ESTRUTURA DE DADOS

Engenharia da Computação

Prof. Renato Matroniani

EDUCAÇÃO METODISTA

ANÁLISE DE ALGORITMOS - INTRODUÇÃO

- criação ou utilização de algoritmos – necessário determinar seu desempenho.
- diferentes algoritmos para uma mesma função podem ter eficiência diferentes.
- eficiência neste caso: uso de memória, recursos do sistema e tempo de execução.

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

ANÁLISE DE ALGORITMOS - INTRODUÇÃO

- De acordo com Cormen (2012), “um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída”.
- Algoritmos e Estrutura de Dados: durante o processo computacional, para se obter uma saída, os algoritmos manipulam dados de entrada.
- Se os dados são dispostos ou manipulados de forma homogênea, dizemos que temos um “tipo abstrato de dados”.

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

ANÁLISE DE ALGORITMOS - INTRODUÇÃO

- Um tipo abstrato de dados é formado por um conjunto de valores e por uma série de funções que podem ser aplicados a esses valores.
- funções + valores → modelo matemático para sistemas reais
- para implementar tipos abstratos de dados em uma linguagem de programação, é necessário representá-los nas formas que já conhecemos de estruturas de dados.
- As estruturas de dados armazenam e organizam os dados para facilitar seu acesso e possibilitar modificações.

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

O QUE É ANÁLISE DE ALGORITMOS?

- De acordo com Cormen (2012), “analisar um algoritmo significa prever os recursos de que ele necessitará. De forma geral, memória, largura de banda, hardware para computação são de grande preocupação, mas com frequência é o tempo de computação que se deseja mensurar”.

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

ANÁLISE DE ALGORITMOS

- Dois tipos distintos de problemas de análise de algoritmos:
  - análise de um algoritmo particular – calcular o “custo” que determinado algoritmo terá para resolver um problema específico.
    - quantidade de vezes que o algoritmo deverá ser executado.
    - quantidade de memória necessária.
  - análise de uma classe de algoritmos – dentro de um grupo de algoritmos, para um determinado problema, determinar qual o de menor “custo”.
    - necessário a colocação de limites de tempo de execução desses algoritmos.

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

ANÁLISE DE ALGORITMOS

- Como avaliar o “custo” tempo?
- O tempo pode ser medido em computador real, porém:
  - depende do compilador (que pode privilegiar certas construções);
  - depende do hardware (processador, barramento etc.)
  - depende da memória disponível.
- Para o caso 2, por exemplo, variáveis devem ser fixadas.

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

ANÁLISE DE ALGORITMOS

- Como avaliar o “custo” tempo?
- Outra forma é através de modelos matemáticos ou modelos de computação genéricos, com um único processador – Máquina de Acesso Aleatório (*Random Access Machine* - RAM também, não confundir).

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

ANÁLISE DE ALGORITMOS

- Características do modelo de RAM
  - as instruções são executadas uma após a outra, sem operações concorrentes;
  - contém instruções aritméticas
  - contém instruções de movimentação de dados
  - contém instruções de controle.
- Por exemplo, nos algoritmos de ordenação, considera-se o número de comparações entre os elementos do conjunto a ser ordenado e desconsideram-se as operações aritméticas, de atribuição e manipulação de índices;
- Outra simplificação é que cada instrução demora um tempo constante.

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

FUNÇÃO COMPLEXIDADE DE TEMPO DO ALGORITMO

- Função de custo T
- T(n): medida do tempo necessário para executar um algoritmo cujo problema possui tamanho n.
- Como funciona isso na prática:
- T(n) não representa diretamente o tempo de execução, mas o número de vezes que certa operação relevante é executada.

