FCT/Unesp – Presidente Prudente
Departamento de Matemática e Computação

Projeto e Análise de Algoritmos
Introdução

Prof. Danilo Medeiros Eler

danilo.eler@unesp.br

Apresentação adaptada (ver referências)

Departamento de Matemática e Computação
Faculdade de Ciências e Tecnologia
Unesp – Universidade Estadual Paulista
Presidente Prudente/SP, Brasil

unesp

DEPARTAMENTO DE MATEMÁTICA E COMPUTAÇÃO

O que é um Algoritmo?

- Procedimentos precisamente definidos para
  - obter soluções
  - resolver problemas
- Método que pode ser usado por um computador para a solução de um problema
- É uma sequência finita de instruções não ambíguas para resolver um problema

Algoritmos

- Algoritmos devem satisfazer os seguintes critérios
  - **Entrada**: uma ou mais quantidades são fornecidas externamente
  - **Saída**: ao menos uma quantidade é produzida
  - **Certeza**: cada instrução é clara e não ambígua

Algoritmos

- Algoritmos devem satisfazer os seguintes critérios
  - **Finito**: se seguirmos as instruções de um algoritmo, então, para todos os casos, o algoritmo termina após um número finito de passos
  - **Efetividade**: cada instrução deve ser simples o suficiente de modo a permitir que uma pessoa utilizando somente lápis e papel realize–a

Porque estudar Algoritmos?

- O estudo de algoritmos é a pedra fundamental da ciência da computação
  - Programas de computadores não existiriam sem algoritmos
- Devemos conhecer um conjunto de algoritmos de diferentes áreas
- Devemos ser capazes de projetar novos algoritmos e analisar suas eficiências

Metodologia para Construção de Algoritmos

- **Passo 1**: Análise do Problema
  - Compreensão
  - Entendimento das entradas e saídas

- **Passo 2**: Projeto do Algoritmo
  - Definir uma estrutura de dados
  - Otimizar tempo e espaço

Metodologia para Construção de Algoritmos

- **Passo 3**: Implementação
  - Codificar em uma linguagem de programação

- **Passo 4**: Verificação
  - Demonstrar que o algoritmo realmente resolve o problema proposto, qualquer que seja sua instância

Eficiência de Algoritmos

- Um determinado algoritmo pode ser construído de diferentes maneiras

- Algoritmos que resolvem o mesmo problema possuem a mesma eficiência?

Eficiência de Algoritmos

- Algoritmos criados para resolver um mesmo problema podem diferenciar de forma quanto a sua eficiência

n	BubbleSort Tradicional	QuickSort	HeapSort	ShellSort	InsertionSort	SelectionSort	MergeSort
100	0	0.002	0	0	0	0	0
200	0.002	0.002	0.001	0	0	0.001	0
300	0.004	0.001	0.001	0.001	0.001	0.002	0.001
400	0.007	0.002	0.001	0	0.002	0.003	0.001
500	0.01	0.003	0.001	0	0.003	0.004	0.001
600	0.015	0.003	0.002	0	0.003	0.007	0.001
700	0.02	0.002	0.001	0.001	0.005	0.009	0.002
800	0.028	0.003	0.002	0.001	0.007	0.011	0.003
900	0.033	0.002	0.002	0.002	0.01	0.015	0.003
1000	0.042	0.003	0.002	0.001	0.011	0.018	0.003
2000	0.173	0.003	0.006	0.003	0.055	0.075	0.007
3000	0.449	0.006	0.009	0.006	0.095	0.155	0.01
4000	0.739	0.007	0.013	0.008	0.167	0.271	0.014
5000	1.18	0.009	0.016	0.009	0.26	0.423	0.017
7000	2.395	0.011	0.024	0.015	0.508	0.826	0.024
8000	3.17	0.012	0.028	0.017	0.658	1.075	0.027
9000	4.058	0.014	0.032	0.019	0.836	1.359	0.032
10000	5.052	0.016	0.036	0.022	1.034	1.677	0.035
20000	21.139	0.033	0.077	0.05	4.053	6.689	0.073
25000	33.122	0.041	0.099	0.065	6.306	10.446	0.092
30000	48.01	0.05	0.121	0.079	9.175	15.037	0.114
40000	86.402	0.068	0.167	0.108	16.101	26.712	0.152

