CES-11
Prof. Armando

2º Exercício: Ordenação

Comparar três métodos de ordenação: BubbleSort, MergeSort e QuickSort
com relação à eficiência de tempo.

-----Parte A

Dado um arquivo contendo nomes de pessoas, colocar os nomes em ordem alfabética.
Escreva um único programa que utiliza os 3 métodos e gera 3 arquivos de saída.
Cada saída deve conter o tempo de execução e o número de comparações feitas.
Detalhes:

Cada nome de pessoa: no máximo 50 letras.
Quantidade de pessoas: sem limite ☺
Arquivo entrada:

```
C:\\Lab2\\entrada2.txt
```

Arquivos de saída:

```
C:\\Lab2\\seunome2bubble.txt
C:\\Lab2\\seunome2merge.txt
C:\\Lab2\\seunome2quick.txt
```

-----Parte B

Você deve rodar os 3 métodos com diversas entradas geradas randomicamente
e elaborar um RELATÓRIO com tabelas e gráficos comparando-os.
Primeiramente:

Rodar cada método até descobrir o tamanho máximo que consegue rodar em 2 segundos:
- BubbleSort
- QuickSort
- MergeSort com vetor T global ou static
- MergeSort com vetor T local

OBS.: Este MergeSort com T local deve ser usado **somente** para o teste dos 2 s.
Em todo o restante use o "melhor" MergeSort . ☺

Após isso:

BubbleSort: rodar com 10 ou mais entradas. Tamanhos múltiplos de mil.
MergeSort: rodar com 15 ou mais entradas. Tamanhos múltiplos de J*mil.
QuickSort: rodar com 15 ou mais entradas. Tamanhos múltiplos de K*mil.
Escolha J e K conforme a velocidade do seu computador, para o gráfico ficar "bom". ☺

No relatório, para cada método, faça:

Tabela: tamanho da entrada / número comparações / tempo
Gráfico para o número de comparações
Gráfico para o tempo de execução
O tipo deve ser "gráfico de dispersão com linhas" ou tipo análogo.

Análise dos dados:

Os resultados estão coerentes com os esperados, $O(n^2)$ e $O(n \cdot \log(n))$?
As relações entre tempo gasto e número de comparações estão coerentes?
Explique a diferença entre as duas versões do MergeSort.
Pode acrescentar outras análises que julgar interessantes.

-----Observações

Use BubbleSort **não** otimizado.
Os algoritmos MergeSort e QuickSort devem ser os **mesmos** vistos em aula,
mas com as adaptações necessárias para ordenar vetor de strings.
Devem-se usar os mesmos BubbleSort, MergeSort e QuickSort nas partes A e B,
mas na parte B podem estar em diferentes arquivos fonte (.CPP).
Inicialmente não é necessário entregar os programas nem as entradas usadas na parte B.
Mas guarde-os. Pois, durante a correção, podem ser solicitados para conferência.
Fazer individualmente.
Significa não ver os relatórios nem os resultados dos outros alunos.

-----Bizus

Tempo gasto: cuidado para NÃO contar o tempo inicial para preencher o vetor
nem o tempo final para escrever no arquivo.
Você receberá, disponível no TIDIA:
- códigos do MergeSort e QuickSort vistos em aula
- exemplo de código para marcar tempo de execução
- programa que gera strings randomicamente.

Observações:
Posso usar os arquivos gerados pelo meu colega?
- Não.

Posso usar um gerador de strings feito pelo meu colega?
- Sim.

No relatório, dizer:
"Usei um gerador feito pelo aluno Fulano" ou
"Usei o gerador fornecido no TIDIA" ou
"Fiz meu próprio gerador...", etc ☺

Número de comparações: contabilizar somente comparações entre strings.
Para isso, no lugar de `strcmp` o programa deve **obrigatoriamente** usar a seguinte função:

```
int compara (const char * a, const char * b)
{
   contador ++;
   return strcmp (a, b);
}
```

-----Entregar pelo TIDIA:

Referente à parte A:
`Seunome2.cpp`
`Seunome2.exe`
Referente à parte B:
`Seunome2relatório.doc` ou `.pdf`