ECM306 – Tópicos Avançados em Estrutura de Dados

Aula 05

Engenharia da Computação – 3ª série

Crescimento Assintótico de Funções

(E1, E2)

2024

Pergunta

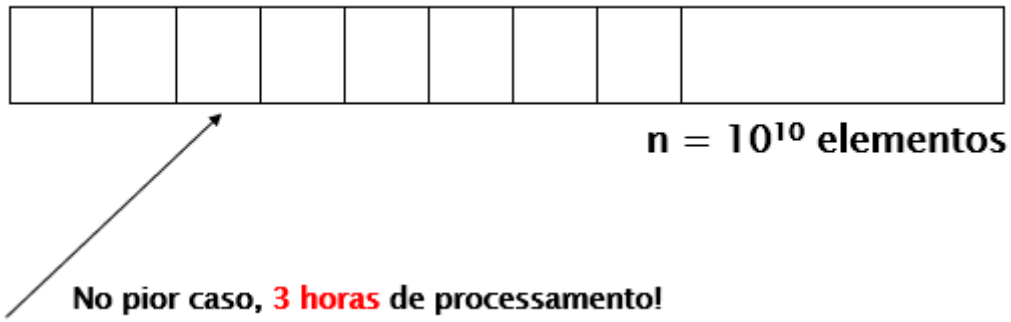
- O que é o mais importante para a execução de uma tarefa computacional?

Resposta

- Para a execução de uma tarefa computacional, o mais importante é projetar um algoritmo correto;
- Um algoritmo é dito correto se ele atende à especificação da tarefa requerida;
- Entretanto, a despeito de ser correto, um algoritmo pode ter execução impraticável;
- Por exemplo, a pesquisa linear em um ***array*** é correta, mas impraticável se o ***array*** tiver 10^{10} elementos.

Exemplo

- Pesquisa Linear em um ***array*** com 10^{10} elementos;
- Tempo para se processar um elemento do array: 10^{-6} segundos;
- No pior caso, serão necessários 10.000 segundos ou cerca de 3 horas para se buscar o elemento arbitrário no ***array***.



Conclusão

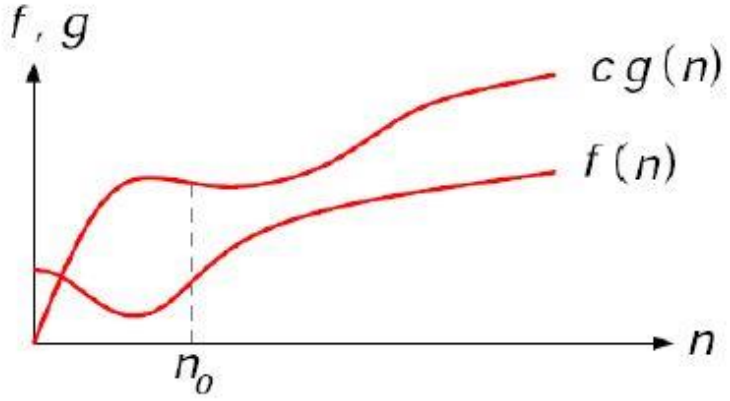
- Assim, algoritmos devem ser corretos mas também eficientes . . .

Introdução

- É difícil se determinar, de forma exata, o tempo de execução de um algoritmo;
- Em geral, cada passo em um pseudocódigo e cada ***statement*** de implementações em Linguagem de Alto Nível, ou *HLL – High Level Language*, corresponde a um pequeno número de operações primitivas que não depende do tamanho da entrada;
- Assim, pode-se executar uma análise simplificada que estima o número de operações primitivas, por meio da contagem dessas operações;

Introdução

- Felizmente, há uma notação que permite caracterizar os principais fatores que afetam o desempenho de um algoritmo, sem levar em conta os detalhes dessas operações primitivas;



$$f(n) = O(g(n))$$

Crescimento Assintótico de Funções

Pergunta

- O que é Análise Assintótica?

