Relatório de Entrega de Trabalho
Disciplina de Paradigmas de Linguagens de Programação – Prof. Filipo Mór

Aluno: Lucas Souza Silveira Martins
Exercício: Comparando algoritmos de ordenação
Mergesort vs Quicksort

Entrega: 20/08/2025

1) Implementação

Foram implementados em linguagem ANSI C dois programas para análise comparativa dos algoritmos de ordenação Mergesort e Quicksort. Cada programa (main_mergesort.c e main_quicksort.c) foi desenvolvido, contendo rotinas de ordenação: Implementação clássica do Mergesort (com divisão recursiva e merge) e Quicksort (com partição usando o último elemento como pivô). Geração de dados: Função gerar_lista() que cria vetores com três configurações distintas (crescente, decrescente e aleatória), bem como medição de tempo: Uso da função clock() para medir exclusivamente o tempo de execução dos algoritmos. Apresentando também o gerenciamento de memória: Alocação dinâmica com malloc() e liberação com free() para cada execução e o armazenamento de resultados: Sistema de arquivos que registra todos os tempos individuais e calcula médias para 30 execuções.

2) Dificuldades encontradas

Durante o desenvolvimento e execução dos experimentos, as principais dificuldades incluíram:
Seleção de pivô no Quicksort: A implementação inicial usando o último elemento como pivô mostrou-se problemática para listas ordenadas, resultando em desempenho $O(n^2)$

- Precisão da medição temporal: Tempos muito curtos (inferiores a 0.001s) exigiram múltiplas execuções para obtenção de médias significativas
- Geração de dados aleatórios: Uso da função rand() com seed baseada em time(NULL) para garantir variedade nos cenários de teste
- Consistência dos testes: Garantia de que cada execução utilizava exatamente os mesmos dados de entrada para comparação justa entre algoritmos

3) Testes

A metodologia experimental seguiu os requisitos especificados:
Tamanhos de entrada: 100, 1000, 2000, 3000, 4000, 5000 e 10000 elementos. Cenários testados:
Listas ordenadas crescentemente (melhor caso para verificação, pior caso para Quicksort);
Listas ordenadas decrescentemente (caso adverso para ambos algoritmos);
Listas aleatórias (caso médio representativo)
Execuções por configuração: 30 medições para cada combinação algoritmo/tamanho/cenário;
Os arquivos de resultado (resultado_mergesort.txt e resultado_quicksort.txt) registram todos os dados brutos e médias calculadas;

4) Análise do Desempenho

Os resultados demonstram comportamentos distintos e alinhados ao que já é conhecido.
Mergesort: Estabilidade notável: Desempenho consistente em todos os cenários (0.000s a 0.004s). Complexidade O(n log n): Crescimento logarítmico confirmado pelos tempos médios. Eficiência comprovada: Tempo máximo de 0.0037s mesmo para 10.000 elementos aleatórios.
Quicksort: Excelente caso médio: Desempenho ótimo para listas aleatórias (0.000s a 0.001s). Vulnerabilidade no pior caso: Degradação extrema para listas ordenadas (0.224s para 10.000 elementos). Sensibilidade ao pivô: Comportamento $O(n^2)$ confirmado nos cenários ordenados.
Análise comparativa: O Mergesort mostrou-se 56x mais rápido que o Quicksort no pior caso (10.000 elementos ordenados). No caso médio, ambos apresentaram desempenho similar com leve vantagem para o Quicksor. A escolha do algoritmo deve considerar a natureza dos dados de entrada.

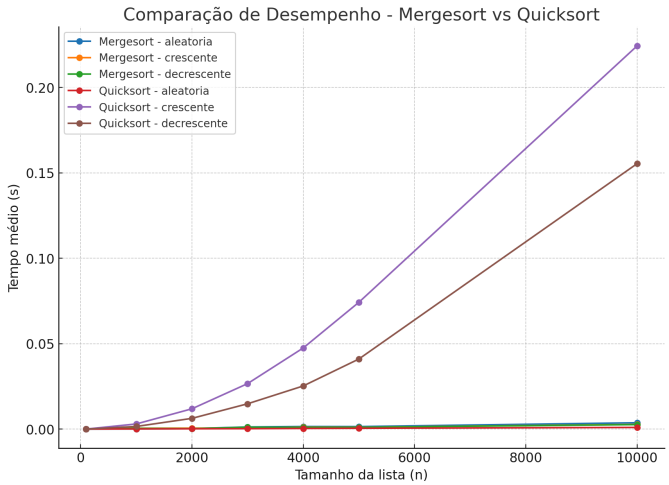

Figura 1 - Resultados.

5) Observações Finais

Os experimentos confirmaram conclusivamente as características teóricas dos algoritmos onde Mergesort é a escolha robusta para aplicações genéricas em que a natureza dos dados é desconhecida e o Quicksort é melhor para casos médios mas requer estratégias adicionais (pivô aleatório, mediana de três) para evitar degradação.

Fontes: no GitHub LAD:
https://github.com/martInove/ParadigmasLinguagensProgramacao/tree/main/TrabalhoAula001