Teoria

Análise Experimental de Algoritmos

Análise Experimental de Algoritmos

Algoritmos:

- Um algoritmo corresponde a uma descrição de um padrão de comportamento, expresso em termos de um conjunto finito de ações (Dijkstra, 1971 citado por Ziviani, 2004);

- Ao executarmos a operação $a + b$ percebemos um padrão de comportamento, mesmo para valores diferentes de a e b;

- Segundo Cormen *et al*. (2002): informalmente, um algoritmo é um procedimento computacional bem definido que toma um conjunto de valores como ***entrada*** e produz algum conjunto de valores como ***saída***.

Análise Experimental de Algoritmos

Quais as características importantes em um algoritmo:

- Performance: Um algoritmo deve ter desempenho, ponto chave de qualquer algoritmo/*software*;

- Simplicidade: Um algoritmo simples é mais fácil de ser implementado corretamente e, por consequência, há menor probabilidade de obter erros;

- Clareza: Um algoritmo deve ser escrito de forma clara e documentada para facilitar sua manutenção;

- Segurança: Um algoritmo deve ser seguro (*safety*/*security*);

Análise Experimental de Algoritmos

Quais as características importantes em um algoritmo:

- Funcionalidade: Um algoritmo deve possuir diversas funcionalidades;
- Modularidade: Permite ao algoritmo uma melhor manutenção, reuso etc.;
- Interface Amigável: Característica fundamental para a maior parte dos usuários;

Análise Experimental de Algoritmos

Quais as características importantes em um algoritmo:

- Corretude: Diz respeito à determinada especificação; Corretude Funcional se refere ao comportamento de entrada/saída do algoritmo (cada entrada produz uma saída correta);

  Segundo Cormen *et al.* (2002), um algoritmo é dito correto se, para cada instância de entrada, ele para com a saída correta (ou informa que não há solução para aquela entrada);

  Um algoritmo correto sempre termina? E ser for oferecido um algoritmo correto que permite obter uma solução, por exemplo em 3 anos?

Análise Experimental de Algoritmos

Eficiência:

- Os computadores são “muito rápidos” atualmente;
- Porém, sempre, os problemas crescem mais rápido do que a velocidade do computador;
- É muito importante levar em consideração a eficiência de um algoritmo ao desenvolver um *software*:
  - Alguns programas executam instantaneamente;
  - Alguns programas executam de um dia para o outro;
  - Alguns programas poder executar por dias, meses, anos etc.

Análise Experimental de Algoritmos

Um problema e muitas estratégias e soluções:

- Tendo um mapa e a meta de determinar a menor rota de um local a outro:
  - O número de rotas pode ser enorme;
  - Diversas estratégias podem ser utilizadas para obter a menor rota:

- Como garantir uma boa rota?
  - Garantindo a corretude do algoritmo!

Análise Experimental de Algoritmos

Eficiência:

- Uma das perguntas comuns em entrevista de emprego no Google:

  "Qual a maneira mais eficiente de ordenar um milhão de inteiros de 32 bits"

- É importante conseguir relacionar classes de problemas e algoritmos com a eficiência com base no tempo de execução;

Análise Experimental de Algoritmos

Eficiência:

- Para conhecer a resposta é preciso:
  - Definir “eficiente”;
  - Definir metodologia padronizada para medir eficiência;
  - Comparar a eficiência entre os algoritmos.

- A eficiência pode ser associada aos recursos computacionais:
  - A quantidade de espaço de armazenamento que utiliza;
  - A quantidade de tráfego que gera em uma rede de computadores;
  - A quantidade de dados que precisam ser movidos da memória não volátil para a memória não volátil;

Análise Experimental de Algoritmos

Eficiência:

- No entanto, para a maior parte dos problemas, a eficiência está relacionada ao tempo de execução em função do tamanho da entrada a ser processada;

- O objetivo sempre será ser capaz de, dado um problema, mapeá-lo em uma classe de algoritmos que o solucionam e encontrar a melhor escolha entre eles, com base na eficiência de cada um.

