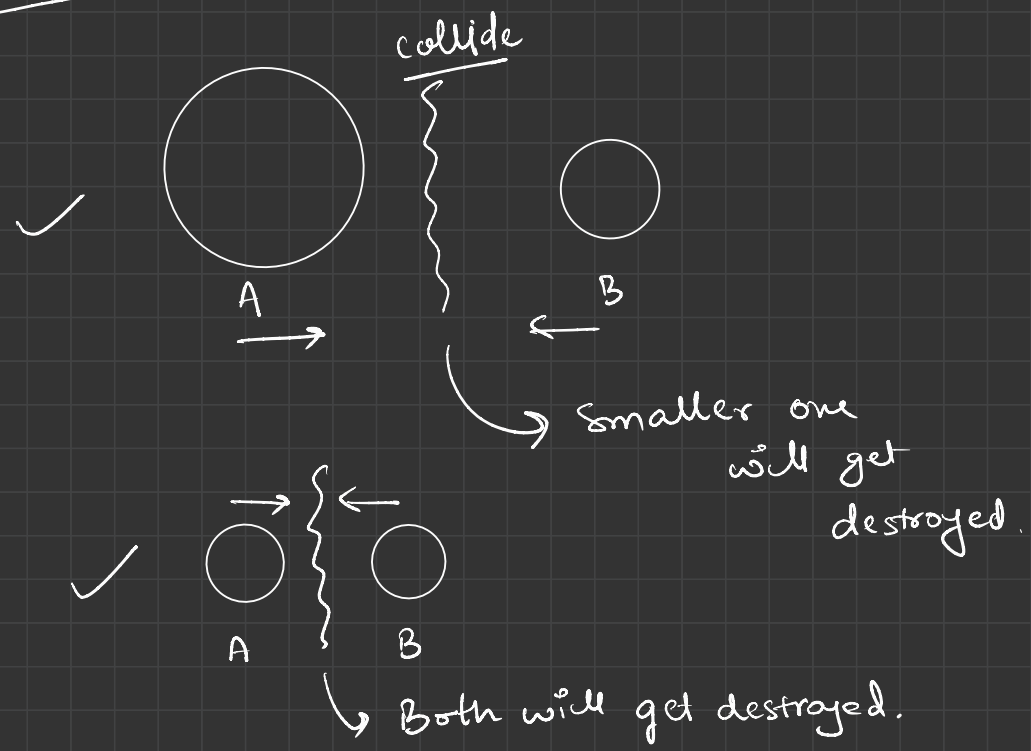
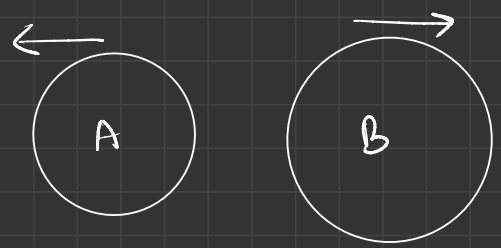
Asteroid Collision:



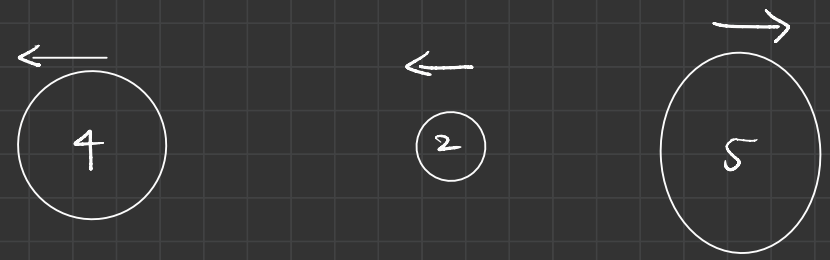
Hence No collision will happen

No collision



No Collision

asteroids[] = { 1, 2, 3, -4, -2, 5, -3 }



{ -4, -2, 5 }

→ Ans

{ Asteroids in system after all the collisions

(+)ve moving towards right }

(−)ve moving towards left

asteroids[] = { -2, 3, 4, -4, 5, -7, 8, 9, -3, -2, 11 }

← 2

← 7 → 8 → 9 → 11

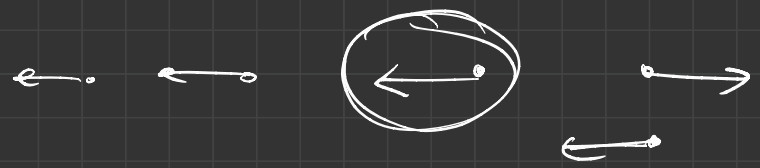
← 2 ← 7 → 8 → 9 → 11

{ -2, -7, 8, 9, 11 } → Asteroids left

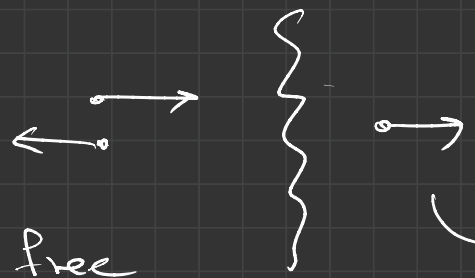
11
9
8
-7
-2

Stack

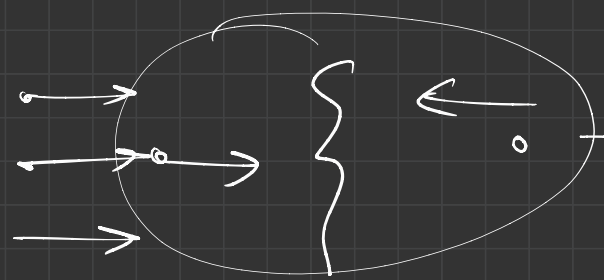
{ -2, -7, 8, 9, 11 }



if No, one in free universe comfy from right that means I will never collide.

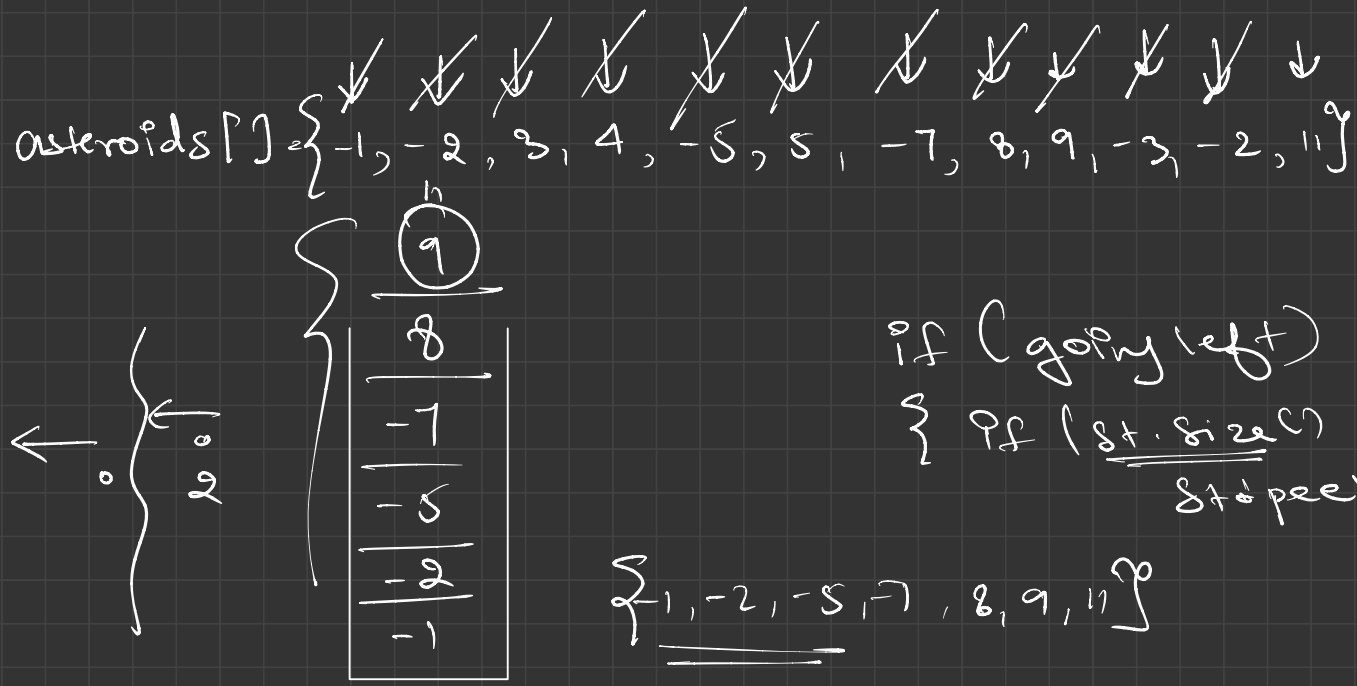


moving towards right directly add him to my free universe.



