

Especificação do Projeto: Análise de Algoritmos de Ordenação

Objetivo

O objetivo deste projeto é que os estudantes desenvolvam um programa em C++ que implemente diferentes algoritmos de ordenação, analisando o desempenho de cada um em termos de tempo de execução e de complexidade. Além disso, é proibido o uso de estruturas vector e variáveis globais, incentivando boas práticas de programação.

Descrição Geral

Os alunos, organizados em duplas ou trios, deverão implementar um programa que carregue três arquivos de entrada contendo 100.000 números inteiros e aplique cinco algoritmos de ordenação diferentes aos dados. Após a execução de cada algoritmo, o programa deve medir e registrar o tempo gasto para ordenar os números e verificar se a ordenação foi realizada corretamente.

Algoritmos de Ordenação a Implementar

1. **Bubble Sort (versão 1 dos slides)**
2. **Bubble Sort Melhorado (versão 3 dos slides)**
3. **Selection Sort**
4. **Insertion Sort**
5. **Quick Sort**

Arquivos de Entrada

Os alunos deverão utilizar três arquivos de entrada, fornecidos pelo professor:

- ***aleat_100000.txt***: Números em ordem aleatória.
- ***cresc_100000.txt***: Números em ordem crescente.
- ***decresc_100000.txt***: Números em ordem decrescente.

Cada arquivo contém 100.000 números inteiros, com um número por linha.

Funcionalidades do Programa

1. **Carregamento dos Dados:**
 - O programa deve ser capaz de carregar os números dos arquivos de entrada para a memória utilizando *arrays* estáticos ou dinâmicos.

2. Execução dos Algoritmos de Ordenação:

- Cada um dos cinco algoritmos de ordenação deve ser aplicado a cada conjunto de dados (aleatório, crescente e decrescente).

3. Medição do Tempo de Execução:

- O tempo gasto por cada algoritmo para ordenar os números deve ser medido utilizando a função *clock()* do C++.

4. Verificação da Correção da Ordenação:

- Após a execução de cada algoritmo, o programa deve verificar se os dados foram ordenados corretamente.

5. Geração de Relatórios:

- O programa deve gerar um relatório em um arquivo de saída *resultados.txt*, contendo o tempo de execução de cada algoritmo, e se os dados foram ordenados corretamente.

6. Saída Uniforme:

- A saída exibida na tela deve ser idêntica à saída registrada no arquivo *resultados.txt*.

Regras Especiais

- **Proibição do Uso de vector:** É obrigatório utilizar arrays tradicionais (*int[]* ou *int **) para armazenar e manipular os dados. O uso da estrutura *vector* do C++ é estritamente proibido.
- **Proibição do Uso de Variáveis Globais:** Todas as variáveis devem ser declaradas dentro do escopo das funções ou passadas como parâmetros.

Boas Práticas de Programação

- **Nomes Significativos:** Utilize nomes claros e descritivos para variáveis e funções, facilitando a compreensão do código.
- **Comentários no Código:**
 - **Cabeçalho:** O código deve incluir comentários no início com os nomes dos integrantes da dupla, a data de submissão e uma breve descrição do programa.

- **Comentários ao Longo do Código:** Inclua comentários para explicar seções importantes do código, como a lógica de cada algoritmo de ordenação, a manipulação de dados e o processo de medição de tempo.

O que são Variáveis Globais?

Variáveis globais são aquelas declaradas fora de qualquer função, tornando-se acessíveis a todas as funções do programa. Embora possam simplificar o acesso a dados compartilhados, seu uso pode levar a dificuldades na manutenção do código, dificuldades de depuração e maior risco de erros, especialmente em programas maiores e mais complexos. O uso excessivo de variáveis globais pode também tornar o código menos modular e mais difícil de entender.

O que deverá ser entregue?

Um arquivo compactado composto de:

1. Código-fonte do Programa (*.cpp):

- O código deve ser bem comentado e organizado de forma clara.
- Deve ser entregue o arquivo com extensão *cpp*.