Resposta

- Ao se deparar com uma Função de Complexidade definida por ***F(n) = n + 10*** ou ***F(n) n^2 + 1***, geralmente pensa-se em valores não muito grandes de ***n***, ou ainda valores próximos de **zero**;
- Na Análise de Algoritmos atua-se, exatamente, de forma contrária:
  - Ignora-se valores pequenos de ***n*** e foca-se em valores grandes, suficientemente grandes, de ***n***;
- Esse tipo de análise é denominada **Análise Assintótica**.

Exemplo

- Considerando-se, então, o número de operações de dois Algoritmos que resolvam um mesmo problema em função de ***n***, onde ***n*** corresponde ao tamanho da entrada:
  - Algoritmo A: $F_1(n) = 2n^2 + 50$ operações;
  - Algoritmo B: $F_2(n) = 500\,n + 4000$ operações.
- Dependendo do valor de ***n***, o Algoritmo A pode requerer mais ou menos operações que o Algoritmo B.

Exemplo

- Algoritmo A: $F_1(n) = 2n^2 + 50$ operações;
  - Para $n = 10$, serão necessárias $2.10^2 + 50 = 250$ operações;
  - Para $n = 100$, serão necessárias $2.100^2 + 50 = 20050$ operações;
- Algoritmo A: $F_2(n) = 500\,n + 4000$ operações;
  - Para $n = 10$, serão necessárias $500.10 + 4000 = 9000$ operações;
  - Para $n = 100$, serão necessárias $500.100 + 4000 = 54000$ operações;

Exemplo

- Assim, o importante é se observar que $F_1(n)$ tem crescimento proporcional à n^2, ou **quadrático**;
- Ao passo que $F_2(n)$ tem crescimento proporcional à n, ou **linear**;
- Sendo assim, um crescimento **quadrático** é **PIOR** que um crescimento **linear**;
- Portanto, na comparação das Funções F_1 e F_2, deve-se preferir o **Algoritmo B**.

Conclusão

- Considerando que é muito difícil se levantar a quantidade exata de operações executadas por um algoritmo, a **Análise de Algoritmos** concentra-se, então, no comportamento assintótico das funções de complexidade, ou seja, deve-se observar a taxa de crescimento da função quando ***n*** é suficientemente grande;
- Em geral, os termos inferiores e as constantes multiplicativas pouco contribuem na análise e podem, dessa forma, serem **descartadas**.
- .

Conclusão

- Para valores suficientemente grandes de ***n***, as funções:

  n^2 , $7/2\ n^2$, $555555\ n^2$, $n^2/8888$, $7\ n^2 + 300\ n + 4$

- Crescem todas com a mesma velocidade e, portanto, do ponto de vista assintótico, são "**equivalentes**";
- Na área de **Análise de Algoritmos**, as funções de Complexidade são classificadas em "**ordens**";
- Todas as funções de uma mesma **ordem** são "**equivalentes**";
- Essas cinco funções apresentadas, portanto, pertencem à mesma ordem, ou seja, **ordem quadrática**.

Crescimento Assintótico de Funções

Pergunta

- O que são Funções Assintoticamente Não Negativas?

Crescimento Assintótico de Funções

Resposta

- Na Análise de Algoritmos restringe-se o estudo para funções assintoticamente não-negativas, ou seja, uma função ***f*** tal que ***f(n) >= 0***, para todo ***n*** suficientemente grande;

- Mais explicitamente, ***f*** é assintoticamente não-negativa se existe um n_0 tal que ***f(n) >= 0***, para todo $n > n_0$.

Exemplo

- Por exemplo, o tempo exato de execução de um algoritmo pode ser dado pela função polinomial $f(n) = 3n^2 + 2n + 3$.
- Neste caso, o tempo aproximado de execução será uma função de n^2, ou seja $f(n^2)$, a mais alta potência de n;
- Dessa forma, pode-se desprezar o coeficiente de n^2, bem como os outros termos da **função polinomial** que define a **complexidade** do algoritmo;

Conclusão

- Assim, para efeito de Análise de Algoritmos, utiliza-se uma notação que seja capaz de exprimir a ordem de grandeza do tempo de execução;
- Essa notação é ***assintótica***, ou seja, representa uma linha que se aproxima da função de complexidade do algoritmo.

Pergunta

- O que é a Notação Big-Oh?

