Unioeste - Universidade Estadual do Oeste do Paraná
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
Colegiado de Ciência da Computação
Curso de Bacharelado em Ciência da Computação

DOCUMENTAÇÃO CODIFICAÇÃO

Guilherme de Oliveira Correia, João Vitor Detoni e Rodrigo Brickmann Rocha

Orientador: Leonardo Medeiros

Cascavel, 2022

SUMÁRIO

1. INTRODUÇÃO

O presente trabalho tem por objetivo o desenvolvimento de 3 estruturas de dados, sendo elas: um TAD Grafo (e suas respectivas funções, como por exemplo, busca, inserção, exclusão, entre outros), um TAD Fila usando Herança de uma Lista Encadeada Simples (utilização de orientação a objeto) e um método de ordenação, neste caso, o Quicksort.

Tratando do conceito de grafos, deve-se saber que eles consistem em um conjunto de nós (vértices) e um conjunto de arcos (arestas), sendo que cada arco é especificado por um par de nós e, se os pares forem ordenados, o grafo será orientado, caso contrário não.

Falando sobre o algoritmo de Fila utilizando Herança, deve-se saber que a ideia geral é a representação que simula uma fila da vida real, sendo muito utilizada na simulação de processos sequenciais. Tratando da relação com herança, que consiste em uma forma de abstração utilizada na orientação a objetos e trata de uma relação entre classes, o algoritmo seguirá uma ideia lógica de apresentação dos conceitos gerais da relação.

Algoritmos de ordenação consistem na ordenação de um conjunto de elementos em uma ordem específica. Sobre o algoritmo de ordenação quicksort, deve-se saber que ela ocorre por troca de partição, mais especificamente, a ideia é dividir o problema de ordenar um conjunto com n itens em dois problemas menores, que são ordenados independentemente e os resultados combinados para formar a resposta final.

Para o desenvolvimento de todo o apresentado foi utilizado o ambiente de desenvolvimento VSCode e, a fim de tornar o trabalho adaptado ao requisitado, foram feitas as devidas mudanças para tornar o código executável no sistema operacional Linux.

2. DESENVOLVIMENTO

Todos os códigos foram desenvolvidos no formato module.h, module.c e main.c, e apresentam um respectivo makefile para compilação e execução. Como mencionado antes, os testes foram realizados para satisfazer o SO, mas também foram testados casos extremos.

Para os 3 códigos é possível perceber a mesma interdependência, de tal modo que o main.c possui o código em si, o module.h possui o header das funções e o module.c, possui as funções completas.

2.1. TAD GRAFO

A primeira implementação foi um TAD Grafo, uma estrutura de dados característica por conter nós (vértices) ligados entre si por arcos (arestas). Entre as operações disponíveis estão: a criação do grafo, alocando espaço para criação de um novo grafo; inserir arestas, o que torna possível fazer a ligação entre dois nós; remover arestas, caso seja necessário remover a ligação entre dois nós; imprimir o grafo, a fim de apresentá-lo visualmente e liberar o grafo, que tem por objetivo liberar a memória que foi alocada para criação do grafo.

Além dessas funções básicas, foram também implementadas duas formas de busca bastante famosas, a busca por profundidade e a busca por largura, que tem por objetivo visitar o grafo de duas maneiras diferentes. Além disso, foi também implementado uma maneira de encontrar o caminho mínimo dentro do grafo.

Quanto a entrada de dados, temos elas demonstradas na figura que segue:

```
Informe o número de vértices: 13

Informe o grau máximo do grafo: 2

O grafo é ponderado?(1 para SIM e 0 para NAO) 1
```

Figura 1 - Ilustração do Funcionamento de Entrada Codificado

Primeiro é informado o número de vértices que o grafo conterá, depois é informado qual o grau máximo para cada vértice e, finalmente, se o grafo apresenta pesos (ponderado) ou não.

