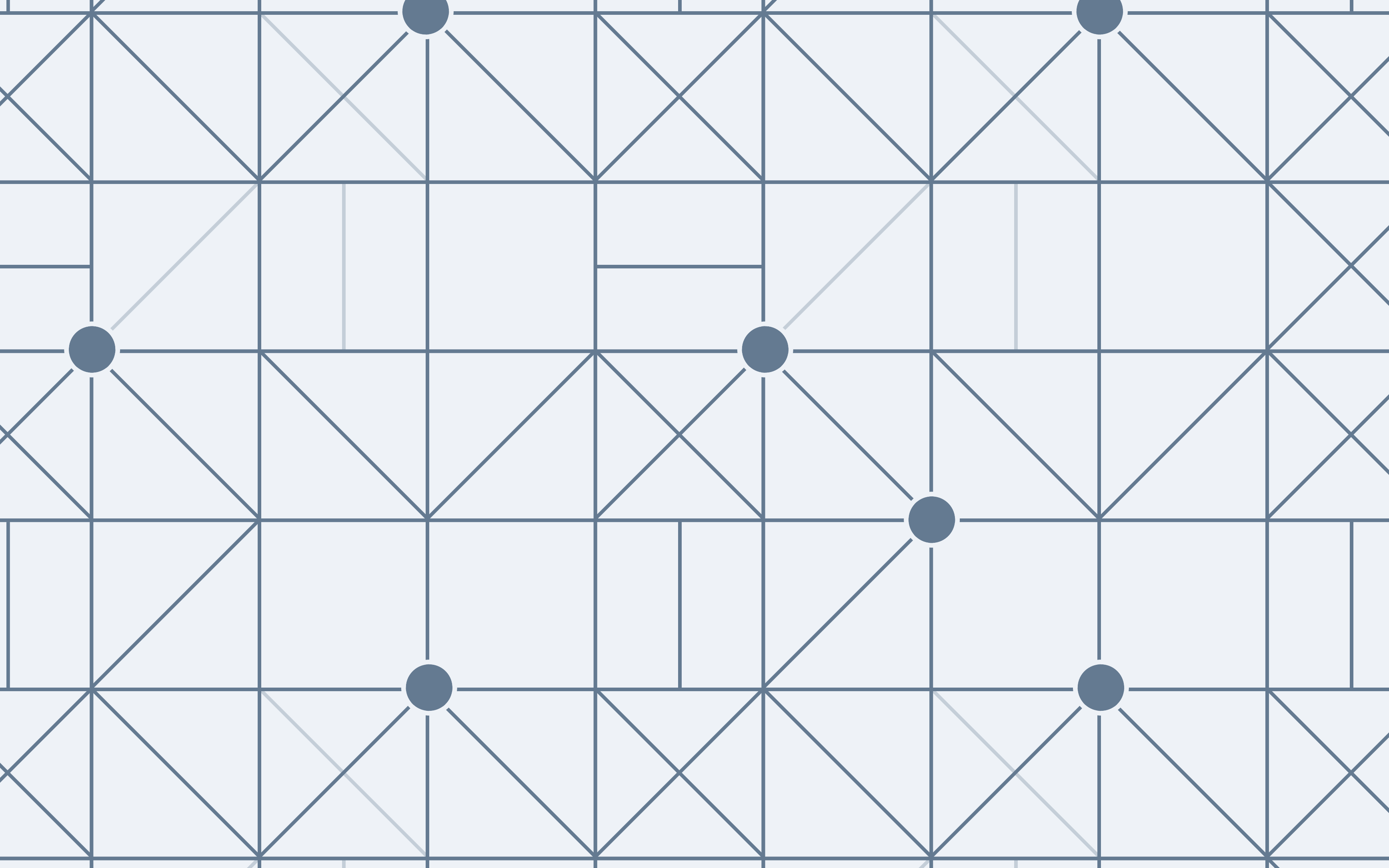


## Sliding Window Maximum.

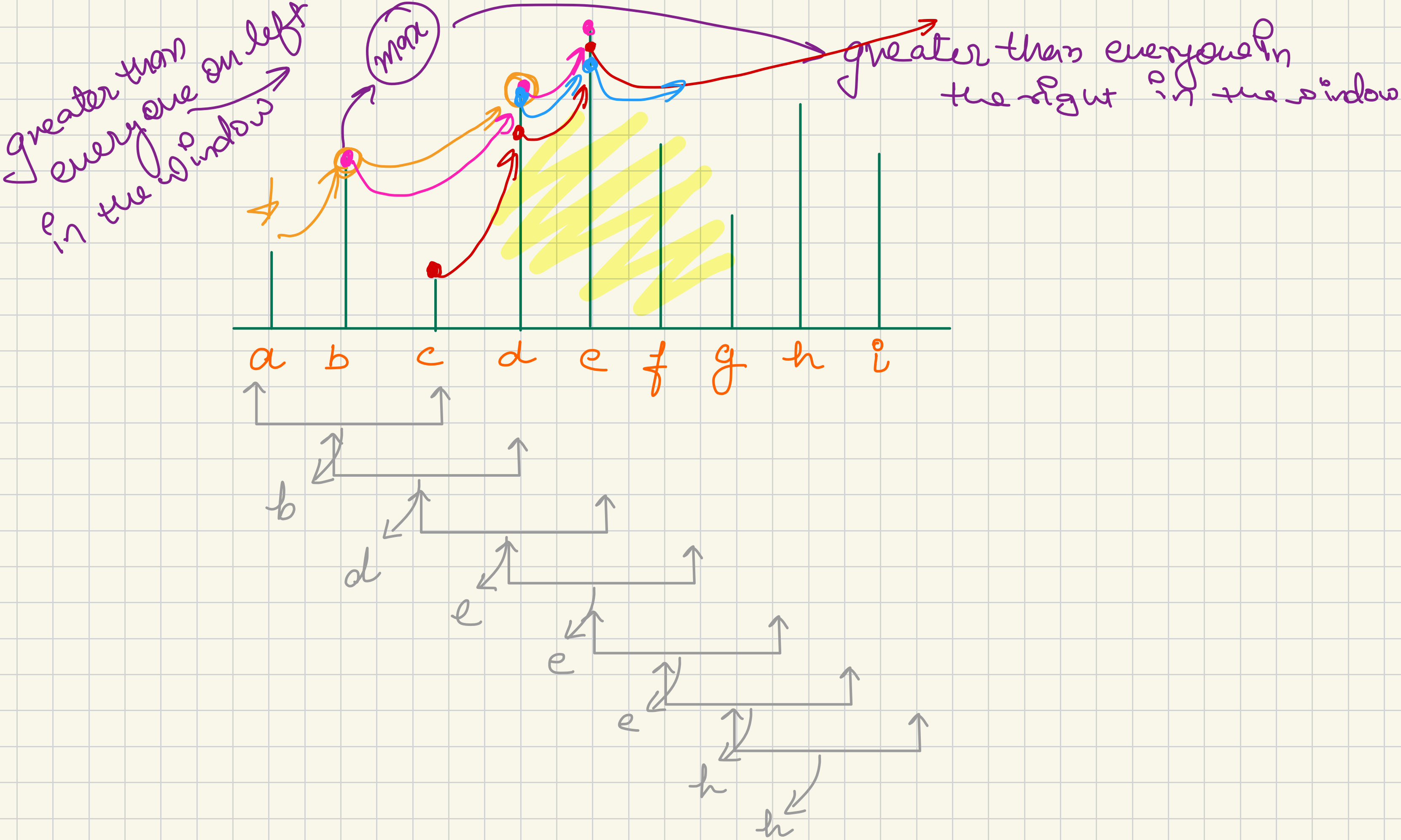
```
             0  1  2  3   4  5  6  7