Eficiência de Algoritmos

- Algoritmos criados para resolver um mesmo problema podem diferenciar de forma drástica quanto a sua eficiência

# fibers	LSP	PLP	PLMP	LAMP	Chen
250,000	...	69.00s	0.40s	14.290s	...
4,096	102.55s	1.59s	0.016s	0.041s	1,125.0s
2,048	15.27s	0.92s	0.014s	0.014s	256.0s
1,024	1.43s	0.87s	0.011s	0.011s	67.0s
512	0.33s	0.48s	0.010s	0.006s	1.2s

Análise de Algoritmos

Análise de Algoritmos

- Analisar um algoritmo significa prever os recursos que um algoritmo necessitará
- A análise nos fornece condições de
  - Comparar diversos algoritmos para o mesmo problema
  - Determinar se o algoritmo é viável para a aplicação
- Como comparar algoritmos?
- Quais são as técnicas de análise?

Análise de Algoritmos

- Possíveis técnica de análise
  - Experimentação
  - Análise assintótica
- A análise foca principalmente no
  - Tempo de execução
  - Consumo de memória
- Porque o foco nesses dois itens?

Análise de Algoritmos

- Tempo é muito precioso

- Atualmente, poucos sistemas se preocupam com o consumo de memória, quando comparado com o passado (anos 80)
  - Preocupação com consumo de memória nos dispositivos móveis ou GPUs

Análise de Algoritmos

- O tempo de execução de um algoritmo depende de muitos fatores
  - Hardware
  - Software
  - Natureza do problema

Análise de Algoritmos

- Considerando o Hardware:
  - Processador utilizado;
  - Quantidade de processadores disponíveis;
  - Ciclos por instrução;
  - Disco;
  - Memória (Primária, Secundária);
  - Dentre outros fatores

Análise de Algoritmos

- Software
  - Sistema operacional
    - Kernel
    - Escalonamento de processos
    - Prioridade do processo
  - Da linguagem utilizada pelo programador
  - Do compilador utilizado pelo programador

Análise de Algoritmos

- **Natureza do Problema**
  - A execução de um algoritmo pode variar de acordo com a entrada de dados
  - Podemos executar o mesmo algoritmo com diferentes dados de entrada e registrar o tempo gasto por cada execução
    - Por exemplo, no caso de algoritmos de ordenação, é necessário variar a ordem com que os dados são fornecidos para o algoritmo
  - Também é necessário varia o tamanho das entradas, para verificar o comportamento do algoritmo

Exemplos de Entradas de Algoritmos

- Considerando um algoritmo de ordenação
  - A entrada é o conjunto de elementos a serem ordenados
    - Entrada

    - Saída

Exemplos de Entradas de Algoritmos

- Exemplo:

Entrada 1:	5
Entrada 2:	12
Entrada 3:	16 2
Entrada 4:	2 12 14
Entrada 5:	12 -5 16 2
Entrada 6:	-5 16 2 12 14
Entrada 7:	12 -5 16 2 12 14
Etc...	

Exemplos de Entradas de Algoritmos

- Considerando o algoritmo escalonador de processos
  - A entrada é o conjunto de processos prontos para execução

