**UNIVERSIDADE FEDERAL DO ESPIRITO SANTO**

**CIÊNCIA DA COMPUTAÇÃO**



**IASMIN MARQUES PEREIRA**

**RAFAEL MENDES MERLO**

**RELATÓRIO TRABALHO – CONTROLE DE ESTOQUE**

**(ÁRVORE RUBRO NEGRA – RECURSIVA)**

**ESTRUTURA DE DADOS II**

SÃO MATEUS

2023

**IASMIN MARQUES PEREIRA**

**RAFAEL MENDES MERLO**

**RELATÓRIO TRABALHO – CONTROLE DE ESTOQUE**

**(ÁRVORE RUBRO NEGRA – RECURSIVA)**

**ESTRUTURA DE DADOS II**

Relatório do trabalho de estrutura de dados II sobre controle de estoque através da estrutura da Árvore Rubro apresentado ao professora Luciana Lee, como requisito para obtenção de nota da Universidade Federal do Espírito Santo – Campus São Mateus.

PROFESSORA: **Luciana Lee**

SÃO MATEUS

2023

**SUMÁRIO**

1. **Justificativa.........................................................................................p.5**
2. **Objetivos..............................................................................................p.5**
3. **Introdução...........................................................................................p.5**
4. **Metodologia.........................................................................................p.5**

**4.1. Método da Bisseção....................................................................p.5**

**4.2. Método de Newton-Raphson.......................................................p.6**

**4.3. Método da Secante.......................................................................p.6**

**4.4. Jupyter Notebook.........................................................................p.7**

1. **Resultados e Discussão.....................................................................p.7 5.1. Exemplo 1 ....................................................................................p.7**

**5.1.5. Vantagens e Desvantagens – Exemplo 1 .....................p.10**

**5.2. Exemplo 2 ...................................................................................p.10**

1. **Conclusão..........................................................................................p.16**
2. **Referências Bibliográficas...............................................................p.16**

**1. Justificativa**

Este trabalho possui como intuito implementar a árvore rubro-negra de forma que os nós não tenham ponteiro para o nó pai, atendendo as seguintes operações: cadastro de um novo produto, exclusão um produto cadastrado, atualização da quantidade de um produto no estoque, listar todos os produtos cadastrados, listar todos os produtos em estoque, impressão da árvore Rubro-Negra.

**2. Objetivos**

* O programa deverá ser implementado em Linguagem C e totalmente comentado.
* O relatório deverá conter a explicação detalhada do programa (forma de compilação do programa, estruturas, funções, formatos de entrada e saída de dados).
* O programa deverá ter as operações de cadastro de novo produto, exclusão de um produto cadastrado, atualização da quantidade de um produto no estoque, listar todos os produtos cadastrados, listar todos os produtos disponíveis no estoque e por fim a impressão da árvore Rubro-Negra.
* A entrega da implementação e do relatório é obrigatória. A falta de qualquer uma dessas partes leva à anulação do trabalho (nota zero).
* A estrutura de um nó da árvore não pode ter ponteiro para o nó pai (e nem para nenhum ancestral na árvore

**3. Introdução**

Visando o objetivo geral e os específicos decidimos dividir o desenvolvimento do trabalho em alguns tópicos, sendo eles: a implementação da árvore rubro negra (sem o ponteiro para o pai), operações de inserção e exclusão do nó da arvore rubro negra e por fim as implementações e adições no código das informações relacionadas ao produto a ser controlado pelo estoque. A organização da implementação foi feita através de um tipo abstrato de dados (TAD), em três arquivos:

* **main.c:** contém o menu que sera exibido para o usuário com as opções disponíveis de controle de estoque, produtos e impressão da árvore rubro negra;
* **rbt.c:** nesse arquivo contém todas as implementações relacionadas a estrutura da árvore rubro negra, inserção, exclusão, balanceamento, rotações, função de listagem de produtos cadastrados, impressão de produtos no estoque, impressão da árvore rubro negra, entre outras funções auxiliares que nos ajudaram nas operações/ funções principais;
* **rbt.h:** onde contém todos os headers/ cabeçalho das funções;

