arr[] : { 1, 2, 3, 2, 2, 4, 3 }

Brute force.

TC : $O(N^2)$

Freq Map

id	freq
1	1
2	3
3	2
4	1

TC : O(N)

if (map.get(id) > 1)
ans.add(id);

↳ TC : O(N)

TC : O(N) S(: O(N)

Design a Circular Deque

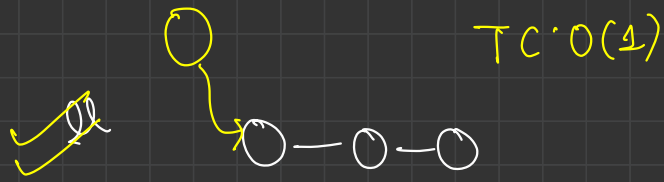
maxCapacity = K

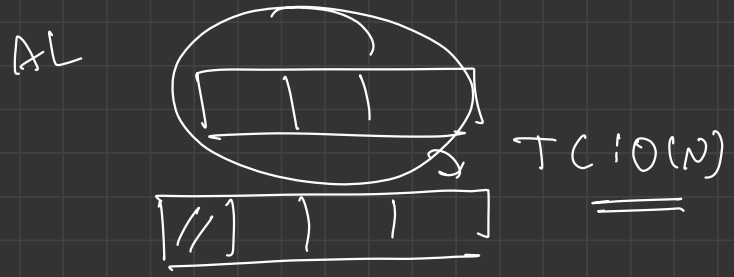
- insertFront(4)
- insertLast(5)
- getFront() ⟶ 4
- deleteFront()
- insertFront(3)
- insertLast(7)
- getRear() ⟶ 7
- removeLast();

Linear ds

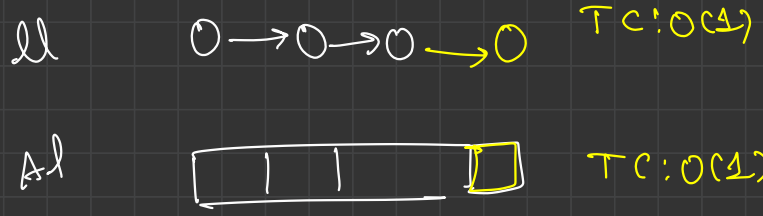
{ ~~Array~~, ArrayList, LinkedList, ~~Stacks~~, ~~Queue~~ }

- insertFirst()

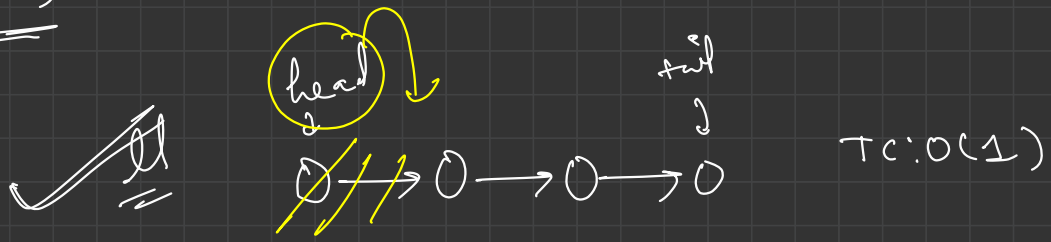




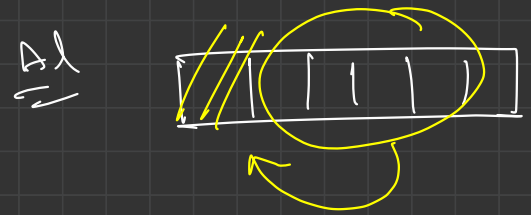
- insertLast()



deleteFront()



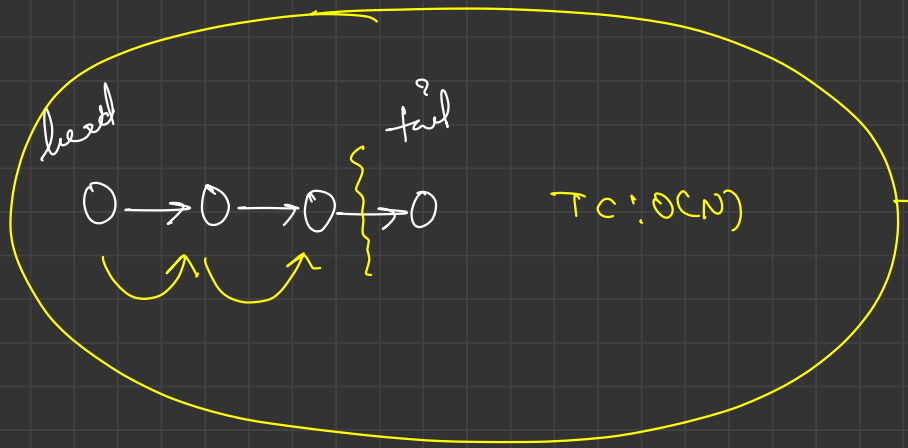
TC: O(1)