Processos em Execução

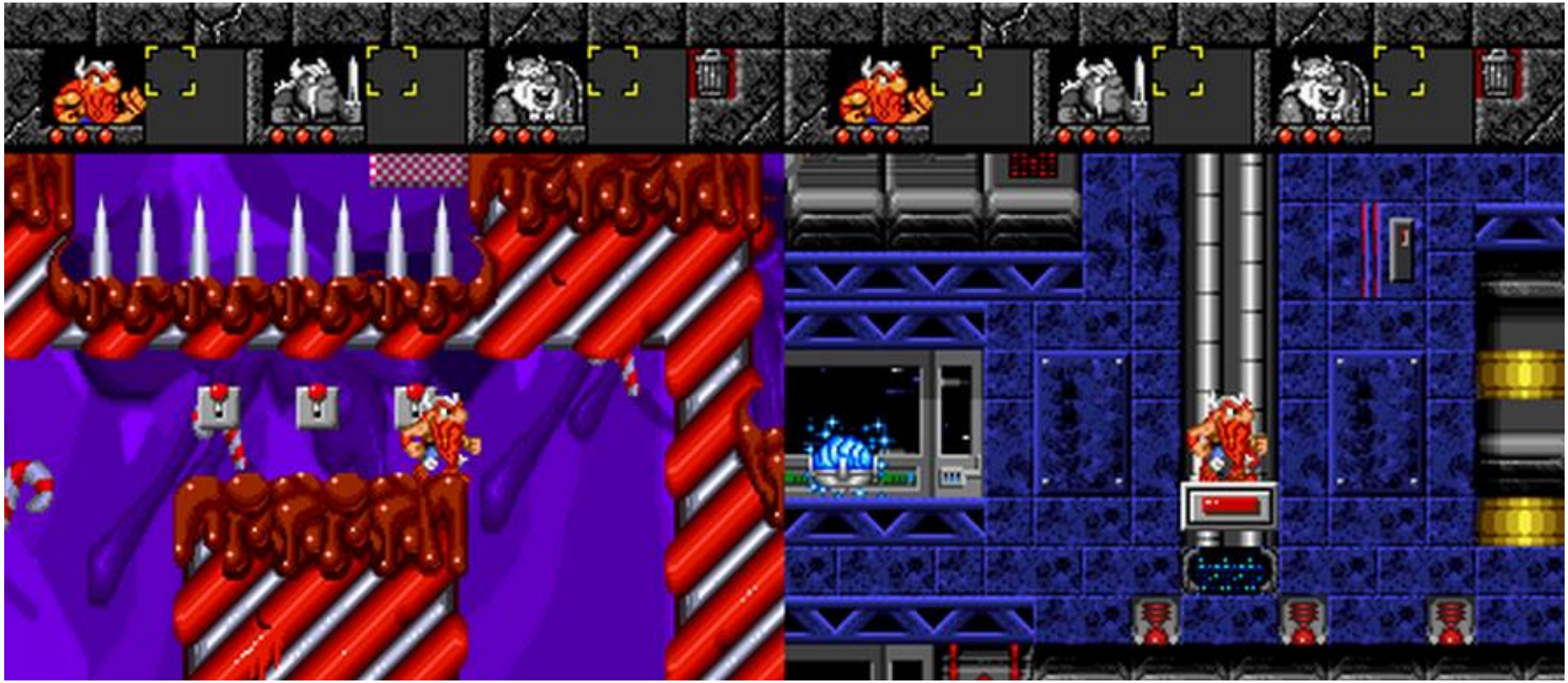
```
top - 19:18:23 up 27 days, 10:47,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 115 total,   1 running, 112 sleeping,   1 stopped,   1 zombie
Cpu(s):  0.2%us,  0.0%sy,  0.0%ni, 99.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1025648k total,   509988k used,   515660k free,   114612k buffers
Swap:  1951856k total,    40252k used,  1911604k free,   185928k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
15117 humberto  20   0  2308 1128  852 R    1  0.1   0:13.32 top
    1 root      20   0  2844 1688  544 S    0  0.2   0:01.14 init
    2 root      15  -5     0    0    0 S    0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S    0  0.0   0:00.04 migration/0
    4 root      15  -5     0    0    0 S    0  0.0   0:00.14 ksoftirqd/0
    5 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/0
    6 root      RT  -5     0    0    0 S    0  0.0   0:00.04 migration/1
    7 root      15  -5     0    0    0 S    0  0.0   0:00.08 ksoftirqd/1
    8 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/1
    9 root      15  -5     0    0    0 S    0  0.0   0:04.86 events/0
   10 root      15  -5     0    0    0 S    0  0.0   0:04.88 events/1
   11 root      15  -5     0    0    0 S    0  0.0   0:00.00 khelper
   46 root      15  -5     0    0    0 S    0  0.0   0:01.98 kblockd/0
   47 root      15  -5     0    0    0 S    0  0.0   0:00.50 kblockd/1
   50 root      15  -5     0    0    0 S    0  0.0   0:00.00 kacpid
   51 root      15  -5     0    0    0 S    0  0.0   0:00.00 kacpi_notify
  130 root      15  -5     0    0    0 S    0  0.0   0:00.00 kseriod
  170 root      20   0     0    0    0 S    0  0.0   0:00.58 pdflush
  171 root      20   0     0    0    0 S    0  0.0   0:09.00 pdflush
  172 root      15  -5     0    0    0 S    0  0.0   0:01.68 kswapd0
  213 root      15  -5     0    0    0 S    0  0.0   0:00.00 aio/0
  214 root      15  -5     0    0    0 S    0  0.0   0:00.00 aio/1
  878 www-data  20   0 26392  13m 4512 S    0  1.3   0:07.36 apache2
  923 www-data  20   0 30512  17m 4412 S    0  1.7   0:05.30 apache2
```

