Relatório: Resolução Do Trabalho 02 Da Disciplina Estruturas De Dados II

Eduardo de Sousa Gomes Vieira
E-mail: eduardosousa@ufpi.edu.br
Estruturas de Dados II - Juliana Oliveira de Carvalho

28 de julho de 2023

Resumo

O relatório, aborda sobre árvore rubro-negra, árvore 2-3 e e árvore 4-5, que são estruturas de dados utilizadas para armazenar informações de maneira eficiente, garantindo operações de busca, inserção e remoção com complexidade balanceada. As árvores rubro-negras, 2-3 e 4-5, são exemplos de árvores de busca binária balanceadas, projetadas para manter o equilíbrio da árvore durante operações de inserção e remoção de elementos.

A problemática surgiu ao se questionar qual seria a diferença de desempenho entre essas três estruturas. O problema abordado neste relatório consiste em implementar estruturas de árvores rubro-negras, 2-3 e 4-5 na linguagem C. O objetivo é criar funções que permitam a inserção, busca e remoção de elementos nessas árvores de forma eficiente, mantendo a propriedade de balanceamento.

Através da implementação das estruturas de árvores rubro-negra, 2-3 e 4-5, foi possível observar que ambas as árvores são capazes de manipular grandes volumes de dados com eficiência, mantendo a altura da árvore em valores próximos ao logaritmo da quantidade de elementos. As operações de busca, inserção e remoção foram otimizadas, resultando em tempos de execução consistentes mesmo com quantidades expressivas de dados.

Palavras-chaves: Árvores Binárias, rubro-negra, 2-3, 4-5, Estruturas de Dados.

Introdução

As estruturas de dados são elementos fundamentais no campo da ciência da computação, permitindo a organização, armazenamento e manipulação eficiente de informações. Contudo, essas estruturas adquirem significado quando associadas a um conjunto de operações (algoritmos) que visam manipulá-las (SANTOS; JÚNIOR; NETO, 2016). Neste relatório, abordaremos três tipos de árvores de busca balanceadas: as árvores rubro-negras, as árvores 2-3 e as árvores 4-5.

Árvores são estruturas de dados que representam uma relação hierárquica entre os dados que a compõem. Cada nó em uma árvore tem uma conexão com um nó superior, chamado de pai, e pode ter um ou mais nós abaixo dele, chamados de filhos. A raiz da árvore é o nó que não possui pai, e os nós subsequentes são organizados em níveis, formando assim subárvores. Os nós que compartilham o mesmo pai são chamados de irmãos, e esses irmãos, por sua vez, são filhos do mesmo nó pai. Essa estrutura de hierarquia permite uma organização eficiente dos dados e facilita a realização de operações como busca, inserção e remoção dentro da árvore (UBER; MALDONADO; COSTA,).

As árvores binárias rubro-negras, árvores 2-3 e árvores 4-5 são três tipos de árvores balanceadas essenciais para resolver problemas computacionais que requerem operações eficientes em estruturas de dados. Essas árvores garantem um desempenho logarítmico para operações fundamentais, tornando-as adequadas para gerenciar grandes conjuntos de dados. Cada uma delas apresenta particularidades que contribuem para o equilíbrio da árvore e, consequentemente, para um desempenho eficiente das operações (ALENCAR; PESSOA; PIRES, 2020).

Este relatório descreve o projeto desenvolvido no âmbito da disciplina de Estruturas de Dados II. Seu objetivo é examinar e comparar o desempenho de três tipos de estruturas de dados - rubro-negras, 2-3 e 4-5 -, bem como as estratégias de busca, inserção e remoção utilizadas. Também serão discutidas as propriedades e vantagens de cada tipo de árvore, juntamente com uma análise da complexidade das operações. O trabalho foi desenvolvido utilizando a linguagem de programação C e as bibliotecas específicas para a implementação das estruturas de dados mencionadas.

Detalhes sobre o projeto desenvolvido serão incluídos nas demais partes deste relatório. As próximas seções irão discutir os aspectos funcionais do programa, descrevendo os algoritmos utilizados, chamadas de sistema pertinentes e tecnologias empregadas. Em seguida, serão apresentados os resultados do programa, juntamente com medições e análises dos dados coletados, normalmente na forma de tabelas para facilitar o entendimento. Por fim, na seção de conclusão, serão reforçadas as informações apresentadas ao longo do relatório.

