Agenda

1. sliding window maximum → LC : H
2. Rotten Oranges → LC : M
3. Asteroid collision → LC : M

sliding window maximum.

0	1	2	3	4	5	6	7

int [] arr = { 1, 3, -1, -3, 5, 3, 6, 7 }   K = 3

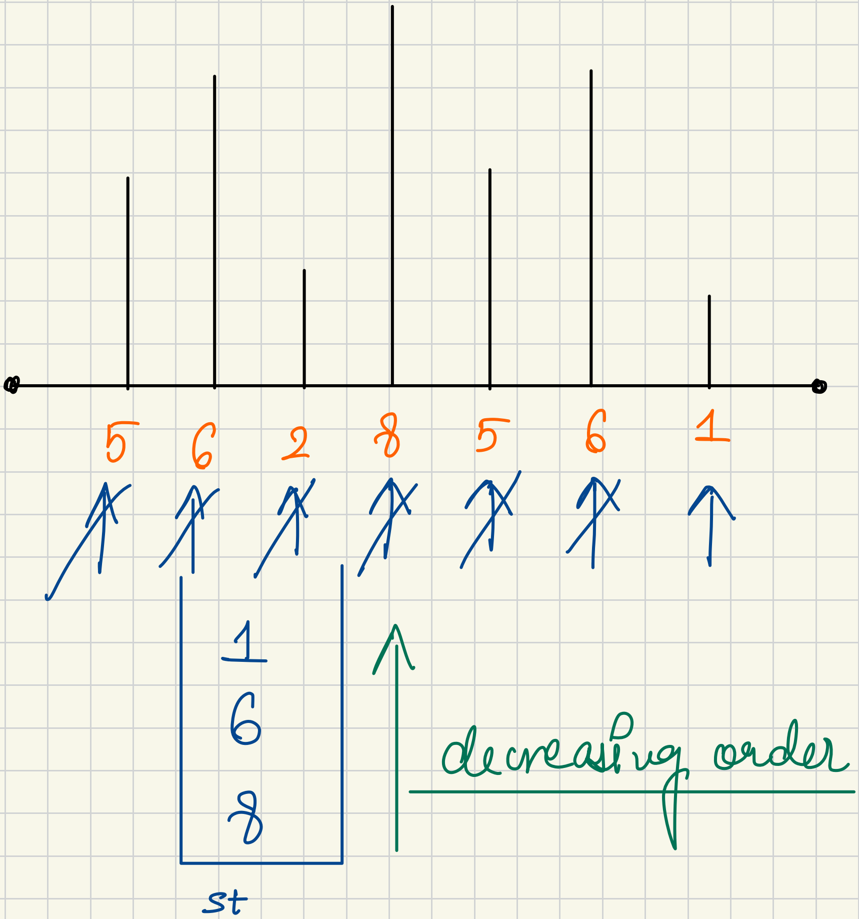
{ max^m value of each window   { 3, 3, 5, 5, 6, 7 }

Brute force.

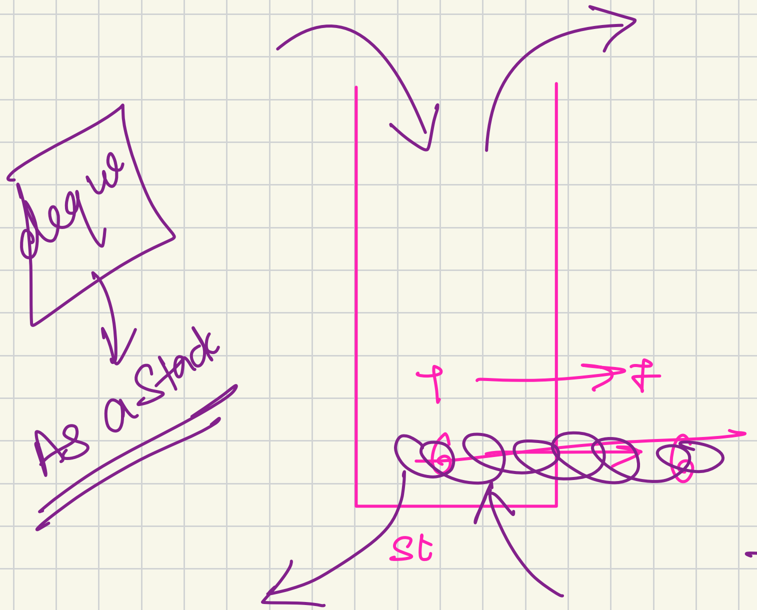
```
for(int i=0; i<=n-K; i++)
{
    int max = -∞;
    for (int j=i; j<i+K; j++)
    {
        max = Math.max(max, arr[j]);
    }
}
```

