025089 – Projeto e Análise de Algoritmos

Aula 01

lizier@dc.ufscar.br

025089 – P.A.A.

- Aulas: 4h/semana (total de 60h)
- Estudos: mínimo de 4h/semana

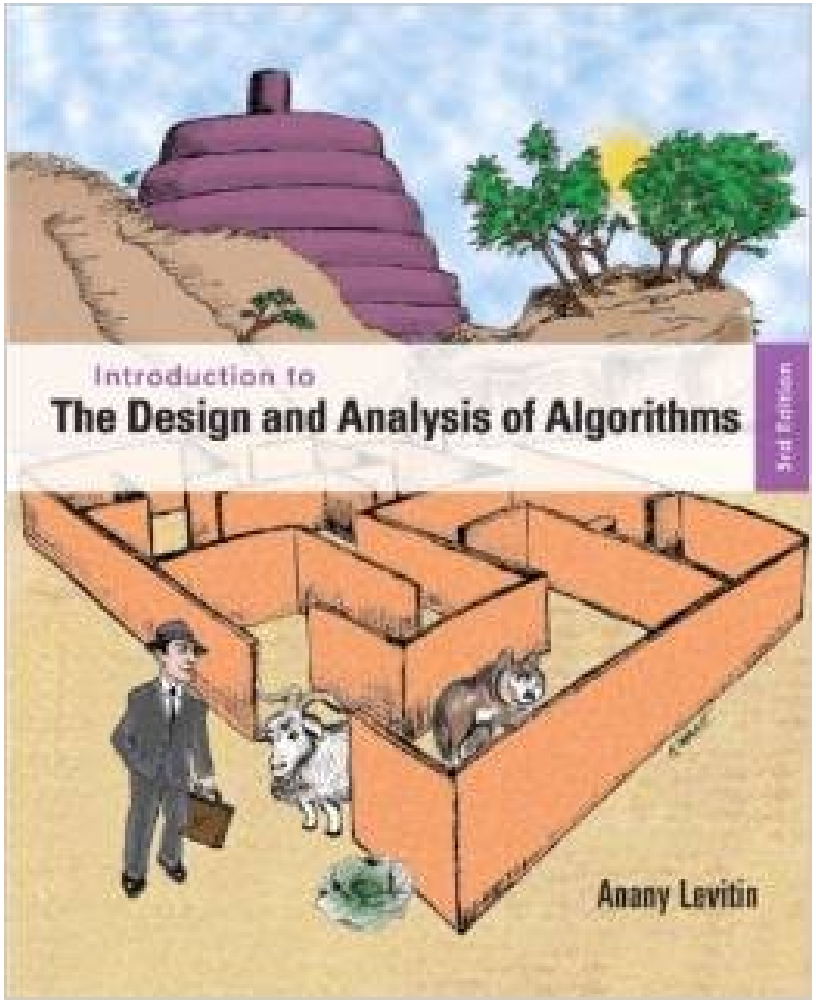
- Objetivos (Projeto Pedagógico / NEXOS):
  - CONSCIENTIZAR O ALUNO SOBRE A NECESSIDADE DE SE PROJETAR ALGORITMOS **EFICIENTES**. HABILITAR O ALUNO A REALIZAR **ANÁLISES** DE **EFICIÊNCIA** E **COMPLEXIDADE** DE ALGORITMOS. ESTUDO DE ALGORITMOS **CLÁSSICOS** PARA CERTAS CATEGORIAS DE PROBLEMAS. INTRODUZIR TÉCNICAS PARA A CONCEPÇÃO DE ALGORITMOS **EFICIENTES.**

Ementa

- **Análise de algoritmos:**
  - Fundamentos e técnicas para análise de algoritmos
  - Análise Amortizada de Algoritmos
- **Projetos de algoritmos:**
  - Força Bruta
  - Ganancioso ou Guloso
  - Redução/Transformação e Conquista
  - Divisão e Conquista
  - Programação Dinâmica
- **Algoritmos Clássicos:**
  - Aleatórios
  - Numéricos
  - Geométricos
  - Grafos
- **Limitações de Algoritmos**