**4. Metodologia**

**4.1 Implementação da Árvore Rubro Negra**

Implementação da Árvore Rubro negra recursiva por conta da limitação de não pode ter um ponteiro para seu pai; Para uma árvore ser considerada uma árvore rubro negra ela precisa preencher alguns requisitos, sendo:

1. Todo nó da árvore é vermelho ou preto;
2. A raiz é sempre preta;
3. Todo nó folha (nó externo/ NULL) é preto;
4. Se um nó é vermelho, então os seus filhos são pretos (Não existem nós vermelhos consecutivos);
5. Para cada nó, todos os caminhos desse nó para os nós folhas descendentes contém;

O balanceamento da Árvore Rubro Negra é feito através da altura preta da árvore, da quantidade de NOS pretos presentes na árvore. Durante o balanceamento da árvore faremos através das rotações e da mudança de cor dos nós caso necessário.

Sobre as rotações, existem apenas duas funções de rotação: rotação à esquerda e a rotação à direita;

* Rotação a Esquerda

Forma

Descrição gerada automaticamente**Texto

Descrição gerada automaticamente**

Figura 1 - Rotação a esquerda

* Rotação à Direita

**Diagrama, Forma

Descrição gerada automaticamenteTela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

Figura 2- Rotação a direita

* Mudança de Cor da Árvore

Podemos ter a necessidade de mudar a cor de um nó e de seus filhos de vermelho para preto e vice-versa. A função implementada só troca a cor dos NOS não a sua informação;

**Uma imagem contendo relógio, objeto, homem

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente**

Figura 3 - Mudar Cores

**4.1.1 Inserção**

Na implementação da inserção dividimos o codigo em duas funções principais, a “inserRb” e a “insertNodeRb”. Sendo a “inserRb” uma função auxiliar da inserção, que sera chamada no arquivo “main.c” quando o usuário digitar a opção número um, de inserir um novo produto; já na “insertNodeRb” é feito o tratamento das possiveis violações de uma arvore rubro negra.

**Texto

Descrição gerada automaticamente**

Figura 4 - Função auxiliar de Inserção

Existem três casos possíveis que podem violar as propriedades da árvore rubro negra, são eles:

Caso 1: o tio de q é Rubro.

**Uma imagem contendo espada, foto, pendurado, diferente

Descrição gerada automaticamente**

Figura 5 - Violação Caso 1

Caso 2: O tio de q é Negro e q é filho a direita de seu pai

Caso 3: o tio de q é Negro e q é é filho a esquerda de seu pai.

**Diagrama

Descrição gerada automaticamente**

Figura 6 - Casos 2 e 3 de violação

**Texto

Descrição gerada automaticamente**

Figura 7 - Função que lida com a Inserção e possíveis violações

**4.1.2 Exclusão**

Na implementação da exclusão dividimos o codigo em duas funções principais, a “removeRb” e a “removeElementRb”. Sendo a “removeRb” uma função auxiliar da remoção, que sera chamada no arquivo “main.c” quando o usuário digitar a opção número dois, de excluir um produto; já na “removeElementRb”, e na função secundária auxiliar “dellBalanceNodes”, é feito o tratamento das possiveis violações de uma arvore rubro negra quando uma chave é excluida.

