ESTRUTURAS DE DADOS

Aula 1 – Introdução à disciplina

Centro Universitário da FEI

Estrutura de Dados

Guilherme Alberto Wachs Lopes

- Graduação em Ciência da Computação (FEI)
- Mestre em Engenharia Elétrica(FEI)
- Doutorando em Engenharia Elétrica (FEI)

Estrutura de Dados

Centro Universitário da FEI

Planejamento

1. Introdução à disciplina. Noções de complexidade
2. Listas lineares e encadeadas
3. Listas duplamente ligadas
4. Listas ligadas circulares
5. Pilhas, filas FIFO e filas com prioridade.
6. Tabelas Hash
7. Árvores
8. Árvores Binárias
9. Árvores AVL: conceitos e implementação
10. Teoria dos Grafos
11. Matrizes Esparsas
12. Outras estruturas de Dados (Heap, Arvores, etc)

Estrutura de Dados

Centro Universitário da FEI

Critérios de Avaliação

$$M = \frac{A + 2P_1 + 3P_2}{6}$$

Onde:

$0 \leq A \leq 10$:Nota de exercícios teóricos e do laboratório.

ESTRUTURA DE DADOS

Revisão e Requisitos para a Disciplina

Estrutura de Dados
Funções e Procedimentos

Centro Universitário da FEI

- Declaração de Procedimento em C
- Declaração de Funções em C
- Passagem de parâmetros

Estrutura de Dados
Ponteiros

- Alocação (malloc)
- Desalocação (free)
- Endereço de memória (&)
- Acesso à memória (*)

Estrutura de Dados
Vetores

- Criação de vetores
- Vetores como ponteiros

Estrutura de Dados
Strings

- O tipo char*
- O tipo string
- O caracter ‘\0’

Estrutura de Dados
Estruturas

- Criação de tipos com *struct*

Estrutura de Dados
Arquivos

- Leitura e gravação textual
- Leitura e gravação binária

Introdução

- O que são algoritmos?
  - "...qualquer procedimento computacional bem definido que tenha algum valor ou um conjunto de valores de entrada e produz algum valor ou um conjunto de valores de saída."

- Para que estudar técnicas de algoritmos ?

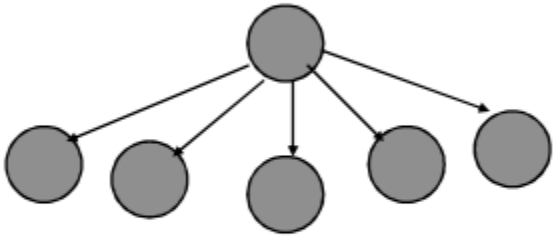
  - Criar um algoritmo para solucionar problemas é MUITO fácil:
    - Programa que cria teses
    - Busca em Largura (testa todas as possibilidades)
  - O difícil é esperar a resposta....

  - O DESAFIO então é criar algoritmos que consigam solucionar problemas em tempo razoável

Introdução

- Por que os algoritmos são ineficientes?
  - Explosão Combinatória

- Exemplo:
  - Um jogo de xadrez tem, a cada jogada, algo em torno de 35 possibilidades (fator de ramificação b=35)



Suponha:
fator de expansão b = 10
1.000 nós gerados/segundo
cada nó ocupa 100 bytes

Profundidade	Nós	Tempo	Memória
0	1	1 milissegundo	100 bytes
2	111	0.1 segundo	11 quilobytes
4	11111	11 segundos	1 megabytes
6	10^6	18 minutos	111 megabytes
8	10^8	31 horas	11 gigabytes
10	10^{10}	128 dias	1 terabyte
12	10^{12}	35 anos	111 terabytes
14	10^{14}	3500 anos	11111 terabytes

Introdução

- Como fazer um algoritmo eficiente ?
  - Analisando a complexidade do problema e do algoritmo proposto para solução do problema
  - Aplicando técnicas de programação

- O que seria, basicamente, essa análise ?
  - Analisar o comportamento do algoritmo face uma número n de entradas

- O que seriam, basicamente, essas tais técnicas de programação ?
  - Estrutura de dados avançada
  - Métodos de solução e divisão de problemas

Estrutura de Dados

- Importância
  - Suponha que eu tenho um conjunto de números para armazenar. Qual a melhor forma de armazená-los ?