1 Hardware utilizado

Todos os testes a seguir foram feitos utilizando o mesmo hardware, com as seguintes especificações da tabela 1 abaixo:

Marca	Lenovo
Sistema Operacional	Windows 11
Processador	Intel Core i5-1035G1
Memória RAM	8GB
Tipo de memória	DDR4 2666MHz

Tabela 1 – Hardware utilizado

2 Seções Específicas

Os tópicos a seguir, serão apresentados os enunciados das questões, juntamente com suas resoluções e testes de desempenho. Isso possibilitará aos leitores compreender a

lógica por trás da solução e avaliar a eficiência do método utilizado. Foram utilizados três tipos de estruturas não lineares Árvore rubro-negra, árvore 2-3 e árvore 4-5.

2.1 Questão 01- Árvore Rubro-negra

Nessa parte da questão foi proposto uma implementação em C que construa uma árvore vermelha-preta, com todas as palavras incluídas a partir de um arquivo texto, e armazene o número de cada linha em que a palavra foi usada.

2.1.1 Estrutura Lista

A estrutura Lista representa um nó de uma lista encadeada que armazena os números de linha em que uma palavra aparece no texto. Ela possui dois campos:

- num linha: um inteiro que armazena o número da linha em que a palavra ocorre. prox: um ponteiro para o próximo nó da lista.
- prox: um ponteiro para o próximo nó da lista.

2.1.2 Estrutura Info

A estrutura Info representa as informações associadas a cada nó da árvore rubro-negra. Ela possui dois campos:

- palavra: um ponteiro para um array de caracteres (string) que armazena a palavra em si.
- num lista: um ponteiro para o início da lista encadeada de números de linha associados à palavra.

2.1.3 Estrutura arvRubroNegra

A estrutura ArvRubroNegra representa um nó da árvore rubro-negra. Ela possui quatro campos:

- info: um ponteiro para uma estrutura Info, que contém as informações da palavra e da lista de números de linha associados a ela.
- cor: um inteiro que representa a cor do nó na árvore rubro-negra (1 para vermelho e 0 para preto).
- esq: um ponteiro para o filho esquerdo do nó na árvore.

  dir: um ponteiro para o filho direito do nó na árvore.

2.1.4 Função para inserir linha

Essa função insere um número de linha em uma lista encadeada de números de linha associados a uma palavra na árvore. Ela recebe como parâmetros um ponteiro para o ponteiro da lista encadeada (no) e o número da linha (num linha) a ser inserido. A função percorre a lista até encontrar um nó vazio, onde cria um novo nó contendo o número da linha. Caso a inserção seja bem-sucedida, retorna 1; caso contrário, retorna 0.

2.1.5 Função para imprimir as linhas

Essa função imprime os números de linha associados a uma palavra em ordem crescente. Recebe como parâmetros o ponteiro para a raiz da lista de números de linha (raiz) e um contador (cont) para controle recursivo. A função percorre a lista encadeada e imprime os números de linha.

2.1.6 Função Cor

Essa função retorna a cor de um nó em uma árvore rubro-negra. Recebe como parâmetro o nó da árvore rubro-negra (raiz) e retorna a cor do nó (0 para preto e 1 para vermelho).

2.1.7 Função para criar nó

Essa função cria um novo nó a ser inserido na árvore rubro-negra. Recebe como parâmetros o ponteiro para o ponteiro da raiz da árvore rubro-negra (raiz), a palavra a ser inserida no novo nó (palavra) e o número da linha a ser associado à palavra (num linha). A função verifica se a árvore está vazia e cria um novo nó contendo as informações da palavra e do número de linha. Caso a criação seja bem-sucedida, retorna 1; caso contrário, retorna 0.

2.1.8 Função de rotação para esquerda

Essa função realiza uma rotação simples à esquerda em uma árvore rubro-negra. Recebe como parâmetro um ponteiro para o ponteiro da raiz da árvore rubro-negra (raiz). A função realiza a rotação simples à esquerda e ajusta as cores dos nós para manter as propriedades da árvore rubro-negra.

2.1.9 Função de rotação para direita

Essa função realiza uma rotação simples à direita em uma árvore rubro-negra. Recebe como parâmetro um ponteiro para o ponteiro da raiz da árvore rubro-negra (raiz). A função realiza a rotação simples à direita e ajusta as cores dos nós para manter as propriedades da árvore rubro-negra.