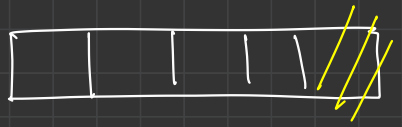
TC: O(N)

deleteLast()

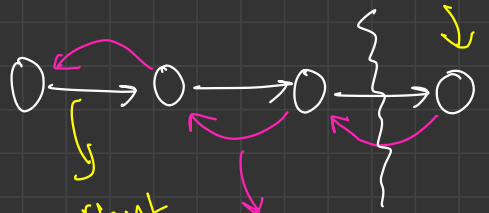
LL



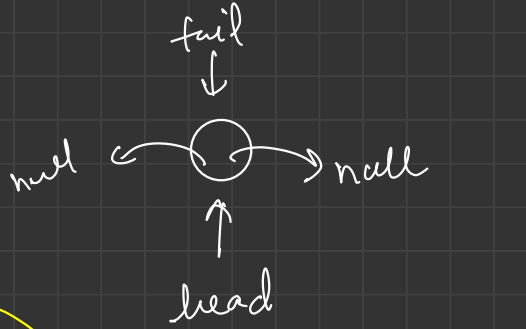
Arr

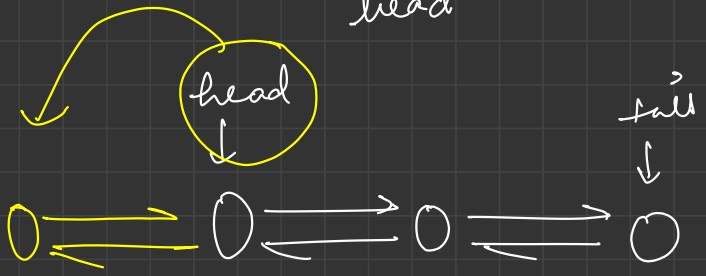


Doubly Linked List

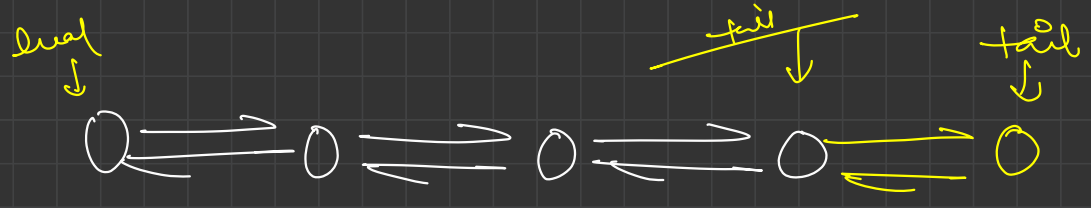


TC: O(1)

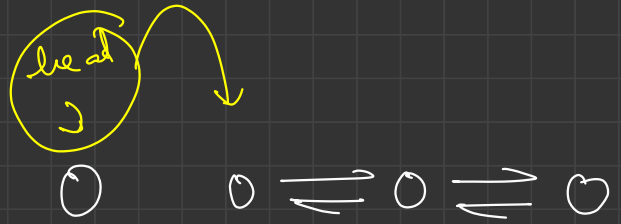




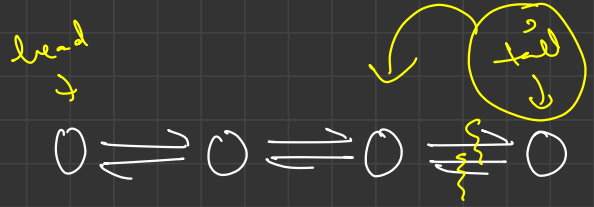
node.next = head
head.prev = node
head = node




```
tail.next = node
node.prev = tail
tail = node
```



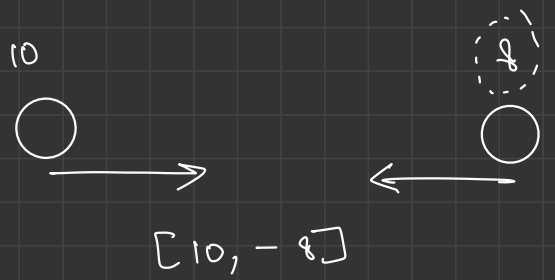

```
head = head.next;
head.prev = null
```