  - Exemplo de duas estruturas
    - Estrutura 1: Vetor Booleano → 10010100101010010100001111O
    - Estrutura 2: Vetor inteiros → [1,4,6,9,11,13,16,18,23,24,25,26]

- A melhor estrutura depende de como irá usar esses números:
  - Se você precisar descobrir se um número está ou não no conjunto, então a estrutura 1 é melhor
  - Se você precisar descobrir QUAIS números fazem parte do conjunto, então a estrutura 2 é melhor

Estrutura de Dados

- Uma estrutura de dados é uma forma sistemática de organizar e acessar dados
- Estruturas de dados fazem a diferença entre um algoritmo eficiente e um ineficiente, na maioria das vezes
- A escolha da estrutura de dados depende:
  - Dos dados que serão armazenados
  - Como esses dados serão manipulados
- Estruturas de dados lineares
  - Pilhas, Filas, Vetores, Listas e Sequências
- Estruturas de dados não-lineares
  - Árvores, Heaps, Hash Tables, etc…

Análise de Algoritmos

- Como analisa Algoritmos?
  - Verificar a quantidade de iterações que o algoritmo irá fazer
  - Por exemplo:
    - Remover elemento de um vetor com n elementos
      - No pior caso, leva n-1 iterações
    - Alterar o elemento numa posição r do vetor
      - Apenas 1 iteração
    - E assim por diante...
  - Muitas vezes, os algoritmos são processos repetidos de loopings (while, for, etc)

ESTRUTURA DE DADOS

Notação O e Notação Assintótica

Análise de Algoritmos

- Costuma-se analisar um algoritmo em termos de tempo e espaço (ou memória). Para o tempo, diversas medidas são em princípio possíveis:
  - Tempo absoluto (em minutos, segundos, etc.) - não é interessante por depender da máquina.
  - Número de operações consideradas relevantes (por exemplo comparações, operações aritméticas, movimento de dados, etc.)

- Três casos podem ser considerados: o melhor caso, o caso médio e o pior caso. O estudo do caso médio depende do conhecimento ou suposição da distribuição das entradas ao problema. Em geral o pior caso é o mais fácil de se analisar.

Exemplo

- Um problema tem duas soluções
  - Solução A) $F(n) = 3n^3 + 2n$ operações
  - Solução B) $F(n) = 400n + 5000$ operações

- Dependendo da entrada (n) a solução A pode ser melhor do que a solução B.

- Para estudarmos comportamento assintótico, interessa um N grande e portanto, a análise pode considerar apenas os termos de maior grau.

Notação O

- Solução A) $F(n)=O(n^3)$
- Solução B) $F(n)=O(n)$

- Dizemos que $F(n) = O(g(n))$ se existir um inteiro m e constante c tais que:

$$F(n) \leq c.g(n) \text{ para } n > m$$

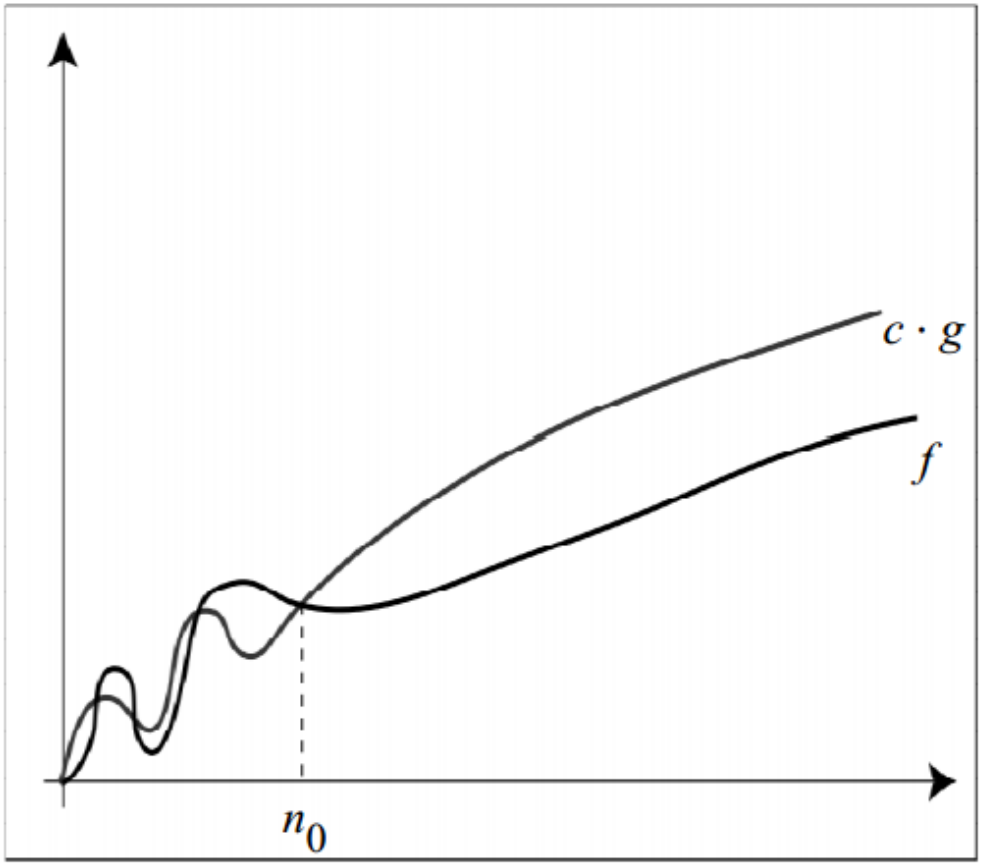
Ou seja, para n grande F(n) não cresce mais rápido que g(n)

Notação Assintótica

- Ordem de Crescimento
  - Quanto uma função f(n) cresce com o crescimento de n ?
  - Há alguma outra função (g(n)) que limite a função f(n) na parte superior e inferior ?
- Notação Assintótica
- Limite assintótico superior (Big-Oh)
  - $O(g(n))$ se $0 \leq f(n) \leq c.g(n)$ para todo $n >= n_0$
- Limite assintótico inferior
  - $\Omega (g(n))$ se $0 \leq c.g(n) \leq f(n)$ para todo $n >= n_0$
- Limite assintóticamente restrito
  - $\Theta (g(n))$ se $0 \leq c1.g(n) \leq f(n) \leq c2.g(n)$ para todo $n >= n_0$
  - $\Theta (g(n))$ se $g(n)$ for $O(g(n))$ e $\Omega (g(n))$ para $f(n)$