2.1.10 Função de troca de cor

Essa função realiza a troca de cor entre um nó e seus filhos em uma árvore rubro-negra. Recebe como parâmetro o nó da árvore rubro-negra (raiz). A função inverte a cor do nó e de seus filhos, mantendo as propriedades da árvore rubro-negra.

2.1.11 Função inserir palavra

Essa função insere uma palavra na árvore rubro-negra e associa um número de linha a ela. Recebe como parâmetros um ponteiro para o ponteiro da raiz da árvore rubro-negra (raiz), a palavra a ser inserida na árvore (palavra) e o número da linha em que a palavra está presente (num linha). A função percorre a árvore buscando a posição correta para inserir a palavra. Caso a palavra já exista na árvore, apenas o número da linha é atualizado. A função também realiza as rotações e ajusta as cores dos nós conforme necessário para manter as propriedades da árvore rubro-negra. O retorno indica o resultado da inserção (0 para palavra não inserida, 1 para palavra inserida e 2 para palavra já existente).

2.1.12 Função inserir rubro

Essa função insere uma palavra na árvore rubro-negra e associa um número de linha a ela. Recebe como parâmetros um ponteiro para o ponteiro da raiz da árvore rubro-negra (raiz), a palavra a ser inserida na árvore (palavra) e o número da linha em que a palavra está presente (num linha). A função converte a palavra para letras minúsculas e chama a função inserirPalavra para realizar a inserção na árvore. Após a inserção, a raiz da árvore é colorida de preto.

2.1.13 Função para ler o arquivo de texto

Essa função lê um arquivo de texto e processa as palavras, inserindo-as na árvore rubro-negra. Recebe como parâmetros um ponteiro para o ponteiro da raiz da árvore rubro-negra (raiz) e o nome do arquivo de texto a ser lido (nome arquivo). A função abre o arquivo, lê cada linha, separa as palavras e chama a função inserirRubro para inseri-las na árvore. O retorno indica se a leitura do arquivo foi bem-sucedida (0 para arquivo não encontrado, 1 para leitura bem-sucedida).

2.1.14 Função para imprimir em ordem

Essa função imprime as palavras da árvore rubro-negra em ordem alfabética, junto com os números das linhas em que cada palavra aparece. Recebe como parâmetros a raiz da árvore rubro-negra (raiz) e um contador para controle recursivo (cont). A função realiza um percurso em ordem na árvore e chama a função imprmrLinhs para imprimir os números de linha associados a cada palavra.

2.1.15 Função buscar palavra

Essa função busca uma palavra na árvore rubro-negra e retorna as informações do nó correspondente (palavra e lista de números de linha). Recebe como parâmetros a raiz da árvore rubro-negra (raiz), a palavra a ser buscada (palavra) e um ponteiro para o nó onde as informações serão armazenadas (No). A função percorre a árvore buscando a palavra e, se encontrá-la, cria um novo nó para armazenar as informações e retorna 1. Caso a palavra não seja encontrada, a função retorna 0.

2.1.16 Função para buscar folha

Essa função busca o maior filho à esquerda de um nó em uma árvore rubro-negra. Recebe como parâmetro o nó da árvore rubro-negra (ultimo) a partir do qual a busca será realizada. A função percorre a árvore até encontrar o maior filho à esquerda do nó (ultimo) e o retorna.

2.1.17 Função para remover palavra

Essa função remove uma palavra da árvore rubro-negra, se ela existir. Recebe como parâmetros um ponteiro para o ponteiro da raiz da árvore rubro-negra (raiz) e a palavra a ser removida (palavra). A função busca a palavra na árvore e, se encontrá-la, remove o nó correspondente. Caso o nó a ser removido tenha filhos, é encontrado o maior filho à esquerda para ocupar a posição do nó removido, mantendo as propriedades da árvore rubro-negra. O retorno indica se a palavra foi removida (0 para palavra não encontrada, 1 para palavra removida).

2.2 Questão 2 - Árvore 2-3

O problema proposto consiste em desenvolver um programa em linguagem C para criar uma estrutura de dados de referência cruzada, utilizando uma árvore 2-3. O programa deve ler um arquivo de texto contendo palavras e números de linha onde essas palavras foram utilizadas. Cada palavra será armazenada em uma árvore vermelha-preta e associada a uma lista encadeada que conterá os números das linhas onde a palavra foi utilizada.

2.2.1 Estrutura Lista

Estrutura que representa uma lista encadeada de números de linha associados a uma palavra na árvore:

- num linha: um inteiro que armazena o número da linha.
- prox: um ponteiro para o próximo nó da lista, formando assim uma lista encadeada.