Em seguida é possível escolher as operações já citadas anteriormente através da inserção do número que aponta a operação:

```
1 - Inserir Aresta
2 - Remove Aresta
3 - Apagar Grafo
4 - Imprime Grafo
5 - Busca menor
6 - Busca em Profundidade
7 - Busca em Largura
```

Figura 2 - Ilustração das Funções Codificadas e Disponíveis

A saída de dados será em um arquivo (não acumulativo), chamado “saida.txt”, e contará com a impressão do grafo em forma de lista de adjacência, como mostra a figura a seguir:

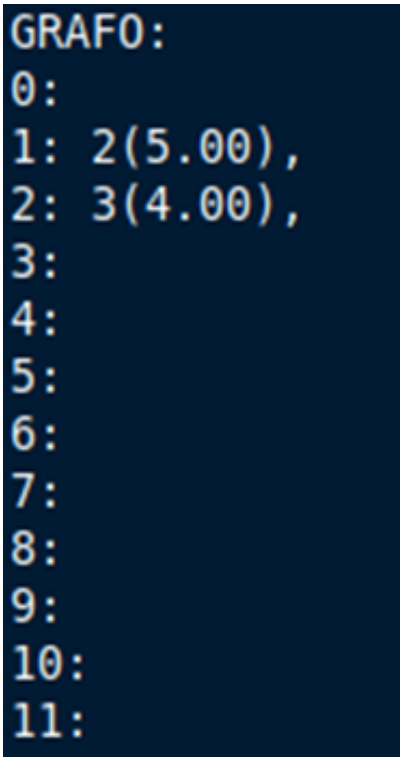

Figura 3 - Ilustração do Resultado no Arquivo de Saída

É possível observar na **Figura 3** um exemplo de teste do algoritmo, em tal caso, foi inserido uma quantidade de vértice igual a 12, com um grau máximo 2 e ponderado.

Para os pares de vértices 1-2 e 2-3 foram inseridas arestas, que possuem, respectivamente, os pesos 5 e 4.

2.2. TAD FILA USANDO HERANÇA

A segunda implementação foi um TAD Fila utilizando o conceito de herança de orientação a objeto, vindo de uma Lista Encadeada Simples. Por se tratar de uma POO, as operações disponíveis estão relacionadas a acessar os dados que os objetos de uma classe recebem.

É importante ressaltar que essa aplicação foi implementada utilizando uma situação aleatória, no caso, um sistema de paciente e doutor de um hospital.

Em relação a entrada de dados, são apresentados prints que guiam bem nas informações que devem ser inseridas, como mostra a figura:

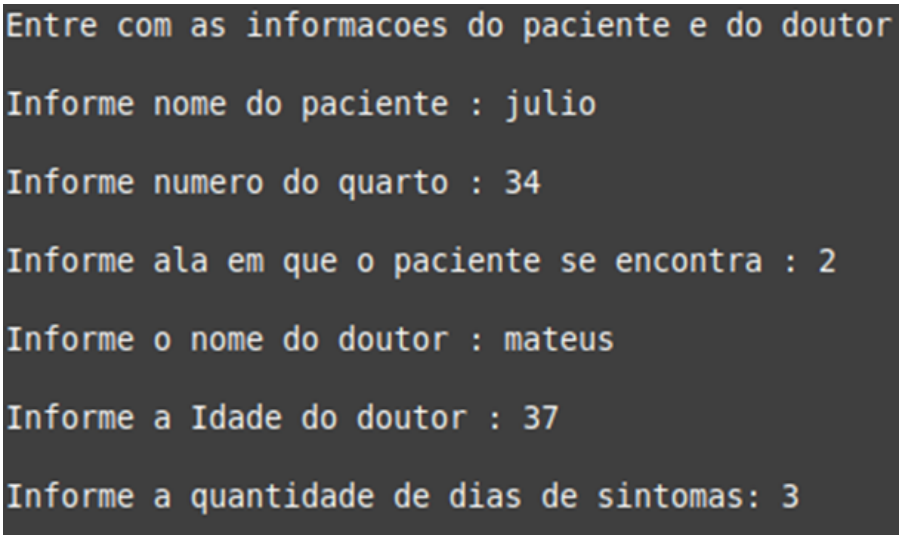

Figura 4 - Ilustração de Entrada do Algoritmo

A saída de dados será em um arquivo (não acumulativo), chamado “saida.txt”, e contará com a impressão das informações sobre o paciente e o doutor, como mostra a figura:

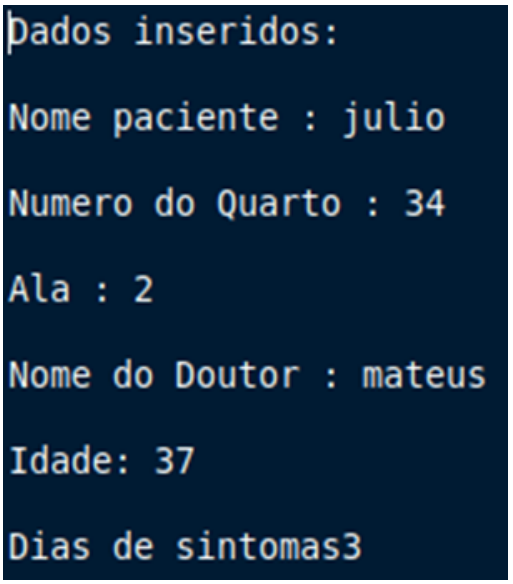

Figura 5 - Ilustração do Resultado no Arquivo de Saída

É possível observar um caso de teste deste algoritmo observando as entradas inseridas na Figura 4 e suas respectivas saídas apresentadas na Figura 5.