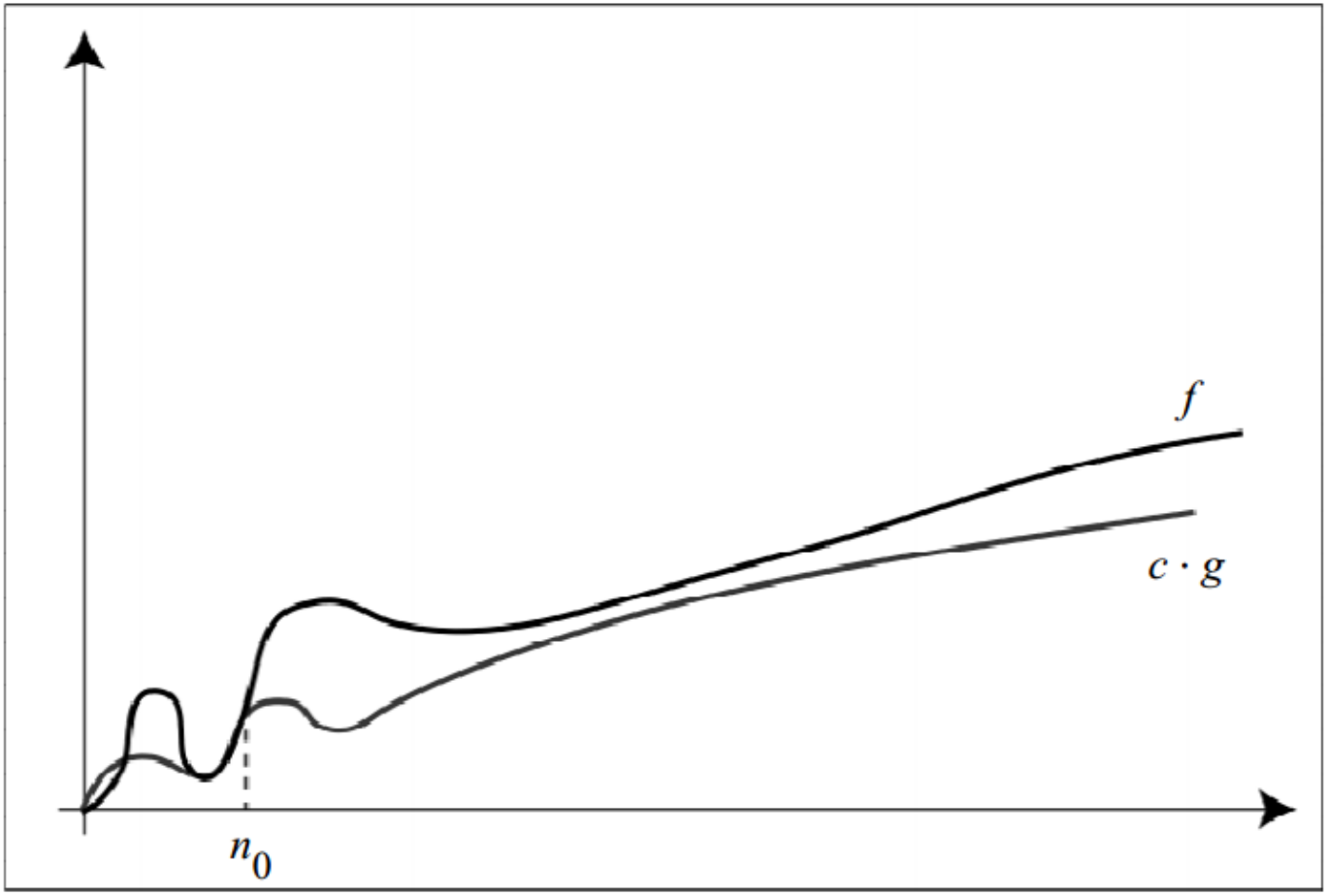
Exemplo

- f(n) = O(g(n))