Exemplos de Entradas de Algoritmos

- Considerando um jogo de computador, a entrada pode ser o número de objetos em uma cena

Exemplos de Entradas de Algoritmos

- Entrada de um Jogo de Computador



Exemplos de Entradas de Algoritmos

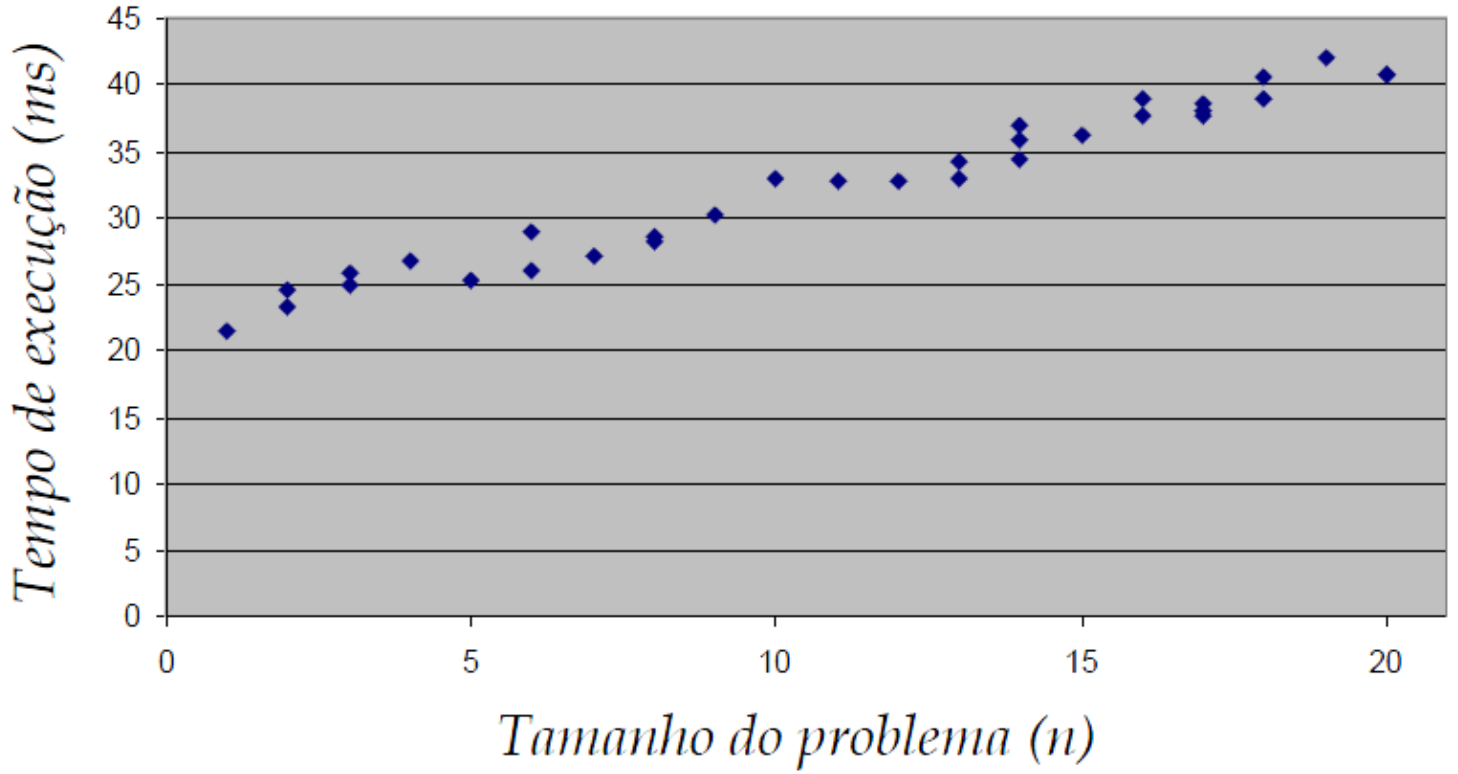
- Considerando o tamanho da entrada, então podemos medir o tempo gasto pelo algoritmo executando-o diversas vezes, com diversas entradas
  - No algoritmo de ordenação, variando o tamanho da entrada
  - No escalonador, variando o número de processos
  - Nos jogos, variando o número de objetos na cena

