1. The MinStack should have built on top of a normal stack to keep the O(1) runtime of the functions. By adding a new value that keeps the minimum of the previous elements we can suffice the min stack condition. With this the push operation will always run in O(1) runtime (we need to compare only the previous minimum with current value), also the pop operation too (each element is keeping their own minimum values). Peek, isEmpty, and min operations only need to run in the last element.
2. The running time of the reverse operation is O(N).

```
public void reverse() {
    Node current = head;
    Node temp;
    while (current != null && current.next != null) {
        temp = current.next;
        current.next = current.next.next;
        add(temp.value);
    }
}
```

3. The sorting algorithm is faster than Inversion-Bound sorting algorithms and it is slower than the Merge Sort by the worst case since the average running time of the sorting algorithm will be **O(N logN)**.

4. all of them black

Num nodes *n*	**Does there exist a red-black tree with *n* nodes, all of which are black?**
1	Yes
2	No
3	Yes
4	No
5	No
6	No
7	Yes

5. exactly one of the nodes is red:

Num nodes *n*	**Does there exist a red-black tree with *n* nodes, all of which are black?**
1	No
2	Yes
3	No
4	Yes
5	No
6	No
7	No