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

FUNÇÃO COMPLEXIDADE DE TEMPO DO ALGORITMO

- A função de complexidade de tempo T(n) é o número de comparações dos elementos do vetor A.
- Quantas comparações serão necessárias?

```
{
        int i, menor;

        menor = A[0];
        for (i = 1; i < n; i++)
        {
                if (A[i] < menor)
                        menor = A[i];
        }
        return menor;
}
```

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

FUNÇÃO COMPLEXIDADE DE TEMPO DO ALGORITMO

- Se A = [2, 4, 5, 7, 3, 9, 1, 0],

então n = 8 e A[0] = 2

portanto:

```
{
        int i, menor;

        menor = A[0];
        for (i = 1; i < n; i++)
        {
                if (A[i] < menor)
                        menor = A[i];
        }
        return menor;
```

n = 8
A[0] = 2
menor = 0

pior T(n) = 8
melhor T(n) = 1

médio $T(n) = \frac{n+1}{2} = 4,5$

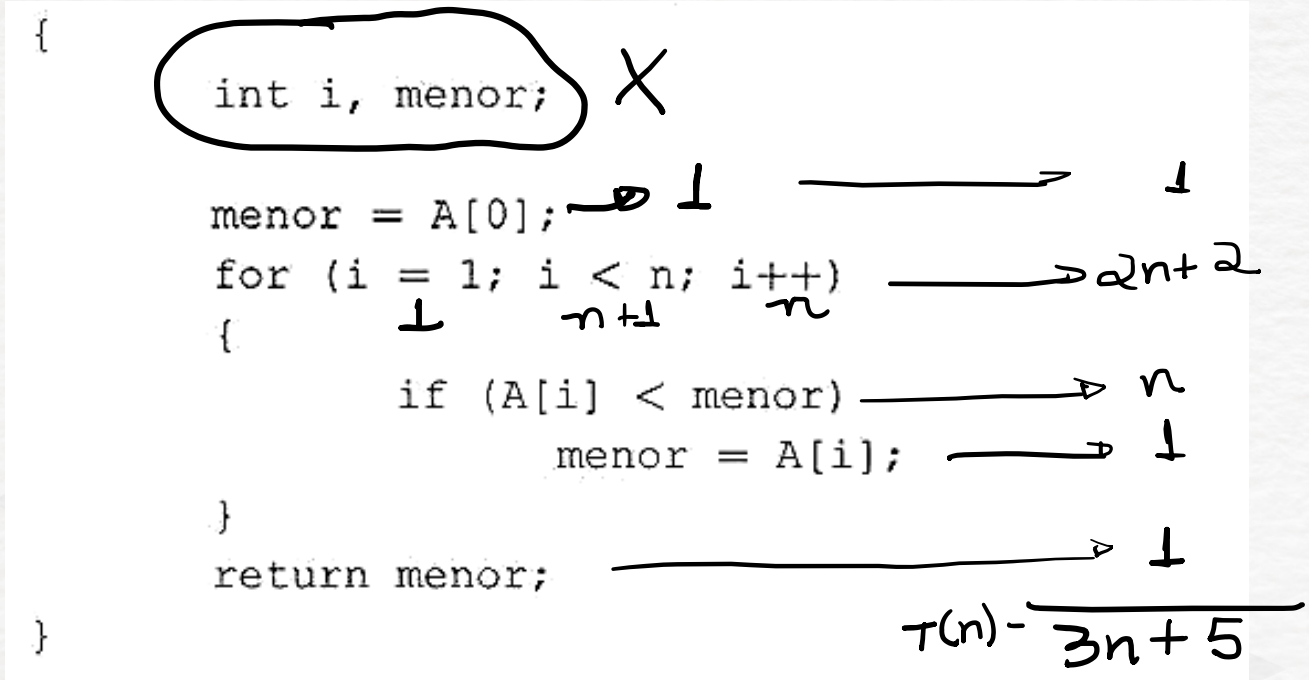
↑ ALG. mais eficiente

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

FUNÇÃO COMPLEXIDADE DE TEMPO DO ALGORITMO

- Se A estiver ordenado, o tempo gasto será o mesmo?
- Neste exemplo, o tempo de execução é uniforme (não necessariamente igual) para qualquer tamanho de entrada n?