✓ TC: O(N*K) }
SC: O(1)

nger



$$\text{int[] arr} = \left\{\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1, & 3, & -1, & -3, & -2, & 3, & 6, & 7 \end{array}\right\} \qquad K = 3$$

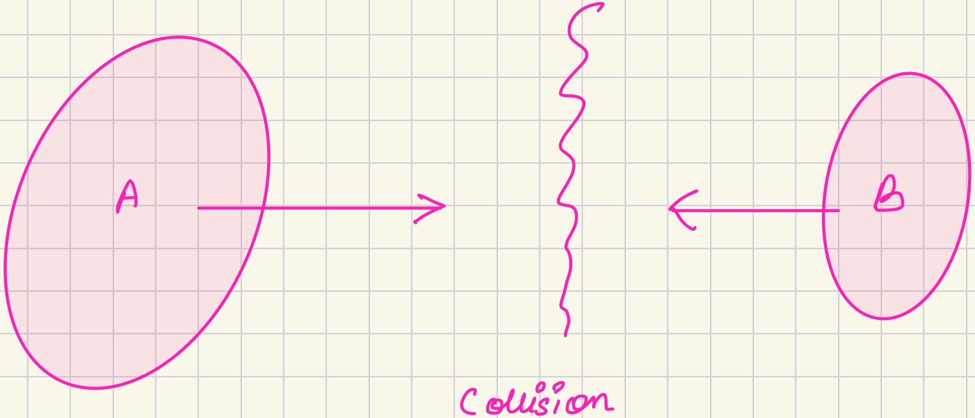


{3, 3, -1, 3, 6, 7}

→ value tc : O(1)

→ remove from here

Asteroid Collision

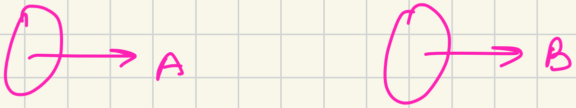


NOTE: at time of collision smaller asteroid gets destroyed and other asteroid will be unaffected

NOTE: if two asteroid of same size collide then both will get destroyed.

