*i* {} 5 ⊙ □

716. Max Stack

Design a max stack data structure that supports the stack operations and supports finding the stack's maximum element.

Implement the MaxStack class:

- MaxStack() Initializes the stack object.
- void push(int x) Pushes element x onto the stack.
- int pop() Removes the element on top of the stack and returns it.
- int top() Gets the element on the top of the stack without removing it.
- int peekMax() Retrieves the maximum element in the stack without removing it.
- int popMax() Retrieves the maximum element in the stack and removes it. If there is more than one maximum element, only remove the **top-most** one.

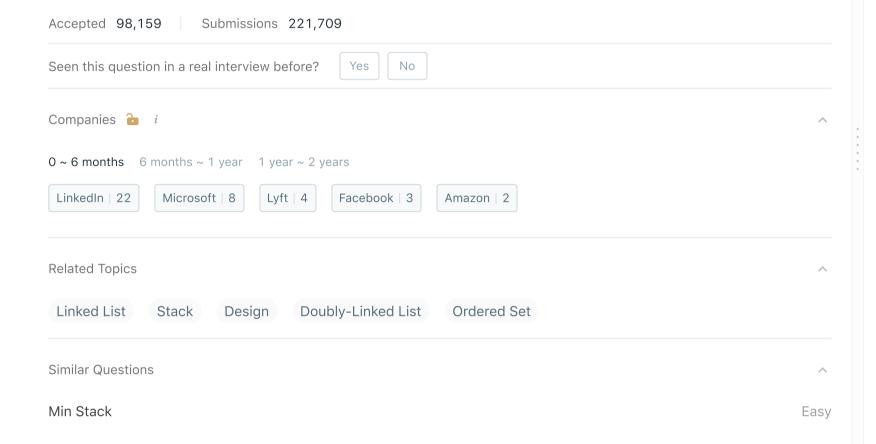
## Example 1:

```
Input
["MaxStack", "push", "push", "top", "popMax", "top", "peekMax", "pop", "top"]
[[], [5], [1], [5], [], [], [], [], [], []]
Output
[null, null, null, 5, 5, 1, 5, 1, 5]
Explanation
MaxStack stk = new MaxStack();
stk.push(5); // [\underline{5}] the top of the stack and the maximum number is 5.
stk.push(1); // [\underline{5}, \mathbf{1}] the top of the stack is 1, but the maximum is 5.
stk.push(5); // [5, 1, 5] the top of the stack is 5, which is also the maximum, because it is
the top most one.
stk.top(); // return 5, [5, 1, 5] the stack did not change.
stk.popMax(); // return 5, [5, 1] the stack is changed now, and the top is different from the
max.
stk.top(); // return 1, [5, 1] the stack did not change.
stk.peekMax(); // return 5, [5, 1] the stack did not change.
stk.pop(); // return 1, [\underline{5}] the top of the stack and the max element is now 5.
stk.top(); // return 5, [5] the stack did not change.
```

## **Constraints:**

- $\bullet$  -10<sup>7</sup> <= x <= 10<sup>7</sup>
- At most 104 calls will be made to push, pop, top, peekMax, and popMax.
- There will be at least one element in the stack when pop, top, peekMax, or popMax is called.

**Follow up:** Could you come up with a solution that supports O(1) for each top call and O(logn) for each other call?



1 ▼ class MaxStack { TreeMap<Integer, List<Node>> map; DoubleLinkedList dll; public MaxStack() { map = new TreeMap();dll = new DoubleLinkedList(); 10 ▼ public void push(int x) { 11 Node node = dll.add(x); 12 if(!map.containsKey(x)) 13 map.put(x, new ArrayList<Node>()); 14 map.get(x).add(node); 15 16 17 ▼ public int pop() { 18 int val = dll.pop(); 19 List<Node> L = map.get(val); 20 L.remove(L.size() - 1); 21 if (L.isEmpty()) map.remove(val); 22 return val; 23 24 25 ▼ public int top() { 26 return dll.peek(); 27 28 29 ▼ public int peekMax() { 30 return map.lastKey(); 31 32 33 ▼ public int popMax() { 34 int max = peekMax(); 35 List<Node> L = map.get(max); 36 Node node = L.remove(L.size() - 1); 37 dll.unlink(node); 38 if (L.isEmpty()) map.remove(max); 39 return max; 40 41 } 42 43 ▼ class DoubleLinkedList { 44 Node head, tail; 45 public DoubleLinkedList() { 46 ▼ 47 head = new Node(0);48 tail = new Node(0); 49 head.next = tail;50 tail.prev = head;51 52 53 ▼ public Node add(int val) { 54 Node x = new Node(val);55 x.next = tail;56 x.prev = tail.prev; 57 tail.prev = tail.prev.next = x;

Autocomplete

i Java