FUNÇÃO COMPLEXIDADE DE TEMPO DO ALGORITMO: EXEMPLO DE BUSCA (INTROD.)

- Basicamente a operação necessária é buscar uma chave dentro de uma outra chave armazenada, fazendo comparações sucessivas (em um registro ou vetor).
- O caso mais comum é a busca sequencial, onde o algoritmo não se comporta uniformemente como no algoritmo de busca do menor valor (por conta da recorrência das rotinas).
- Temos então, baseados em complexidade de tempo, o pior, o melhor e o caso médio. Se a entrada tem tamanho n, então:
- pior caso $T(n) = n$
- melhor caso $T(n) = 1$
- caso médio $T(n) = (n+1)/2$

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

FUNÇÃO COMPLEXIDADE DE TEMPO DO ALGORITMO: EXEMPLO DE BUSCA (INTROD.)

- O tempo de execução do algoritmo aumenta à medida que aumenta a entrada n.
- Para valores baixos de n, o algoritmo (qualquer que seja) será considerado eficiente.
- A análise é válida para tamanhos “grandes” de n
- Para isso, estuda-se o **comportamento assintótico** da função complexidade de tempo dos algoritmos, sua **eficiência assintótica**.
- Como o tempo de execução de um algoritmo aumenta, dependendo do aumento do tamanho da entrada.

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

FUNÇÃO COMPLEXIDADE DE TEMPO DO ALGORITMO: EXEMPLO

- Cálculo da soma de cubos

```
inteiro somaCubos (inteiro n)
                    inteiro i, somaParcial;
                    início
1                         somaParcial = 0;
2                         para i de 1 até n faça
3                            somaParcial = somaParcial + i * i * i;
4                         fim para
5                         retorne somaParcial;
                    fim
```

- Quais linhas tomam tempo e quais não tomam?

Texto base/fonte: https://lapix.ufsc.br/ensino/estrutura-de-dados/complexidade-de-algoritmos/

EDUCAÇÃO METODISTA

NÇÃO COMPLEXIDADE DE TEMPO DO GORITMO: EXEMPLO

As declarações das variáveis e a inha 4 não tomam tempo.

As linhas 1 e 5 contabilizam uma unidade de tempo cada.

A linha 3 conta 4 unidades de empo, pois há duas multiplicações, uma adição e uma atribuição) e como é executada n vezes, temos um total de 4n unidades de tempo.

A linha 2 possui custos de inicializar o i, testar se é menor que n e fazer ++. Contamos 1 unidade para sua nicialização, n + 1 para todos os estes e n para todos os ncrementos, o que perfaz 2n + 2 unidades de tempo.

O total perfaz 6n + 4 unidades de empo, o que indica que o algoritmo é O(n), da Ordem de Complexidade a, ou seja, linear.

- Cálculo da soma de cubos

```
inteiro somaCubos (inteiro n)
                    inteiro i, somaParcial;
                    início
1       1                       somaParcial = 0;
2       2n+2                      para i de 1 até n faça
3       4n                              somaParcial = somaParcial + i * i * i;
4                               fim para
5       1                       retorne somaParcial;
                fim
```

- Quais linhas tomam tempo e quais não tomam?

EDUCAÇÃO METODISTA

Texto base/Fonte: https://lapix.ufsc.br/ensino/estrutura-de-dados/complexidade-de-algoritmos/