collision will happen only in this case!

asteroids[] = {-1, -2, 3, 4, -5, 5, -7, 8, 9, -3, -2, 11}



if (going left)
{ pf (st.size() ||
st.peek() < 0)

{-1, -2, -5, -7, 8, 9, 11}

→ free universe

```java
public int[] asteroidCollision(int[] asteroids) {
  //Write code here
  Stack<Integer> st = new Stack<>();

  for (int asteroid : asteroids) {
    if (asteroid > 0) {
      st.push(asteroid);
    } else {
      while (st.size() > 0 && st.peek() > 0 && st.peek() < -asteroid) {
        st.pop();
      }

      if (st.size() > 0 && st.peek() > -asteroid) {
        // he destroyed me, hence no space for me in free universe
        continue;
      } else if (st.size() > 0 && st.peek() == -asteroid) {
        // we both got destroyed
        st.pop();
      } else {
        st.push(asteroid);
      }
    }
```

~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ ~~10~~ ~~11~~ 1

{-1, -2, 3, 4, -5, 5, -7, 8, 9, -3, -2, 11}

11
9
8
-7
-5
-2
-1

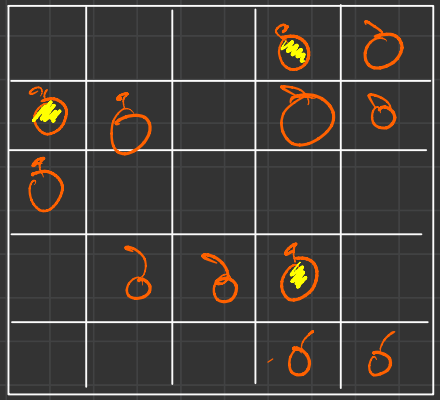
st.

[-1, -2, -5, -7, 8, 9, 11]

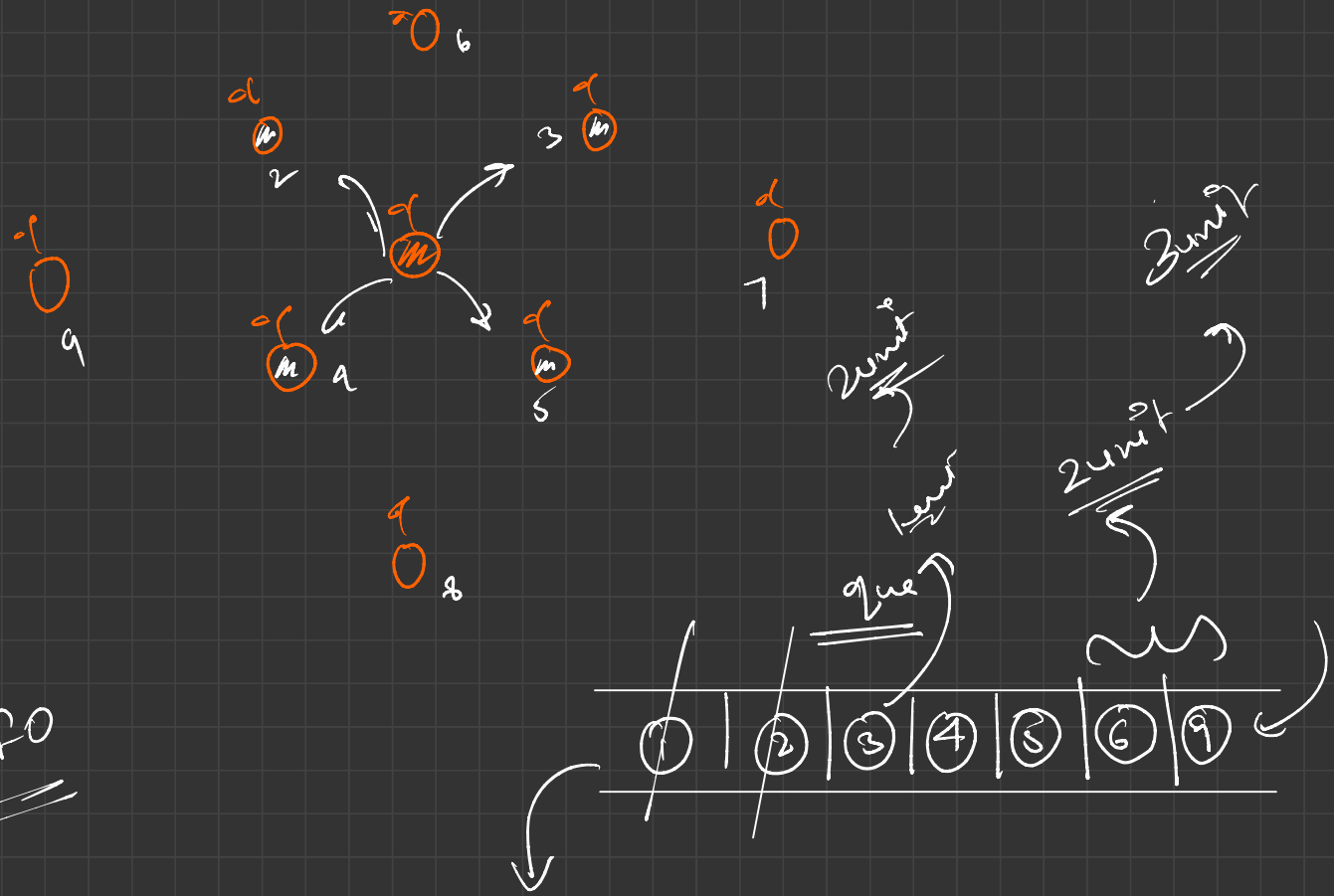
TC: O(N)