2.3. QUICKSORT

A terceira e última implementação foi um método de ordenação, o Quicksort, que tem por característica escolher um pivô e particionar os números, de modo que os números menores que o pivô estejam à esquerda dele e os maiores à sua direita.

Como esse algoritmo tem apenas um propósito, a ordenação, há apenas uma operação disponível, a própria ordenação. No entanto ela pode ser subdividida em operações menores, a de particionamento, para quebrar os números em duas partes (os à esquerda e os à direita) e a troca de troca de posição, a fim de ordenar os valores.

Quanto a entrada de dados, demonstra-se na seguinte figura:

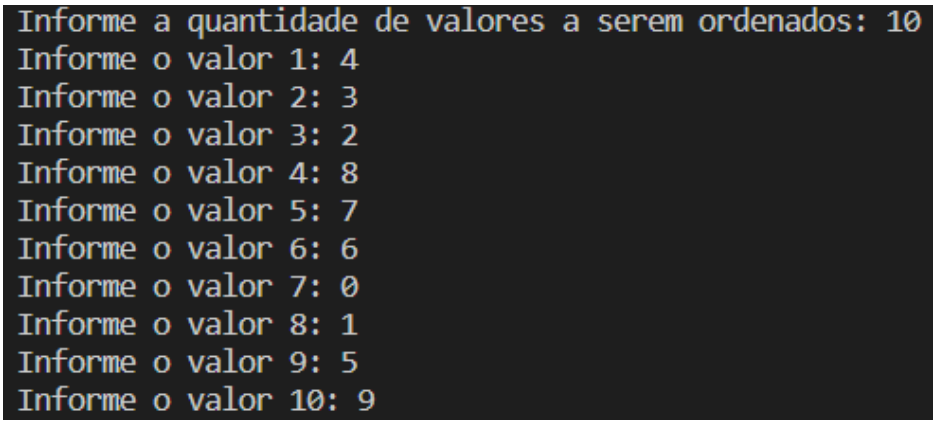

Figura 6 - Ilustração de Entrada

É informado a quantidade de números que devem ser ordenados e, logo em seguida, é requisitado, um a um, os números que devem ser ordenados.

A saída de dados será em um arquivo (não acumulativo), chamado “saida.txt”, e contará com a impressão dos números na ordem em que foram inseridos e logo após todos ordenados em ordem crescente:

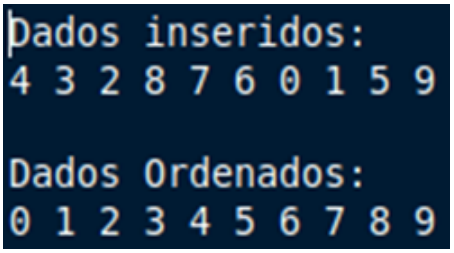

Figura 7 - Ilustração do Resultado no Arquivo de Saída

Os casos de testes podem ser os mais variados, não importando se forem números positivos ou negativos, como podemos observar no caso seguinte:

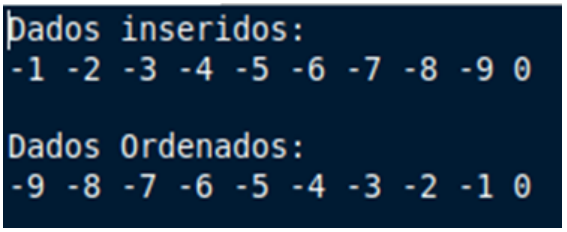

Figura 8 - Ilustração do Resultado no Arquivo de Saída

3. CONSIDERAÇÕES FINAIS

Através do trabalho proposto, foi possível absorver os conceitos teóricos e práticos dos três algoritmos solicitados e implementados, bem como pensar em planos de testes para serem executados, analisando se todo o comportamento irá ocorrer da maneira correta em cada um dos casos, garantindo, por consequência, uma boa qualidade do projeto. Visto que são pontos muito importantes para a área de computação, serão de grande importância para futuras implementações, visto que a visão de desenvolvimento já estará mais evoluída.