Análise Experimental de Algoritmos

Eficiência: Análise Experimental

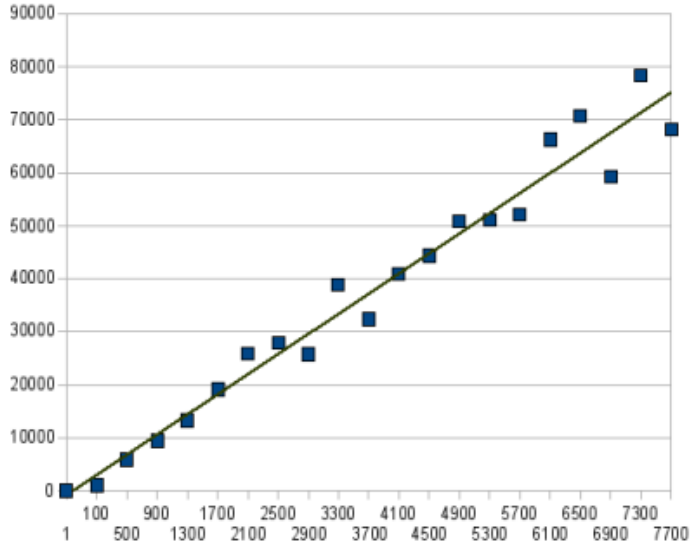
- Computar o tempo total de execução de um programa, por exemplo:

```
#include <stdio.h>
#include <time.h>

int main (void) {
  time_t t1, t2, total;

  t1 = time(NULL); // retorna hora atual do sistema

  /* algoritmo */

  t2 = time(NULL);
  total = difftime(t2,t1); // retorna a diferenca t2-t1
  printf("\n\nTotal: %ld seg.\n", total);
  return 0;
 }
```

Análise Experimental de Algoritmos

Gráfico de Análise Experimental

- Realizando-se diversos experimentos e computando o tempo para entradas diferentes, pode-se traçar um gráfico como o abaixo, onde, no eixo *x*, está o tamanho da entrada e, no eixo *y*, está o tempo, em milissegundos (*ms*) que o algoritmo demora para processar a entrada.



Análise Experimental de Algoritmos

Eficiência: Análise Experimental

- Desvantagens e limitações:
  - É preciso implementar e testar o algoritmo com diversas entradas diferentes;
  - A análise será feita por um conjunto limitado de dados;
  - O tempo dependerá de diversos fatores, como *hardware*, sistema operacional, linguagem de programação, compilador, sistema computacional etc.

Análise Experimental de Algoritmos

Eficiência: Análise Experimental

- Analise a afirmação abaixo, por exemplo:

"Desenvolvi um novo algoritmo chamado *TripleX* que leva 14,2 segundos para processar 1.000 números, enquanto o método *SimpleX* leva 42,1 segundos"

- Você trocaria o *SimpleX*, que já roda na sua empresa, pelo *TripleX* ?

Para responder, há vários fatores envolvidos, por exemplo, o *TripleX* também seria mais rápido para processar quantidades maiores que 1.000 números?

Análise Experimental de Algoritmos

Eficiência: Passos Básicos e Tamanho da Entrada

- Abordagem:

  O número de passos básicos necessários, em função do tamanho da entrada que o algoritmo recebe:

  - Descorrelaciona a performance da máquina na performance do algoritmo;
  - Reduz a análise ao desempenho, em função do tamanho da entrada.

Análise Experimental de Algoritmos

Eficiência: Passos Básicos e Tamanho da Entrada

- Tamanho da Entrada:

  Depende do problema, mas geralmente é relativo ao número de elementos da entrada, que são processados pelo algoritmo:

  - O número de elementos em um arranjo, lista, árvore etc.;
  - O tamanho de um inteiro, que é passado por parâmetro.