Crescimento Assintótico de Funções

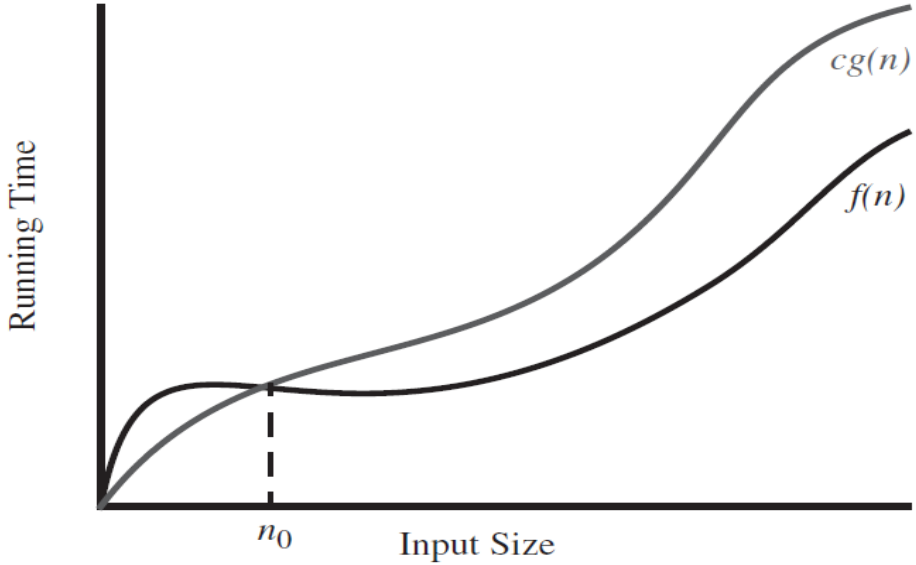
Resposta

- Seja ***f(n)*** e ***g(n)*** funções que mapeiam inteiros não negativos para números reais;
- Diz-se que ***f(n)*** é ***O(g(n))*** se existir uma constante real ***c > 0*** e uma constante inteira $n_o \geq 1$ tal que $f(n) \leq cg(n)$ para todo inteiro $n \geq n_0$;

Crescimento Assintótico de Funções

Resposta

- Essa definição é frequentemente dita: "***f(n) é Big-Oh de g(n)***" ou "***f(n) é ordem g(n)***":



The function $f(n)$ is $O(g(n))$, for $f(n) \leq c \cdot g(n)$ when $n \geq n_0$.

ECM306 – Tópicos Avançados em Estrutura de Dados

Referências bibliográficas

- CORMEN, T.H. et al. Algoritmos: Teoria e Prática (Caps. 13). Campus. 2002.
- ZIVIANI, N. Projeto de algoritmos: com implementações em Pascal e C (Cap. 1). 2.ed. Thomson, 2004.
- FEOFILOFF, P. Minicurso de Análise de Algoritmos, 2010. Disponível em:

  http://www.ime.usp.br/~pf/livrinho-AA/
- DOWNEY, A.B. *Analysis of algorithms* (Cap. 2), Em: *Computational Modeling and Complexity Science.* Disponível em:

  http://www.greenteapress.com/compmod/html/book003.html
- ROSA, J.L. Notas de Aula de Introdução a Ciência de Computação II. Universidade de São Paulo. Disponível em:

  http://coteia.icmc.usp.br/mostra.php?ident=639

ECM306 – Tópicos Avançados em Estrutura de Dados

Referências bibliográficas

- GOODRICH, Michael T. et al: *Algorithm Design and* Applications. Wiley, 2015.
- LEVITIN, Anany. *Introduction to the Design and Analysis of Algorithms*. Pearson, 2012.
- SKIENA, Steven S. *The Algorithm Design Manual.* Springer, 2008.
- Série de Livros Didáticos. *Complexidade de Algoritmos.* UFRGS.
- BHASIN, Harsh. *Algorithms – Design and Analysis.* Oxford University Press, 2015.
- FREITAS, Aparecido V. de – 2022 – Estruturas de Dados: Notas de Aula.
- CALVETTI, Robson - 2015 – Estruturas de Dados: Notas de Aula.

Aula 05

FIM