int[] arr = {1, 3, -1, -3, 5, 3, 6, 7};        K=3

             index minimum
            {3, 3, 5, 5, 6, 7}
```

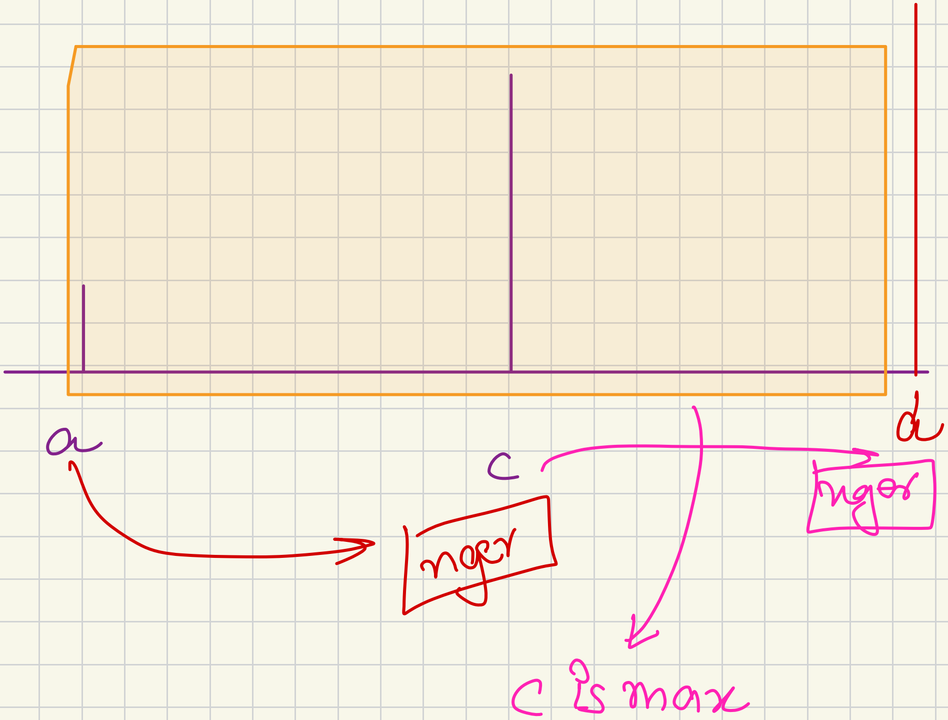
## Brute Force.

