INSTITUTO FEDERAL DE GOIÁS
CÂMPUS GOIÂNIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
TEORIA GERAL DA ADMINISTRAÇÃO

Nome do Aluno: Luiz Antônio Rodrigues dos Santos
Data: 02/04/2018
Prof. Renan Rodrigues

Análise Empírica

1. O custo de um algoritmo pode ser analisado de forma empírica, onde o algoritmo é executado em um computador real, sendo o tempo de execução medido diretamente. As medidas de tempo obtidas desta forma podem ser inadequadas e os resultados não devem ser generalizados. Cite no mínimo três vantagens e três desvantagens deste tipo de análise de algoritmos.

Vantagens	Desvantagens
- Avaliar o desempenho em uma determinada configuração de computador/linguagem; - Considera custos não aparentes (ex: o custo de alocação de memória); - Comparar computadores; - Comparar linguagens; - Pode ser feita dentro do sistema onde o algoritmo se encontra; - Permite ter uma "noção" da ordem de complexidade; - Permite "otimização de constantes", ou seleção de um algoritmo mais adequado por meio de comparações sem análise matemática;	- Necessidade de implementar o algoritmo, que depende da habilidade do programador; - Resultado pode ser mascarado pelo hardware ou software e eventos ocorridos no momento da avaliação, depende das características específicas do ambiente de teste - Qual a natureza dos dados, dados reais, aleatórios que avaliam desempenho médio e o pior caso - Não permite comparações genéricas com algoritmos cujos testes foram realizados em outro ambiente

2. Considere dois programas A e B com tempos de execução $1.000n$ e $5n^2$, respectivamente, qual é o mais eficiente?

Fazendo uma análise empírica constatei que enquanto n for menor que 200 o programa mais eficiente é o B, quando n for igual a 200 os dois programas serão equivalentes e quando n for maior que 200 o programa A é o mais eficiente.

3. Sejam dois computadores:
• C1 que executa 10^7 instruções por segundo (10 milhões);
• C2 que executa 10^9 instruções por segundo (1 bilhão);
Considere o seguinte algoritmo de ordenação:
• A: linguagem de máquina cujo código exige $2n^2$ instruções para ordenar n números.
Quanto tempo C1 e C2 gastam para ordenar um milhão de números usando o algoritmo A?

C1 irá gastar 200 mil segundos - $2(1.000.000)^2 / 10^7$
C2 irá gastar 2 mil segundos - $2(1.000.000)^2 / 10^9$

4. Considerando a implementação parcial do programa em C disponível no Moodle, faça uma análise empírica dos seguintes algoritmos de ordenação: Bubble Sort, Quicksort e Insertion Sort. O código fonte já disponibiliza a implementação dos dois primeiros algoritmos e uma estratégia para contar a
quantidade de trocas para realizar a ordenação e a determinação do tempo de execução dos algoritmos. Pesquise na Internet uma implementação do algoritmo Insertion Sort (não é necessário fazer sua própria implementação) e realize as alterações necessárias para realizar o experimento.

Plano para a Análise Empírica de Algoritmos

a) Descreva a configuração da máquina onde o experimento foi realizado.

Placa-mãe Asus P8H61-M LX3 R2.0 Intel(R) Core(TM) i5-2310 CPU @ 2.90GHz Clock 100MHz Memória Ram ddr3 16GB 1.333 MHz (0,8 ns) Cache size 6144 KB Sistema Operacional Ubuntu 16.04 x64

b) Descreva o objetivo do experimento.

Avaliar o desempenho dos algoritmos Bubble Sort, QuickSort e InsertSort.

c) Descreva a métrica de eficiência a ser medida e a unidade de medida.

Tempo que cada algoritmo demora para executar; Número de trocas de posições no vetor;

d) Descreva como a amostra das entradas foram geradas.

Amostra de entradas foram geradas por um algoritimo de geração de números inteiros aleatórios.

e) Breve descrição das características de cada algoritmo.

Algoritmo de Ordenação	Características do Algoritmo
Bubble Sort	Percorre o vetor comparando pares de elementos adjacentes e os troca de lugar se estiverem na ordem errada, o processo se repete até que nenhuma troca seja necessária, ou seja, quando todos os elementos estão ordenados.
Quicksort	Um elemento é escolhido como pivô, os dados são particionados e rearranjados com valores menores do que o pivô colocados antes dele e os maiores depois, recursivamente ordena as duas partições.
Insertion Sort	Ordena um elemento de um vetor colocando-o em seu devido lugar em ordem, ou seja recebe um elemento e já o coloca em seu devido lugar.

f) Anotações do Experimento.

Algoritmo de Ordenação	Tamanho da Entrada							
	1.000		10.000		100.000		1.000.000	
	Tempo (ms)	Qtde. de Trocas	Tempo (ms)	Qtde. de Trocas	Tempo (ms)	Qtde. de Trocas	Tempo (ms)	Qtde. de Trocas
Bubble Sort	2.242	241.946	289.554	24.906.220	31436.7	2.480.580.091	3.19923e+06	247.489.647.661
Insertion Sort	1.082	241.946	107.144	24.906.220	10268.1	2.480.580.091	1.02826e+06	247.489.647.661
Quick Sort	0.106	3.511	0.835	52.253	9.365	692.721	93.183	8.626.930

g) Análise dos Dados Obtidos.

De acordo com os resultados obtidos no experimento para análise empírica, constatei que para os algoritmos usados o Quick Sorte é o mais eficiente, tanto para entradas pequenas quanto entradas muito grandes, devido a executar sempre em um menor espaço de tempo e com menor quantidade de trocas, constatei que o crescimento do algoritmo é sempre linear tanto para trocas quanto para o tempo de execução. Os algoritmos Bubble Sort e Insertion Sort o crescimento é sempre exponencial tanto para tempo de execução quanto para trocas, apesar do tipo de ordenação dos dois algoritmos acontecer de modo diferente tiveram o mesmo número de trocas, mas o algoritmo Insert Sort teve melhor desempenho no tempo de execução comparado ao Bubble Sort, analisando os resultados constatei que o tempo de execução do algoritmo Bubble Sort cresce exponencialmente em relação ao Insertion Sort.