ANÁLISE ASSINTÓTICA – ELEMENTOS

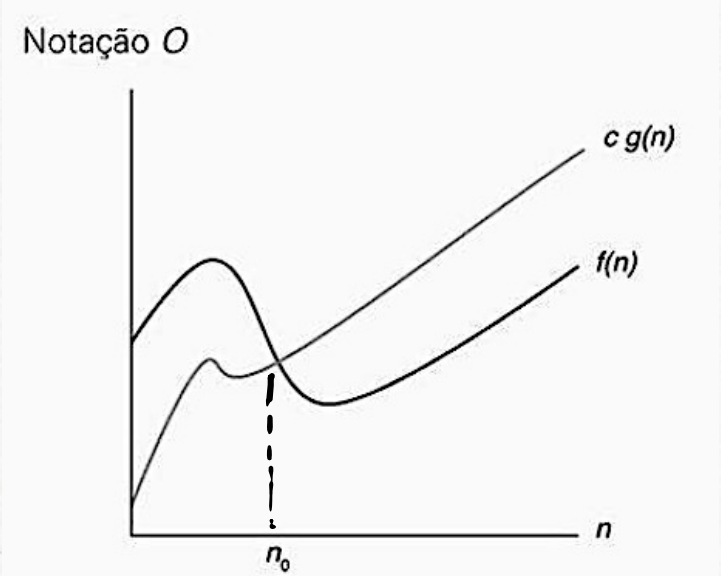
- As notações assintóticas são utilizadas para representar o comportamento assintótico das funções de complexidade de tempo dos algoritmos, bem como relacionar o comportamento das funções de complexidade de dois algoritmos.
- "Uma função $g(n)$ domina assintoticamente outra função $f(n)$ se existem duas constantes positivas c e n_0 tais que, para $n \geq n_0$, temos que $|f(n)| \leq c.|g(n)|$" (ver gráfico):

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

ANÁLISE ASSINTÓTICA – ELEMENTOS

- Comparação de funções que representam a complexidade do tempo de dois algoritmos. Estudar o comportamento/taxa de crescimento.
- Existem outras notações, além da O.



Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

ANÁLISE ASSINTÓTICA – EXEMPLOS

- Exemplo 1: verificar a assintoticidade de $|n| \leq c.|-(n^2)|$ para $n > c \; e \; n > 1$, para qualquer c.

$n > c$

$n > 1$

$f(n) = |n|$

$g(n) = c|-(n^2)|$

p/ n

$c = 2$

$n = 2$

$f(2) = 2$

$g(2) = 2.|-(2^2)|$

$g(2) = 2 \cdot 4 = 8$

$f(3) = 3$

$g(3) = 2.|-(3^2)|$

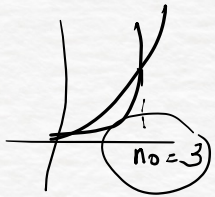
$g(3) = 2 \cdot 9 = 18$

,

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

ANÁLISE ASSINTÓTICA – EXEMPLOS



- Exemplo 2: Considere $f(n) = (n+1)^3$ e $g(n) = n^3$. Provar que $g(n)$ domina assintoticamente $f(n)$, para $c = 3$, $n_0 = 3$ e $n \geq 3$. E para $c = 1$ e $n_0 = 1$ e $n \geq 1$?