SC: O(N)

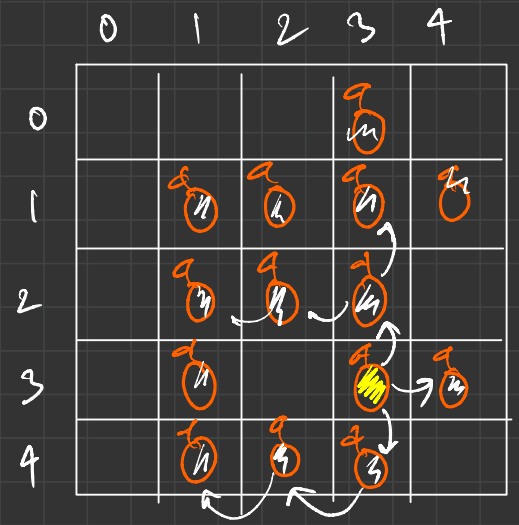
Rotten Oranges.



min amt of time?



FIFO



time = 1

4

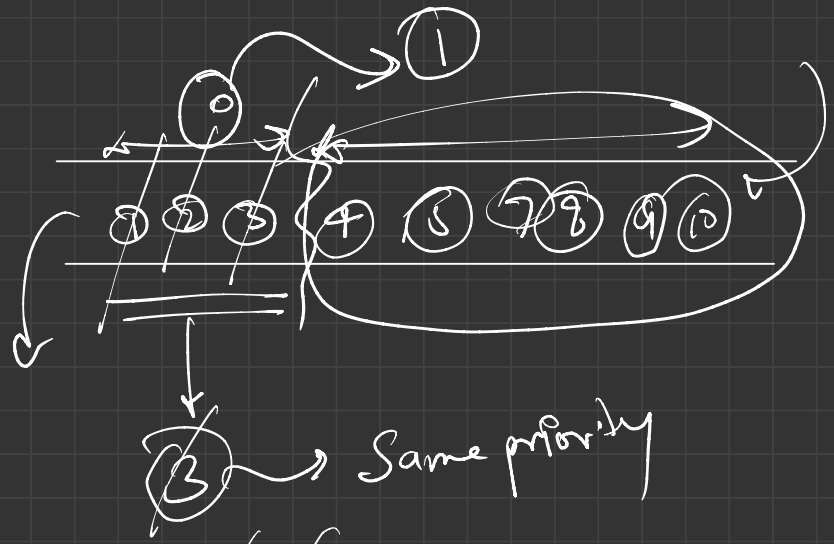
2

3

(1,2) (0,3)

(1,4) (2,1)

(4,1)



