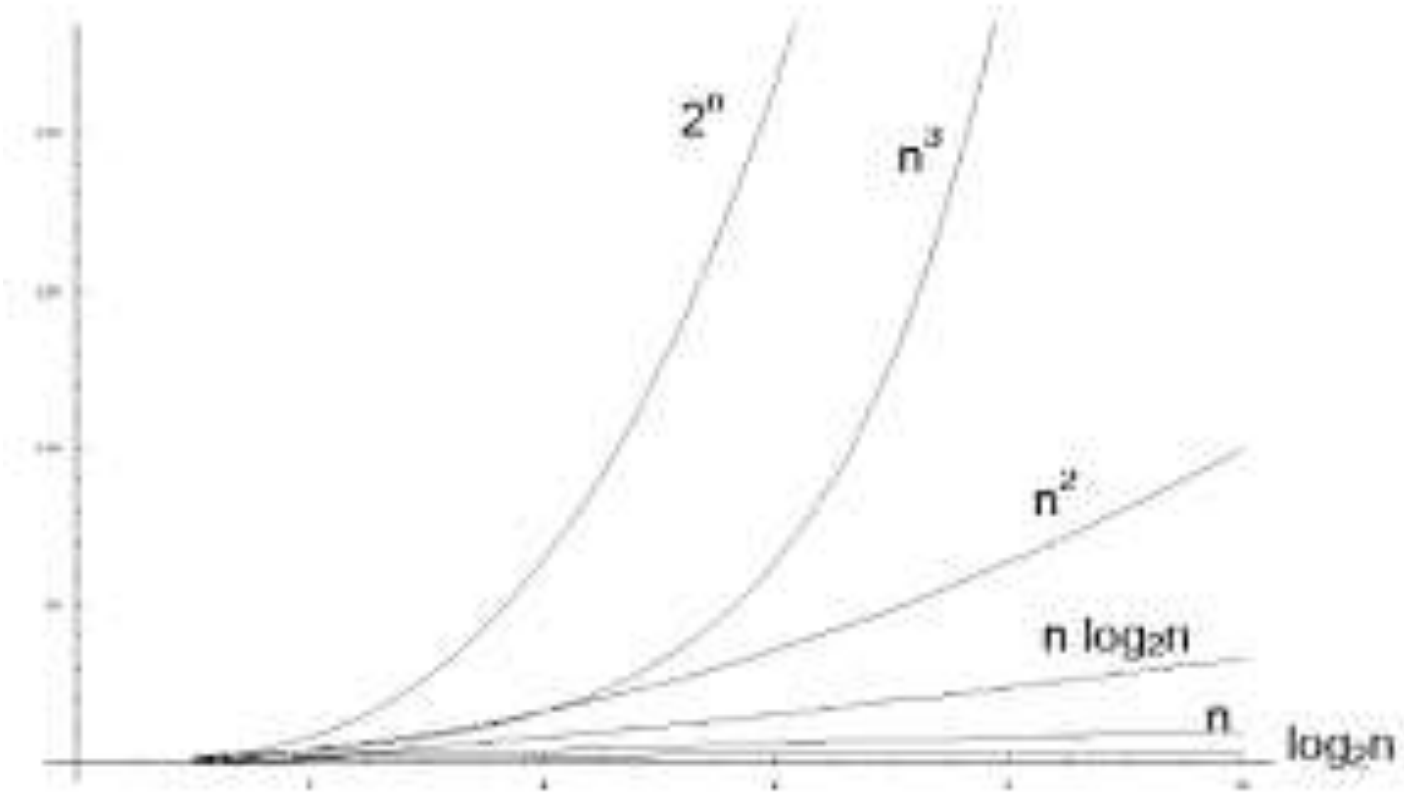
Complexidade Assintótica

- Operações Fundamentais
  - Aritméticas
    - Soma, subtração, multiplicação, divisão, resto, piso, teto,...
  - Movimentação de Dados
    - Carregar, armazenar, copiar, atribuições
  - Controle
    - Desvio condicional e incondicional, chamada e retorno de sub-rotinas
- Cada uma dessas instruções demora um período constante.

Classes de Comportamento Assintótico



Classes de Comportamento Assintótico



Complexidade

- Exercício
  - Calcule a complexidade do seguinte trecho de código
    - Qual a função de complexidade?
    - Qual a complexidade assintótica?

```
algoritmo1 (v: vetor; n: integer);
início
      para  i ← 1 até n faça
      Início
            soma ← v[1];
            para j ← 2 até i faça
            início
                  soma ← soma + v[j]
            fim
      fim
fim
```

Exercício

```
algoritmo3 (n: inteiro);
inicio
    para i de 1 até n faça
    inicio
        para j de i até 7*n faça
        inicio
            para k de 1 até 2*n faça
            inicio
                escreva(“@”);
            fim
        fim
    fim
fim
```

Exercício

MÁXIMO

Entrada Uma seqüência de números $a_1, \ldots, a_n$ com $n > 0$.

Saída O máximo $m = \max_i a_i$.

```
1  m := a_1
2  for i := 2, ..., n do
3     if a_i > m then
4        m := a_i
5     end if
6  end for
7  return m
```

Exercício

BUSCA SEQÜENCIAL

Entrada Uma seqüência de números $a_1, \ldots, a_n$ com $n > 0$ e um chave c.

Saída A primeira posição p tal que $a_p = c$ ou $p = \infty$ caso não existe tal posição.

```
1  for i := 1, ..., n do
2     if a_i = c then
3        return i
4     end if
5  end for
6  return ∞
```

Exercício

MULTIPLICAÇÃO DE MATRIZES

Entrada Duas matrizes $A = (a_{ij}) \in \mathbb{R}^{m \times n}$, $B = (b_{jk}) \in \mathbb{R}^{n \times o}$.

Saída O produto $C = (c_{ik}) = AB \in \mathbb{R}^{m \times o}$.

```
1  for i := 1, ..., m do
2     for k := 1, ..., o do
3        c_ik := 0
4        for j := 1, ..., n do
5           c_ik := c_ik + a_ij b_jk
6        end for
7     end for
8  end for
```

Exercício

```
1  int MaxMin1(int* A, int n, int* pMax, int* pMin) {
2     int i;
3     *pMax = A[0];
4     *pMin = A[0];
5     for(i = 1; i < n; i++)
6        if(*pMax < A[i]) // Comparação envolvendo os elementos
7           *pMax = A[i];
8        if(*pMin > A[i]) // Comparação envolvendo os elementos
9           *pMin = A[i];
10 }
```

Exercício

```
1  void Procedimento2() {
2     int i, j, k, x;
3     x = 0;
4     for(i = 1; i <= n; i++) {
5        for(j = 1; j <= n; j++)
6           for(k = 1; k <= j; k++)
7              x = x + j + k;
8     x = i;
9  }
```

Exercício

```
1  void Procedimento1(int n) {
2     int i, j, x, y;
3     x = y = 0;
4     for(i = 1; i <= n; i++) {
5        for(j = i; j <= n; j++)
6           x = x + 1;
7        for(j = 1; j < i; j++)
8           y = y + 1;
9     }
10 }
```

Referências online

- http://www.decom.ufop.br/reinaldo/site_media/uploads/2013-02-bcc202/aula_05_-_analise_de_algoritmos_(parte_2)_(v2).pdf

- http://www.inf.ufrgs.br/~mrpritt/lib/exe/fetch.php?media=cmp155:ca-notas-2561.pdf