2.2.2 Estrutura Info

Estrutura que armazena as informações associadas a uma palavra na árvore:

- palavra: um ponteiro para uma string (char) que armazena a palavra.
- num lista: um ponteiro para o início da lista encadeada de números de linha (Lista) associados à palavra.

2.2.3 Estrutura arvRubroNegra

Estrutura que representa um nó da árvore 2-3:

- info1: um ponteiro para a estrutura Info, que contém a primeira informação associada ao nó.
- info2: um ponteiro para a estrutura Info, que contém a segunda informação associada ao nó (ou NULL, caso o nó tenha apenas uma informação).
- numInfo: um inteiro que indica a quantidade de informações presentes no nó (1 ou 2).
- esq: um ponteiro para o filho esquerdo do nó (outro nó da árvore).
- centro: um ponteiro para o filho central do nó (outro nó da árvore, ou NULL, caso o nó tenha apenas uma informação).
- dir: um ponteiro para o filho direito do nó (outro nó da árvore, ou NULL, caso o nó tenha apenas uma informação).

2.2.4 Função criar info

Esta função cria uma nova estrutura de informação (Info) contendo uma palavra representada por uma string e uma lista de números de linha em que a palavra aparece. Ela recebe três parâmetros: palavra (string que representa a palavra a ser armazenada na estrutura Info), lista (um ponteiro para uma lista encadeada que contém números de linha em que a palavra aparece) e num linha (o número da linha a ser inserido na lista, caso a lista seja nula).

2.2.5 Função para criar nó

Essa função cria um novo nó para a árvore 2-3, que contém uma única informação (Info), apontada pelo ponteiro info. O novo nó pode ter até dois filhos, representados pelos ponteiros noEsq, noCentro e noDir. Ela recebe quatro parâmetros: info (um ponteiro para a informação que será armazenada no novo nó), noEsq (um ponteiro para o nó filho esquerdo do novo nó), noCentro (um ponteiro para o nó filho central do novo nó) e noDir (um ponteiro para o nó filho direito do novo nó).

2.2.6 Função adiciona nó

Esta função adiciona uma nova informação (Info) a um nó da árvore 2-3 e atualiza seus ponteiros de filhos. A função compara a ordem lexicográfica entre a nova informação e a informação existente no nó para determinar a posição correta de inserção e atualizar os ponteiros de filhos do nó. A nova informação é adicionada ao nó, e o nó filho direito é atualizado com o ponteiro novo. Os parâmetros dessa função são: raiz (um ponteiro para o ponteiro da raiz da árvore 2-3), info (um ponteiro para a nova informação a ser adicionada no nó) e novo (um ponteiro para o nó que será o filho direito do nó após a inserção).

2.2.7 Função para verificar se é folha

Essa função verifica se um nó específico da árvore 2-3 é uma folha, ou seja, se ele não possui filhos. Ela retorna um valor inteiro que indica se o nó é uma folha ou não. Se o nó for uma folha, a função retorna 1; caso contrário, retorna 0. O parâmetro passado é raiz (um ponteiro para o nó que será verificado).

2.2.8 Função inserir linha

Essa função insere um número de linha em uma lista encadeada, representada por um ponteiro para o ponteiro da lista encadeada. Se a lista estiver vazia (ponteiro para a lista for NULL), a função criará um novo nó para armazenar o número de linha fornecido. Caso contrário, a função chamará recursivamente a si mesma para inserir o número de linha no próximo nó da lista. Os parâmetros dessa função são: no (um ponteiro para o ponteiro da lista encadeada) e num linha (o número da linha a ser inserido na lista).

2.2.9 Função para quebrar nó

Esta função divide um nó da árvore 2-3 em dois nós e retorna o nó criado com as informações maiores. Ela recebe cinco parâmetros: raiz (um ponteiro para o ponteiro da raiz da árvore 2-3), filho (um ponteiro para o nó filho direito do nó sendo dividido), info (um ponteiro para a nova informação a ser inserida na árvore), infoSobe (um ponteiro para um ponteiro que armazenará a informação que será propagada para cima na árvore após a divisão).