Análise de Algoritmos

- Na análise experimental, estamos interessados em determinar a dependência do tempo de execução com respeito ao tamanho/distribuição da entrada fornecida para o algoritmo

Análise de Algoritmos

- Podemos visualizar os resultados da execução de um algoritmo

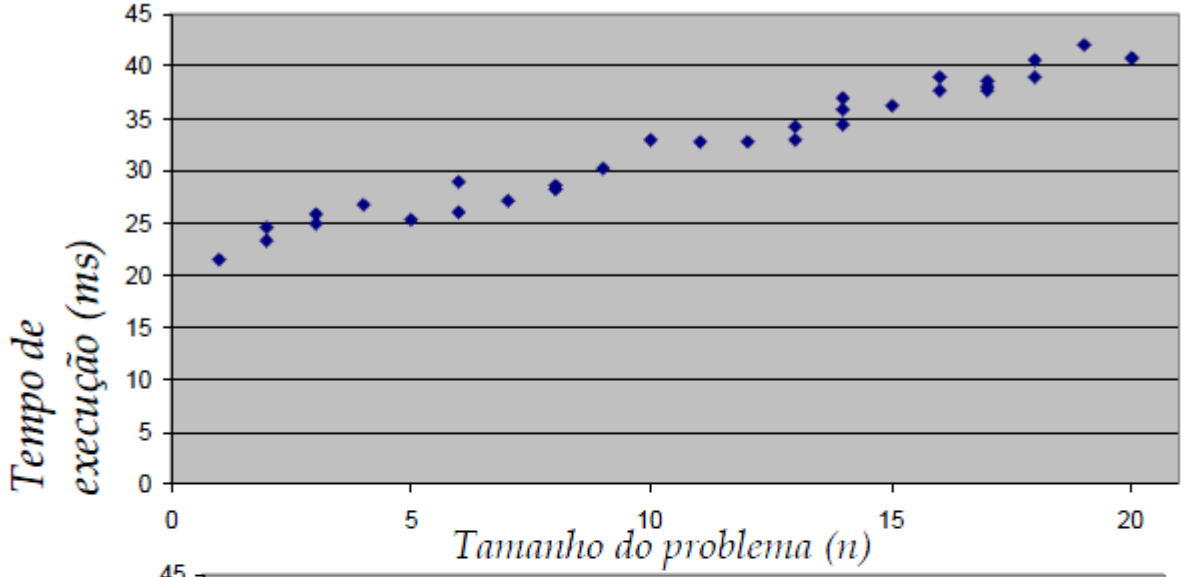


Análise de Algoritmos

Pentium 4

3.0 GHz

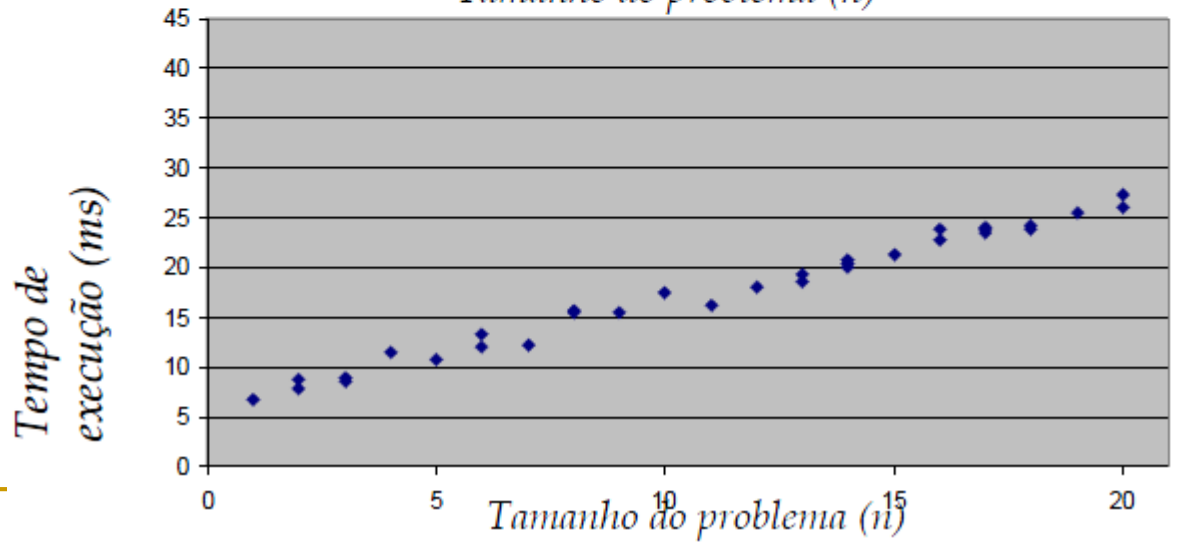
2 GB RAM



Core2Quad

2.0 GHz

3 GB RAM



Análise de Algoritmos

- Análise por tempo de execução exige
  - Um bom conjunto de entradas
  - Um número de testes suficientes deve ser executado
    - Para que possam ser feitas afirmações válidas
      - Sob um ponto de vista estatístico
    - Pois muita coisa vai influenciar em cada execução

Análise de Algoritmos

- Estudos experimentais dos tempos de execução são úteis
- Possuem algumas limitações e/ou dificuldades
- Experimentos podem ser feitos apenas em um **número limitado de entradas de teste**
  - Para a grande maioria dos problemas

Análise de Algoritmos

- Geralmente, a quantidade de entradas possível é exponencial
  - Em função do tamanho da entrada do problema
- É difícil comparar a eficiência de dois algoritmos
  - Os experimentos para a obtenção de seus tempos de execução devem ser feitos com o mesmo hardware e software
  - Além disso, o S.O. é dinâmico

Análise de Algoritmos

- Exemplo
  - Algoritmo *Triplo X* executou três vezes mais lento que o algoritmo *X*, considerando uma entrada de 1000 elementos
  - Qual dos dois algoritmo você prefere utilizar?

Análise de Algoritmos

- Exemplo
  - Algoritmo *Triplo X* executou três vezes mais lento que o algoritmo *X*, considerando uma entrada de 1000 elementos
  - Qual dos dois algoritmo você prefere utilizar?