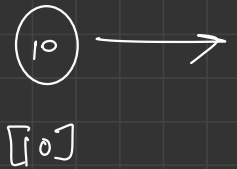
tail = tail.prev

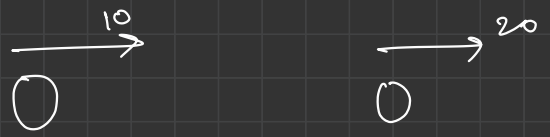
tail.next = null

Moving Bombs

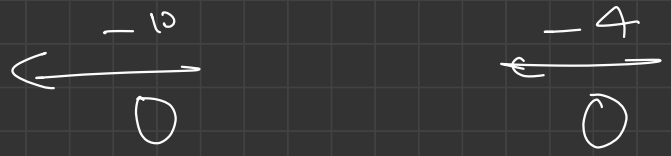


After Collision



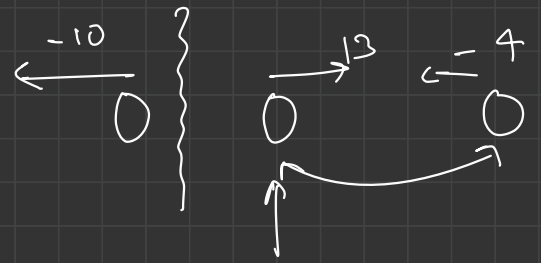


[10, 20]



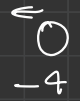
[-10, -4]

[-10, -5]



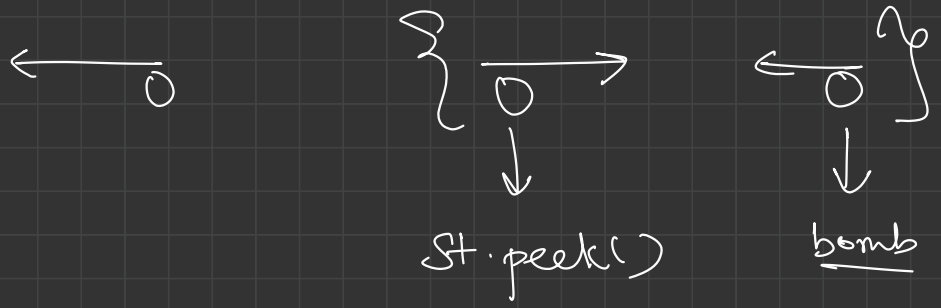
Stack LIFO

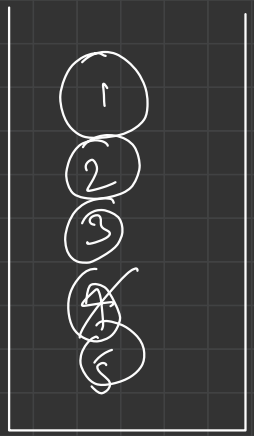
bombs[] : [-10, 3, 2, -1, 3, -3, 3, -4]

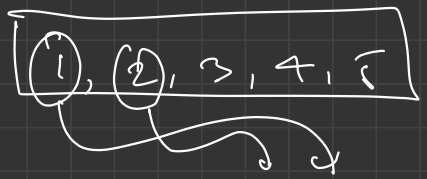


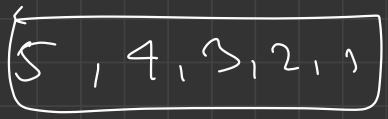
-4, -10

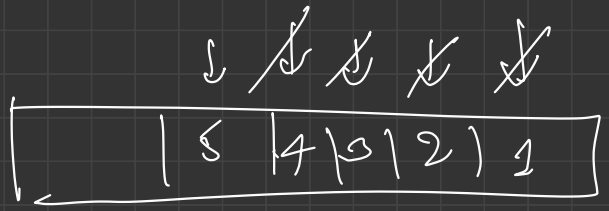
[-10, -4]


st.peek()
bomb


1
2
3
4
5


1, 2, 3, 4, 5


5, 4, 3, 2, 1


5 4 3 2 1

```java
// TC: O(N), SC: O(N)
public int[] MovingBombs(int[] bombs) {
    // Write your code here

    Stack<Integer> st = new Stack<>();
    for (int bomb : bombs) {
        if (bomb > 0) {
            st.push(bomb);
        } else {
            while (st.size() > 0 && st.peek() > 0 && st.peek() < -bomb) {
                st.pop();
            }

            if (st.size() > 0 && st.peek() == -bomb) {
                st.pop();
            } else if (st.size() > 0 && st.peek() > -bomb) {
                continue;
            } else {
                st.push(bomb);
            }
        }
    }

    int[] ans = new int[st.size()];
    int i = st.size() - 1;
    while (st.size() > 0) {
        ans[i] = st.pop();
        i--;
    }

    return ans;
}
```

bombs[] = [-10, 3, 2, -1, 3, -3, 3, -4]

Stack

[-10, -4]