Universidade Federal de Itajubá
Instituto de Matemática e Computação
Algoritmos e Estruturas de Dados II
COM-112
Vanessa Souza

1º Trabalho Prático

Turma 2 : SISTEMAS DE INFORMAÇÃO

Assunto : Ordenação
Valor : 20 pontos

7 GRUPOS DE 5 PESSOAS

Informações Gerais

Este trabalho consiste em analisar o desempenho de diferentes algoritmos de ordenação e foi inspirado no trabalho da professora Jussara Almeida, do DCC da Universidade Federal de Minas Gerais. Esta análise consistirá em comparar os algoritmos considerando três métricas de desempenho : número de comparações de chaves, número de cópias de registros realizadas e o tempo total gasto para ordenação (tempo de processamento e não o tempo de relógio). As entradas deverão ser conjuntos de elementos com chaves aleatoriamente geradas.

Os grupos deverão comparar a eficiência do algoritmo para ordenar vetores gerados aleatoriamente e para ordenar vetores ordenados (pior caso), com as seguintes variações de tamanho do vetor N = 50, 1000, 50000 e 500000 ou mais elementos. Para cada valor de N, realize experimentos com 5 sementes diferentes e avalie os valores médios do tempo de execução, do número de comparações de chaves e do número de cópias de registros.

O programa principal deve ser organizado da seguinte maneira:

- Recebe a semente do gerador de números aleatórios e os nomes dos arquivos de entrada e saída. Estes parâmetros devem ser passados pela linha de comando (argc e argv em C). Por exemplo:

quicksort 10 entrada.txt saida10.txt,

onde entrada.txt tem o formato:

```
7 -> número de valores de N que se seguem, um por linha
1000
5000
10000
50000
100000
500000
1000000
```

Ao ler o arquivo de entrada, o programa deverá gerar cada um dos conjuntos de elementos, ordenar, contabilizar as estatísticas de desempenho e armazenar essas estatísticas no arquivo de saída.

Ao final basta processar os arquivos de saída referentes a cada uma das sementes, calculando as médias de cada estatística, para cada valor de N e estrutura de dados considerados.

Universidade Federal de Itajubá
Instituto de Matemática e Computação
Algoritmos e Estruturas de Dados II
COM-112
Vanessa Souza

Os grupos 1 a 4 apresentarão os resultados em sala de aula (mini seminário) dia **09/04/2015**.
Os grupos 5 a 7 apresentarão os resultados em sala de aula (mini seminário) dia **15/04/2015**.

Os resultados deverão ser apresentados em forma de gráficos e/ou tabelas para as três métricas pedidas, discutindo os resultados. Na data da apresentação, os grupos deverão entregar o código implementado e um relatório escrito, descrevendo as estratégias de implementação. Os grupos deverão apresentar também a configuração da máquina onde rodou os experimentos. Lembrem-se de sempre limpar a memória antes de rodar o experimento para obter os melhores tempos possíveis.

GRUPOS 1 a 4

Os grupos deverão realizar a análise de desempenho conforme a tabela 1:

Tabela 1 : Atividades dos grupos 1 a 4.

Grupo 1	Comparação de Desempenho : QuickSort e QuickSort Mediana
Grupo 2	Comparação de Desempenho : QuickSort recursivo e QuickSort iterativo
Grupo 3	Comparação de Desempenho : QuickSort e QuickSort Empilha_Inteligente
Grupo 4	Comparação de Desempenho : QuickSort e QuickSort Aleatório

Sendo:

- QuickSort Mediana : esta variação do QuickSort recursivo escolhe o pivô para a partição como sendo a mediana de k elementos do vetor, aleatoriamente escolhidos. Experimente com k=3 e k=5.
- QuickSort Iterativo : esta variação do QuickSort escolhe o pivô como elemento do meio, mas não é recursiva, mas sim uma versão iterativa do QuickSort.
- QuickSort Empilha_Inteligente : esta variação otimizada do QuickSort Recursivo processa primeiro o lado menor da partição.
- QuickSort Aleatório : esta variação do QuickSort escolhe o pivô aleatoriamente dentro do vetor.

GRUPO 5

O grupo 5 irá implementar o Algoritmo de Ordenação Shellsort, que consiste em uma extensão do algoritmo de ordenação por inserção e comparar o desempenho de ambos e o ganho (se houver, com o Shellsort).

Na apresentação dos resultados, o grupo deverá fazer uma explanação sobre o algoritmo para a sala.

GRUPO 6

O grupo 6 irá implementar o Algoritmo de Ordenação HeapSort, que consiste em uma extensão do algoritmo de ordenação por seleção e comparar o desempenho de ambos e o ganho (se houver).

Na apresentação dos resultados, o grupo deverá fazer uma explanação sobre o algoritmo para a sala.

GRUPO 7

O grupo 7 verificará, para qual tamanho de entrada, os algoritmos MergeSort e QuickSort são melhores que os iterativos bolha, bolha inteligente, seleção e inserção. Nesse caso, o grupo não deverá utilizar os tamanhos de entrada acima citados, mas encontrar esse número para cada algoritmo iterativo.

Medindo o tempo de execução de uma função em C:

O comando getrusage() é parte da biblioteca padrão de C da maioria dos sistemas Unix. Ele retorna os recursos correntemente utilizados pelo processo, em particular os tempos de processamento (tempo de CPU) em modo de usuário e em modo sistema, fornecendo valores com granularidades de segundos e microssegundos. Um exemplo que calcula o tempo total gasto na execução de uma tarefa é mostrado abaixo:

```
#include <stdio.h>
#include <sys/resource.h>

void main () {
  struct rusage resources;
  int    rc;
  double utime, stime, total_time;

  /* do some work here */

  if((rc = getrusage(RUSAGE_SELF, &resources)) != 0)
    perror("getrusage failed");

  utime = (double) resources.ru_utime.tv_sec
    + 1.e-6 * (double) resources.ru_utime.tv_usec;
  stime = (double) resources.ru_stime.tv_sec
    + 1.e-6 * (double) resources.ru_stime.tv_usec;
  total_time = utime+stime;
  printf("User time %.3f, System time %.3f, Total Time %.3f\n",
         utime, stime, total_time);
}
```

FONTE : Profa. Jussara Almeida – UFMG
http://www2.dcc.ufmg.br/disciplinas/aeds2_turmaA1/aeds2.html