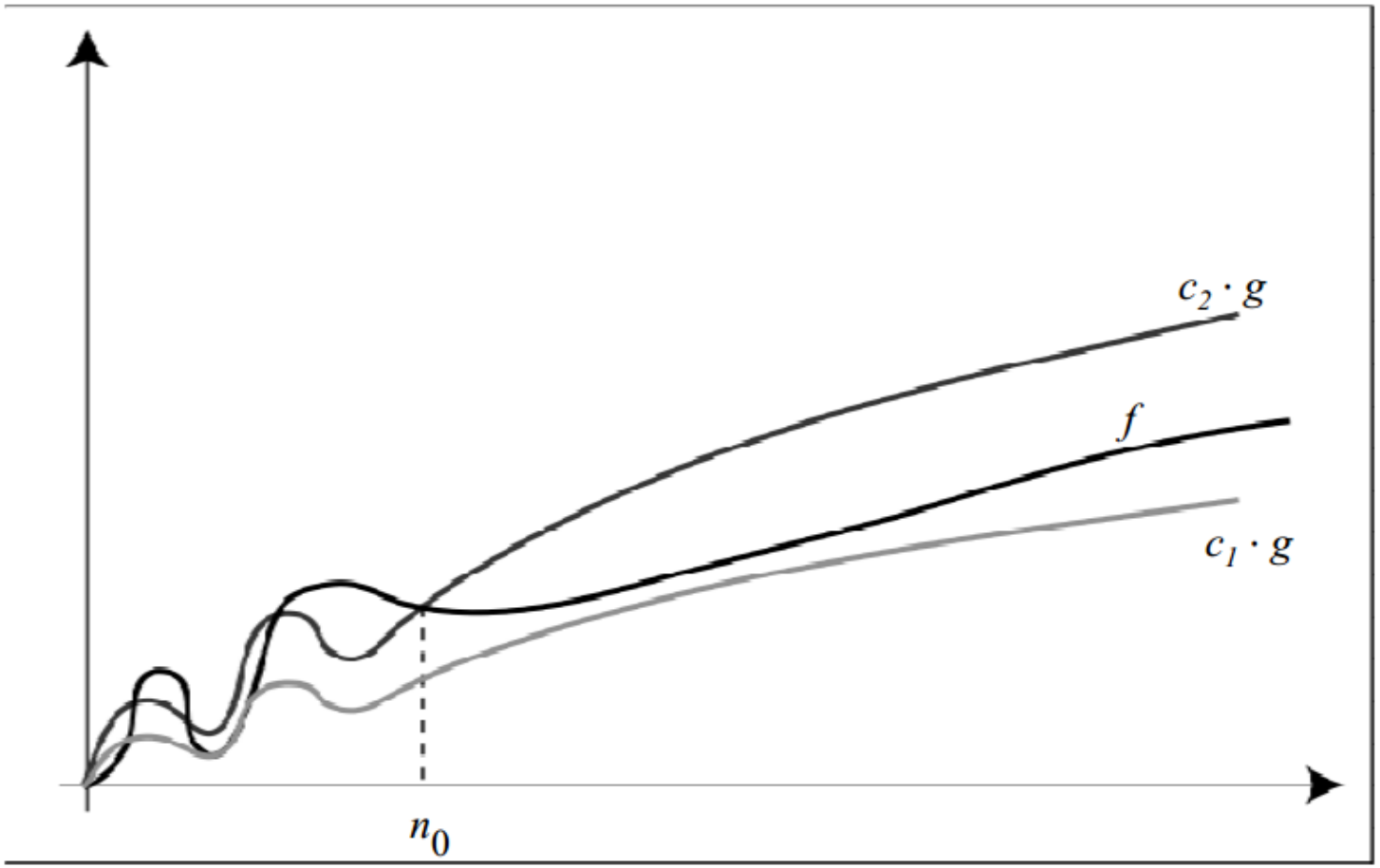
Exemplo

- $f(n) = \Omega(g(n))$