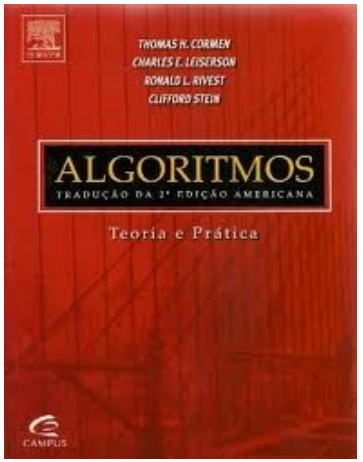
Bibliografia - Principal

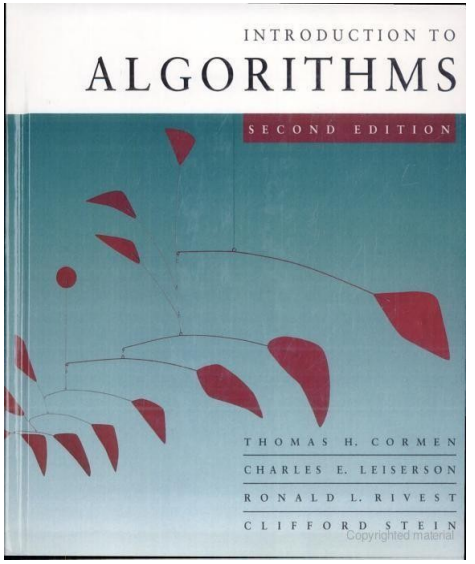
- A. LEVITIN - Introduction to The Design & Analysis of Algorithms, Pearson (2011)

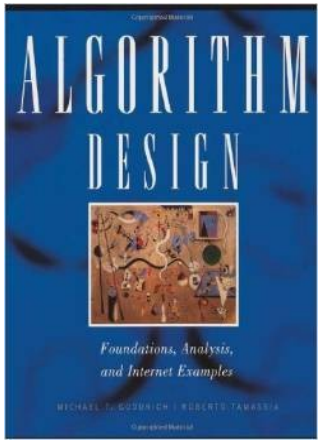


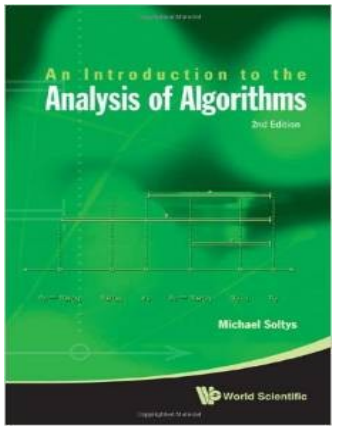
Bibliografia - outros

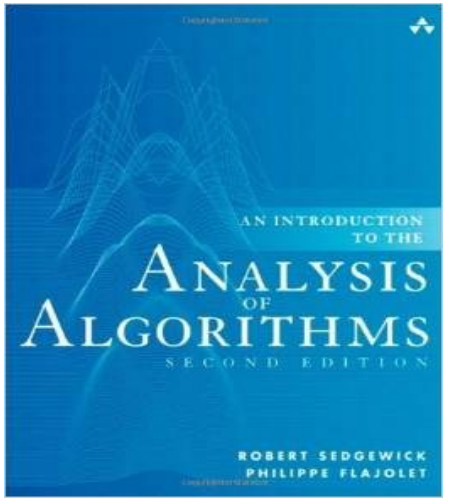
- T.H. CORMEN, C.E. LEISERSON e R.L. RIVEST - Introduction to Algorithms, The MIT Press (2009)
- M. SOLTYS - An Introduction to the Analysis of Algorithms, World Scientific (2012)
- M.T. GOODRICH e R. TAMASSIA - Algorithm Design: Foundations, Analysis, and Internet Examples, Wiley (2001)
- R. SEDGEWICK e P. FLAJOLET - An Introduction to the Analysis of Algorithms, Addison Wesley (2013)
- T.H. CORMEN, C.E. LEISERSON, R.L. RIVEST e C. STEIN - Algoritmos: Teoria e Prática, Campus (2012)
- N. ZIVIANI - Projeto de Algoritmos Com implementações em Java e C++, Thomson Learning (2007)

