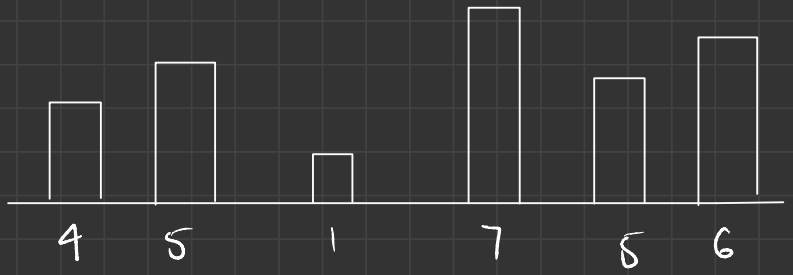


(index − ngeli) = dist

(5 − 3) * (5 − 3) = 2 × 2 = 4

(dist)

0	1	2	3	4	5
4	2	1	5	2	3
✗	✗	✗	✗	✗	↑

max score = ~~0 2 3~~ 4

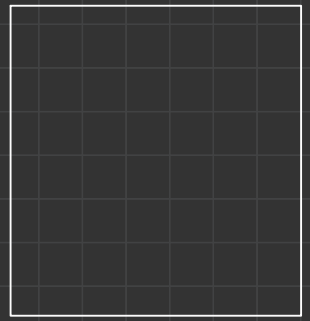
(2 − 1) * 1 = 1

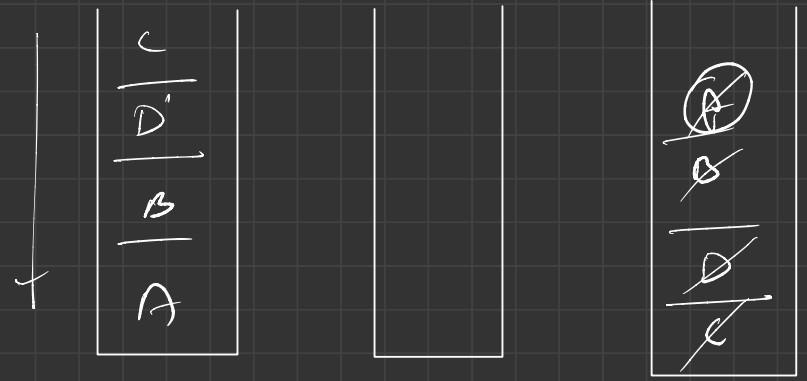
(4 − 2) * (1) = 2

(5 − 2) × 1 = 3 * 1 = 3

Text Editor

write (A)
write (B)
write (C)
undo ()
write (D)
undo ()
redo ()
redo ()
read()





A B D C

[3, 4, 2, 3, 2, 3, 1, 2, 1, 2, 1]

attack = 3 ≠ 5 ≠ 6 ≠ 8

captain Monster

[3, 4, (2), 3]

before captain dies

attack 7

2, 1, 2

Any → defeat when captain dead!