Exemplo

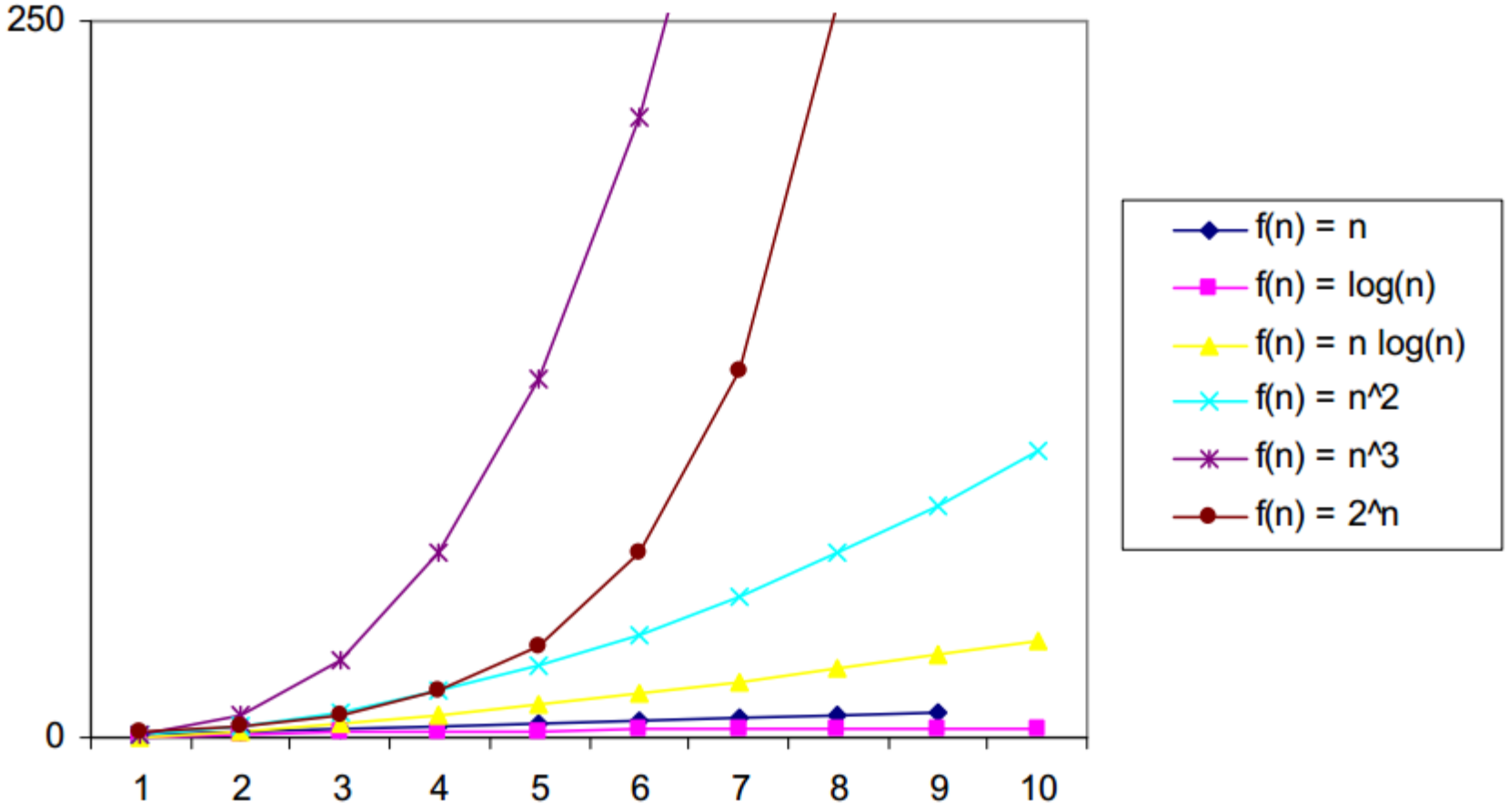
- f(n) = Θ(g(n))



