Sachin Singh
M24CSE033

Q. Implement a function to find the k-th smallest and k-th largest elements in an array. Use the Red Black Tree approach. Also, Implement a deterministic linear time algorithm to find the median of an array.

This is implemented using a red-black tree augmented to keep the size of each subtree. This enables one to find the k-th smallest or largest element with very high efficiency.

Functions:

1. **Node *createNode(int data)**:
    - **Description**: It creates a new node with the given data and initializes the subtree size to 1, and sets its color to RED.
    - **Parameters**:
        - int data: The value to be stored in the node.
    - **Returns**: A pointer to the newly created node.
2. **void leftRotate(RedBlackTree *tree, Node *x)**:
    - **Description**: It performs a left rotation on the node x. And also adjusts the subtree sizes during the rotation.
    - **Parameters**:
        - RedBlackTree *tree: A pointer to the Red-Black Tree.
        - Node *x: The node on which the left rotation is performed.
3. **void rightRotate(RedBlackTree *tree, Node *y)**:
    - **Description**: Performs a right rotation on the node y. Adjusts the subtree sizes during the rotation.
    - **Parameters**:
        - RedBlackTree *tree: A pointer to the Red-Black Tree.
        - Node *y: The node on which the right rotation is performed.
4. **void fixInsert(RedBlackTree *tree, Node *z)**:
    - **Description**: It avoids the violations in the Red-Black Tree that may occur after inserting a new node. It ensures that the tree maintains its properties.
    - **Parameters**:
        - RedBlackTree *tree: A pointer to the Red-Black Tree.
        - Node *z: The newly inserted node.

5. **void insert(RedBlackTree *tree, int data)**:
    - **Description**: Inserts a new node with the given data into the Red-Black Tree. Adjusts the subtree sizes during insertion and fixes any violations using fixInsert.
    - **Parameters**:
        - RedBlackTree *tree: A pointer to the Red-Black Tree.
        - int data: The value to be inserted.
6. **Node *findKth(Node *root, int k)**:
    - **Description**: Recursively finds the k-th smallest element in the tree using the size of the subtrees.
    - **Parameters**:
        - Node *root: The root of the subtree.
        - int k: The rank of the element to find (1-based).
    - **Returns**: A pointer to the k-th smallest node.
7. **int findKthSmallest(RedBlackTree *tree, int k)**:
    - **Description**: Finds the k-th smallest element in the Red-Black Tree.
    - **Parameters**:
        - RedBlackTree *tree: A pointer to the Red-Black Tree.
        - int k: The rank of the element to find (1-based).
    - **Returns**: The value of the k-th smallest element.
8. **int findKthLargest(RedBlackTree *tree, int k)**:
    - **Description**: Finds the k-th largest element in the Red-Black Tree by converting it into a "k-th smallest" problem.
    - **Parameters**:
        - RedBlackTree *tree: A pointer to the Red-Black Tree.
        - int k: The rank of the largest element to find (1-based).
    - **Returns**: The value of the k-th largest element.

Sachin Singh
M24CSE033

Example

```
RedBlackTree tree = {NULL};

// Insert elements into the Red-Black Tree

insert(&tree, 20);
insert(&tree, 15);
insert(&tree, 25);
insert(&tree, 10);
insert(&tree, 5);
insert(&tree, 4);
insert(&tree, 2);
insert(&tree, 19);
insert(&tree, 17);

printf("Enter the vlue of k : ");
int k;
scanf("%d", &k); // input for finding the kth smallest and kth largest elements
printf("k-th Smallest Element: %d\n", findKthSmallest(&tree, k));
printf("K-th Largest Element: %d\n", findKthLargest(&tree, k));
```

Output

```
********** Sachin Singh **********
********** M24CSE033 **********
--------------------------------------------------

Enter the vlue of k : 3
k-th Smallest Element: 5
K-th Largest Element: 19


...Program finished with exit code 0
Press ENTER to exit console.
```

Sachin Singh
M24CSE033

Deterministic Linear Time Algorithm (Median of Medians)

This is where the Median of Medians algorithm applies to determine the k-th smallest or median of a given array in linear deterministic time. This algorithm considers breaking the array into groups of 5, finding their medians, and then using those medians to pick an approximate pivot.

Functions:

1. **void swap(int *a, int *b)**:
    - **Description**: Swaps values of two integers.
    - **Parameters**:
        - int *a: Pointer to the first integer.
        - int *b: Pointer to the second integer.
2. **int partition(int arr[], int l, int r, int pivot)**:
    - **Description**: Partitions the array around the given pivot element.
    - **Parameters**:
        - int arr[]: The array to be partitioned.
        - int l: The left boundary of the array.
        - int r: The right boundary of the array.
        - int pivot: The pivot element.
    - **Returns**: The index of the pivot after partitioning.
3. **int findMedian(int arr[], int l, int n)**:
    - **Description**: Finds the median of a small group of size n. This is used to get medians of groups of 5 in the Median of Medians algorithm.
    - **Parameters**:
        - int arr[]: The array to find the median from.
        - int l: The starting index of the group.
        - int n: The size of the group.
    - **Return**: Median of the group.
4. **int kthSmallest(int arr[], int l, int r, int k)**:
    - **Description**: Find the k-th smallest element in the array using the Median of Medians algorithm.
    - **Parameters**:

- int arr[]: The input array.
- int l: The leftmost element of the array.
- int r: The rightmost element of the array.
- int k: The k-th smallest element (1-based).

- **Returns**: The element which is k-th small in the given array.

5. **int findMedianOfArray(int arr[], int n)**:
   - **Description**: This function calculates the median of the array using the function kthSmallest. The median is the middle element of a sorted list.
   - **Parameters**:
     - int arr[]: The array.
     - int n: The size of the array.
   - **Returns**: The median of the array.

Example:

```
int arr[] = {12, 3, 5, 7, 4, 19, 26};
int n = sizeof(arr) / sizeof(arr[0]);

printf("Median of the array is %d\n", findMedianOfArray(arr, n));
```

Output:

```
********** Sachin Singh **********
************ M24CSE033 ************
--------------------------------------------------

Median of the array is 8


...Program finished with exit code 0
Press ENTER to exit console.
```