Universidade Federal do Rio Grande do Norte
Instituto Métropole Digital
Bacharelado em Tecnologia da Informação

Implementação da estrutura de dados: Árvore Binária de Busca

Carlos Eduardo dos Santos Silva Tertuliano
Luís Eduardo de Oliveira Sales
Luiz Sergio Medeiros Maia

Natal-RN
Maio 25, 2023

1 Relatório

Inicialmente foi criado a função para leitura do arquivo de configuração, depois foi criado a estrutura básica da árvore com chave, nó esquerdo e nó direito. Em seguida, superamos algumas dificuldades técnicas específicas da linguagem usada (c++) com ponteiros, sendo superado após pesquisas por referências. Na próxima etapa implementamos as funções primárias de **busca**, **inserção** e **remoção**, sendo a etapa de inserção independente da busca. Dividimos as funções restantes de forma igual entre os membros do grupo, com prazos para finalizar. Luiz Sergio ficou com a parte de **Enésimo elemento** e **Posição**. Carlos ficou com **imprimir árvore** e **pré-ordem**. Luís Eduardo ficou com **Média** e **Mediana**. As funções **ehCheia** e **ehCompleta** foram feitas em grupo. Tentamos inicialmente ler os arquivos de entrada diretamente do terminal, porém após alguns problemas, resolvemos abrir os arquivos de forma separada dentro do código.

A função **mediana** utiliza o percurso de ordem simétrica para ir atualizando uma variável chamada *count* e saber qual posição ele atualmente se encontra. Quando a posição chegar no elemento do meio ele retorna seu valor.

Para as funções de gerar a **média** também foi utilizado uma técnica de *overload* de funções. A primeira das funções desenvolvidas recebe os parâmetros do nó da raiz da árvore e a chave específica ao qual você quer tirar a média da subárvore dessa chave. Dentro dessa função é primeiro criado uma variável auxiliar para receber o valor da média depois de feito os cálculos, em seguida uma verificação do nó recebido para garantir que não aponta para o nada, em caso de apontar a função retorna um valor de falha na busca da média. Em caso de falha ela transforma o nó do parâmetro no nó da chave por meio de uma busca na árvore, logo ela chama a segunda função que receberá o nó específico que iniciará o cálculo de média e a variável auxiliar *average* para receber a soma de todos os elementos da subárvore. Em seguida a segunda função verifica se o nó acessado aponta para o nada e retorna 0 em caso dessa condição ser verdadeira. Após isso a variável auxiliar *average* somará seu valor a chave do nó que está sendo acessado e em seguida chama recursivamente a segunda função em formato de pré-ordem sendo primeiramente a

filha à esquerda do nó acessado em seguida a filha à direita do nó acessado. Após feito todos os acessos retorna-se o valor da variável auxiliar "media". Por fim a primeira função recebe essa soma e retorna esse valor recebido dividido pela quantidade de elementos na subárvore da chave usada na primeira chamada da primeira função por meio do cálculo media/(node->numberOfChildrenLeft + node->numberOfChildrenRight + 1).

A função **pré-ordem** foi simples de implementar, pois a vimos em sala de aula. Foi necessário fazer apenas algumas alterações para realizar a concatenação dos nodes em uma única *string*. A função **imprime Árvore**, assim como solicitado, foi dividida em dois formatos. No formato 1, utilizamos os níveis em que cada nó estava para fazer a indentação e para adicionar os "-" após imprimir o valor da chave, utilizamos um valor constante subtraído pela quantidade de espaços colocado no início e a quantidade de dígitos que cada valor armazenado possui, esse valor constante precisa ser alterado caso a árvore exceda um determinado número de níveis, isso poderia ter sido evitado usando uma função para calcular a quantidade total de nós na árvore, porém a cada alteração na quantidade de nós teríamos uma quantidade de "-" diferentes e para manter um padrão preferimos utilizar de uma constante. Essa função (**imprimeFormato1**) foi feita recursivamente, utilizando o percurso de pré-ordem para imprimir da forma como a árvore é armazenada. O formato 2, também foi feito de forma recursiva e utilizando o percurso de pré-ordem, abrindo parênteses a cada nova chamada da função e fechando os parênteses quando a função chegasse ao fim.

A função **enésimo elemento** igualmente a **mediana** utiliza o percurso de ordem simétrica para ir atualizando uma variável chamada *count* e saber qual posição ele atualmente se encontra. Quando a posição chegar no elemento procurado retorna seu valor.

A função **posicao** igualmente a **mediana** e **enésimo elemento** utiliza o percurso de ordem simétrica para ir atualizando uma variável chamada *count* e saber qual posição ele atualmente se encontra. Quando a posição chegar no elemento procurado retorna seu valor.

A função **ehCheia** calcula a altura previamente para saber se os nós nulos que encontrar se encontram no último nível, caso não se encontrem retorne falso;

A função **ehCompleta** funciona de maneira quase idêntica a função ehCheia, com a diferença que ele retorna sempre verdadeiro no penúltimo nível, pois não faz diferença se o nó for nulo ou não nulo no penúltimo nível.

Análises assintóticas:

insert - depende da altura, N no pior caso e log N se estiver balanceada;

search - depende da altura, N no pior caso e log N se estiver balanceada;

getNode - depende da altura, N no pior caso e log N se estiver balanceada;

findPredecessor - depende da altura, N no pior caso e log N se estiver balanceada;

removeKey - depende da altura, N no pior caso e log N se estiver balanceada;

countChildren - N, onde N é o número de elementos da árvore;

enesimoElemento - N, onde N é o número de elementos da árvore;

posicao - N, onde N é o número de elementos da árvore;

getMedian - depende da altura, N no pior caso e log N se estiver balanceada;

getAverage - depende da altura, N no pior caso e log N se estiver balanceada;

pre_ordem - N, onde N é o número de elementos da árvore;

getHeight - N, onde N é o número de elementos da árvore;

calculaNivel - N, onde N é o número de elementos da árvore;

imprimeFormato - N, onde N é o número de elementos da árvore (porém as constantes delas podem ser altas dependendo do valor de N);

imprimeArvore - N, onde N é o número de elementos da árvore (semelhante a imprimeFormato);

ehCompleta - N, onde N é o número de elementos da árvore (pior caso é quando ela é completa, que tem que percorrer toda a árvore);

ehCheia - N, onde N é o número de elementos da árvore (semelhante a ehCompleta);

calcQtdDigitos - log N, onde N é o número armazenado no nó.