- Qual seria o comportamento dos algoritmos para entradas maiores?
- Em alguns casos, a experimentação sozinha não é suficiente para analisar algoritmos

Análise de Algoritmos

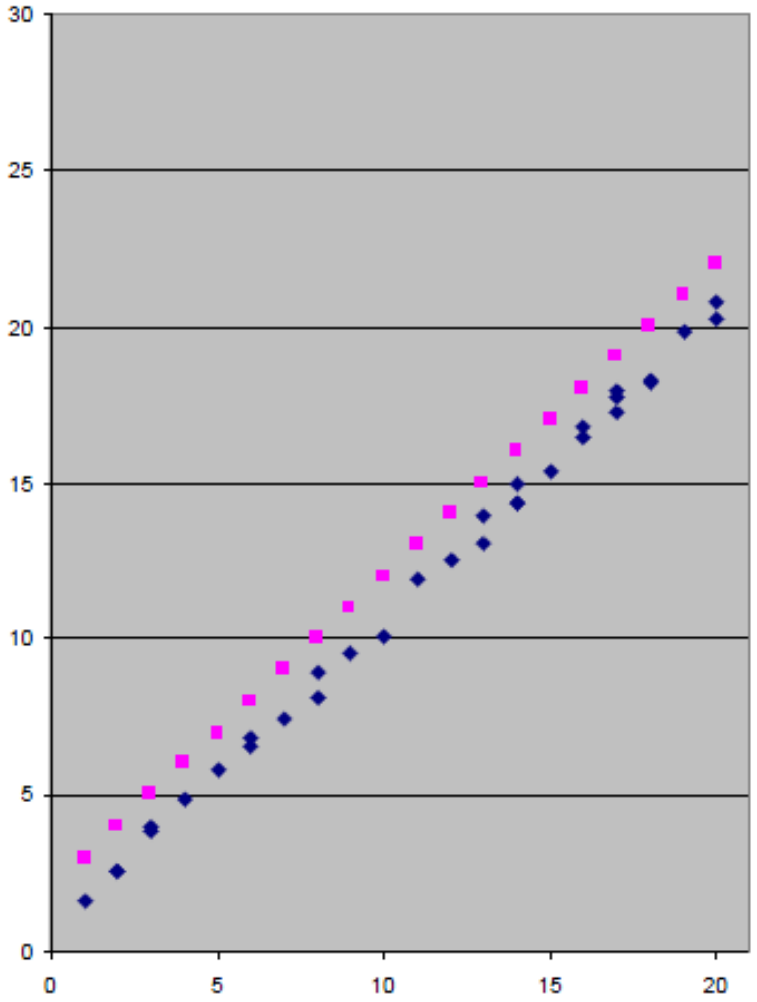
- Portanto, desejamos uma metodologia analítica que
  - Leve em conta **TODAS** as entradas possíveis
  - Permita-nos avaliar a eficiência relativa de dois algoritmos
    - Independente do hardware e do software
  - Possa ser feita através do estudo de uma descrição em alto nível do algoritmo
    - Sem que seja necessário implementá-lo ou executá-lo

Análise de Algoritmos

- Esta metodologia simplificada **associa uma função *f(n)* a cada algoritmo**
  - Descreve o crescimento do tempo de execução ***f*** em função do tamanho da entrada ***n***
- Exemplo: A função pode ser um polinômio de primeiro grau
  - *f(n) = c1*n + c2*
  - O que pode determinar na prática os valores de *c1* e *c2*?