**Texto

Descrição gerada automaticamente**

Figura - Função auxiliar de remoção

**Texto

Descrição gerada automaticamente**

Figura 9 - Função que lida com a Exclusão e possíveis violações

A remoção é um pouco mais complicada de lidar, comparando a com a inserção, os casos de violação da remoção são:

Caso 1: No removido preto com filho vermelho

**Diagrama

Descrição gerada automaticamente**

Figura – Exemplo de Caso 1 da remoção

Caso 2: Irmão Preto e Sobrinho Preto

**Uma imagem contendo objeto, relógio, desenho

Descrição gerada automaticamente**

Figura - Exemplo de Caso 2 da remoção

**Uma imagem contendo objeto, relógio

Descrição gerada automaticamente**

Figura - Exemplo de Caso 2 da Remoção

Caso 3: Irmão Preto e Sobrinho(s) Vermelho(s)

**Uma imagem contendo objeto, relógio

Descrição gerada automaticamente**

Figura - Exemplo do caso 3 de remoção

**Uma imagem contendo objeto, relógio, relógio de pulso

Descrição gerada automaticamente**

Figura - Exemplo de caso 3 de remoção

Caso 4: O Irmão do Nó Removido É Vermelho

**Uma imagem contendo diferente, foto, ar, programa

Descrição gerada automaticamenteUma imagem contendo diferente, foto, ar, programa

Descrição gerada automaticamente**

**Imagem de vídeo game

Descrição gerada automaticamente com confiança média**

Figura - Exemplo de caso 4 de remoção

**Texto

Descrição gerada automaticamente**

Figura - Função auxiliar de balanceamento

Também é utilizado para auxiliar no balanceamento da árvore rubro negra após a exclusão de um produto as funções de mover um no vermelho para esquerda e para direita.

**Texto

Descrição gerada automaticamente**

Figura 17 - Implementação da função moveRedToLeft

**Uma imagem contendo objeto, relógio, display

Descrição gerada automaticamente**

Figura 18 - Exemplo de utilização da função moveRedToLeft

**Texto

Descrição gerada automaticamente**

Figura 19 - Implementação da função moveRedToRight

Uma imagem contendo objeto, relógio

Descrição gerada automaticamente

Figura 20 - Exemplo de utilização da função moveRedToRight

E para encontrar com mais facilidade o maior e menor no de uma determinada arvore, criamos duas funções que percorrerm a arvore e suas sub-árvores para encontrar o maior e menor elemento; sendo a “searchSmallest” para encontrar o menor elemento e a “searchLargest” para encontrar o maior elemento.

**Texto

Descrição gerada automaticamente**

Figura 21- Implementação da função searchSmallest

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

Figura 22- Implementação da função searchLargest

**4.1.3 Funções Auxiliares (geral)**

**Texto

Descrição gerada automaticamente**

Figura - Implementação da função createRbTree

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

Figura - Implementação da função freeNode

**Texto

Descrição gerada automaticamente**

Figura - Implementação da função freeRbTree

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

Figura - Implementação da função searchElement

**Texto

Descrição gerada automaticamente**

Figura - Implementação da função returnQuant

**Texto

Descrição gerada automaticamente**

Figura - Implementação da função getColor

**Texto

Descrição gerada automaticamente**

Figura - Implementação da função changeInfo

**4.1.4 Funções de Impressão**

**Texto

Descrição gerada automaticamente**

Figura - Implementação da função de impressão da árvore

**Texto

Descrição gerada automaticamente**

Figura - Implementação da função de impressão de um produto

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

Figura - Implementação da função de impressão de produtos disponíveis no estoque

**Texto

Descrição gerada automaticamente**

Figura - Implementação da função de impressão dos produtos cadastrados

* 1. **Arquivos**

**4.2.1 Rbt.h**

Texto

Descrição gerada automaticamente

Figura - arquivo rbt.h

* + 1. **Main.c**

Texto

Descrição gerada automaticamente

Figura - arquivo main.c

* + 1. **RBT.c**

Uma imagem contendo Texto

Descrição gerada automaticamente

Figura - arquivo rbtc.c

**5. Resultados e Discussão**

Fizemos alguns testes e algumas observações, quando usuário insere alguma String com espaço, na hora da impressão pode aparecer alguns caracteres aleatórios, mesmo tendo feito o tratamento de String no programa; também tivemos alguns problemas com o controle de Nós nulos, como por exemplo rotações com os nos nulos, rotações com Nós folhas, a solução que encontramos para isso foi verificar a existência do nó antes de qualquer operação. Por exemplo:

Texto

Descrição gerada automaticamente

Figura - Exemplo de implementação de verificação

**5.1 Exemplo 1 – Inserção Invalida**

**Texto

Descrição gerada automaticamente**

Figura - Exemplo de inserção invalida

**5.2 Exemplo 2 – Não há elementos na árvore**

**Texto

Descrição gerada automaticamente Texto

Descrição gerada automaticamente Texto

Descrição gerada automaticamente**

Figura – Exemplo de Operações com uma Árvore Vazia

**5.3 Exemplo 3 - Código digitado do produto invalido/ inexistente**

**Texto

Descrição gerada automaticamente Texto

Descrição gerada automaticamente**

Figura - Exemplo de código invalido

**5.4 Exemplo 4 – Inserção válida**

**Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente**

Figura - Exemplo de Inserção válida

**5.5 Exemplo 5 – Impressão produtos e produtos em estoque**

**Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente**

Figura - Exemplo de Impressão de produtos cadastrados e em estoque

**5.6 Exemplo 6 - Alterar quantidade de um produto**

**Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente**

Figura - Exemplo de alteração de quantidade de um produto

**5.7 Exemplo 7 – Excluir produto**

**Texto

Descrição gerada automaticamente**Texto

Descrição gerada automaticamente Texto

Descrição gerada automaticamente

Figura – Exemplo 1 de exclusão de produto

**5.8 Exemplo 8 – Excluir produto (com mais de três produtos cadastrados)**

**Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente**

Figura – Exemplo 2 de exclusão de produto

Apesar de não violar nenhuma das regras rubro negra, na exclusão de uma chave (exemplo, uma arvore com quatro chaves) o balanceamento correto de acordo com simulações que fizemos, a árvore deveria ficar com todos os nós pretos restantes, na arvore em que desenvolvemos a arvore retorna com a raiz preta e os dois filhos vermelhos já pronta para outra inserção ou remoção de chave, não violando nenhuma das regras que torna a árvore rubro negra.

**5.9 Exemplo 9 – Impressão de arvore rubro negra**

**Texto

Descrição gerada automaticamente**

Figura - Exemplo de impressão de árvore rubro negra

1. **Conclusão**

Concluiu-se que todos os objetivos foram alcançados com sucesso, o desenvolvimento das funções de cadastro de novo produto, exclusão de um produto cadastrado, atualização da quantidade de um produto no estoque e mostra-lo quando ele está disponível, listar todos os produtos cadastrados, listar todos os produtos disponíveis no estoque e a impressão da árvore Rubro-Negra, todas as funções foram implementadas sem a utilização de um ponteiro para o nó pai, nem para nenhum ancestral da árvore, sendo assim o desenvolvimento da arvore ficou todo recursivo, assim como a maioria das funções que auxiliam a construção da nossa árvore Rubro-Negra.

**7.Referências Bibliográficas**

<https://www.inf.ufsc.br/~aldo.vw/estruturas/simulador/RB.html>  (Acessado em: 25/06/2023 às 20:50)

<https://carbon.now.sh/>  (Acessado em: 05/07/2023 às 21:00)

<https://www.facom.ufu.br/~backes/gsi011/Aula12-ArvoreRB.pdf>  (Acessado em: 23/06/2023 às 22:10)

<https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/st-redblack.html#Node>  (Acessado em: 27/06/2023 às 15:57)

<http://www.ulysseso.com/livros/ed2/ApF.pdf> (Acessado em: 27/06/2023 às 16:45)

<http://wiki.foz.ifpr.edu.br/wiki/index.php/Caracteres_e_String_em_C>  (Acessado em: 01/07/2023 às 19:23)