2. Relatório de Análise (*.pdf):

- Um documento em formato PDF explicando os resultados obtidos, discutindo as diferenças de desempenho entre os algoritmos, e a eficácia de cada um com base nos diferentes tipos de entrada (aleatório, crescente e decrescente).
- O relatório deve incluir gráficos e tabelas que apresentem de forma visual e organizada os tempos de execução dos algoritmos em cada um dos casos testados.

3. Arquivo *resultados.txt*:

- Este arquivo deve conter os tempos de execução e a validação da ordenação para cada combinação de algoritmo e arquivo de entrada.

Critérios de Avaliação

- **Corretude da Implementação:** O programa deve implementar corretamente cada um dos algoritmos de ordenação.
- **Organização e Clareza do Código:** O código deve ser bem estruturado, de fácil compreensão e conter os comentários exigidos.
- **Análise Crítica dos Resultados:** O relatório deve demonstrar uma análise crítica e bem fundamentada dos resultados obtidos, utilizando gráficos e tabelas para melhor visualização.

- **Eficiência do Código:** A implementação deve ser eficiente em termos de tempo de execução, especialmente para algoritmos mais complexos como o Quick Sort.
- **Cumprimento das Regras:** O uso de *arrays* tradicionais e a proibição do uso de variáveis globais e a uniformidade na saída do programa devem ser respeitados. A não observância dessas regras resultará em penalização.

Prazo de Entrega

O projeto deve ser entregue até o dia 17/09/2024 pelo SIGAA, por apenas um dos integrantes do grupo.


RESPONDER TAREFA

Nome da Tarefa: Projeto 1 - Algoritmos de ordenação

Descrição: Leia atentamente a especificação do projeto com o seu grupo (dupla ou trio). Apenas um integrante deverá entregar o arquivo (compactado).

Nos comentários, escreva os nomes dos integrantes do grupo.

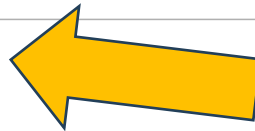
Período: Inicia em 04/09/2024 às 00h00 e finaliza em 17/09/2024 às 23h59


Arquivo:  Nenhum arquivo escolhido

(Selecione o arquivo a ser enviado como resposta. Tamanho máximo: 10 MB)

Comentários que podem ser visualizados pelo professor:

FULANO
BELTRANO
SICRANO



 Itens de Preenchimento Obrigatório

Apêndice: resultados.txt

aleat_100000.txt

Não está ordenado!
quicksort: 0.019 segundos
Ordenou!

Não está ordenado!
insertionsort: 7.179 segundos
Ordenou!

Não está ordenado!
selectionsort_otimizado: 16.842 segundos
Ordenou!

Não está ordenado!
selection_sort: 16.545 segundos
Ordenou!

Não está ordenado!
bubblesort_melhorado: 40.481 segundos
Ordenou!

Não está ordenado!
bubblesort: 50.818 segundos
Ordenou!

cresc_100000.txt

Está ordenado!
quicksort: 0.01 segundos
Ordenou!

Está ordenado!
insertionsort: 0 segundos
Ordenou!

Está ordenado!
selectionsort_otimizado: 11.99 segundos
Ordenou!

Está ordenado!
selection_sort: 12.045 segundos
Ordenou!

Está ordenado!
bubblesort_melhorado: 0 segundos
Ordenou!

Está ordenado!
bubblesort: 29.455 segundos
Ordenou!

decresc_100000.txt

Não está ordenado!
quicksort: 0.004 segundos
Ordenou!

Não está ordenado!
insertionsort: 15.354 segundos
Ordenou!

Não está ordenado!
selectionsort_otimizado: 12.045 segundos
Ordenou!

Não está ordenado!
selection_sort: 12.386 segundos
Ordenou!

Não está ordenado!
bubblesort_melhorado: 26.5 segundos
Ordenou!

Não está ordenado!
bubblesort: 36.516 segundos
Ordenou!