f(n) = (n+1)^3 →

f(1) = 8

f(2) = (2+1)^3 = 9

f(3) = (3+1)^3=64

f(4) = (4+1)^3 = 125

f(5) = 216

g(1) = 3

g(2) = 3.(2^3) = 24

g(3) = c.n^3=3.(3^3) = 81

g(4) = 3.(4^3) = 192

g(5) = 375

$f(n) = (n+1)^3$ $\quad g(n) = 3.n^3$

$f(3) = (3+1)^3 = 64$ $\quad g(3) = 3.3^3 = 81$

$f(2) = (2+1)^3 = 27$ $\quad g(2) = 3.2^3 = 24$

$f(n) = (n+1)^3$ $\quad g(n) = 1.n^3$

$f(1) = (1+1)^3 = 8$ $\quad g(1) = 1.1^3 = 1$

$f(2) = (2+1)^3 = 27$ $\quad g(2) = 1.2^3 = 8$

$f(3) = (3+1)^3 = 64$ $\quad g(3) = 1.3^3 = 27$

$f(4) = (4+1)^3 = 125$ $\quad g(4) = 1.4^3 = 64$

$f(5) = (5+1)^3 = 216$ $\quad g(5) = 1.5^3 = 125$

$f(6) = (6+1)^3 = 343$ $\quad g(6) = 1.6^3 = 216$

c=1

n≥1

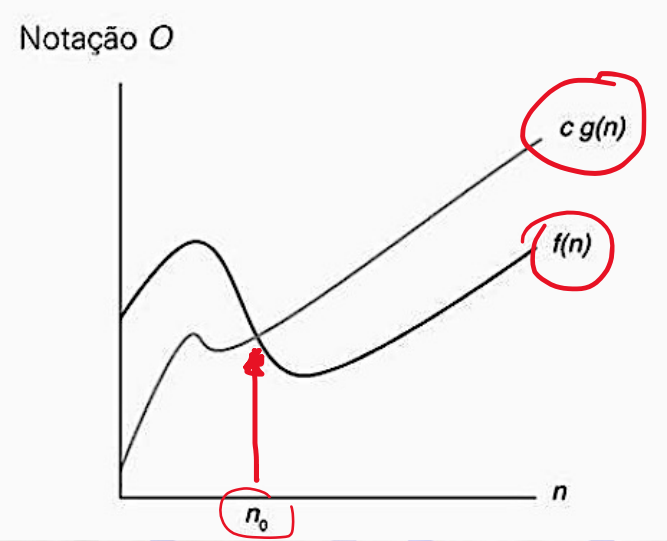
n_0=1

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.

EDUCAÇÃO METODISTA

ANÁLISE ASSINTÓTICA – APLICAÇÃO EM ANÁLISE DE ALGORITMOS

- Notação O (Big-Oh): é utilizada para dar um limite assintótico superior dentro de um fator constante. A partir de um certo valor n_0, $c.g(n)$ estará acima de $f(n)$.



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

$g(n) = 2n^2$

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S. e https://lapix.ufsc.br/ensino/estrutura-de-dados/complexidade-de-algoritmos/

ANÁLISE ASSINTÓTICA – APLICAÇÃO EM ANÁLISE DE ALGORITMOS

- A Notação Big-O me fornece a Ordem de Complexidade ou a Taxa de Crescimento de uma função .
- Para isso, não consideramos os termos de **ordem inferior** da complexidade de um algoritmo, apenas o **termo predominante**. Exemplo: Um algoritmo tem complexidade $T(n) = 3n^2 + 100n$.
- Nessa função, o segundo termo tem um peso relativamente grande, mas a partir de $n_0 = 11$, é o termo n_2 que "dá o tom" do crescimento da função: uma parábola. A constante 3 também tem uma influência irrelevante sobre a taxa de crescimento da função após um certo tempo. Por isso dizemos que este algoritmo é da ordem de n^2 ou que tem complexidade $O(n^2)$.

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S. e https://lapix.ufsc.br/ensino/estrutura-de-dados/complexidade-de-algoritmos/

ANÁLISE ASSINTÓTICA – NOTAÇÃO O

- A Notação Big-O me fornece a Ordem de Complexidade ou a Taxa de Crescimento de uma função .
- Para isso, não consideramos os termos de **ordem inferior** da complexidade de um algoritmo, apenas o **termo predominante**. Exemplo: Um algoritmo tem complexidade $T(n) = 3n^2 + 100n$.
- Nessa função, o segundo termo tem um peso relativamente grande, mas a partir de $n_0 = 34$, é o termo n^2 que "dá o tom" do crescimento da função: uma parábola. A constante 3 também tem uma influência irrelevante sobre a taxa de crescimento da função após um certo tempo. Por isso dizemos que este algoritmo é da ordem de n^2 ou que tem complexidade $O(n^2)$.