2.2.10 Função para inserir palavra

Essa função insere uma palavra em uma árvore 2-3, representada por um ponteiro para a raiz da árvore. Ela realiza a inserção de acordo com as regras da árvore 2-3 e é chamada recursivamente para percorrer a árvore e inserir a palavra corretamente. A função recebe seis parâmetros: raiz (um ponteiro para o ponteiro da raiz da árvore 2-3), palavra (a palavra a ser inserida na árvore), num linha (o número da linha onde a palavra ocorre

no arquivo), pai (um ponteiro para o nó pai do nó atual. NULL caso o nó atual seja a raiz), infoSobe (um ponteiro para o ponteiro de informação que será propagada para cima em caso de divisão de nós) e inseriu (um ponteiro para um inteiro que indica o resultado da inserção).

2.2.11 Função para ler o arquivo texto

Esta função lê um arquivo de texto e insere cada palavra em uma árvore 2-3, representada por um ponteiro para a raiz da árvore. O arquivo é lido linha por linha, e cada linha é tokenizada para obter as palavras individuais. As palavras são convertidas para letras minúsculas antes da inserção na árvore. A função recebe dois parâmetros: raiz (um ponteiro para o ponteiro da raiz da árvore 2-3) e nome arquivo (o nome do arquivo de texto a ser lido).

2.2.12 Função para buscar palavra

Essa função busca uma palavra na árvore 2-3 e imprime suas informações caso a palavra seja encontrada. A função utiliza uma abordagem recursiva para percorrer a árvore e comparar as palavras nos nós com a palavra buscada. Se a palavra for encontrada em um nó da árvore, a função imprime a palavra e a lista de números de linha em que ela aparece. O resultado da busca é armazenado no ponteiro resultado. Se a palavra for encontrada, o valor apontado por resultado é alterado para 1. A função recebe três parâmetros: raiz (um ponteiro para o nó raiz da árvore 2-3), palavra (a palavra a ser buscada na árvore) e resultado (um ponteiro para um inteiro que armazena o resultado da busca).

2.2.13 Função para imprimir em ordem

Esta função imprime os nós de uma árvore 2-3 em ordem. Cada nó da árvore representa uma palavra e seus números de linha associados. A função utiliza uma abordagem recursiva para percorrer a árvore em ordem, imprimindo as palavras e suas respectivas listas de números de linha. Ela recebe dois parâmetros: raiz (um ponteiro para o nó raiz da árvore 2-3) e cont (um contador que auxilia no controle da recursão e formatação da impressão).

2.2.14 Função para imprimir linhas

Esta função imprime os números de linha de uma lista encadeada. A lista contém os números de linha em que uma palavra aparece. A função utiliza uma abordagem recursiva para percorrer a lista encadeada e imprimir os números de linha em ordem. Ela recebe dois parâmetros: no (um ponteiro para o nó da lista encadeada) e cont (um contador que auxilia no controle da recursão e formatação da impressão).

2.2.15 Função para remover linha

Esta função é responsável por remover um elemento específico de uma lista encadeada. Ela recebe um ponteiro para o ponteiro da lista e o valor do elemento a ser removido. A função percorre a lista até encontrar o elemento a ser removido. Se o elemento for encontrado, ele é removido da lista e a memória é liberada.

2.2.16 Função Maior Info Remove

Esta função encontra o maior valor menor ou igual à palavra na subárvore esquerda e o substitui. Ela é utilizada durante a remoção de uma palavra na árvore 2-3 quando a palavra a ser removida não é uma folha. A função percorre a subárvore esquerda e encontra o maior valor menor ou igual à palavra. Em seguida, realiza a troca das palavras e chama a função removerArv23 para remover o valor encontrado. A função redistribuiArv também é chamada para realizar a redistribuição dos nós após a remoção.

2.2.17 Função para redistribui nós

Esta função redistribui os nós de uma árvore 2-3 após a remoção de uma palavra. A redistribuição é realizada para manter as propriedades da árvore 2-3 após a remoção. A função percorre a árvore 2-3 e realiza as redistribuições de acordo com os casos específicos de remoção.

2.2.18 Função de remover palavra

Esta função remove uma palavra e seu número de linha associado da árvore 2-3. A função utiliza uma abordagem recursiva para percorrer a árvore em busca da palavra a ser removida. Caso a palavra seja encontrada, a função remove o número de linha da lista associada a essa palavra. Se a lista de números de linha ficar vazia após a remoção, a função realiza a remoção da palavra propriamente dita da árvore. A remoção de uma palavra pode causar uma redistribuição dos nós da árvore. O parâmetro "pai"é um ponteiro para o ponteiro do pai do nó atual. Ele é usado para realizar a redistribuição dos nós após a remoção.