Análise Experimental de Algoritmos

Eficiência: Passos Básicos e Tamanho da Entrada

- Passos Básicos:

  Referem-se às operações primitivas utilizadas pela máquina:

  - Operações aritméticas;
  - Comparações;
  - Chamadas às funções;
  - Retorno das funções;
  - Etc.

Análise Experimental de Algoritmos

Eficiência: Passos Básicos e Tamanho da Entrada

- No exemplo apresentado, assumindo-se:
  - O tamanho da entrada como sendo n;
  - Cada operação leva aproximadamente o mesmo tempo constante;
  - *TripleX*: para uma entrada de tamanho n, o algoritmo realiza $n^2 + n$ operações; Em função, $t(n) = n^2 + n$
  - *SimpleX*: para uma entrada de tamanho n, o algoritmo realiza $2000n$ operações; Em função, $s(n) = 2000 \, . \, n$

Análise Experimental de Algoritmos

Eficiência: Passos Básicos e Tamanho da Entrada

- No exemplo apresentado, assumindo-se:
  - Calculando o número de operações, em função da entrada:

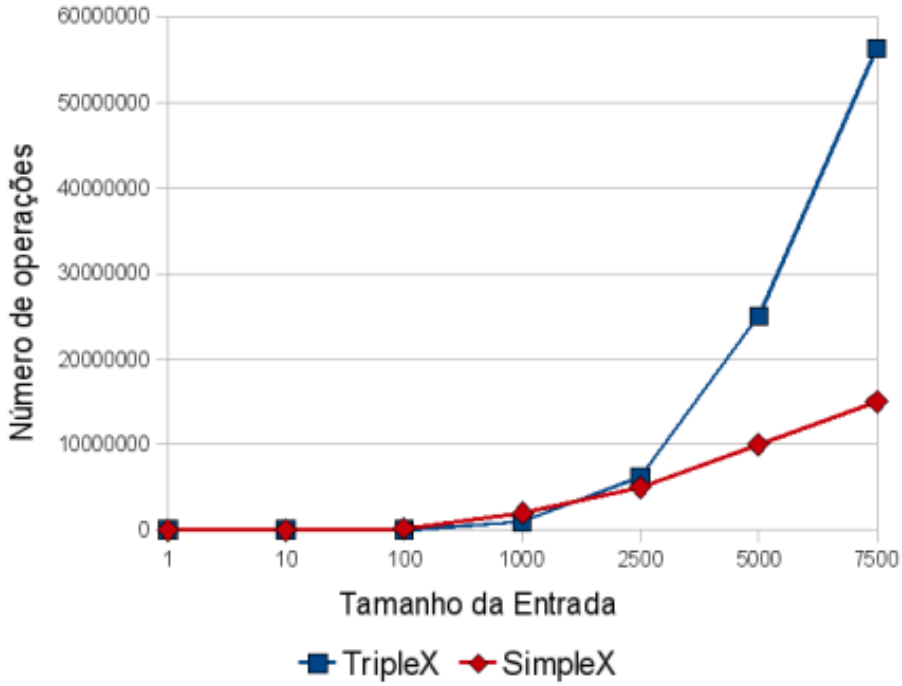
n	1	10	100	1.000	10.000
$t(n) = n^2 + n$					
$s(n) = 2000 \cdot n$					

Análise Experimental de Algoritmos

Eficiência: Passos Básicos e Tamanho da Entrada

- No exemplo apresentado, assumindo-se:
  - Calculando o número de operações, em função da entrada:

n	1	10	100	1.000	10.000
$t(n) = n^2 + n$	2	110	10.100	1.001.000	**100.010.000**
$s(n) = 2000 \cdot n$	2.000	20.000	20.000	2.000.000	20.000.000



Bibliografia

- CORMEN, T.H. et al. **Algoritmos: Teoria e Prática** (Caps. 1–3). 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`

Teoria

FIM