UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
Bacharelado em Tecnologia da Informação

JOSE BEN HUR NASCIMENTO DE OLIVEIRA

Disciplina: Estruturas de dados básicos II

NATAL
2023

1
A árvore de busca binária possui algumas vantagens e motivos para sua utilização:
Eficiência de busca: A árvore de busca binária permite uma busca rápida e eficiente de elementos. A complexidade de busca é O(log n) no caso médio, o que significa que o tempo de busca aumenta de forma logarítmica com o número de elementos na árvore. Isso é muito mais eficiente do que uma busca linear em uma lista não ordenada, que teria uma complexidade de O(n).
Ordenação automática: A árvore de busca binária mantém os elementos ordenados automaticamente. Cada elemento é inserido em sua posição correta na árvore, garantindo que a estrutura esteja sempre ordenada. Isso é particularmente útil quando se trabalha com grandes conjuntos de dados ou quando se precisa manter uma sequência ordenada de elementos.
Inserção e remoção eficientes: Embora a complexidade de inserção e remoção no pior caso possa ser O(n), no caso médio a complexidade é O(log n), tornando essas operações computacionalmente eficientes. Além disso, a árvore de busca binária permite uma manipulação flexível dos elementos, possibilitando a adição e remoção de elementos de forma fácil e eficiente.
A escolha pela forma como foi feita por mim se baseia nas aulas da disciplina e nos materiais da disciplina, livros e slides.

2
Inserir (insert): A complexidade média e no caso médio é O(log n), onde n é o número de elementos na árvore. No pior caso, quando a árvore está desbalanceada, a complexidade é O(n).
Remover (remove): A complexidade média e no caso médio é O(log n), onde n é o número de elementos na árvore. No pior caso, quando a árvore está desbalanceada, a complexidade é O(n).
Busca (search): A complexidade média e no caso médio é O(log n), onde n é o número de elementos na árvore. No pior caso, quando a árvore está desbalanceada, a complexidade é O(n).
Média (calculate_average): A complexidade é O(n), onde n é o número de elementos na árvore. Isso ocorre porque é necessário percorrer todos os nós da árvore para calcular a soma total.
Mediana (calculate_median): A complexidade é O(n log n), onde n é o número de elementos na árvore. Isso ocorre porque é necessário realizar uma travessia em ordem para coletar todos os valores e, em seguida, ordená-los para encontrar a mediana.
Posição (position): A complexidade média e no caso médio é O(log n), onde n é o número de elementos na árvore. No pior caso, quando a árvore está desbalanceada, a complexidade é O(n).
Enésimo elemento (enesimoElemento): A complexidade média e no caso médio é O(log n), onde n é o número de elementos na árvore. No pior caso, quando a árvore está desbalanceada, a complexidade é O(n).
Imprimir (imprimir): A complexidade é O(n), onde n é o número de elementos na

árvore. Isso ocorre porque é necessário percorrer todos os nós da árvore para imprimir seus valores.
Cheia (isFull): A complexidade é O(n), onde n é o número de elementos na árvore. Isso ocorre porque é necessário percorrer todos os nós da árvore para verificar se todos eles têm dois filhos ou nenhum.
Completa (isComplete): A complexidade é O(n), onde n é o número de elementos na árvore. Isso ocorre porque é necessário percorrer todos os nós da árvore para verificar se a árvore está completa.