USN: 1RV24MC032

GB Pandurangа

Topic: MinMax (Searching & Sorting)

Problem Definition: Given an array of integers, find the minimum & maximum elements using:

① Searching (Iterative approach)

② Sorting (Using a sorting algorithm)

① Using Searching (Iterative approach)

Algorithm:

ⓐ Initialize min & max to the first element of the array.

ⓑ Traverse the array from the second element to the end:

→ If the current element is smaller than min, update min,

→ If the current element is larger than max, update max.

ⓒ Return min & max.

Time Complexity:

→ Best, Worst & Average Case: O(n) (Single pass through the array)

→ Only requires one traversal making it efficient

Space Complexity:

→ O(1) (constant space for min & max variables)

② Using Sorting approach

Algorithm:

ⓐ Sort the array using a sorting algorithm (eg. QuickSort or MergeSort)

ⓑ The first element of the sorted array is the minimum element

(c) The last element of the sorted array is the maximum element. (2)

Time Complexity:

→ Sorting: O(n log n)
→ Fetching min & max: O(1)
→ Overall: O(n log n)

Space Complexity:

→ O(1) if sorting in place, otherwise O(n) for additional space in sorting algorithms

~~Code explanation:~~

~~① Iterative~~

~~Steps~~

Tracing:

For input array: [3, 1, 4, 1, 5, 9, 2]

① Initialize min = 3, max = 3
② Compare each element:
- 1 < 3 → min = 1
- 4 > 3 → max = 4
- 1 is ignored
- 5 > 4 → max = 5
- 9 > 5 → max = 9
- 2 is ignored

③ Final Result: min = 1, max = 9.

Test Cases:

① Input: [3, 1, 4, 1, 5, 9, 2]
Expected Output: Min = 1, Max = 9

② Input: [7, 7, 7, 7, 7]
Expected Output: Min = 7, Max = 7

③ Input: [-1, -5, -3, -8, -2]
Expected Output: Min = -8, Max = -1

④ Input: [5]
Expected Output: Min = 5, Max = 5.

USN: 1RV24MC032

G B Pandranga

Topic: Finding Kth smallest element using BST (Binary Search Tree).

Problem Definition: Given a Binary Search Tree (BST) & an integer 'K', find the Kth smallest element in the BST

Algorithm (In-Order Traversal):

① Perform in-order traversal on the BST
② Keep a counter to track the number of visited nodes.
③ When the counter equals K, the current node's value is the Kth-smallest element.

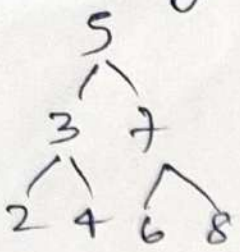
Time Complexity:

→ Best, Worst and Average case: O(n) where 'n' is the number of nodes.
→ Traverses the number of nodes.
→ Traverses the tree once.

Space Complexity:

→ O(h) where h is the height of the tree for recursion stack in Balanced BST.
→ In worst case (unbalanced), O(n).

Tracing:



Find the 3rd smallest element:

① Traverse left subtree of 5 → visit 2 → visit 3 → visit 4 (counter = 3)

② Return value 4 as the 3rd smallest element.

Test cases:

① Input: [5, 3, 7, 2, 4, 6, 8], k = 3
Expected output = 4

② Input: [5, 3, 7, 2, 4, 6, 8], k = 1
Expected output: 2

③ Input: [5, 3, 7, 2, 4, 6, 8], k = 7
Expected output = 8

④ Input: [5], k = 1
Expected output: 5

⑤ Input: [5, 3, 7], k = 5
Expected output: Invalid k value.