**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 balancemento

* **Move 2 Esquerda RED**

**Texto

Descrição gerada automaticamente**

**Uma imagem contendo objeto, relógio, display

Descrição gerada automaticamente**

* O no atual -> H: é tratado como PAI/ Avô
* Recebe um no H e troca as cores dele e de seus filhos
* se o filho ESQUERDO do FILHO DIREITO é vermelho: aplica a rotação a DIREITA no filho Direito
* e uma rotação ESQUERDA no PAI
* Troca as cores do NO PAI e de seus filhos
* **Move 2 Direita RED**

Uma imagem contendo objeto, relógio

Descrição gerada automaticamente

**Texto

Descrição gerada automaticamente**

**Texto

Descrição gerada automaticamente**

**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

**4.1.3 Funções Auxiliares (geral)**

**Texto

Descrição gerada automaticamente**

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

**Texto

Descrição gerada automaticamente**

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

**Texto

Descrição gerada automaticamente**

**Texto

Descrição gerada automaticamente**

**Texto

Descrição gerada automaticamente**

**4.1.4 Funções de Impressão**

**Texto

Descrição gerada automaticamente**

**Texto

Descrição gerada automaticamente**

**Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente**

**Texto

Descrição gerada automaticamente**

**4.2 Controle de Estoque**

**Texto

Descrição gerada automaticamente**

* Cadastrar um novo produto
* Excluir produto cadastrado
* Atualizar quantidade de um produto no estoque
* Listar produtos cadastrados
* Listar produtos em estoque
* Imprimir a árvore rubro negra

galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

* Arquivos
  + Rbt.h

Texto

Descrição gerada automaticamente

* Main.c

Texto

Descrição gerada automaticamente

**5. Resultados e Discussão**

Foi escolhido três funções *(“f(x)”),* calculado a derivada *(“f ’(x)”)* de cada função*,* e escolhido dois determinados pontos *(“a”* e “*b”)* para que o gráfico pudesse ser gerado. Através destes parâmetros pode ser calculado através dos métodos de da Bisseção, Newton – Raphson e Secante as raízes, números de iterações e os erros.

**5.1 Exemplo 1**

* Função escolhida:

*f(x) = sen(2x)*

* Derivada da função:

*f'(x) = cos (2x)\*2*

* Intervalos:

*a = 1*

*b = 2*

**5.1.5 Vantagens e Desvantagens – Exemplo 1**

Com a função exemplo numero um pode se observar que o melhor método em termo de rapidez foi o método de Newton -Raphson, mas esse método não possui a precisão que o método da bisseção possui. Porem método da Bisseção possui uma desvantagem o tempo de geração do resultado, gerando assim muitas iterações. E tento como método da Secante o segundo método mais rápido.

**5.2 Exemplo 2**

* Função escolhida:

*f(x) = cos(x)*

* Derivada da função:

*f'(x) = - sen (x)*

\* Inserção

Duas funções principais:

- insertNodeRb

- insertRb

Auxiliares:

- getColor

- rotateRightRb

- rotateLeftRb

\* Exclusão

Duas principais

- removeElementRb

- removeRb

Auxiliares:

- searchElement

- getColor

- moveRedToLeft

- rotateRightRb

- rotateLeftRb

- moveRedToRight

- searchLargest

- dellBalanceNodes

\* Auxiliares no Geral

createRbTree

freeNode

freeRbTree

searchElement

returnQuant

getColor

changeColor

changeInfo

\* Impressão

printTree

printTreeHelper

printProd

printProdHelper

printProdEstoqueAux

printProdEstoque

printProdCastrados

printProdCastradosHelper

**6. Conclusão**

Concluiu-se que todos os objetivos foram alcançados com sucesso, a definição das três funções, implementações dos três métodos propostos e o cálculo das raízes das funções escolhidas. Pode se observar um padrão entre as três funções escolhidas, todas as três funções o método mais eficiente em termo de rapidez foi o método de Newton -Raphson. O método mais preciso foi o método da bisseção e o segundo método mais rápido foi o método da Secante.

**(????) Vantagens e Desvantagens**

**7.Referências Bibliográficas**

Simulador de Árvore Rubro Negra: <https://www.inf.ufsc.br/~aldo.vw/estruturas/simulador/RB.html>  (Acessado em: 25/06/2023 ás 20:50)

Code Picture/ Foto do Código:

<https://carbon.now.sh/>  (Acessado em: )

Referências sobre Árvore rubro negra

<https://www.facom.ufu.br/~backes/gsi011/Aula12-ArvoreRB.pdf>

<https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/st-redblack.html#Node>

<http://www.ulysseso.com/livros/ed2/ApF.pdf>

<http://wiki.foz.ifpr.edu.br/wiki/index.php/Caracteres_e_String_em_C>