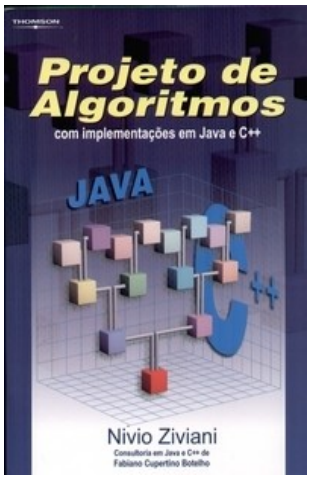










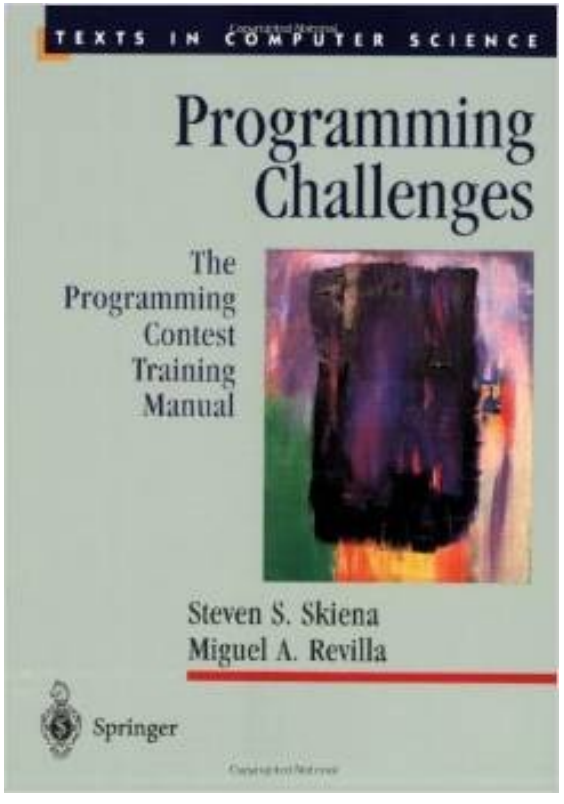


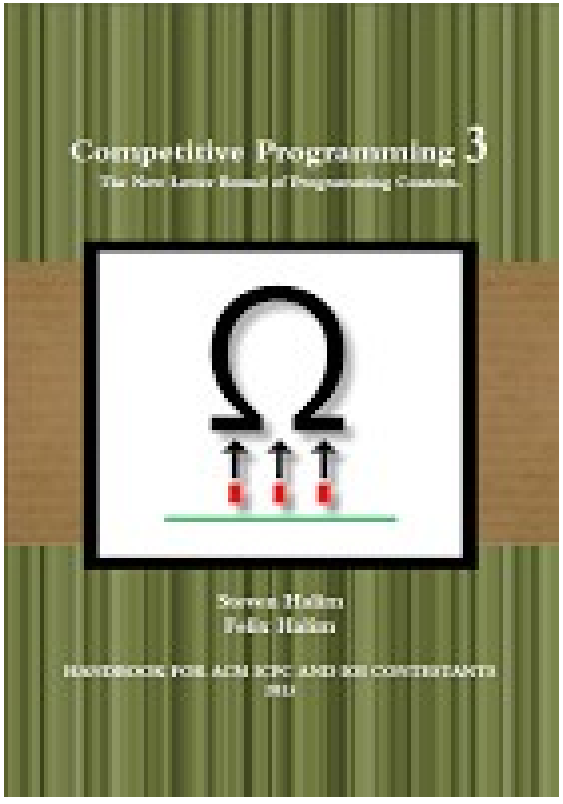
Bibliografia - Gratuita

- J. ERICKSON - Algorithms, (2013)
  - http://web.engr.illinois.edu/~jeffe/teaching/algorithms/
- P. FEOFILOFF - Minicurso de Análise de Algoritmos (2013)
  - http://www.ime.usp.br/~pf/livrinho-AA/
- S. HALIM e F. HALIM - Competitive Programming 1 (2011)
  - https://sites.google.com/site/stevenhalim/
- D. EVANS - Introduction to Computing, (2013)
  - http://www.computingbook.org/

Bibliografia
Problemas desafiadores

- S.S. SKIENA e M.A. REVILLA - Programming Challenges: The Programming Contest Training Manual, Springer (2003)
- S. HALIM e F. HALIM - Competitive Programming 3: The New Lower Bound of Programming Contests, Lulu (2013)





Disciplina

- Sala de aula:
  - 217 AT9
- Horário:
  - Turma A: Segundas e Sextas (14h-16h)
  - Turma B: Segundas e Sextas (16h-18h)
- Plantão de dúvidas:
  - Terças (13h-14h) na sala G01
- Moodle:
  - *http://moodle.dc.ufscar.br*
- Atividades:
  - Aulas + Provas + Exercícios

