UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE SOBRAL

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

FRANK WILLIAM ARAUJO SOUZA - 473269

RELATÓRIO - PARTE 2 DO TRABALHO

SOBRAL-CE

2022

1 INSTRUÇÕES E PROJETO DEV-C++

1.1 Instruções

2. Implemente os algoritmos BubbleSort, InsertionSort, QuickSort e HeapSort e calcule o tempo de cada um para ordenar vetores de tamanho $10^3, 10^4, 10^5$ e 10^6 com os elementos dispostos de três formas: crescente, decrescente, e aleatória. Elabore um pequeno relatório com os dados obtidos (no máximo, com três páginas – formato PDF).

1.2 Projeto DEV-C++

Para atender as instruções solicitadas pelo trabalho, foi desenvolvido um projeto na IDE Dev-C++ com o objetivo de organizar o código, afim deixar o código com uma boa legibilidade. Temos que a estrutura do código segue o seguinte formato:

- Arquivo main.c: responsável por inicializar o programa e chamar as funções de processamento dos testes dos algoritmos passando o tamanho dos vetores de teste;
- Arquivo ordenacao.h: cabeçalho das funções presentes em ordenacao.c, nesse arquivo são definido as assinaturas das funções de processamento dos métodos de ordenação;
- Arquivo ordenacao.c: neste arquivo estão contido todos os métodos de ordenação, porém a partir de main.c só é possível acessar as funções presentes em ordenacao.h.

A organização do código segue as recomendações de criação de tipos abstratos de dados, além da utilização de funções seguindo as recomendações de código limpo. Ao executar o código disponível, é recomendado a utilização de um computador com 8 Gbs de RAM, devido ao grande uso de memoria nas pilhas de recursividade de algumas técnicas de ordenação. Ademais, para fins de verificação rápida desse trabalho é recomendado que comente as linhas que processam os testes dos métodos BubleSort e InsertionSort com vetores de tamanho 10^6.

2 TESTES E CONCLUSÕES

2.1 Testes

No método main foram configurado os devidos tamanhos de vetores a serem ordenados como requisitado nas instruções do trabalho e o código executado, ao fim da execução foi obtida uma lista com o calculo de tempo em milissegundos impressa no terminal. Ademais, os dados exibidos no terminal foram organizados pela técnica de ordenação de vetores e adicionados na Tabela 1.

2.2 Conclusão

Na Tabela 1 é possível observa que os métodos QuickSort e HeapSort tiveram um tempo significativamente menor para ordenar os vetores de tamanho de 10^6, porém a medida em que o tamanho dos vetores de testes tiveram os seus tamanhos defasados, os métodos como BubbleSort começam a ser convidativos dados que sua complexidade de implementação é menor.

O método InsertionSort apresenta o seu melhor funcionamento em vetores já ordenados e o seu pior funcionamento em vetores em ordem descrente como é visto na literatura presente.

Outrossim é que o método HeapSort apresenta um tempo pequeno se comparado com as outras técnicas de ordenação. A superioridade do HeapSort é observada tanto para vetores da ordem de tamanho 10^3 e de 10^6 e em todos os estados de organização dos vetores utilizados nesse trabalho. Já em posição oposta temos o BubbleSort que é a solução mais simples implementada em termos de quantidade de linhas digitadas, porém a medida em que os vetores submetidos a essa técnica de ordenção crescem seu desempenho é defasado.

Com isso é possível averiguar que o comportamento dos métodos de ordenação implementados seguiram o comportamento esperado baseando-se na literatura.

Técnica de Ordenação	Tamanho do Vetor(N)	Ordem do Vetor	Tempo(Ms)
BubbleSort	10^6	Crescente	1.22721e+006
BubbleSort	10^6	Descrente	2.03853e+007
BubbleSort	10^6	Aleatório	1.2183e+006
BubbleSort	10^5	Crescente	12939
BubbleSort	10^5	Decrescente	21118
BubbleSort	10^5	Aleatório	12749
BubbleSort	10^4	Crescente	127
BubbleSort	10^4	Decrescente	212
BubbleSort	10^4	Aleatório	131
BubbleSort	10^3	Crescente	1
BubbleSort	10^3	Decrescente	2
BubbleSort	10^3	Aleatório	1
InsertionSort	10^6	Crescente	3
InsertionSort	10^6	Decrescente	2.35867e+006
InsertionSort	10^6	Aleatório	1.16749e+006
InsertionSort	10^5	Crescente	0
InsertionSort	10^5	Decrescente	24084
InsertionSort	10^5	Aleatório	12423
InsertionSort	10^4	Crescente	0
InsertionSort	10^4	Decrescente	234
InsertionSort	10^4	Aleatório	117
InsertionSort	10^3	Crescente	0
InsertionSort	10^3	Decrescente	2
InsertionSort	10^3	Aleatório	1
QuickSort	10^6	Crescente	54
QuickSort	10^6	Decrescente	53
QuickSort	10^6	Aleatório	98
QuickSort	10^5	Crescente	4
QuickSort	10^5	Decrescente	4
QuickSort	10^5	Aleatório	8
QuickSort	10^4	Crescente	0
QuickSort	10^4	Decrescente	1
QuickSort	10^4	Aleatório	1
QuickSort	10^3	Crescente	0
QuickSort	10^3	Decrescente	0
QuickSort	10^3	Aleatório	0
HeapSort	10^6	Crescente	168
HeapSort	10^6	Decrescente	163
HeapSort	10^6	Aleatório	194
HeapSort	10^5	Crescente	15
HeapSort	10^5	Decrescente	14
HeapSort	10^5	Aleatório	18
HeapSort	10^4	Crescente	1
HeapSort	10^4	Decrescente	1
HeapSort	10^4	Aleatório	1
HeapSort	10^3	Crescente	0
HeapSort	10^3	Decrescente	0
HeapSort	10^3	Aleatório	0

Tabela 1 – Resultados obtidos.