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

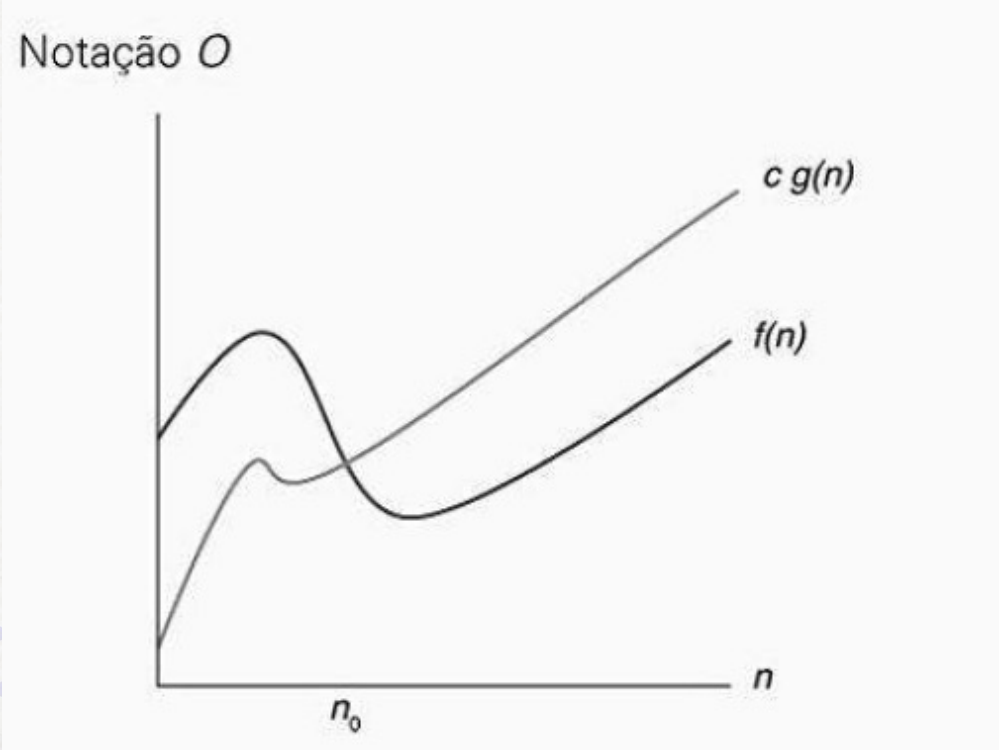
ANÁLISE ASSINTÓTICA – NOTAÇÃO O

- A Notação Big-O é também chamada de “Ordem de Complexidade”.
- Na análise assintótica de algoritmos precisamos sempre pensar no “pior caso”.
- a complexidade de tempo é limitada superiormente por um polinômio da forma cn^2, c sendo uma constante.
- Conforme Ascencio e Araújo (2010), a notação O é utilizada para dar um limite assintótico superior sobre uma função, dentro de um fator constante.
- Na figura do próximo slide, para todos os valores de n à direita de n_0, o valor da função f(n) é igual oou está abaixo de g(n).

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ANÁLISE ASSINTÓTICA – NOTAÇÃO O



Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ANÁLISE ASSINTÓTICA – NOTAÇÃO O

- Exemplos
- se $f(n) = \frac{1}{3}n^2 - 3n$ então $f(n)$ é $O(n^2)$, quando $c = \frac{1}{3}$ e $n_0 = 1$.
- se $f(n) = (n+1)^2$ então $f(n)$ é $O(n^2)$.
- se $f(n) = 2n^3 + 3n^2 + n$ então $f(n)$ é $O(n^3)$.

- a escolha das duas constantes c e n_0 devem sempre existir.

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ANÁLISE ASSINTÓTICA – NOTAÇÃO O

- A partir disso:
- “uma função $f(n)$ é $O(g(n))$ se existem duas constantes positivas, c e n_0 tais que:

$$f(n) \leq cg(n) \text{ para todo } n \geq n_0\text{”}$$

- Podemos dizer que a função O representa a “ordem de complexidade” ou “taxa de crescimento” de uma função.

