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

1º Trabalho Prático

Assunto : Ordenação
Valor : 3 pontos
Entrega : 09/04/2022

8 GRUPOS DE 5 PESSOAS
1 DUPLA

1. Especificações do Trabalho

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 conjuntos de dados aleatórios, vetores ordenados e vetores em ordem decrescente e com as seguintes variações de tamanho do vetor N = 10000, 100.000, 500.000 e 1.000.000 ou mais elementos. Para cada valor de N, realize experimentos com 10 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 deverá gerar cada um dos conjuntos de elementos, ordenar, contabilizar as estatísticas de desempenho e armazenar essas estatísticas em um arquivo de saída. Quanto mais automatizado o processo, melhor pontuado será o trabalho.

Com os resultados das execuções, os grupos devem:

1. Gerar os gráficos comparativos das médias das métricas de desempenho. Para tanto, estudem a melhor forma de apresentar esses resultados, de maneira que fique explícito as diferenças. Os gráficos devem comparar as métricas entre os algoritmos estudados.
2. Discutir os resultados obtidos considerando os conceitos aprendidos em sala.
    a. Os resultados obtidos foram os esperados? Por quê?
    b. Existe diferença entre os conjuntos de dados? Essa diferença é esperada?

ATENÇÃO : Lembrem-se de sempre limpar a memória antes de rodar o experimento para obter os melhores tempos possíveis.

Os grupos deverão comparar o desempenho dos algoritmos segundo a Tabela 1. Os grupos serão sorteados.

Tabela 1 : Comparação de desempenho entre algoritmos de ordenação. Grupos de Trabalho.

Grupo 1	QuickSort, QuickSort Aleatório e QuickSort Mediana
Grupo 2	QuickSort, QuickSort Aleatório e QuickSort Empilha_Inteligente
Grupo 3	Inserção, QuickSort, MergeSort e Shellsort
Grupo 4	Seleção, QuickSort, MergeSort e HeapSort
Grupo 5	Seleção, QuickSort, MergeSort e SmoothSort
Grupo 6	Seleção, QuickSort e "An Optimal Sorting Algorithm for Mobile Devices[1]"
Grupo 7	Inserção, QuickSort, MergeSort e TimSort
Grupo 8	Counting Sort e E-Counting Sort[2]

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 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.

2. Especificações do Trabalho para a DUPLA

Existe um conceito chamado "estabilidade". A dupla deve estudar o conceito e implementar um programa que mostre de forma explícita quando um algoritmo é estável e quando não é.

3. Entregas

No dia da entrega os grupos devem apresentar um relatório técnico contendo:

- **Introdução:** Contextualização sobre os algoritmos utilizados
- **Referencial Teórico:** Apresentação dos algoritmos utilizados
- **Metodologia:** Detalhes importantes da implementação devem ser apresentados e explicados.
- **Resultados:** exemplo do arquivo de saída, gráficos e discussões. Configuração da máquina de testes. Link para um vídeo onde o grupo mostre a execução do algoritmo e do arquivo de saída. Link para o código no GitHub.
- **Considerações Finais:** opinião do grupo sobre a execução do trabalho.

Os grupos que fizerem os relatórios em Latex terão uma pontuação extra de 0.5pts no trabalho. Neste caso, devem compartilhar o link do overleaf comigo.

Neste link tem um exemplo de trabalho publicado comparando execuções do QuickSort para servir de inspiração: https://arxiv.org/ftp/arxiv/papers/2109/2109.01719.pdf.

Cada grupo terá 10 minutos para apresentar um breve resumo do trabalho para a turma: Apresentar o algoritmo estudado, Demonstrar o algoritmo, Mostrar os resultados.

[1] https://dx.doi.org/10.17485/ijst/2015/v8iS8/71491
[2] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.681.4955&rep=rep1&type=pdf

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

Outras opções são as funções time() e clock() da biblioteca time.h. Essas funções retornam os valores em segundos. Caso o resultado esteja dando 0, você precisa tratar a saída para milissegundos ou microssegundos.