FIFO

```java
int time = 0;
while (que.size() > 0) {
    int size = que.size();

    while (size > 0) {
        Pair rnode = que.remove();

        int cr = rnode.row;
        int cc = rnode.col;

        if (cr - 1 >= 0 && grid[cr - 1][cc] == 1) {
            grid[cr - 1][cc] = 2;
            que.add(new Pair (cr - 1, cc));
        }

        if (cr + 1 < n && grid[cr + 1][cc] == 1) {
            grid[cr + 1][cc] = 2;
            que.add(new Pair (cr + 1, cc));
        }

        if (cc - 1 >= 0 && grid[cr][cc - 1] == 1) {
            grid[cr][cc - 1] = 2;
            que.add(new Pair (cr, cc - 1));
        }

        if (cc + 1 < m && grid[cr][cc + 1] == 1) {
            grid[cr][cc + 1] = 2;
            que.add(new Pair (cr, cc + 1));
        }

        size--;
    }

    time++;
}
```

	0	1	2	3
0	0	2	0	1
1	2	2	2	0
2	2	2	0	2
3	0	2	0	2

time-1

1 unit

(1,1) (0,3) (2,3) (0,1) (2,1) (1,0) (1,2) (3,3)

time = 0 1 2

3

(2,0) (3,1)

Sliding Window Maximum

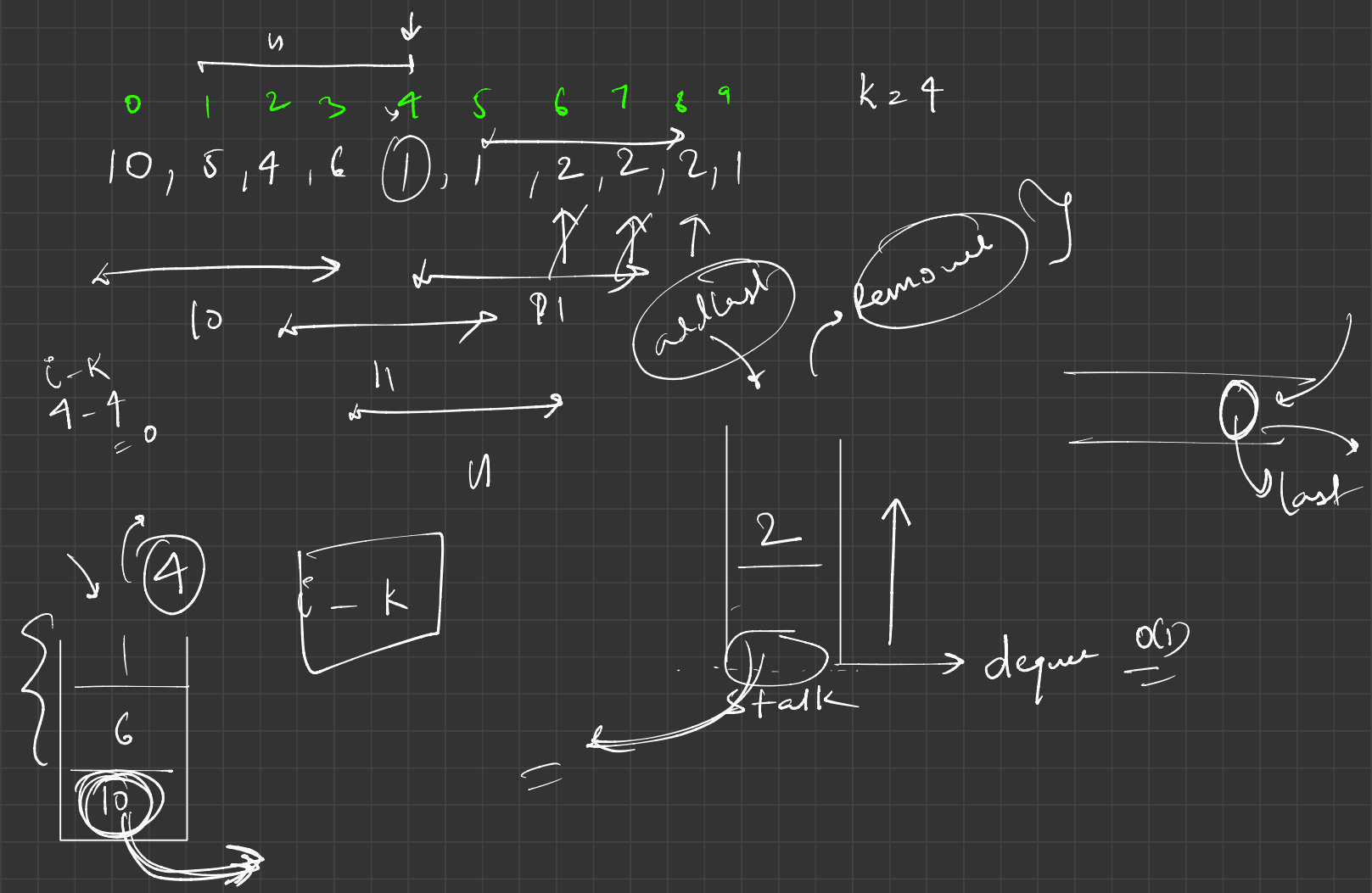
$$arr[] = \{10, 3, -1, -3, 5, 3, 6, 7\}$$