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S., 2010

ANÁLISE ASSINTÓTICA – NOTAÇÃO O

- Por um lado, estamos dizendo que f(n) está limitada por g(n) de cima para baixo, ou que f(n) é uma função "menor que" g(n). Outro modo formal de dizer isso é que f(n) é **assintoticamente limitada** por g(n).”

EDUCAÇÃO METODISTA

Texto base/Fonte: Tanenbaum (2008)

NOTAÇÃO O – CONSIDERAÇÕES

- Embora uma função possa ser assintoticamente limitada por várias outras funções, como por exemplo:

$$f(n) = 10n^2 + 37n + 153 \text{ é}$$
$$O(n^2), O(10n^2), O(37n^2 + 10n) e\ O(0{,}05n^2).$$

- Em geral procuramos um limite assintótico que seja um único termo com um coeficiente inicial de 1 e que se aproxime o máximo possível.
- Sendo assim, diríamos que $f(n) = 10n^2 + 37n + 153$ é $O(n^2)$, embora seja também assintoticamente limitado por várias outras funções.

EDUCAÇÃO METODISTA

Texto base/Fonte: Tanenbaum (2008)

NOTAÇÃO O – CONSIDERAÇÕES

- Se f(n) é uma constante ou um polinômio, isso pode ser sempre feito usando seu termo mais alto com um coeficiente de 1.
- Entretanto, para funções mais complexas, nem sempre é possível encontrar um ajuste tão perfeito.

EDUCAÇÃO METODISTA

Texto base/Fonte: Tanenbaum (2008)

NOTAÇÃO O – CONSIDERAÇÕES

- Funções de ordem polinomial e exponencial:
- As funções que são $O(n^k)$ são consideradas de **ordem polinomial**, enquanto as que são $O(d^n)$ para algum $d > 1$ mas não $0(n^k)$ para qualquer k são consideradas de **ordem exponencial**.

EDUCAÇÃO METODISTA

Texto base/Fonte: Tanenbaum (2008)

NOTAÇÃO O – CONSIDERAÇÕES

- A diferença entre as funções de ordem polinomial e as de ordem exponencial é muito importante.
- Mesmo uma pequena função de ordem exponencial, como 2^n, cresce bem mais do que qualquer função de ordem polinomial, como n^k, independentemente do tamanho de k.
- Exemplo: 2^{10} é igual a 1024, mas que 2^{100} (ou seja, 1024^{10}) é maior que o número formado por um número 1 seguido por 30 zeros.
- O menor k para o qual 10^k excede 2^{10} é 4, mas o menor k para o qual 100^k excede 2^{100} é 16.
- À medida que n se torna maior, são necessários valores maiores de k para que n^k acompanhe 2^n.

EDUCAÇÃO METODISTA

Texto base/Fonte: Tanenbaum (2008)

NOTAÇÃO O – CONSIDERAÇÕES

- Devido à elevada taxa de crescimento das funções de ordem exponencial, os problemas que exigem algoritmos de tempo exponencial para sua solução são considerados **intratáveis** no atual ambiente de computação, ou seja, tais problemas não podem ser solucionados com precisão, **exceto nos casos mais simples.**

EDUCAÇÃO METODISTA

Texto base/Fonte: Tanenbaum (2008)

NOTAÇÃO O – CLASSES DE PROBLEMAS

- Se f(n) = O(1), temos algoritmos com complexidade constante, pois independe do valor de n. O algoritmo possui instruções e/ou rotinas que são executadas um número fixo de vezes.
- Se f(n) = O(log n), temos um algoritmo com complexidade logarítmica, onde os algoritmos “se dividem” em algoritmos de menor complexidade.
- Por exemplo: Considere $log_2 n = 10$ e $log_2 n = 20$. Qual o valor de n em cada um dos casos? Qual o comportamento de n? Se eu mudar a base para 10, para um valor alto de n, a mudança é significativa?