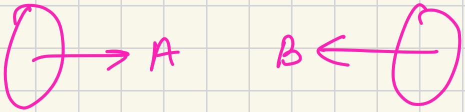
Conditions for collision

Case 1



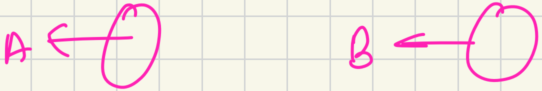
no collision

Case 2



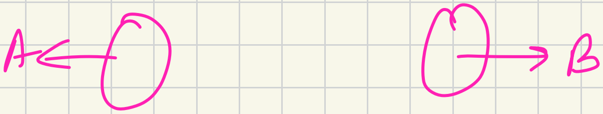
collision

Case 3



no collision

Case 4

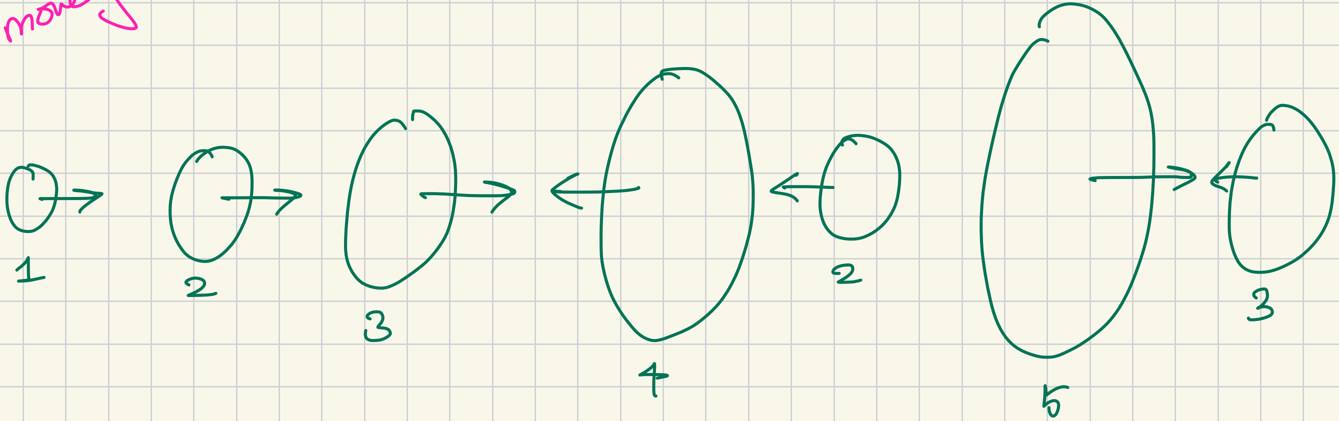


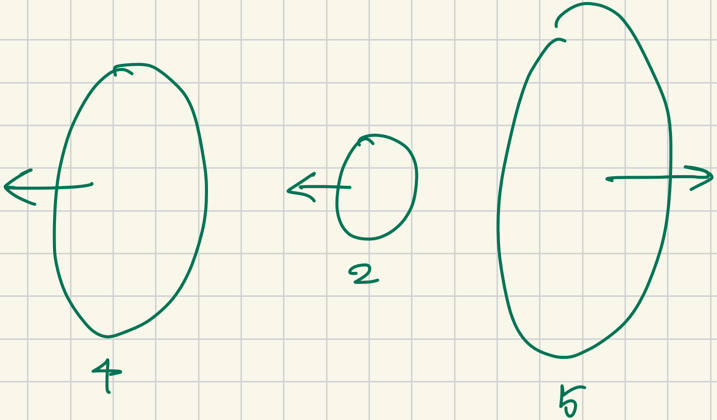
no collision

```
                       0    1    2    3    4    5    6
int[] asteroids = {  1 ,  2 ,  3 , -4 , -2 ,  5 , -3 }
```