attack 1

4, 2, 3, 2

attack 2

2, 3, 2, 3

attack 3

3, 2, 3, 1

attack 4

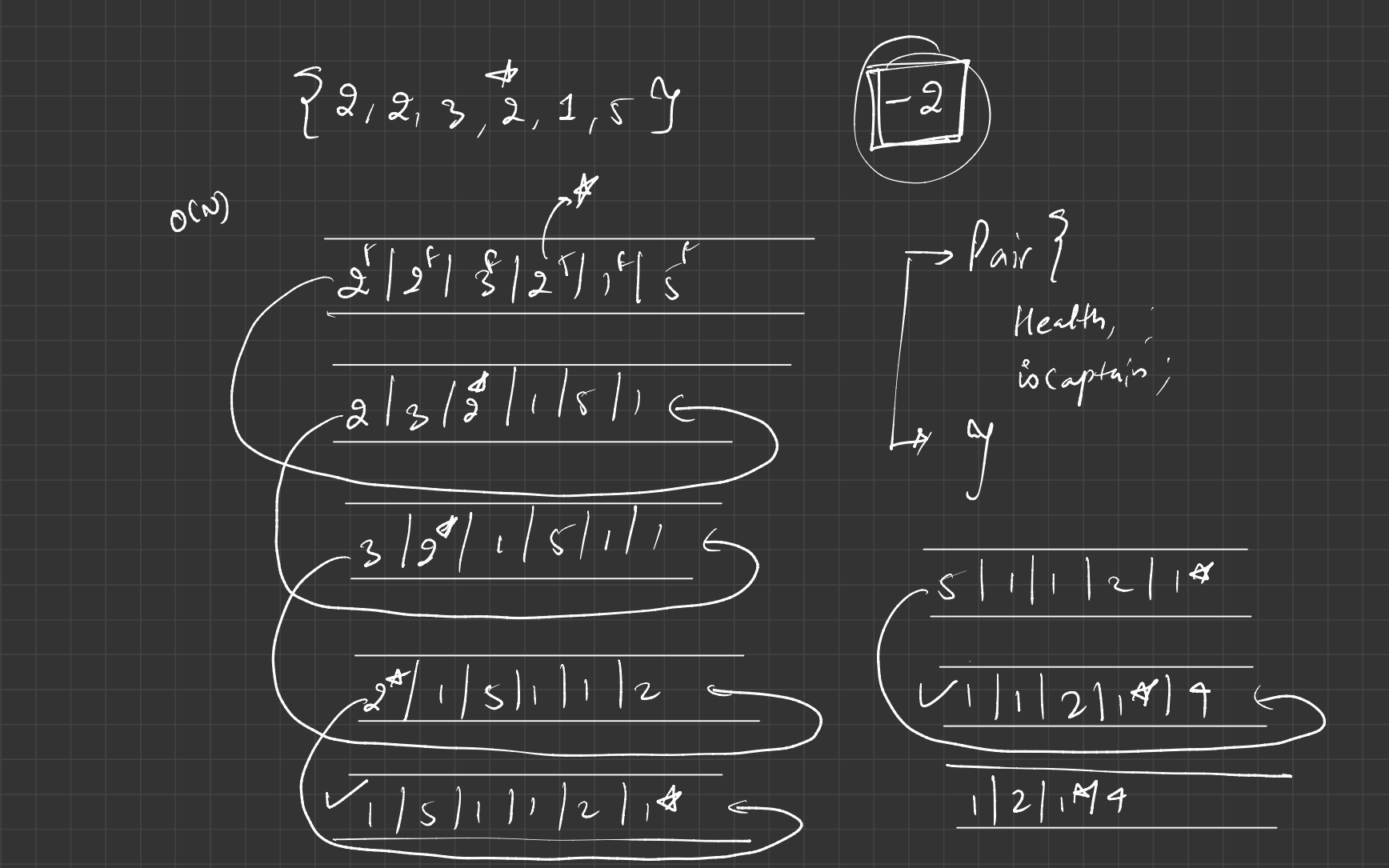
2, 3, 1, 2

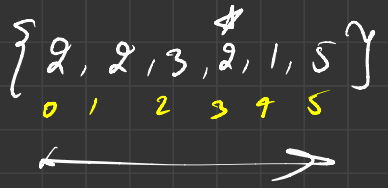
attack 5

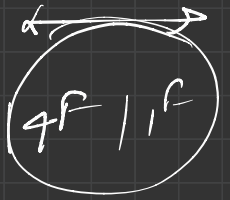
3, 1, 2, 1

attack 6

1, 2, 1, 2


{2, 2, 3, 2, 1, 5}
-2
O(N)
Pair {
Health,
isCaptain;
q



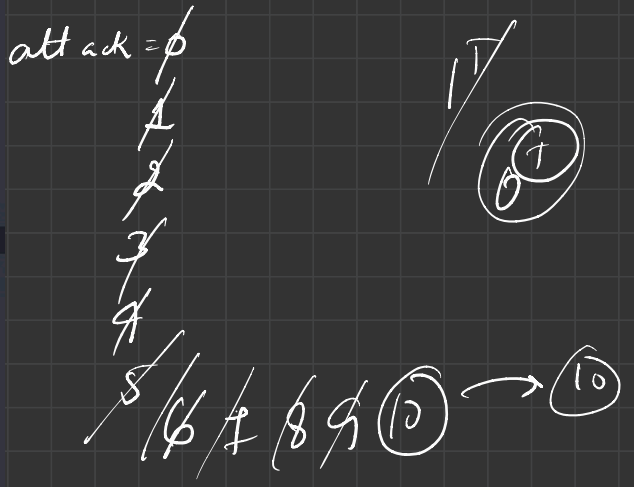


```java
public static int minAttacksNeeded(int[] health, int k) {
  int n = health.length;

  Queue<Pair> que = new ArrayDeque<>();

  for (int i = 0; i < n; i++) {
    if (i == k) {
      que.add(new Pair(health[i], true));
    } else {
      que.add(new Pair(health[i], false));
    }
  }

  int attacks = 0;
  while (true) {
    Pair person = que.remove();
    attacks += 1;
    person.myHealth -= 1;
    if (person.myHealth <= 0) {
      if (person.isCaptain == true) {
        return attacks;
      }
    } else {
      que.add(person);
    }
  }
}
```



TC: O(N*K)

↓

health of Captain

SC: O(N)