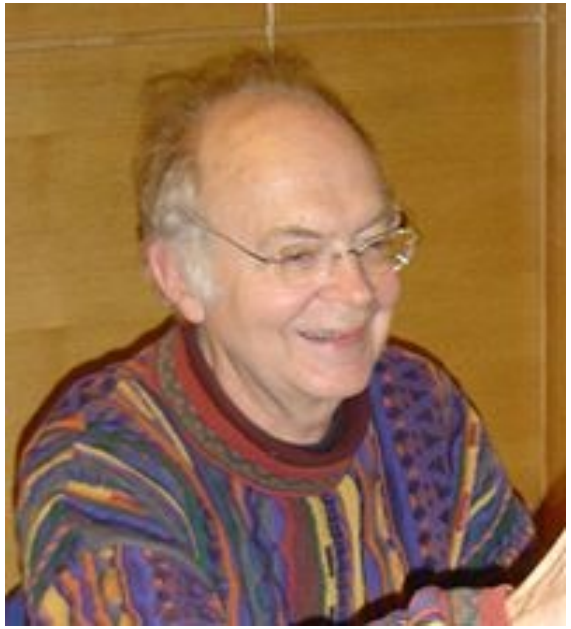
Disciplina

- Frequência:
  - Obrigatório ter pelo menos 75%
  - Apenas o DeAMO/DICA são habilitados para analisar atestados
- Provas:
  - P1: 19/09
  - P2: 31/10
  - P3: 08/12
  - Sub: 15/12
    - Apenas para quem perdeu uma das provas
- Exercícios:
  - 3 listas de exercícios (L1,L2 e L3)
  - ML = (L1 + L2 + L3)/3
- Média final:
  - MF = P1*0.3 + P2*0.3 + P3*0.3 + ML*0.1

Projeto e Análise de Algoritmos

- Porque fazer esta disciplina?
- Por que analisar um algoritmo?
- É importante para todos?
- Vamos programar?
- Vamos utilizar POO?
- ...
- Outras perguntas?
- ...