(+ve) → moving right

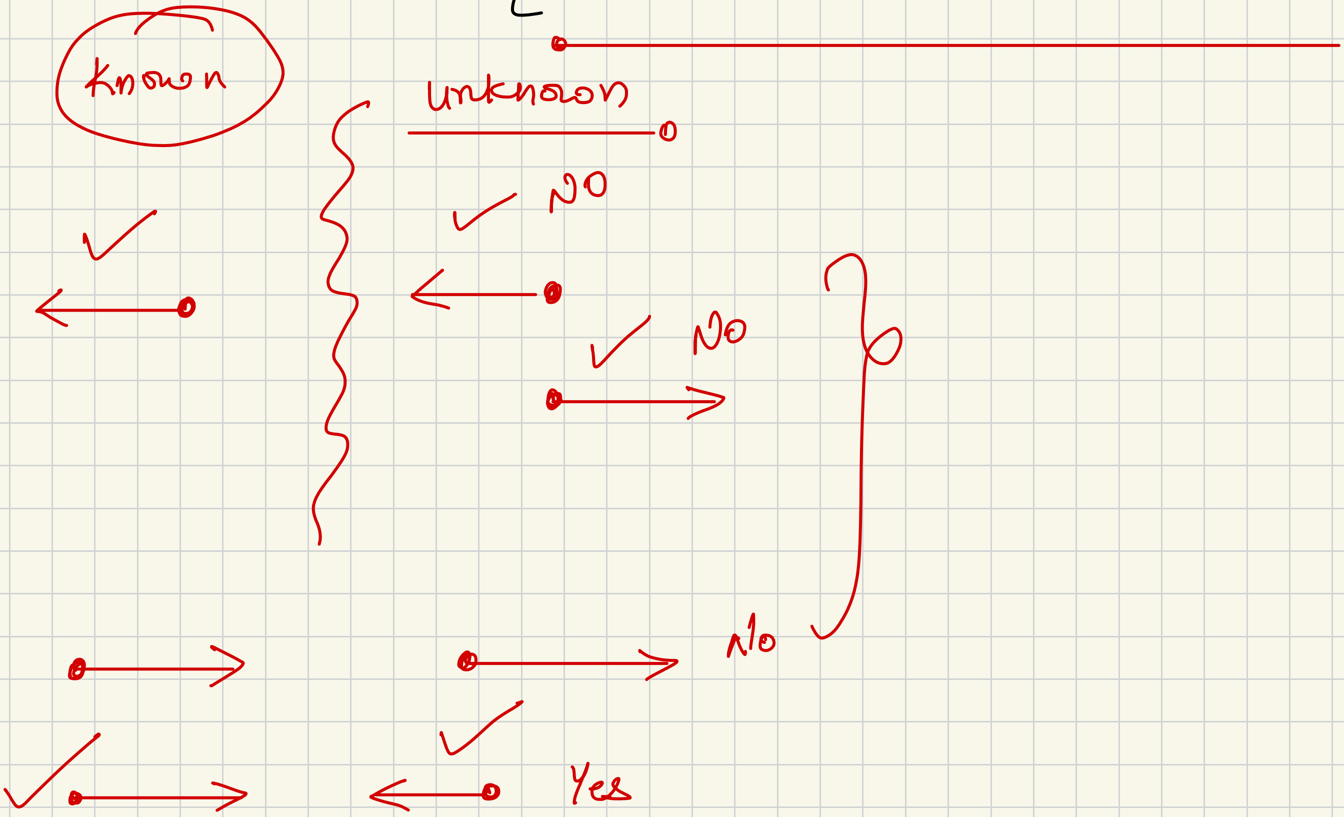
(-)ve → moving left

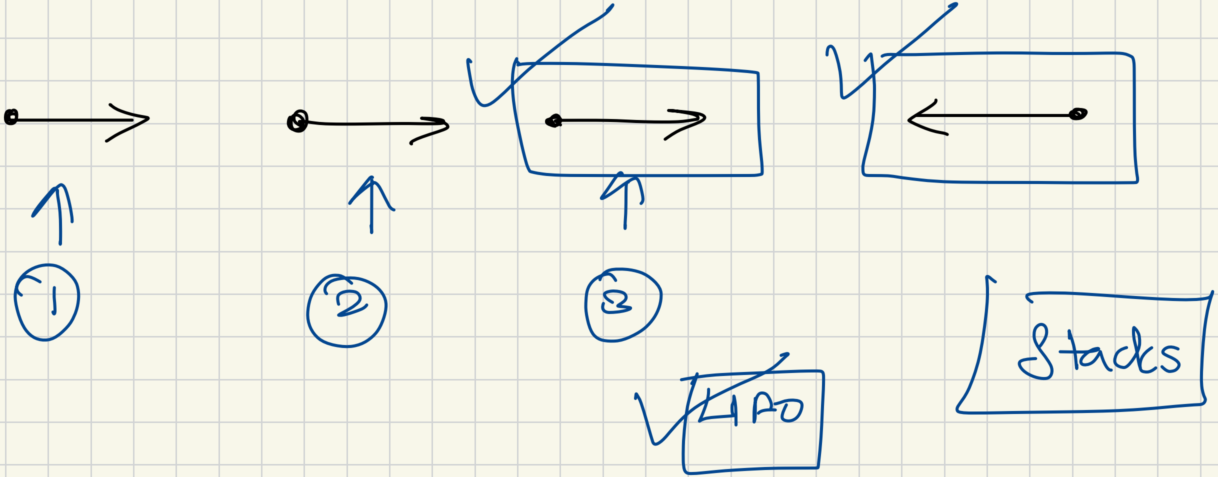




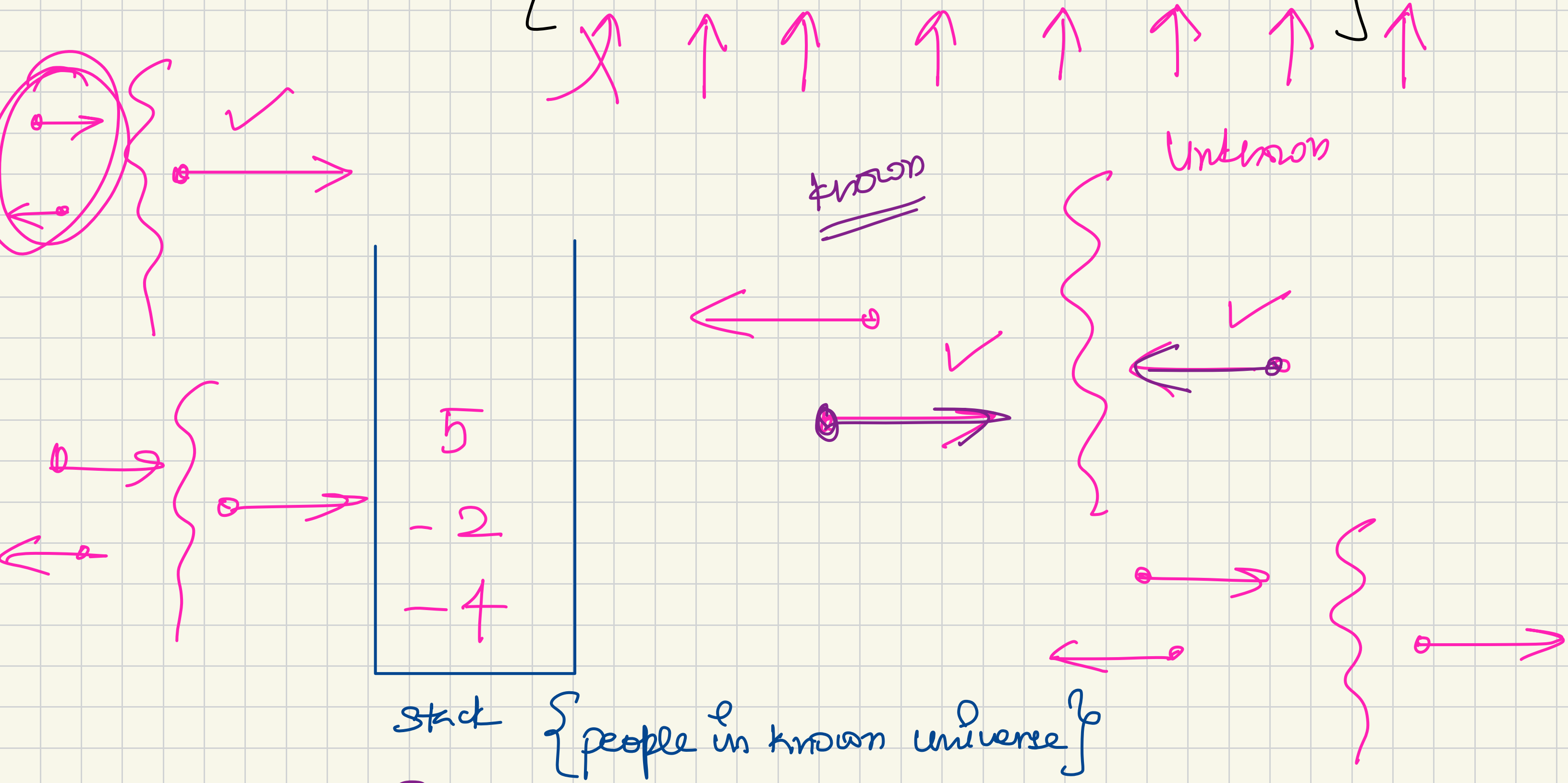
$\checkmark \{-4, -2, 5\} \rightsquigarrow \text{stable state of universe} \}$

```
                        0    1    2    3    4    5    6
int[] asteroids  =  {  1 ,  2 ,  3 , -4 , -2 ,  5 , -3  }
```



