Pseudocode:

The pseudo code of red-black tree is similar as the slide, so I list the function pseudo code that I add.
(Insert=O(lgn), Delete=O(lgn), Rotate=O(1), Insert-Fixup=O(lgn), Delete –Fixup= O(lgn))

Find_smallest(largest)_sameID: O(nlgn)

```
While (predecessor->value == current->value){ O(n)
    current = predecessor; O(lgn)
}
```

Find_tree_smallest(largest): O(lgn)

```
while (current->leftchild->ID != 0) { O(lgn)
        current = current->leftchild;
}
```

Find_Successor(Predecessor): O(lgn)

```
if (current->rightchild->ID != 0) { O(1)
   return Leftmost(current->rightchild);
}
new_node = current->parent
while (new_node->ID != 0 && current == new_node->rightchild) { O(lgn)
   current = new_node;
   new_node = new_node->parent;
}
```

SearchK: O(lgn)

```
if (VALUE < current->value)
   current = leftchild;
else if (VALUE > current->value)
   current = rightchild;
else if (VALUE == current->value)
   current = current;
else {
   if (current->value < VALUE)    //if the input doesn't exit
        current = Predecessor;
}
```

output mode 1,2 ID: O(nlgn)

```
print inorder of Rbtree and count;
if(counter == rank){ O(n)
```

```
        output mode2 result;
    }
    Find_smallest_sameID();O(nlgn)
    Output mode1 result;
output mode 1,2 rank: O(nlgn)
    while(have successor){ O(n)
      current = successor; O(lgn)
      if(current->value == value we want)O(1)
          push into stack;
      }
    mode1 result = stack.bottom
    mode2 result = stack.top & stack.bottom
output N nearest: O(nlgn)
    SearchK(); O(lgn)
    for(0 to N) O(nlgn)
        if (K->successor exist)
          push K into stack;
          K = K->successor;
    K = original_K
    for(0 to N) O(nlgn)
        if (K-> predecessor exist)
          push K into stack;
          K = K->successor;
    sort stack to find min(K->value – input_value) O(nlgn)
    for(0 to N) O(n)
        output stack

Main:
    Create empty red-black tree; (empty struct)
    While(input case = "?"){ run total "n" command
        'I': Insert node; O(lgn)
        'D': Delete node; O(lgn)
        'r': output mode 1 ID; O(nlgn)
        'R': output mode 2 ID; O(nlgn)
        'v': output mode 1 rank; O(nlgn)
        'V': output mode 2 rank; O(nlgn)
        'K': output N nearest; O(nlgn)
    }
```

Time complexity analysis:

Assume:
Red black tree has insert “n” node O(nlgn)+
Run total “m” command O(m*
Each command at most cost nlogn)

Time complexity = O(nlgn) + O(m*nlogn) = O(m*nlogn) = O(n^2logn)

Experimental results:

Example:

Input:

```
I 5 17.2
I 2 8.6
I 16 12.5
I 11 4.1
I 4 4.1
r 4
D 16 12.5
R 4
I 6 15.9
I 7 0.1
v 0.1
V 4.1
V 8.6
I 16 4.1
I 17 4.1
I 18 8.6
K 4.5 5
```

Output:

```
r 4 4.1
R 11 4.1
v 6
V 4 5
V 3 3
K 4 11 16 17 2
```

My own data:

Input:

```
I 1 9.5
I 2 13.7
I 3 2.2
I 4 5.9
I 5 13.7
I 6 9.5
I 7 8.3
I 8 9.5
I 9 2.2
I 10 20.0
D 1 9.5
I 11 9.5
D 6 9.5
I 12 9.5
v 9.5
V 9.5
r 5
R 5
K 30.0 5
```

Output:

```
v 4
V 4 6
r 6 9.5
R 11 9.5
K 10 2 5 6 11
```