2.3 Questão 3 - Árvore 4-5

O problema proposto envolve o desenvolvimento de um programa em linguagem C, utilizando uma árvore 4-5, para uma loja de calçados. As informações de todos os sapatos da loja serão armazenadas em um arquivo chamado "calçados". Cada calçado possui as seguintes informações: código, tipo, marca, tamanho, quantidade e preço.

2.3.1 Função para ler o arquivo texto

Lê um arquivo de texto contendo informações de calçados e insere os dados em uma árvore 4-5. O arquivo é lido linha por linha, e cada linha é tokenizada para obter as informações do calçado. As palavras-chave são convertidas para letras minúsculas antes da inserção na árvore. Recebe dois parâmetros: um ponteiro para o ponteiro da raiz da árvore 4-5 e um ponteiro para a informação a ser inserida na árvore em caso de subida.

2.3.2 Função para criar info

Cria uma nova informação (Info) com os dados de um calçado. Aloca memória para uma nova estrutura de informação e inicializa seus campos com os dados fornecidos. Retorna um ponteiro para a nova estrutura Info.

2.3.3 Função para criar nó

Cria um novo nó para a árvore 4-5. Aloca memória para um novo nó e inicializa seus campos. O novo nó pode conter uma ou duas informações, dependendo do nó pai. Retorna um ponteiro para o novo nó criado.

2.3.4 Função para verificar se é folha

Verifica se um nó da árvore 4-5 é uma folha. Retorna 1 se o nó é uma folha (não possui filhos) ou 0 caso contrário.

2.3.5 Função adiciona nó

Esta função adiciona uma nova informação a um nó da árvore 4-5 e atualiza seus ponteiros de filhos. A função verifica a ordem do código da informação para determinar a posição correta de inserção e atualiza os ponteiros de filhos do nó. Dependendo da quantidade de informações no nó, ele pode conter uma ou duas informações. A função recebe como parâmetros um ponteiro para o ponteiro do nó que será modificado pela função, a nova informação a ser adicionada e um ponteiro para o nó filho que será o filho do nó após a inserção.

2.3.6 Função quebra nó

Esta função é chamada quando um nó da árvore 4-5 está cheio (contém quatro informações) e precisa ser quebrado para manter a árvore balanceada. Ela recebe um nó, uma nova informação, um ponteiro para a informação que será "subida"para o nó pai e um nó filho que será o filho direito do novo nó após a quebra. A função realiza a quebra do nó em dois novos nós, mantendo a ordem dos códigos das informações, e retorna o maior nó criado após a quebra.

2.3.7 Função de inserir na 4-5

Esta função insere uma nova informação na árvore 4-5 e realiza as devidas quebras e atualizações nos nós conforme necessário para manter a árvore balanceada. Se a árvore estiver vazia, a função cria um novo nó raiz com a nova informação. Caso contrário, ela navega pela árvore para encontrar a posição correta para a inserção e realiza as quebras conforme necessário. A função também recebe ponteiros para o nó pai do nó atual sendo analisado (NULL se o nó atual for a raiz) e um ponteiro para a variável "flag"que indica se a inserção foi realizada com sucesso.

2.3.8 Função para imprimir informações

Imprime as informações de um calçado contidas na estrutura Info. Recebe um ponteiro para a estrutura Info do calçado e imprime seus campos na saída padrão.

2.3.9 Função para mostrar informações

Mostra as informações contidas em todos os nós da árvore 4-5. Recebe um ponteiro para a raiz da árvore 4-5 e um nível, e imprime as informações contidas em todos os nós da árvore. A função percorre a árvore em ordem, imprimindo as informações em cada nível da árvore.

2.3.10 Função buscar produto

Busca um produto na árvore 4-5 pelo código. Recebe um ponteiro para a raiz da árvore 4-5 e o código do produto a ser buscado. Retorna um ponteiro para a estrutura Info do produto encontrado ou NULL se o produto não for encontrado. A função realiza uma busca recursiva pela árvore, comparando o código do produto com os códigos das informações em cada nó.

2.3.11 Função para atualizar arquivo

Atualiza as informações de um produto no arquivo. Recebe um ponteiro para a estrutura Info do produto a ser atualizado. A função abre o arquivo "lojaCalcados.txt", localiza a linha correspondente ao produto pelo número de linha armazenado na estrutura Info e sobrescreve os dados do produto no arquivo.