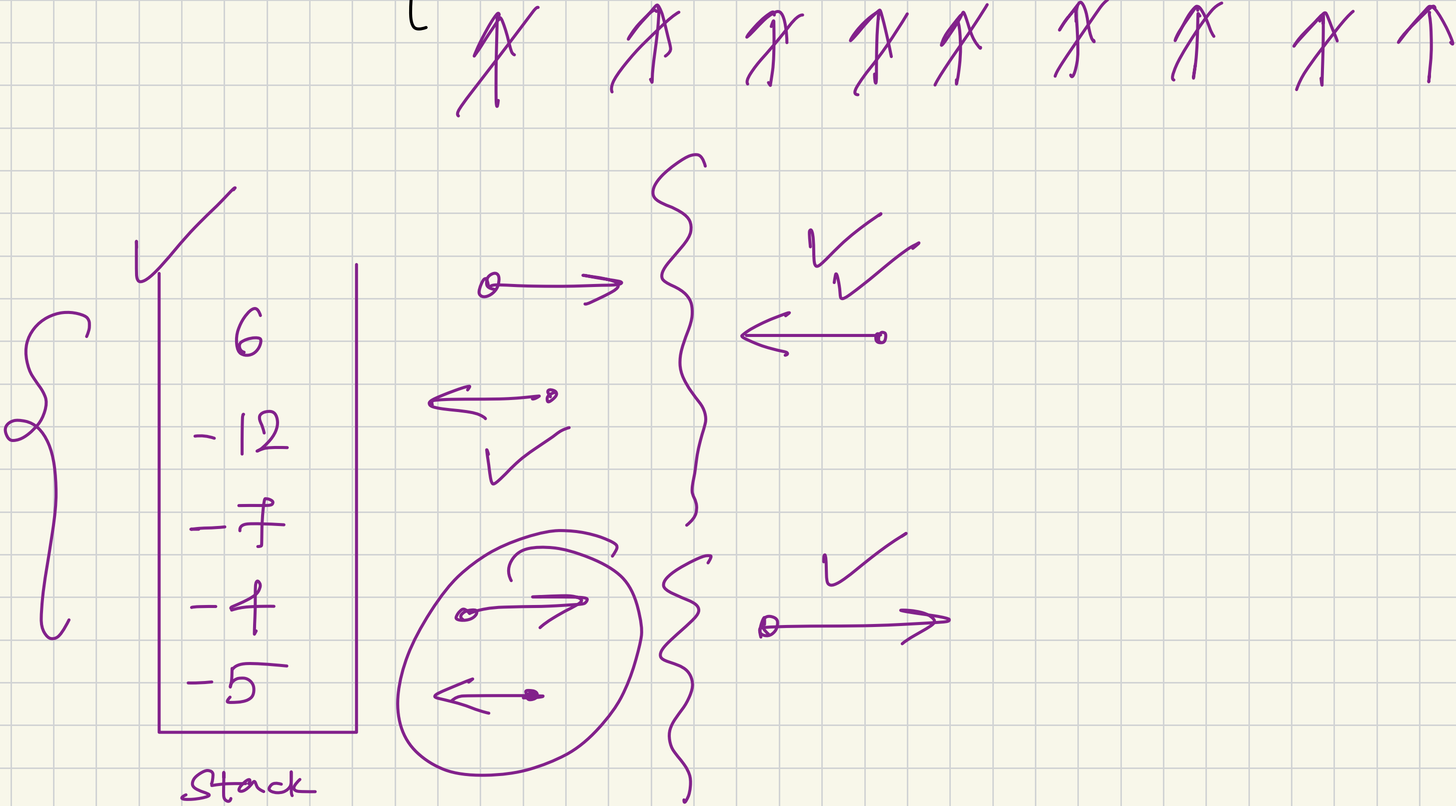

1
2
3
LIFO
Stacks

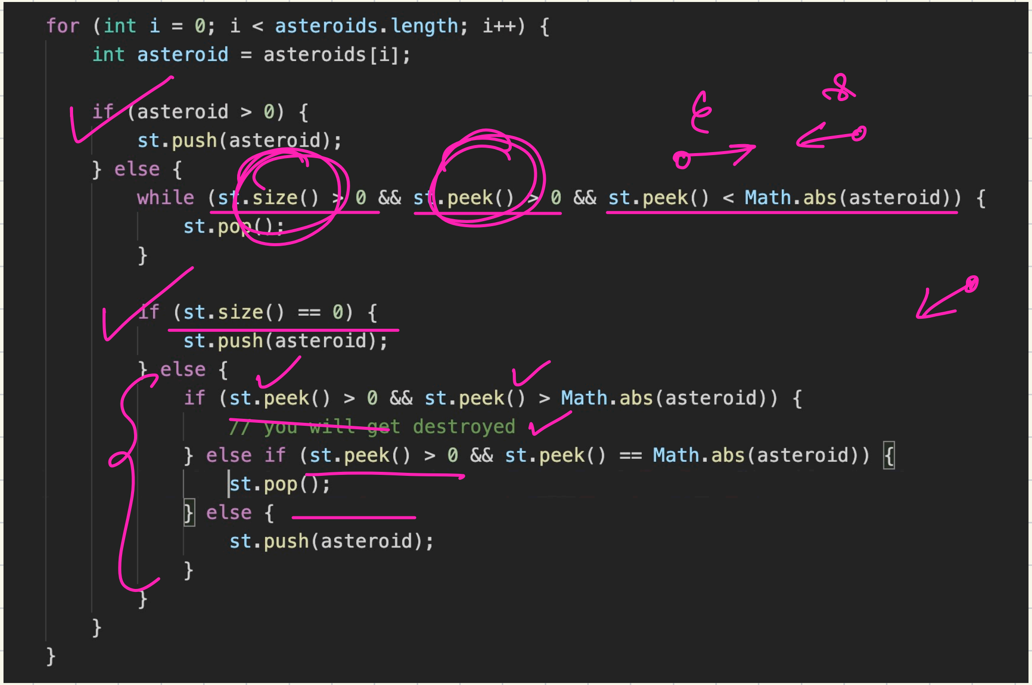
```
                     0   1   2   3   4   5   6
int[] asteroids = {  1 , 2 , 3 , -4, -2, 5 , -3 }
```