Estruturas de Dados

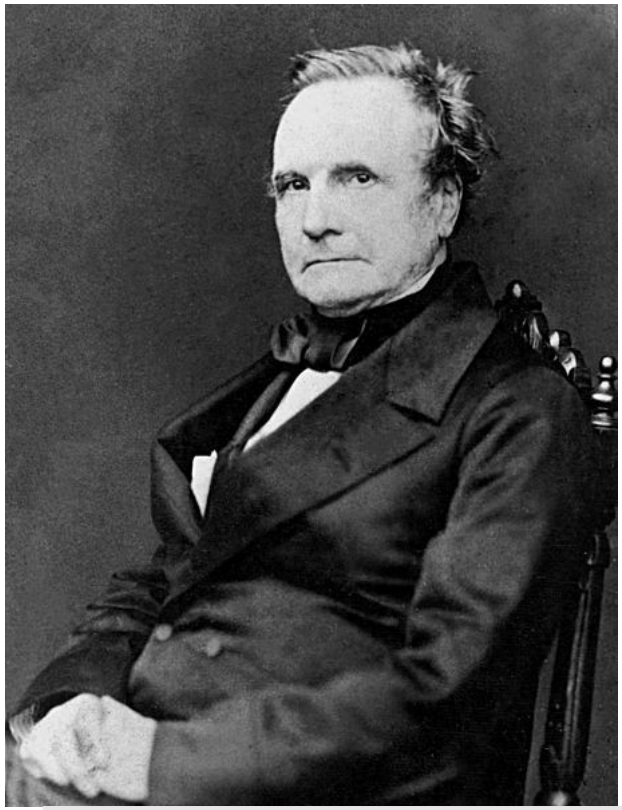


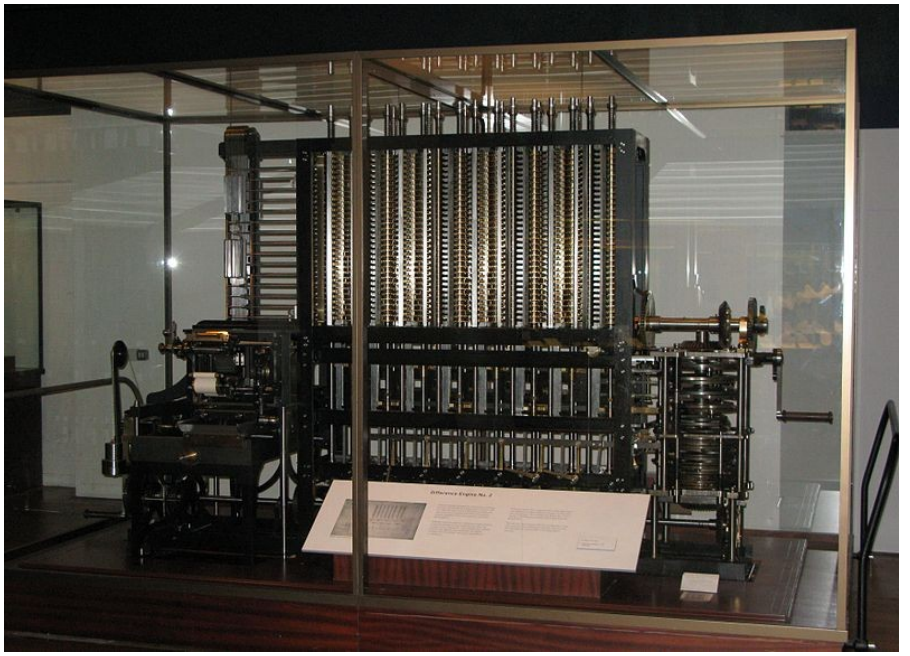
Principal desenvolvedor da análise teórica de complexidade de algoritmos.

"A person well-trained in computer science knows how to deal with algorithms: how to construct them, manipulate them, understand them, analyze them." - Donald Knuth

"It has often been said that a person does not really understand something until after teaching it to someone else. Actually, a person does not really understand something until after teaching it to a computer." - Donald Knuth

Estruturas de Dados

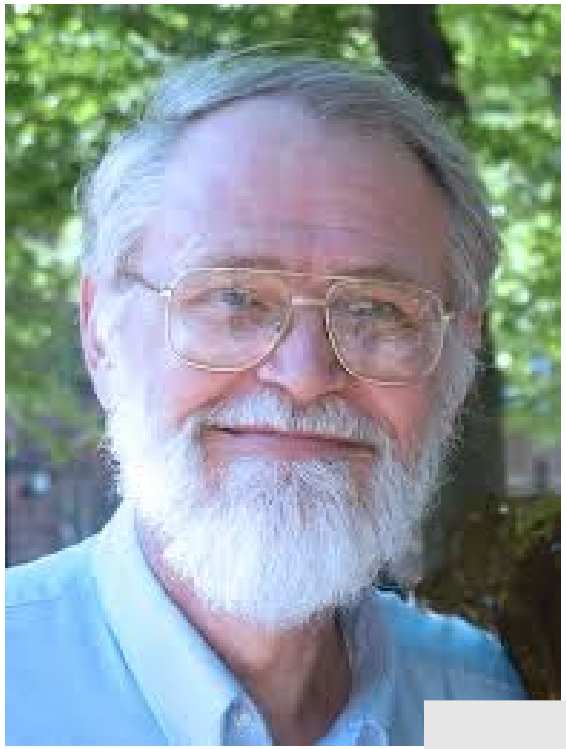




Desenvolvedor do primeiro computador mecânico (considerado um dos “pais” da computação).

"As soon as an Analytic Engine exists, it will necessarily guide the future course of the science. Whenever any result is sought by its aid, the question will arise - By what course of calculation can these results be arrived at by the machine in the shortest time?" - Charles Babbage (1864)

Estruturas de Dados



Famoso cientista da computação.
Autor de vários livros e linguagens.

"Controlling complexity is the essence of computer programming." - Brian Kernighan

PAA

- Análise:
  - Eficiência/Complexidade de tempo ou espaço
  - Adequação de recursos computacionais
  - Estudo de viabilidade de soluções
  - Comparação de algoritmos
- Projeto (design/estratégia/paradigma):
  - Abordagens para solução de problemas
  - Instanciação de problemas novos em categorias já conhecidas
  - Modelos genéricos de solução

Algoritmos

- Um bom algoritmo pode ser a diferença entre resolver ou não um problema prático
  - Um algoritmo correto ineficiente pode ser tão útil quanto um algoritmo incorreto

- Um algoritmo eficiente pode ser tão simples quanto um ineficiente
  - Não existe qualquer relação entre número de linhas, *loops* e *ifs* para determinar a complexidade de um algoritmo

