Trabalho Avaliativo 02 – Estrutura de Dados

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

8 de julho de 2022

**Sumário**

**Sumário** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **1**

**1 Introdução** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **1**

**2 Desenvolvimento** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **1**

**2.1 TAD Grafo** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **2**

**2.2 TAD Fila usando Herança** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3**

**2.3 Quicksort** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4**

**3 Considerações Finais** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **6**

**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.

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

**2 Desenvolvimento**

Todos os 3 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 inteiro 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:

Texto

Descrição gerada automaticamente

**Figura 1**

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 apresentará pesos (ponderado) ou não.

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

Texto

Descrição gerada automaticamente

**Figura 2**

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:

Uma imagem contendo Texto

Descrição gerada automaticamente

**Figura 3**

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 relacionas 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:

Texto

Descrição gerada automaticamente

**Figura 4**

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:

Texto

Descrição gerada automaticamente

**Figura 5**

É possível observar um caso de teste desse 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 a 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 a esquerda e os a direita) e a troca de troca de posição, a fim de ordenar os valores.

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

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**Figura 6**

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:

Texto

Descrição gerada automaticamente

**Figura 7**

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

Texto

Descrição gerada automaticamente

**Figura 8**

**2 Considerações Finais**

Através do trabalho proposto, foi possível absorver os conceitos teóricos e práticos dos 3 algoritmos implementados. Visto que são pontos muito importantes para a área de computação, serão de grande importância para futuras implementações.