UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade de Computação
Ave. João Naves de Ávila, 2121, Bloco 1B - Bairro Santa Mônica, Uberlândia/MG, CEP 38400-902
Telefone: +55 (34) 3239-4218 - www.facom.ufu.br

DISCIPLINA: ALGORITMOS E PROGRAMAÇÃO II [FACOM32201]
DOCENTE: PROF. DR. CLAUDINEY RAMOS TINOCO
VALOR: 20 PONTOS

Trabalho de Conclusão de Disciplina (TCD)

Objetivo

Implementar e comparar o desempenho de algoritmos de busca (linear e binária) e ordenação (Insertion Sort, Bubble Sort, Selection Sort, Merge Sort, Quick Sort e um algoritmo adicional), analisando seu comportamento em diferentes cenários.

Descrição do Programa

Desenvolva um sistema interativo via linha de comando (CLI) com as seguintes funcionalidades (*esse menu deverá aparecer para o usuário*):

```
1. Carregar arquivo de dados
2. Buscar elemento (linear ou binária)
3. Ordenar dados (Insert., Bubble, Selection, Merge, Quick, EXTRA)
4. Gerar relatório (Log)
5. Sair
```

Obs.:

- A opção 1 pede o nome de um arquivo que está na pasta corrente;
- As opções 2 e 3 só poderão ser executadas se um arquivo já foi carregado;
- As opções 2 e 3 devem abrir submenus para escolha dos algoritmos;
- A opção 4 só poderá ser executada se algum algoritmo já foi executado; e,
- A opção 5, antes de ser executada, deverá liberar toda a memora alocada.

Carregamento de Arquivos:

- O usuário deve informar o nome de um arquivo de texto que está na pasta corrente;
- Cada linha do arquivo representa um elemento (números inteiros);
- A quantidade de elementos em um arquivo pode variar, logo o programa deve estar preparado para mudar o tamanho do vetor de armazenamento durante a execução (*leitura deve ser feita até o arquivo acabar*); e,
- Exemplo de arquivo (`dados.txt`):

```
42
17
89
5
...
```

Operações de Busca *(Entrada: Vetor de elementos, Tamanho vetor, Elemento da busca)*:

- *Busca Linear:* Funciona em dados não ordenados;
- ***Busca Binária:*** Exige dados ordenados (se o vetor estiver desordenado, solicitar ordenação – faça uma função específica para verificar ordenação); e,
- Registrar e exibir:
  - Se o elemento foi encontrado e sua posição; e,
  - Tempo de execução da busca (9 casas decimais de precisão).

Operações de Ordenação *(Entrada: Vetor de elementos e o Tamanho vetor)*:

- Implementar os 5 algoritmos de ordenação vistos em sala de aula: BubbleSort, InsertionSort, SelectionSort, MergeSort e QuickSort.

- Escolher um algoritmo de ordenação extra para ser implementado;
  Exemplos: (https://en.wikipedia.org/wiki/Sorting_algorithm)
  Buble Sort Otimizado: BubleSort modificado para diminuir a qtd. de comparações;
  Counting Sort: Ideal para valores inteiros em intervalos limitados;
  Radix Sort: Eficiente para números inteiros com dígitos;
  Shell Sort: Versão otimizada do Insertion Sort com gaps;
  Gnome Sort: Algoritmo simples baseado em comparações e trocas.

  Importante: "TOTEM DA IMUNIDADE"!!
  O grupo que conseguir o melhor tempo de ordenação com o algoritmo EXTRA ganhará o totem. Com ele será possível "escolher tirar Total" ou na P1, ou na P2 ou em todos os labs!

  1. *Para validar o tempo de execução, todo o trabalho deverá ser implementado;*
  2. *Os programas serão avaliados no mesmo ambiente e nas mesmas condições; e,*
  3. *O algoritmo CountingSort e derivados não fazem parte do desafio.*

- Justificar a escolha do algoritmo extra;
- Registrar e exibir o tempo de execução de cada algoritmo; e,
- Opção para salvar o vetor ordenado em um novo arquivo.

Arquivo de Log:
Gerar automaticamente um arquivo `log.txt` com os tempos de ordenação e busca para cada algoritmo. *Para evitar outliers, executar cada algoritmo 100 vezes e salvar a respectiva média aritmética.*

Informações Adicionais:

1. O trabalho deverá ser feito, impreterivelmente, em grupos de 3 a 4 pessoas (*Após formar o grupo, um dos membros do grupo deve enviar para o prof., por e-mail até dia 06/02/2026 às 23h59, os nomes completos e matrículas de todos que estão no grupo*);

2. Todos os membros do grupo devem conhecer todos os detalhes de implementação do trabalho (i.e., perguntas direcionadas sobre qualquer algoritmo serão realizadas);

3. O programa deve ser feito exclusivamente em linguagem C;
4. O programa deve estar organizado, indentado, modularizado e comentado;
5. Programas copiados de outros grupos serão zerados;
6. **Programas feitos por Inteligência Artificial Generativa serão zerados**;

7. Os trabalhos devem ser enviados para o e-mail do prof. <*claudiney.tinoco@ufu.br*>; com o assunto: "*TCD_APII-20251_Grupo-X*", onde '*X*' é o número do seu respectivo grupo; até o dia 06/03/2026 às 23h59 (*trabalhos entregues após esse horário não serão aceitos*); e,

8. O e-mail deve conter os código-fontes, um arquivo *readme* (explicando como compilar seu código – compilação por linha de comando), e um pequeno relatório com os tempos de execução encontrados pelo grupo (as médias para cada algoritmo implementado).