2.3.12 Função para vender

Realiza a venda de um produto da árvore 4-5. Recebe um ponteiro para a raiz da árvore, a quantidade desejada para venda, o código do produto a ser vendido e um ponteiro para uma string "resultado". O resultado da venda é armazenado na string "resultado"para ser exibido posteriormente. A função busca o produto na árvore utilizando a função "buscarProduto"e atualiza a quantidade do produto após a venda utilizando a função "atualizarArquivo".

2.3.13 Função para repor produtos

Repõe o estoque de um produto da árvore 4-5. Recebe um ponteiro para a raiz da árvore. A função exibe as informações de todos os produtos da árvore utilizando a função "mostrarInfos". Em seguida, o usuário seleciona um produto pelo código e insere a quantidade a ser adicionada ao estoque. A função busca o produto na árvore utilizando a função "buscarProduto"e atualiza a quantidade do produto após a reposição utilizando a função "atualizarArquivo".

2.3.14 Função quantidade de linhas

Obtém a quantidade de linhas em um arquivo. Abre o arquivo "lojaCalcados.txt"e conta a quantidade de linhas presentes no arquivo. A função retorna o número total de linhas contadas no arquivo.

2.3.15 Função adicionar calçados

Adiciona novos calçados ao estoque na árvore 4-5 e no arquivo. Recebe um ponteiro para a raiz da árvore, um ponteiro para a informação que será "subida"para o nó pai em caso de quebra, além dos dados do novo calçado. A função adiciona o novo calçado à árvore utilizando a função "inserir45"e também o insere no arquivo "lojaCalcados.txt". O resultado da inserção é armazenado em uma string "resultado"para ser exibido posteriormente.

3 Resultados da Execução do Programa

O projeto em questão foi desenvolvido utilizando três tipos diferentes de árvores: árvore rubro-negra, árvore 2-3 e árvore 4-5. Essa etapa culminou em um ponto crucial de

nosso trabalho: a realização de testes para avaliar os resultados dos tempos de busca e o caminho para encontrar uma palavra em cada uma das árvores. Na árvore 4-5, o caminho escolhido foi buscar um item específico (no caso, "calcado").

Os testes foram elaborados da seguinte forma: no primeiro teste, a palavra "all"foi buscada nas árvores rubro-negra e 2-3; no segundo teste, a palavra "people"foi o alvo da busca; e no terceiro teste, a palavra "santos"foi procurada. Essas buscas foram repetidas 30 vezes, e ao final, calculou-se a média de tempo para cada caso de busca nas árvores rubro-negra e 2-3. No experimento da árvore 4-5, foram realizadas 30 buscas de itens na árvore, e também foi registrado o número de nós percorridos para encontrar cada informação.

Os dados coletados estão disponíveis nas tabelas e no gráfico a seguir, proporcionando uma visão clara do desempenho das duas árvores em diferentes cenários de busca, assim como a quantidade de nós percorridos nesses processos. Para fornecer uma visão mais detalhada das médias exatas obtidas nos testes, elaboramos tabelas complementares.

A Tabela 2 mostra os tempos dos experimento buscando a palavra all nas árvores rubro-negra e 2-3 e a sua quantidade de nós para busca.

A Tabela 3 mostra os tempos dos experimento buscando a palavra people nas árvores rubro-negra e 2-3 e a sua quantidade de nós para busca.

A Tabela 4 mostra os tempos dos experimento buscando a palavra santos nas árvores rubro-negra e 2-3 e a sua quantidade de nós para busca.

A Tabela 5 mostra os tempos dos experimento buscando 30 itens na árvore 4-5 e a sua quantidade de nós para busca.

Árvore	Média de tempo da busca	Caminho para busca
rubro-negra	0.01370 ms	13 nós
2-3	0.00260 ms	08 nós

Tabela 2 – Experimento buscando a palavra all.

Árvore	Média de tempo da busca	Caminho para busca
rubro-negra	0.01000 ms	42 nós
2-3	0.00260 ms	13 nós

Tabela 3 – Experimento buscando a palavra people.

Árvore	Média de tempo da busca	Caminho para busca
rubro-negra	0.01340 ms	52 nós
2-3	0.00300 ms	09 nós

Tabela 4 – Experimento buscando a palavra santos

O gráfico da Figura 1 mostra que o tempo médio de busca nas árvores rubro negra e 2-3.

Essas informações são essenciais para entender o comportamento e eficiência das árvores, permitindo identificar a melhor opção para cada cenário de busca específico.