{-4, -2, 5} ✓

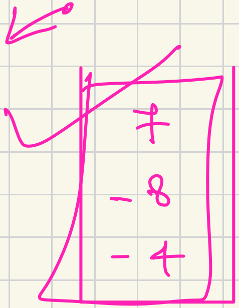
int[] asteroids = {-5, -4, -7, 8, 10, -7, -4, -12, 6}

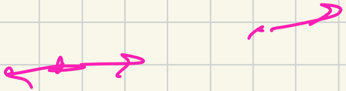


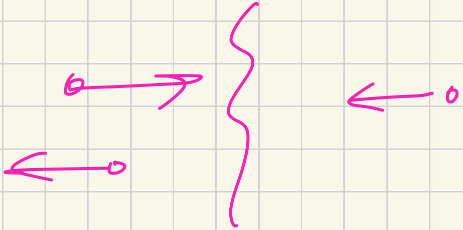


```
for (int i = 0; i < asteroids.length; i++) {
    int asteroid = asteroids[i];

    if (asteroid > 0) {
        st.push(asteroid);
    } else {
        while (st.size() > 0 && st.peek() > 0 && st.peek() < Math.abs(asteroid)) {
            st.pop();
        }

        if (st.size() == 0) {
            st.push(asteroid);
        } else {
            if (st.peek() > 0 && st.peek() > Math.abs(asteroid)) {
                // you will get destroyed
            } else if (st.peek() > 0 && st.peek() == Math.abs(asteroid)) {
                st.pop();
            } else {
                st.push(asteroid);
            }
        }
    }
}
```

