Universidade Federal de Itajubá
Instituto de Matemática e Computação
Algoritmos e Estruturas de Dados II
CTCO02 Vanessa Souza

1º Trabalho Prático

Assunto : Ordenação e Análise de Complexidade

Metodologia de Ensino

Nesse trabalho, a metodologia de ensino utilizada é a **Aprendizagem Baseada em Problemas** (ABP). A ABP é uma metodologia de ensino ativa e, nesta abordagem, os alunos são apresentados a um problema desafiador, que requer a aplicação de conceitos e habilidades aprendidas previamente. Os alunos são incentivados a trabalhar em equipe, colaborar, investigar, analisar informações relevantes, formular hipóteses, desenvolver estratégias de resolução e tomar decisões informadas para encontrar soluções para o problema apresentado.

Esse trabalho deverá ser desenvolvido em grupo formado por 5 (cinco) pessoas.

Problema: Desenvolvimento de *Benchmarking* para Algoritmos de Ordenação

Você foi contratado por uma empresa de desenvolvimento de software para criar um sistema de *benchmarking* para avaliar o desempenho de algoritmos de ordenação. O objetivo é comparar o tempo de execução e a eficiência de diferentes algoritmos em lidar com conjuntos de dados de tamanhos e ordenações variados.

No link abaixo há um exemplo de *benchmarking* desenvolvido na linguagem java. Nesse exemplo os autores utilizaram apenas diferentes tamanhos de conjuntos de dados.

https://github.com/arisath/Benchmarking-Sorting-Algorithms

Descrição do problema:

A empresa deseja desenvolver um sistema que permita comparar os seguintes algoritmos de ordenação:

- Inserção
- Seleção
- MergeSort
- QuickSort Tradicional
- Um novo algoritmo: um algoritmo que não foi visto em aula e será sorteado para cada grupo, dentre os listados na Tabela 2.

O sistema deve ser capaz de receber conjuntos de dados de diferentes tamanhos e ordenações (aleatório, ordem crescente, quase ordenados (10% fora da ordem) e ordem decrescente) e executar cada algoritmo de ordenação sobre esses conjuntos.

O sistema deve analisar o desempenho dos diferentes algoritmos de ordenação considerando três métricas de desempenho: número de comparações de chaves, número de cópias de

registros realizadas e o tempo total gasto para ordenação. Ao final fornecer análises comparativas para determinar quais algoritmos são mais eficientes em diferentes cenários.

Tarefas

1. **Projetar e implementar um sistema de *benchmarking* que seja capaz de receber conjuntos de dados de entrada e executar os algoritmos de ordenação especificados.**
   - O sistema deve ser implementado utilizando a linguagem C.
   - Deve ser modularizado e incluir as boas práticas de programação (código limpo, bem comentado e indentado).
   - Cada teste deve contemplar um tamanho de vetor e quatro diferentes ordenações: aleatório, ordem crescente, ordem decrescente, quase ordenado.
   - Para os conjuntos aleatório e quase ordenado, o resultado do teste deve ser uma média de 5 repetições com conjuntos diferentes.
   - Ao final dos testes, o sistema deverá gerar uma tabela para cada métrica de avaliação, conforme exemplo da Tabela 1, que considera o tempo de processamento.

Tabela 1: Exemplo de saída do *benchmarking* para a métrica tempo de processamento.

Teste	**Tamanho Entrada**	**Tempo (ms)**			
		Aleatório (média de 5 repetições)	Ordem crescente	Ordem decrescente	Quase ordenado (média de 5 repetições)
1	1000				
2	10000				
3	100000				
4	1000000				
5	10000000				

2. **Desenvolver algoritmos de geração de conjuntos de dados de diferentes tamanhos e ordenações para fins de teste.**
   - O sistema deve gerar conjuntos de dados aleatórios de diferentes tamanhos. Para tanto, deve haver uma função com a assinatura abaixo, onde *tam* é a quantidade de elementos que deve ser gerada e a *semente* é o inteiro que permite que esse conjunto de dados seja reproduzido posteriormente. A função retorna o vetor previamente preenchido.