EDUCAÇÃO METODISTA

Texto base/Fonte: Nívio Ziviani, UFMG

NOTAÇÃO O – CLASSES DE PROBLEMAS

- Se f(n) = O(n), temos algoritmos com complexidade linear.
- Se f(n) = O(n.log n), temos algoritmos que além de se dividirem em algoritmos mais simples, juntam as soluções dos mais simples depois.
- Calcule $nlog_2 n$ para n = 1 milhão e 2 milhões. Qual a diferença?
- Se f(n) = $O(n^2)$, temos algoritmos com complexidade quadrática, que são algoritmos que processam as entradas de dados aos pares.
- Se f(n) = $O(n^3)$, temos algoritmos com complexidade cúbica, e assim por diante.
- Se f(n) = $O(2^n)$, temos algoritmos com complexidade exponencial.

EDUCAÇÃO METODISTA

Texto base/Fonte: Nívio Ziviani, UFMG

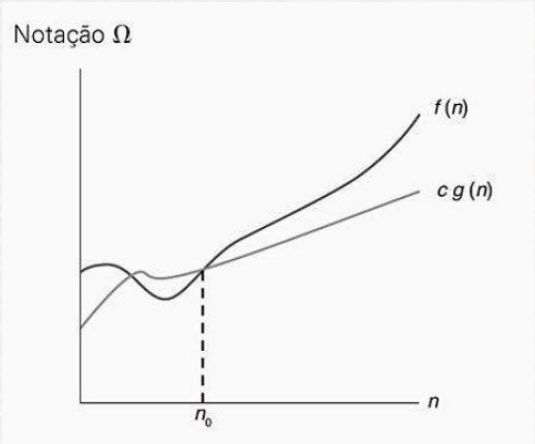
NOTAÇÃO O – CLASSES DE PROBLEMAS

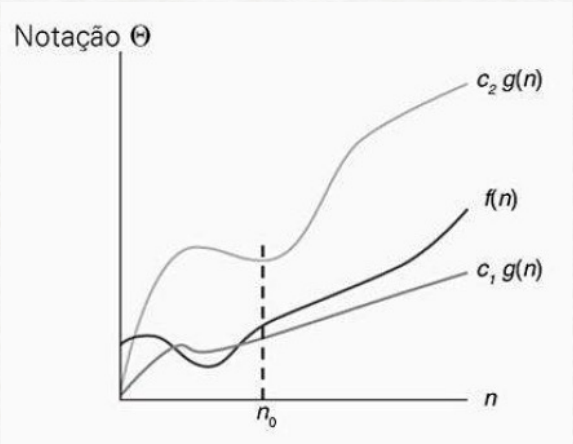
Função de custo	Tamanho n					
	10	20	30	40	50	60
n	0,00001 s	0,00002 s	0,00003 s	0,00004 s	0,00005 s	0,00006 s
n^2	0,0001 s	0,0004 s	0,0009 s	0,0016 s	0,0.35 s	0,0036 s
n^3	0,001 s	0,008 s	0,027 s	0,64 s	0,125 s	0.316 s
n^5	0,1 s	3,2 s	24,3 s	1,7 min	5,2 min	13 min
2^n	0,001 s	1 s	17,9 min	12,7 dias	35,7 anos	366 séc.
3^n	0,059 s	58 min	6,5 anos	3855 séc.	10^8 séc.	10^{13} séc.

Fonte: Nívio Ziviani, UFMG

EDUCAÇÃO METODISTA

OUTRAS NOTAÇÕES





Notação Ω: utilizada para dar limite assintótico inferior.

Notação Θ: utilizada para dar limite assintótico firme (entre duas funções).

Notação o: fornece limite assintótico superior não restrito.

Notação ω: fornece limite assintótico inferior não restrito.

EDUCAÇÃO METODISTA

Texto base/Fonte: ASCENCIO, A. F. G., ARAÚJO, G . S.