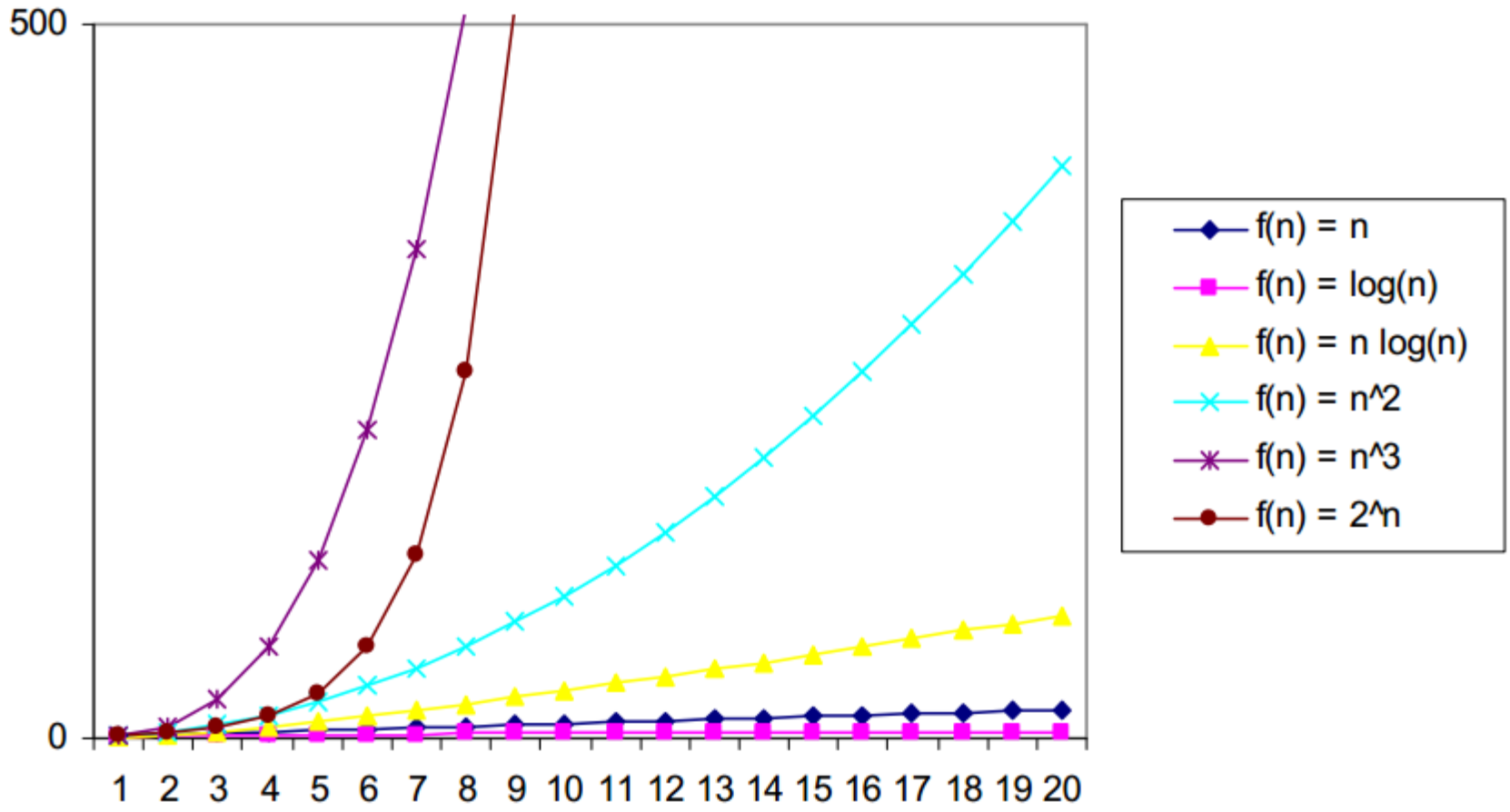
Análise Assintótica

- A análise assintótica estuda como os valores das funções convergem para o infinito conforme seus argumentos
- Ela ignora constantes e o comportamento da função para argumentos pequenos ($<n_0$)
- Isso é aceitável no contexto da análise de algoritmos, pois os algoritmos são rápidos para entradas pequenas e o crescimento do tempo de processamento é o que realmente importa.
- A análise assintótica permite, portanto, comparar eficiências de algoritmos para situações de entrada

Complexidade na Prática



Complexidade na Prática



Ordem de Complexidade

- Hierarquia das funções

$$1,\ \log_2 n,\ \sqrt[3]{n},\ \sqrt{n},\ n,\ n\log_2 n,\ n\sqrt{n},\ n^2,\ n^3,\ 2^n,\ n!,\ n^n$$

- Cada função é limite assintótico superior da anterior

- Perceba que as funções polinomiais (nx) são assintóticamente menores que as funções exponenciais (xn)…
- Isso irá importar quando determinarmos a complexidade de problemas em aulas mais adiante

Bibliografia

- CLRS (2ª. Edição)

- Capitulos
  - Capitulo 2
    - 2.1 – Ordenação por inserção
    - 2.2 – Análise de Algoritmos
  - Capitulo 3
    - 3.1 – Notação Assintótica

Referência de slides

- Aula 1 de Algoritmos: Tonindandel, Flavio

Referências dos slides

Referências dos slides

Os slides das aulas foram, em grande parte, retirados da lista de slides abaixo:

- Roberto Tamassia - Brown University - Slides of Analysis of Algorithms
  Slides names: Chapter 2 - Basic Data Strutures
  Vectors

- Larry F. Hodges - Geogia Tech - Slides of Basic Mathematics
  Slides names: Functions and Sets
  Sequences & Summations
  Doing Sums
  The Growth of Functions
  Mathmatical Inductions
  Induction Practice

- Douglas C. Szajda - University of RichMond - All Slides of CMSC 315 Class
  Available in 2005: http://oncampus.richmond.edu/~dszajda/classes/cs315/Spring_2007/lecture_notes.html

- Kevin Wayne - Univeristy of Princeton - Slides of Data Structures.
  Slides Name: Binary and Binomial Heaps
  Fibonacci Heaps

- Norbert Zeh - Dalhousie university - All Slides of CSCI 3110 - Introduction to Algorithms
  Available in 2005: http://users.cs.dal.ca/~nzeh/Teaching/Summer2004/3110/

- Fawzi Emad and Chau-Wen Tseng - University of Maryland. Slide of CMSC 132 - Maps & Hashing

- David Luekbe - University of Virginia - All Slides of CS332 - Algorithms Classes
  Available in 2005: http://www.cs.virginia.edu/~luebke/cs332.fall00/index.html