Universidade Federal do Piauí – Campus Senador Helvídio Nunes de Barros – CSHNB Curso de Sistemas de Informação Disciplina: Programação Funcional

Professora: Juliana Oliveira de Carvalho

Relatório de Estruturas de Dados II

Autor:

Bryan Victor da Costa Martins

Picos PI, favereiro de 2023

- **Resumo:**

Esse relatório tem como objetivo demostrar a pratica e implementação Estrutura de Arvores 2-3 e para resolução de alguns problemas práticos propostos, utilizando a Linguagem de programação C.

- **Introdução:**

A Problematica envolve duas questões, onde na primeira questão vamos gerenciar blocos de memória do sistema operacional ultizando árvore 2-3, a segunda questão é uma analise de dados sobre a execução da questão anterior.

- **Seções Específicas**

Informações técnicas:

Para o desenvolvimento e testes deste projeto foi utilizado um notebook com um processador AMD Ryzen™ 7 5700U 4.3GHz, oito gigas de memória RAM ddr4, sistema operacional com a arquitetura de 64 bits, Windows 11 Pro.

Os códigos foram feitos na IDE e editor de texto Visual Studio Code e compilados pelo compilador gcc (MinGW.org GCC-6.3.0-1) 6.3.0.

Problema 1: Será citado as funções principais do código.

int ehfolha (): Função que vai retornar um interior caso o nó seja folha, função possui uma variavel flag que vai receber 1 caso o nó não seja nulo e não tiver nenhum filho, sendo assim um nó folha, caso não flag vai receber e retornar 0.

void criarNo (): Função vai receber três variaveis type(Arvore23), uma varivel que vai definir a estrutura bloco. Função vai alocar de forma dinamica um nonvo nó, inicializa as infos do nó (info1, info2), Ninfos(Quantidade de info), esq, cen, dir e retorna o ponteiro para o novo nó.

void adicionaNo (): Função vai receber duas variaveis type(Arvore23) e duas variaveis para a estrutura bloco, função adiciona uma nova info ao nó caso tenha espaço.

void quebraNo (): Função vai receber um variavel type(Arvore23) e duas variaveis para a estrutura bloco, função vai quebrar o nó quando a quantidade de informações passe de duas.

int balanceiaArvore () : Função vai receber duas variaveis type(Arvore23) e uma varivel inteira, função vai balancear a arvore caso seja necessario após remover um nó.

int remove_folha (): Função vai receber duas variaveis type(Arvore23) e uma variavel inteira, vai remover o nó somente caso ele seja um nó folha.

int remove_NoFilhos_Folha (): Função vai reeber duas variaveis type(Arvore23) e uma variavel inteira, função vai remover o nó caso seus filhos desse nó sejam nós folhas.

int remove_MenorNo (): (): Função vai reeber duas variaveis type(Arvore23) e uma variavel para estrutura bloco, função vai remover a menor info a partir do nó recebido como parâmentro.

int remover(): Funçao vai receber duas variáveis type(Arvore23) e duas variaveis inteira, função vai controlar a remoção de um bloco da

árvore, na qual chama as funções **remove_MenorNo(), remove_folha()** e **remove_NoFilhos_Folha()** e determina as condições que elas serão utilizadas.

void imprime_bloco_info(): Função vai receber um variavel para a estrutura bloco, função vai imprimir as informações contidas em um bloco.

void concatenaInfo (): Função vai receber duas variaveis type(Arvore23), função vai juntar infos dentro de um mesmo bloco, vai ocupar os espaços vazios com as novas info.

void ocupa_espaco() : Função vai receber duas variaveis type(Arvore23) e duas variaveis inteira, função vai ocupar os espaços disponiveis do bloco com valores informados.

struc b* insere(): Função do tipo arvore23, função vai receber duas variaveis type(Arvore23) e duas variaveis para a estrutura bloco,

função utilizada para inserir novos blocos na árvore.