{-4, 5, 6, 2, 3, -5, -8, 7}







{ TC : O(N)   SC : O(N) }

✓ Rotten Oranges

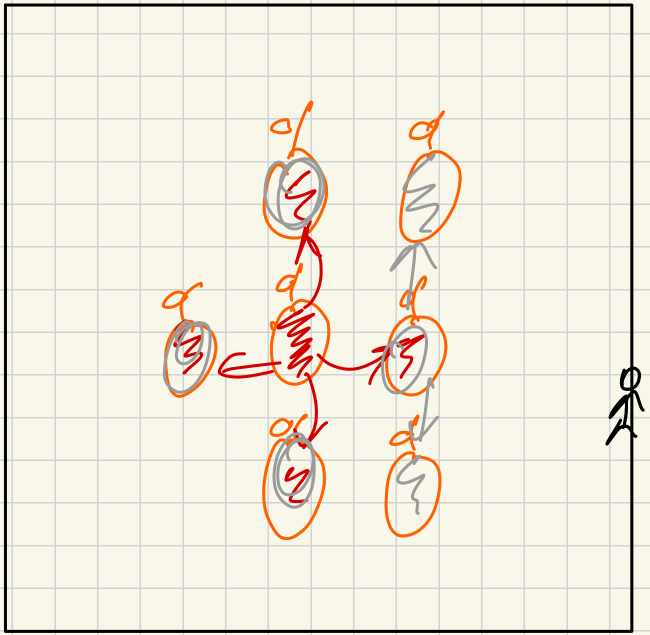
→ BFS {Breadth first Search}

→ Queues

✓ Basic

✓ Binary Trees

✓ M4 → BFS ✓



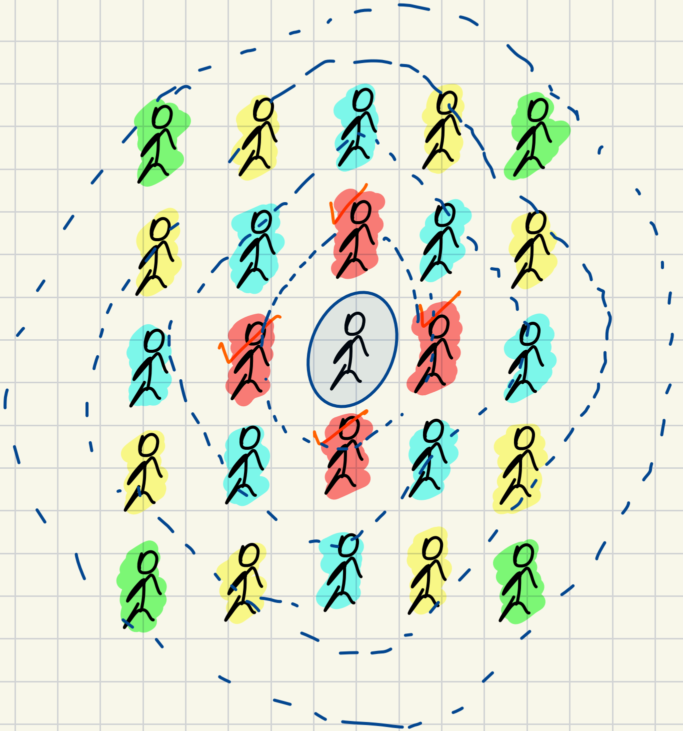
at each 1 unit of time a rotten Orange will other fresh oranges present in all 4 dir, and at 1 unit of dist.

2 unit of times, whole basket got rotten.

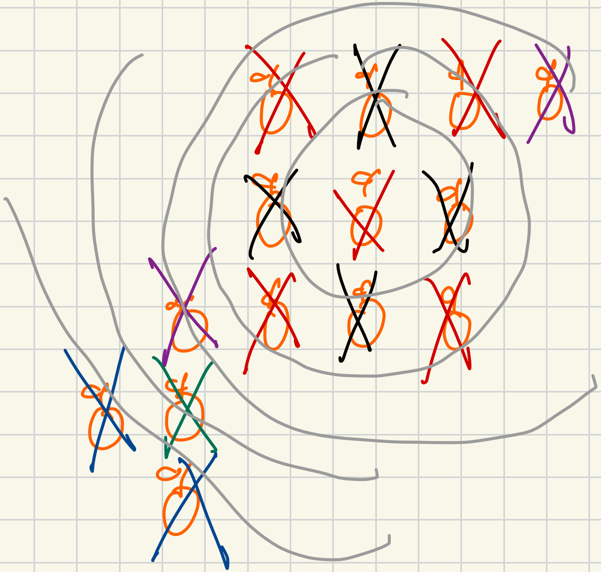
Covid

- move radially

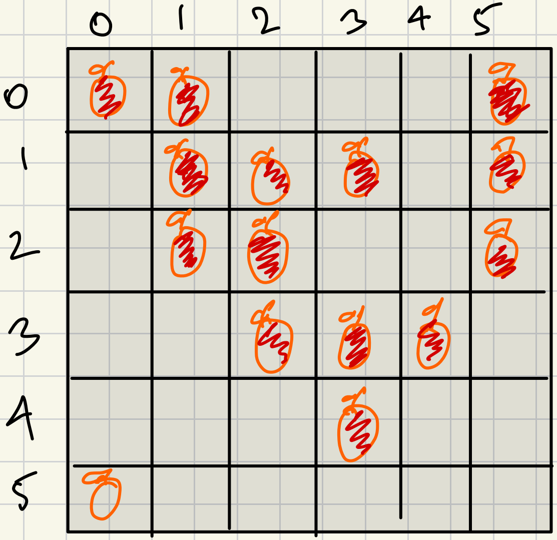
BFS



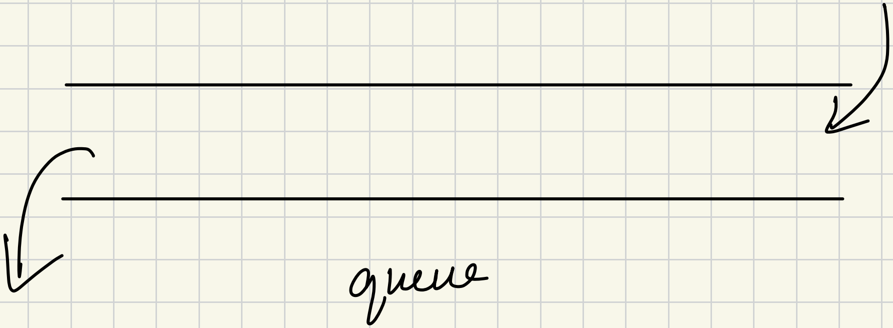
time = ~~1~~ ~~2~~ ~~3~~ 4



time = 5 units of time



level = ~~0~~ ~~1~~ ~~2~~ 3 ✓



size = ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0

time = 2

✓ level - 1