**Documentação Trabalho Prático Extração-Z**

**Estruturas de Dados**

**Nome: Heloiza Aparecida dos Santos**

**Matrícula: 2017086953**

Departamento de Ciencia da Computação – Universidade Federal de Minas Gerais (UFMG) Belo Horizonte – MG – Brazil

heloizas@ufmg.br

1. **Introdução**

Este trabalho prático tem como objetivo desenvolver conhecimentos a cerca dos métodos de ordenação estudados nas aulas da disciplina Estruturas de Dados DCC205, incluindo os algoritmos de ordenação **XXXXX**. Foi solicitado a ordenação de arquivos contendo nomes de planetas e a distância deles em saltos hiperespaciais do mais distante para o mais perto.

1. **Implementação**

* A estratégia utilizada para desenvolver o programa controlador de robôs foi a implementação de classes e dos métodos de ordenação **XXXXX**. A classe principal é a Ordenacao, deve receber 2 argumentos: *arquivo.txt* e *quantidadeLinhas*. As classes estão organizadas da seguinte maneira:

|  |  |
| --- | --- |
| **Classes:** | **Principais responsabilidades:** |
|  |  |
|  |  |
|  |  |
|  |
|  |  |

No arquivo Main do programa um objeto da classe Ordenacao é instanciado: ordenacao(arquivo, qtdLinhas). No construtor dessa classe o arquivo é lido tendo como base a quantidade de linhas solicitadas. Na classe Ordenacao existem funções que instanciam a classe do método de ordenação solicitado.

* O código está organizado seguindo a seguinte estrutura de pastas:

projeto

- bin: executáveis gerados após compliação.

- src: códigos .cc.

- obj: arquivos .o.

- include: headers .h.

Makefile

* O trabalho foi desenvolvido e testado utilizando a seguinte configuração:  
  Sistema Operacional: macOS Catalina 10.15.7  
  Linguagem de programação implementada: C++  
  Compilador utilizado: GCC  
  Dados do processador: Intel Iris Plus Graphics 640  
  Memória RAM: 8 GB

1. **Instruções de compilação e execução**

Na pasta principal do programa, executar o comando make. Na pasta bin rodar o programa fornecendo os arquivos na linha de comando.

|  |
| --- |
| make  cd bin  ./run.out arquivo.txt qtdLinhas\* |

\*qtdLinhas é um inteiro 1<=qtdLinhas>=200000.

1. **Análise de complexidade**

Análise de complexidade das principais funções:

|  |  |
| --- | --- |
| **Funções:** | **Complexidade:** |
| gerarMapa() | **Complexidade de Tempo:**  Nessa função o mapa no arquivo *mapa.txt* é mapeado em uma matriz. Essa é a função onde o custo de tempo é maior, pois contém dois laços aninhados, cuja duração se baseia na iteração de toda a entrada possuindo duração Θ(n²).  **Complexidade de Espaço:**  O algoritmo utiliza memória baseada na matriz com n lados e m colunas, possuindo complexidade de espaço de Θ(n²). |
| encontrarPonto(int, int) | **Complexidade de Tempo:**  Complexidade constante Θ(1). Existe um if que compara o ponto solicitado com o ponto da matriz e retorna uma string.  **Complexidade de Espaço:**  Complexidade constante Θ(1). |
| adicionarPonto(int, int) | **Complexidade de Tempo:**  Complexidade constante Θ(1). Apenas atribui um caracter ‘.’ no ponto solicitado.  **Complexidade de Espaço:**  Complexidade constante Θ(1). |
| gerarComandos() | **Complexidade de Tempo:**  Itera sobre o arquivo de *comandos.txt* de tamanho n com um while. Complexidade assintotica de tempo Θ(n).  **Complexidade de Espaço:**  Complexidade de espaço Θ(n). |
| tipoComando(string) | **Complexidade de Tempo:**  Complexidade constante Θ(1). Realiza comparações para descobrir se é ordem de comando, comando direto ou prioritário.  **Complexidade de Espaço:**  Complexidade constante Θ(1). |
| executarComando(string) | **Complexidade de Tempo:**  Complexidade constante Θ(1). Realiza comparações e também chama funções com ordem de complexidade constante.  **Complexidade de Espaço:**  Complexidade constante Θ(1). |
| adicionarComando(string) | **Complexidade de Tempo:**  Complexidade constante Θ(1). Chama o método enfileirar de uma fila circular de arranjo.  **Complexidade de Espaço:**  Complexidade constante Θ(1). |
| adicionarComandoPrioritario(string) | **Complexidade de Tempo:**  Complexidade constante Θ(1). Chama o método desenfilerar e enfileirar de uma fila circular de arranjo.  **Complexidade de Espaço:**  Complexidade constante Θ(1). |
| executarRobo(mapa) | **Complexidade de Tempo:**  Complexidade Θ(n). de acordo com o tamanho da fila de execução do robô.  **Complexidade de Espaço:**  Complexidade constante Θ(1). |

1. **Conclusão**

Esse trabalho lidou com o problema de ordenar arquivos utilizando os métodos de ordenação aprendidos. Durante a implementação da solução para o problema, houveram importantes desafios a serem superados, por exemplo a alocação dinâmica de memória.

**Referências**

* Chaimowicz, L. and Prates, R. (2021). Slides virtuais da disciplina de estruturas de dados. Disponibilizado via moodle. Departamento de Ciencia da Computação. Universidade Federal de Minas Gerais. Belo Horizonte
* <http://www.cplusplus.com/reference/fstream/fstream/>
* <http://www.cplusplus.com/reference/string/>
* <http://www.cplusplus.com/reference/string/stoi/>