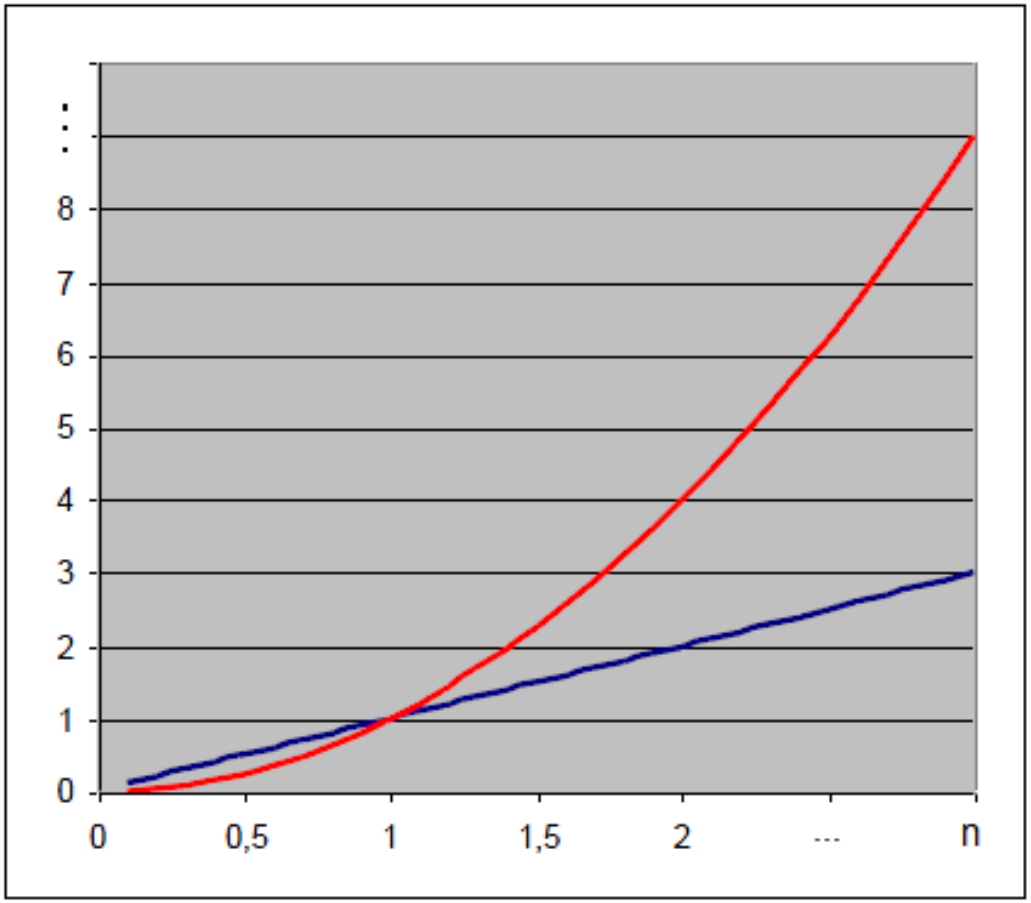
Análise de Algoritmos

- Por exemplo, o algoritmo A é executado em tempo proporcional a **n**
  - f(n) = c1*n + c2
- Variando as constantes teríamos diferentes tempos, mas o comportamento é similar



Análise de Algoritmos

- Dados dois algoritmos e suas funções de tempo de execução
  - A (azul)
  - V (vermelho)
- Qual algoritmo você prefere utilizar?



Referências Bibliográficas

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; (2002). Algoritmos –Teoria e Prática. Tradução da 2ª edição americana. Rio de Janeiro. Editora Campus
- TAMASSIA, ROBERTO; GOODRICH, MICHAEL T. (2004). Projeto de Algoritmos -Fundamentos, Análise e Exemplos da Internet
- ZIVIANI, N. (2007). Projeto e Algoritmos com implementações em Java e C++. São Paulo. Editora Thomson

Referências de Material

- Adaptado do material de
  - Professor Alessandro L. Koerich da *Pontifícia Universidade Católica do Paraná (PUCPR)*
  - Professor Humberto Brandão da *Universidade Federal de Alfenas (Unifal-MG)*
  - Professor Ricardo Linden da *Faculdade Salesiana Maria Auxiliadora (FSMA)*
  - Professor Antonio Alfredo Ferreira Loureiro da *Universidade Federal de Minas Gerais (UFMG)*