     int geraAleatorios(int tam, int semente);*

   - O sistema deve gerar conjuntos de dados quase ordenados, que contém cerca de 10% dos dados desordenados 90% ordenados. Para tanto, deve haver uma função com a assinatura abaixo, onde *tam* é a quantidade de elementos que

deve ser gerada e a *porcentagem* é o inteiro que determina a porcentagem de dados desordenados. A função retorna o vetor previamente preenchido.

int geraQuaseOrdenados(int tam, int porcentagem);*

- O sistema deve gerar conjuntos de dados ordenados de forma crescente e decrescente. Para tanto, deve haver uma função com a assinatura abaixo, onde *tam* é a quantidade de elementos e *ordem* define o tipo de ordenação: 0 para ordem crescente e 1 para ordem decrescente. A função retorna o vetor previamente preenchido.

  int geraOrdenados(int tam, int ordem);*

3. **Acrescentar ao sistema os algoritmos estudados em sala: inserção, seleção, mergesort e quicksort.**
   - Cada algoritmo deve ser chamado por uma função própria.

4. **Estudar e implementar o algoritmo novo**
   - O algoritmo sorteado para o grupo deve ser estudado e implementado.
   - Questões como 'Qual sua complexidade?', 'Qual seu melhor caso?', 'Qual seu pior caso?', 'Ele é estável?', 'Em quais situações reais ele poderia ser utilizado?' devem ser estudadas pelo grupo.
   - O algoritmo deve ser implementado em C e acrescentado ao sistema

5. **Realizar testes extensivos para coletar dados de desempenho para cada algoritmo de ordenação em diferentes cenários.**
   - Além do tempo, conforme demonstrado na Tabela 1, o sistema deverá comparar os algoritmos utilizando também a quantidade de comparações e trocas executadas pelos algoritmos.
   - Lembrem das boas práticas para realizar testes empíricos como esse!

6. **Analisar os resultados dos testes e produzir relatórios detalhados que comparem o desempenho de cada algoritmo.**
   - O grupo deverá produzir um relatório em formato de artigo, onde descrevem principalmente o algoritmo novo e os resultados obtidos com os testes. Além das tabelas, é importante incluir gráficos. Outro ponto importante é a discussão dos resultados obtidos. Eles estão de acordo com o que se esperava diante da conhecida complexidade assintótica?
   - Identificar padrões e tendências nos resultados e fornecer recomendações para seleção de algoritmos com base nos requisitos específicos de uma determinada aplicação.

7. **Apresentar os resultados**
    - O grupo deverá apresentar para a turma o algoritmo novo. Para tanto, devem preparar uma apresentação de cerca de 15 minutos.
    - O algoritmo novo deve ser 'ensinado', de forma clara e didática para a turma.
    - Os resultados do *benchmarking* devem ser apresentados, mostrando a eficiência dos algoritmos analisados.
    - Por ser uma metodologia ativa, o grupo deverá concluir sua apresentação mostrando para a turma como o trabalho foi dividido entre a equipe e quais os principais aprendizados.

Algoritmo Novo

Cada grupo deverá comparar o desempenho de algoritmos vistos em sala de aula, com um algoritmo que não foi visto. Para tanto, o grupo deverá estudar o funcionamento desse algoritmo e implementá-lo em C. A Tabela 2 apresenta os algoritmos não vistos em sala. Haverá um sorteio para definir qual grupo ficará com qual algoritmo.

Tabela 2 : Algoritmo Novo.

1	Counting Sort
2	Introsort
3	MergeSort in-place
4	Tree sort
5	Shellshort
6	HeapSort
7	TimSort (Python)
8	Dual-Pivot Quick-Sort (JAVA array.sort)
9	Cube sort
10	Block Sort

Papeis e Responsabilidades:

Em atividades que utilizam a Abordagem Baseada em Problemas, é importante que haja a divisão de papeis e responsabilidades. Ao ter papeis e responsabilidades bem definidos, espera-se que haja uma colaboração eficaz e que as tarefas sejam concluídas de forma eficiente e responsável. Isso contribui para o sucesso geral do projeto e para o desenvolvimento das habilidades de trabalho em equipe dos membros da equipe.