Exemplo

- Busca em uma sequência ordenada:
  - Entrada: uma sequência de n números ($a_0, a_1, a_2, \ldots, a_{n-1}$) ordenados, ou seja, $a_0 \leq a_1 \leq a_2 \leq \ldots \leq a_{n-1}$, e um número ***key*** a ser procurado;
  - Saída: índice da posição de ***key*** na sequência de entrada, ou o valor ***-1*** caso ***key*** não esteja presente na sequência de entrada;
- Algoritmos propostos:
  - Busca sequencial
  - Busca binária

Algoritmo: Busca sequencial

- Ideia base: começar do primeiro elemento e ir comparando um a um até encontrar (ou encontrar um elemento maior)

```
int bsequencial( T vetor[], T key, int n )
{
    int i = 0;
    while( (i < n) && (vetor[i] < key) )
        i++;
    if ( (i < n) && (vetor[i] == key ) )
        return i;
    else
        return -1;
}
```

Busca binária

- Ideia base: comparar o elemento do meio para continuar a busca na metade certa

```
int bbinaria( T vetor[], T key, int n )
{
    int imax = n-1;
    int imin = 0;
    while( imax >= imin )
    {
        int imid = imin + ((imax - imin) / 2);
        if( key > vetor[imid] )
            imin = imid + 1;
        else if( key < vetor[imid])
            imax = imid - 1;
        else
            return imid;
    }
    return -1;
}
```

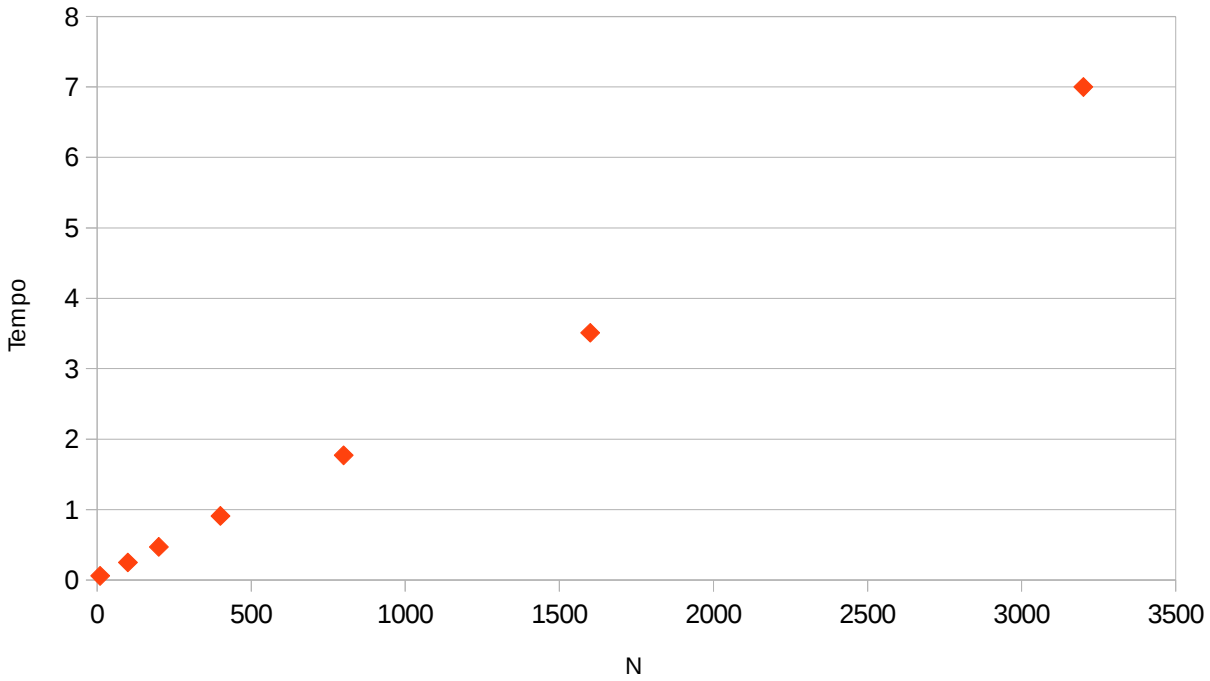
Testes

- Vamos preencher um vetor com números aleatórios, ordenar e executar os algoritmos de buscas propostos
  - Exemplos disponíveis no *moodle*.