Código do item	Tempo da busca (ms)	Caminho para busca (nós)
05	0.00110	05
10	0.00290	05
15	0.00820	04
20	0.00250	05
25	0.00110	05
30	0.00110	05
35	0.00180	05
40	0.00090	05
45	0.00190	03
50	0.00150	05
55	0.00190	05
60	0.00200	04
65	0.00200	05
70	0.00190	05
75	0.01340	05
80	0.00100	05
85	0.00220	05
90	0.00220	03
95	0.00260	05
100	0.00230	05
105	0.00120	04
110	0.00120	05
115	0.00240	05
120	0.00180	04
125	0.00220	05
130	0.00130	05
135	0.01040	02
140	0.00220	05
145	0.00180	05
150	0.00190	04

Tabela 5 – Experimento buscando 30 itens(calçados).

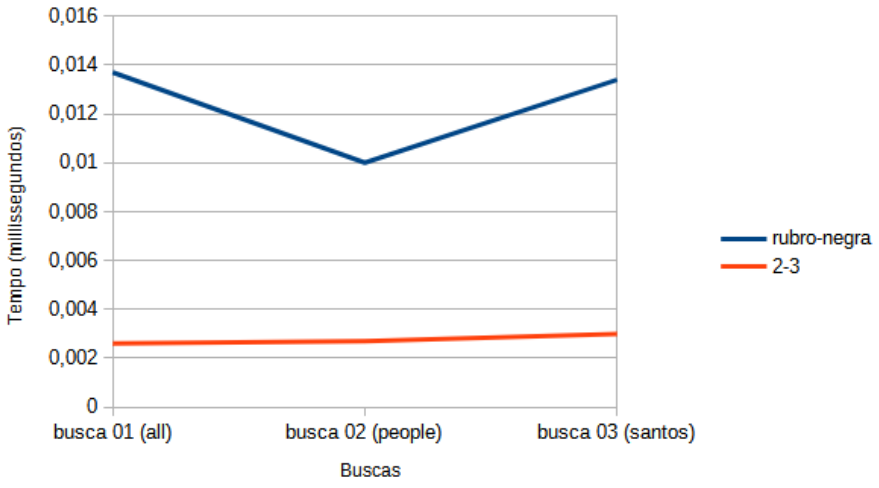

Figura 1 – Médias de busca nas árvores rubro-negra e 2-3.

4 Conclusão

Em resumo, o objetivo deste projeto foi analisar e comparar o desempenho das árvores rubro-negra, 2-3 e 4-5 na implementação de programas. Os resultados obtidos revelaram diferenças significativas entre essas três estruturas de árvore em relação às operações de busca e à quantidade de nós percorridos em cada busca.

Ao avaliar os testes de busca, ficou claro que a árvore 2-3 apresenta tempos de busca superiores em relação à árvore rubro-negra. Além disso, a quantidade de nós percorridos na árvore 2-3 demonstrou maior eficiência em comparação com a árvore rubro-negra, que apresentou lentidão devido ao caminho de busca ser muito extenso nos testes realizados.

Por outro lado, os testes de busca na árvore 4-5 mostraram-se altamente eficientes, tanto em termos de tempo de busca quanto no número de nós percorridos para encontrar cada item. Observou-se que o menor caminho em uma busca foi de apenas 2 nós e o maior caminho foi de 5 nós, evidenciando uma excelente eficiência de tempo, tornando a árvore 4-5 uma opção de busca altamente recomendada.

Em conclusão, este projeto proporcionou informações perspicazes sobre as diferenças de desempenho entre as árvores rubro-negras, 2-3 e 4-5. Esses resultados são valiosos para orientar decisões informadas na escolha das estruturas de dados, considerando as demandas e restrições específicas de cada cenário. Com base nessas análises, pode-se escolher a árvore mais adequada para atender aos requisitos específicos de desempenho e eficiência do programa em questão.

Referências

ALENCAR, L.; PESSOA, M.; PIRES, F. Um jogo educacional para exercitar propriedades de árvores binárias de busca. In: SBC. *Anais dos Workshops do IX Congresso Brasileiro de Informática na Educação*. [S.l.], 2020. p. 226–231. Citado na página 2.

SANTOS, M.; JÚNIOR, G. N. d. S.; NETO, O. P. d. S. Estruturas de dados. 2016. Citado na página 1.

UBER, F. R.; MALDONADO, Y.; COSTA, G. da. Árvores. Citado na página 2.