No entanto, a abordagem baseada em problemas valoriza a contribuição de todos os participantes, e é importante que todos tenham a oportunidade de se envolver ativamente na investigação, análise e discussão do problema. Nesse trabalho, em especial, é importante que todos compreendam a lógica dos algoritmos estudados e se envolvam no desenvolvimento do código.

Para esse trabalho, são sugeridos os seguintes papéis e responsabilidades:

1. **Líder do Grupo e Arquiteto de Software:** Responsável por coordenar as atividades do grupo, garantir que todos os membros estejam engajados e focados no objetivo, e facilitar a comunicação. Responsável por definir a arquitetura geral do sistema, projetar

a estrutura de alto nível, identificar os componentes principais e as interações entre eles, controlar versões do sistema.

2. **Desenvolvedor Principal:** Encarregado de coordenar o desenvolvimento do código-fonte do sistema e garantir que o código seja claro, eficiente e bem documentado.
3. **Pesquisador e Testador de Qualidade de Software:** Encarregado de realizar pesquisas sobre o problema em questão, coletar informações relevantes, buscar exemplos e casos semelhantes e encontrar recursos que possam ajudar na resolução do problema. Concomitantemente, é responsável por criar casos de teste, realizar testes, identificar e relatar bugs.
4. **Desenvolvedor e Mantenedor de código:** Além de escrever código novo, o desenvolvedor também é responsável por manter e atualizar o código existente conforme necessário. Isso pode envolver a correção de bugs, a otimização de desempenho, a refatoração de código e a implementação de novas versões do software e manter o repositório do código atualizado.
5. **Registrador e Documentador Técnico:** Responsável por documentar as descobertas do grupo, registrar ideias, anotar *insights*, manter registros claros das discussões e contribuições de cada membro e preparar relatórios ou apresentações. Ademais, fica responsável por escrever documentação técnica detalhada para o software.

Entregas

O grupo deverá entregar, via SIGAA, o relatório produzido na Etapa 6. O código desenvolvido deve ser disponibilizado em um repositório de códigos, como o GitHub. O link para o repositório deve ser disponibilizado no relatório. O relatório deve ser entregue, impreterivelmente, até o dia **29/04, meio-dia**.

Além do relatório, o grupo deverá apresentar seu trabalho para a turma, conforme Etapa 7. A ideia dessa apresentação é compartilhar com a turma o funcionamento do algoritmo novo, que só o seu grupo conhece. Portanto, não precisa apresentar os algoritmos vistos em sala. Além de apresentar o algoritmo novo, o grupo deve apresentar também o *benchmarking* desenvolvido e discutir seus resultados. A ordem de apresentação será sorteada no dia 29/04, em sala de aula. Ou sejam, todos devem estar com as apresentações prontas no dia 29/04.

Nota

A nota do trabalho é uma média das notas obtidas em cada item da rubrica de avaliação apresentada na Tabela 3.

Caso o grupo tenha alguma dúvida sobre o que se espera em algum dos itens, devem questionar antes do prazo de entrega do trabalho.

Não há negociação no prazo de entrega. Trabalhos não entregues até a data serão zerados. Trabalhos entregues fora do SIGAA serão zerados. O resultado da avaliação é entregue pela professora no SIGAA.

Universidade Federal de Itajubá
Instituto de Matemática e Computação
Algoritmos e Estruturas de Dados II
CTCO02 Vanessa Souza

Rubricas de Avaliação

Tabela 3: Rubrica de avaliação.

