SCHOOL OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

DATA STRUCTURE AND ALGORITHMS

COMP 225 CAT 2

NAME: SHIKUKU EMMANUEL NABWANA
ADM: COM/025/21
PHONE: +254757321358
EMAIL: enshikuku39@gmail.com

E.N Shikuku
COM/025/21

SHIKUKU EMMANUEL NABWANA COM/025/21

1. What is AVL trees?

AVL trees are self-balancing binary search trees in which the heights of the left and right subtrees of every node differ by at most one.

2. When do we say a tree is height balanced?

a tree is said to be height-balanced when the heights of its left and right subtrees differ by at most one, and both the left and right subtrees are also height-balanced.

3. Write an algorithm for the construction of an area tree.

Inputs: A set of rectangles R = {r1, r2, ..., rn}

Outputs: An area tree T representing the arrangement of the rectangles in R.

Algorithm:

I. Initialize T to be an empty area tree.

II. Sort the rectangles in R by their x-coordinates in ascending order.

III. For each rectangle r in R: a. Insert r into T, starting at the root node. b. Traverse the tree to find the node that corresponds to the region covered by r. c. Split any intersected nodes along the vertical line passing through r's left edge and insert any newly created nodes into T.

IV. Sort the rectangles in R by their y-coordinates in ascending order.

V. For each rectangle r in R: a. Traverse the tree to find the node that corresponds to the region covered by r. b. Split any intersected nodes along the horizontal line passing through r's bottom edge and insert any newly created nodes into T.

VI. Return the constructed area tree T.

4. Rules for deciding the type of rotation used in AVL trees

I. If a node's left subtree is taller by two or more levels than its right subtree, and the left child's left subtree is taller than its right subtree (Left-Left case), a right rotation is performed on the node.

II. If a node's left subtree is taller by two or more levels than its right subtree, and the left child's right subtree is taller than its left subtree (Left-Right case), a left rotation is performed on the left child, followed by a right rotation on the node.

III. If a node's right subtree is taller by two or more levels than its left subtree, and the right child's right subtree is taller than its left subtree (Right-Right case), a left rotation is performed on the node.

IV. If a node's right subtree is taller by two or more levels than its left subtree, and the right child's left subtree is taller than its right subtree (Right-Left case), a right rotation is performed on the right child, followed by a left rotation on the node.

After performing the rotation, the heights of the affected nodes are updated, and the new subtree is checked for balance. If the balance is still not correct, additional rotations may be needed to restore the balance.

5. What is sorting technique?

Sorting technique refers to the process of arranging items in a collection or dataset in a specific order based on some criterion, such as numerical or alphabetical order.

6. Factors considered in choosing a sorting technique

I. Time Complexity: This refers to the amount of time it takes for the algorithm to sort the data, and it is usually expressed in terms of the number of data elements. Sorting algorithms with faster time complexity are preferred over slower ones.
II. Space Complexity: This refers to the amount of memory or storage space required by the algorithm to perform the sorting. Sorting algorithms with lower space complexity are preferred over those with higher space complexity.
III. Stability: This refers to the ability of the algorithm to maintain the relative order of equal elements in the input data. Stable sorting algorithms are preferred in situations where preserving the relative order of equal elements is important.
IV. Adaptability: This refers to the ability of the algorithm to perform efficiently when the input data is partially sorted or nearly sorted. Adaptive sorting algorithms are preferred when the input data is likely to be partially sorted.
V. Nature of the data: The type and size of the data being sorted can influence the choice of sorting algorithm. For example, some algorithms are better suited for sorting large datasets, while others are more efficient for sorting small datasets.
VI. Implementation ease: The ease of implementing the sorting algorithm can also be a factor in choosing a sorting technique, especially when the algorithm needs to be customized or modified to suit specific requirements.

7. Describe the different sorting techniques

I. Bubble Sort: This sorting technique works by repeatedly swapping adjacent elements if they are in the wrong order, until the entire array is sorted.
II. Selection Sort: This sorting technique works by finding the minimum element from the unsorted portion of the array and swapping it with the first element of the unsorted portion, until the entire array is sorted.
III. Insertion Sort: This sorting technique works by iterating through the array and inserting each element into its correct position in a sorted subarray, until the entire array is sorted.
IV. Quick Sort: This sorting technique works by selecting a pivot element from the array and partitioning the array into two subarrays based on the pivot, recursively sorting each subarray until the entire array is sorted.
V. Merge Sort: This sorting technique works by recursively dividing the array into halves, sorting each half, and then merging the sorted halves together to produce a sorted array.
VI. Heap Sort: This sorting technique works by building a heap from the array and repeatedly extracting the maximum element from the heap and placing it at the end of the array, until the entire array is sorted.
VII. Counting Sort: This sorting technique works by counting the number of occurrences of each element in the array, and using this information to place each element in its correct position in a sorted output array.
VIII. Radix Sort: This sorting technique works by sorting the elements of the array based on their digits, starting with the least significant digit and moving towards the most significant digit, until the entire array is sorted.