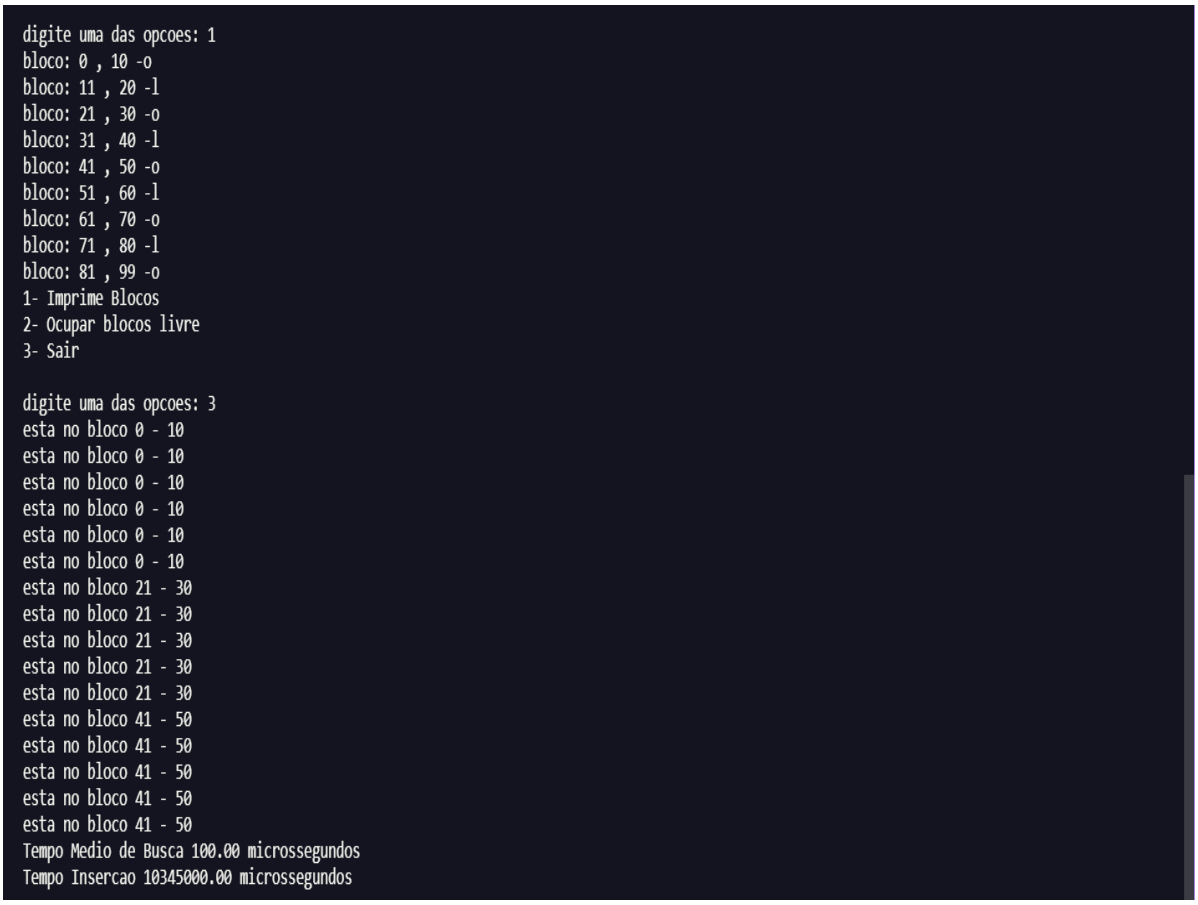
void insere_bloco_NaMemoria(): Funçao vai receber duas variaveis type(Arvore23) e uma variavel inteira, função vai receber os valores do blocos a serem inseridos, utilizada para administrar a função **insere().**

void imprimir(): Função vai receber a Raiz da arvore e imprime a arvore completa.

void limpa_buffer(): Função vai limpar os buffers de memória

void liberaNo(): Função vai receber uma variavel type(Arovre23), função libera o espaço de memória ocupado pelo nó que está sendo alterado na árvore.

EXECUÇÂO:



5. Análise de desempenho ÁRVORE 23 Questão 2

Tempo de Inserção 10345000.00 microssegundos

Tempo de Inserção máximo 9.765 microssegundos

Tempo de Busca Mínimo 0.000 microssegundos

Tempo de Busca Máximo 7.000 microssegundos

Tempo Medio de Busca 100.00 microssegundos

6 **Conclusão:**

Este trabalho teve como objetivo a aplicação dos conhecimentos adquiridos em sala de aula sobre processamento de dados, manipulação de ponteiros e estruturas de dados, com foco na árvore 2-3. Foi possível observar que, em comparação com outras estruturas de dados, a inserção na árvore 2-3 pode ser um pouco mais lenta devido à sua complexidade e necessidade de garantir propriedades específicas, como equilíbrio e ordem dos elementos. No entanto, a busca na árvore 2-3 é geralmente mais rápida, pois sua estrutura é menos profunda, o que a torna mais eficiente. Além disso, a atividade prática proporcionou uma maior compreensão sobre a estruturação da árvore 2-3, bem como a sua implementação e manipulação de seus nós e ponteiros.