Item Avaliado	Atendeu Totalmente	Atendeu Parcialmente	Não Atendeu
Desenvolvimento do sistema como um todo	Sistema está modularizado, com código limpo, bem comentado e indentado. Sistema está disponível em um repositório de código e o *link* foi disponibilizado no relatório final. Nota:10,0	Sistema apresenta erros na modularização e/ou na limpeza e comentário dos códigos. Nota: 7,0 Sistema não está disponível em um repositório de código e/ou *link* não foi disponibilizado previamente. Nota: 5,0	Nota: 0,0
Geração do conjunto de dados	Sistema atende todos os requisitos especificados na Tarefa 2 Nota: 10,0	Sistema atende parcialmente os requisitos: - não fez todas as funções - as funções não seguem a assinatura definida Nota: 5,0	O sistema não gera os conjuntos de dados. Nota: 0,0
Implementação dos algoritmos vistos em sala	Sistema atende todos os requisitos especificados na Tarefa 3 Nota: 10,0	Sistema atende parcialmente os requisitos: - não fez todas as funções - fica claro que o código não é dos alunos (copiado) Nota: 4,0 a 6,0 (depende do caso)	O sistema não implementa todos os algoritmos vistos em sala Nota: 0,0

Universidade Federal de Itajubá
Instituto de Matemática e Computação
Algoritmos e Estruturas de Dados II
CTCO02 Vanessa Souza

Implementação do algoritmo novo	O grupo consegui implementar por conta própria o algoritmo que estudou Nota: 10,0	O grupo adaptou o código do algoritmo novo Nota: 5,0	O grupo copiou o código do algoritmo novo, ou não fez Nota: 0,0
Benchmarking	O grupo gerou o *benckmarking* com as três métricas de avaliação, com todos os algoritmos e tamanhos de entrada e ordenações solicitados. Os resultados foram apresentados em forma de tabelas e gráficos bonitos e de fácil leitura. Nota: 10,0	O grupo não avaliou alguma das métricas ou não avaliou todos os algoritmos solicitados. Mas avaliou o algoritmo sobre a responsabilidade do grupo. e/ou o grupo não apresentou adequadamente os resultados. Nota: 5,0	O grupo não gerou o *benchmarking* ou não incluiu na análise o algoritmo novo. Nota: 0,0
Análise dos resultados	O grupo discutiu adequadamente os resultados, incluindo o que era esperado de cada algoritmo segundo suas complexidades no melhor caso, pior e caso médio. Os resultados obtidos com o *benchmarking* fazem sentido com aquilo que é esperado para cada algoritmo. Nota: 10,0	O grupo apresenta os resultados, mas não aprofunda a discussão. e/ou Há erros na discussão dos resultados. Nota: 5,0	O grupo não discute os resultados. Nota: 0,0
Apresentação para a turma	- A apresentação é organizada. - O grupo é didático para explicar o algoritmo novo para a turma.	- Apresentação desorganizada. - O grupo tem dificuldade de explicar o algoritmo novo e/ou não responde bem eventuais perguntas feitas durante a apresentação.	- Não foi possível compreender o algoritmo novo com a explicação dada pelo grupo.

	- O grupo cumpre o tempo de 15 minutos. O grupo resume bem os resultados, ressaltando as conclusões mais importantes. - O grupo relata suas impressões com a metodologia ativa e divisão de papéis. Nota: 10,0	- Grupo não cumpre o tempo estipulado (para mais ou menos). - Grupo deixa de relatar suas impressões com a metodologia ativa e divisão de papéis. Nota: 4,0 a 6,0 (depende do caso)	-Grupo desorganizado. Nota: 0,0
Relatório ***1 ponto extra para o grupo que fizer em látex, utilizando o modelo de artigo do curso[1]***	- Relatório bem escrito, sem erros de português. - Figuras legíveis e chamadas no texto. - Presença de referências bibliográficas e citações no texto. - Conteúdo adequado. - O grupo cria casos autorais para explicar o algoritmo. - Relatório possui as seções: Introdução, Algoritmo 'novo', *Benchmarking*, Discussões e Considerações Finais. Nota: 10,0	Relatório apresenta falhas na organização, escrita e/ou regras da escrita acadêmica (figuras e referências não são chamadas no texto) Nota: 4,0 a 6,0 (depende do caso)	Relatório não foi entregue ou não atende minimamente os requisitos. Nota: 0,0

[1] https://www.overleaf.com/read/fycfdcfxzwfj