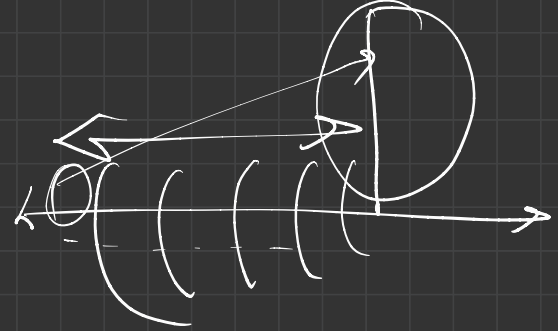
$\boxed{k = 3}$

3 5 7

3 5 6

10, 6, 5, 4, 11, 1, 2, 2, 2, 1


k = 4
0 1 2 3 4 5 6 7 8 9
10, 5, 4, 6, 1, 1, 2, 2, 2, 1
10
11
11
i - k
4 - 4
= 0
addlast
removel
last
4
1
6
10
i - k
2
stalk
deque
O(1)

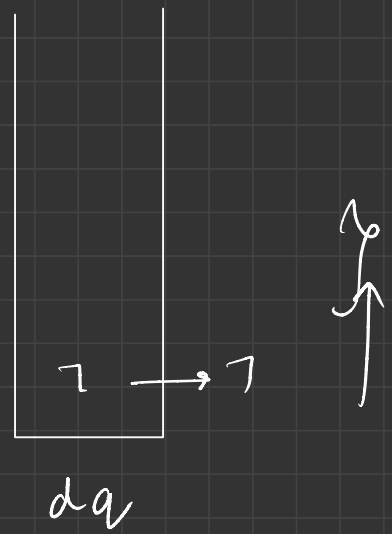


```java
static int[] SlidingWindowMaximum(int n, int k, int[] arr) {
  // write code here
  int[] ans = new int[n - k + 1];
  int itr = 0;

  Deque<Integer> dq = new ArrayDeque<>();
  for (int i = 0; i < n; i++) {
      if (dq.size() > 0 && dq.getFirst() == i - k) {
          dq.removeFirst();
      }

      while (dq.size() > 0 && arr[dq.getLast()] <= arr[i]) {
          dq.removeLast();
      }

      dq.addLast(i);

      if (i >= k - 1) {
          ans[itr] = arr[dq.getLast()];
          itr++;
      }
  }

  return ans;
}
```

K = 3

0 1 2 3 4 5 6 7

int[] = [1, 3, -1, -3, 5, 3, 6, 7]



$\boxed{3, 3, 5, 5, 6, 7}$

```java
static int[] SlidingWindowMaximum(int n, int k, int[] arr) {
    // write code here
    int[] ans = new int[n - k + 1];
    int itr = 0;

    Deque<Integer> dq = new ArrayDeque<>();
    for (int i = 0; i < n; i++) {
        if (dq.size() > 0 && dq.getFirst() == i - k) {
            dq.removeFirst();
        }

        while (dq.size() > 0 && arr[dq.getLast()] <= arr[i]) {
            dq.removeLast();
        }

        dq.addLast(i);

        if (i >= k - 1) {
            ans[itr] = dq.getLast();
            itr++;
        }
    }

    return ans;
}
```

arr[dq.getfirst()]

0 1 2 3 4 5 6 7 8 9

10, 5, 4, 6, 1, 1, 2, 2, 2, 1

K = 4

1-4

first

last

4

8 9

2 1

dec

[10, 6, 6, 6, 2, 2, 2]