Algumas execuções
(números aleatórios)

- Busca sequencial

N	Tempo em micro-segundos (10^{-6})
10	0.06
100	0.25
1000	2.21
10000	21.88
100000	219.79
1000000	?

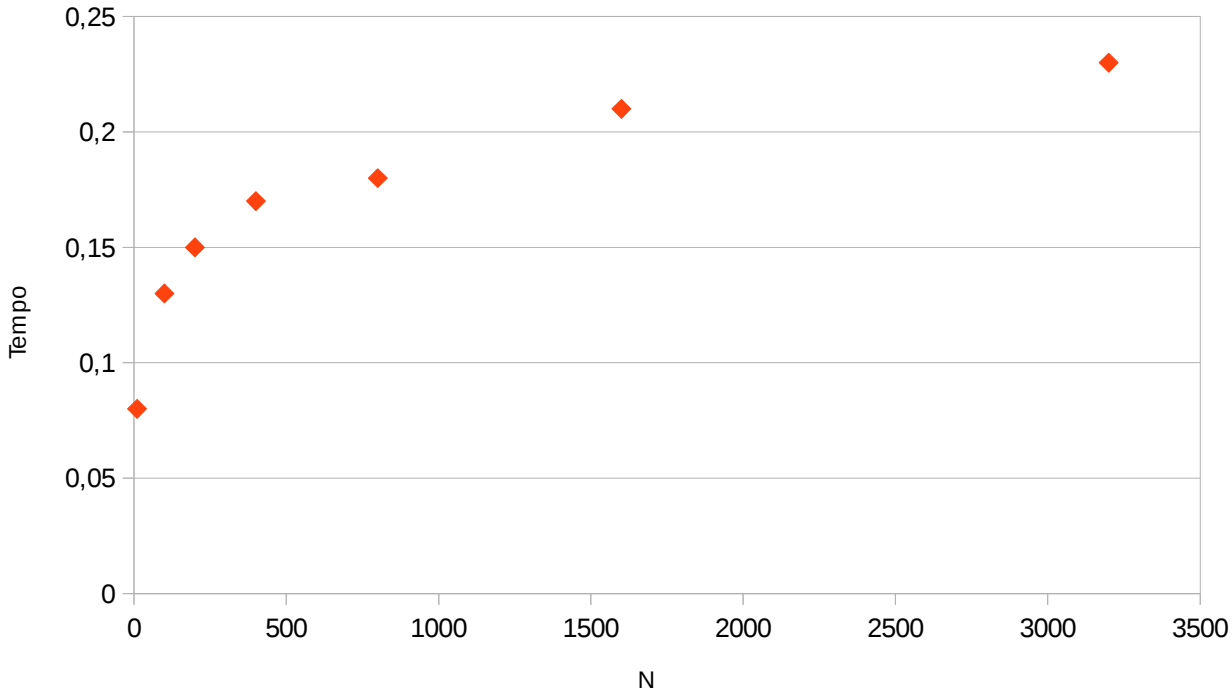


CPU: Intel Core i3-2310M – 2.1GHz

Algumas execuções (números aleatórios)

- Busca binária

N	Tempo em micro-segundos (10^{-6})
10	0.08
100	0.13
1000	0.20
10000	0.27
100000	0.35
1000000	0.55



CPU: Intel Core i3-2310M – 2.1GHz

Conclusões

- Algoritmo importa!
  - Na verdade:
    - Muito mais do que um computador mais moderno
    - Algoritmo é tecnologia
- Como profissionais em computação:
  - Conseguir resolver problemas complexos
    - Grande volume de dados –- E –- problemas complicados
  - Tornar viável a solução para problemas realmente grandes
    - “mas funciona” não é suficiente!!! mais que uma questão de desempenho, uma solução ineficiente pode ser inviável!
  - Conhecer o maior número de soluções já conhecidas
    - Maior correspondência entre uma ótima solução para um novo problema!

Practical implications of order-of-growth

growth rate	problem size solvable in minutes			
	1970s	1980s	1990s	2000s
1	any	any	any	any
log N	any	any	any	any
N	millions	tens of millions	hundreds of millions	billions
N log N	hundreds of thousands	millions	millions	hundreds of millions
N^2	hundreds	thousand	thousands	tens of thousands
N^3	hundred	hundreds	thousand	thousands
2^N	20	20s	20s	30

Bottom line. Need linear or linearithmic alg to keep pace with Moore's law.