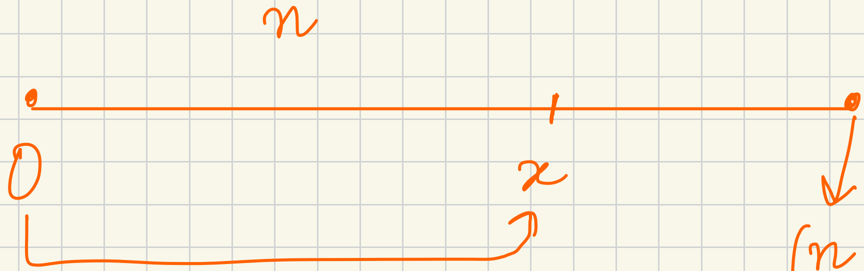
```
for (int i=0 ⟶ n - K)
{   int max = -∞;
    for (int j=i ; j < i+K ; j++)
    {
        max = Math.max(max, arr[j]);
    }
} ans.add(max);
```

TC: O(N*K)

SC: O(1)


greater than everyone on left in the window
max
greater than everyone in the right in the window
a
b
c
d
e
f
g
h
i
b
d
e
e
e
h
h


a
c
d
ngr
ngr
c is max



$(n-1) = x + K - 1$

✓ $\boxed{n - K = x}$

✓ $\boxed{0 \rightarrow n-k}$

$\rightarrow (n-K+1)$ windows |

$$\begin{array}{lccccccccl} & & \swarrow \overset{0}{j} & & & & & & & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \\ \text{int[] arr} = \{ & 1, & 3, & -1, & -3, & 5, & 3, & 6, & 7 & \} \\ & & \uparrow & & & & & & & \\ & & \overset{0}{i} & & & & & & & \end{array}$$

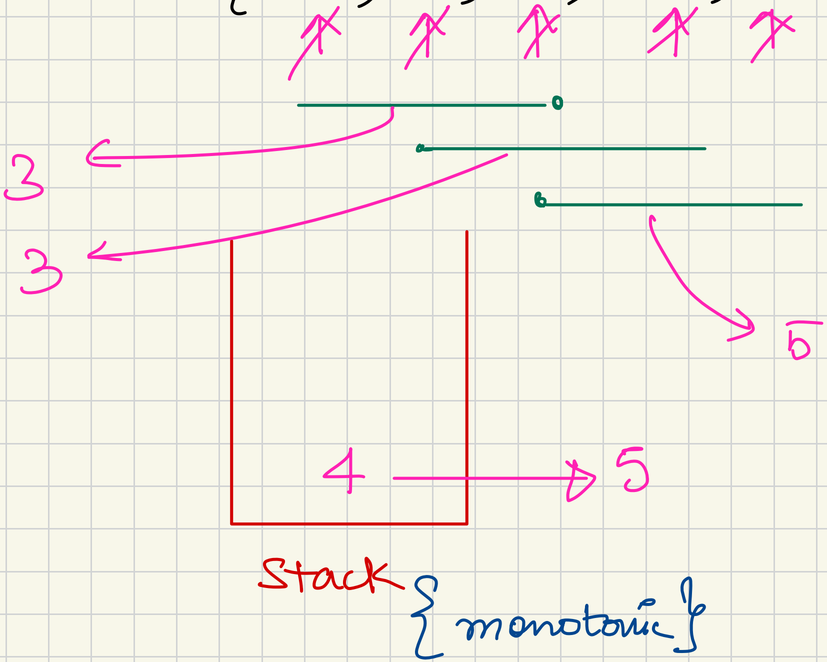
$K=3$

3, 3

nger[] = {1, 4, 4, 4, 6, 6, 7, 8}

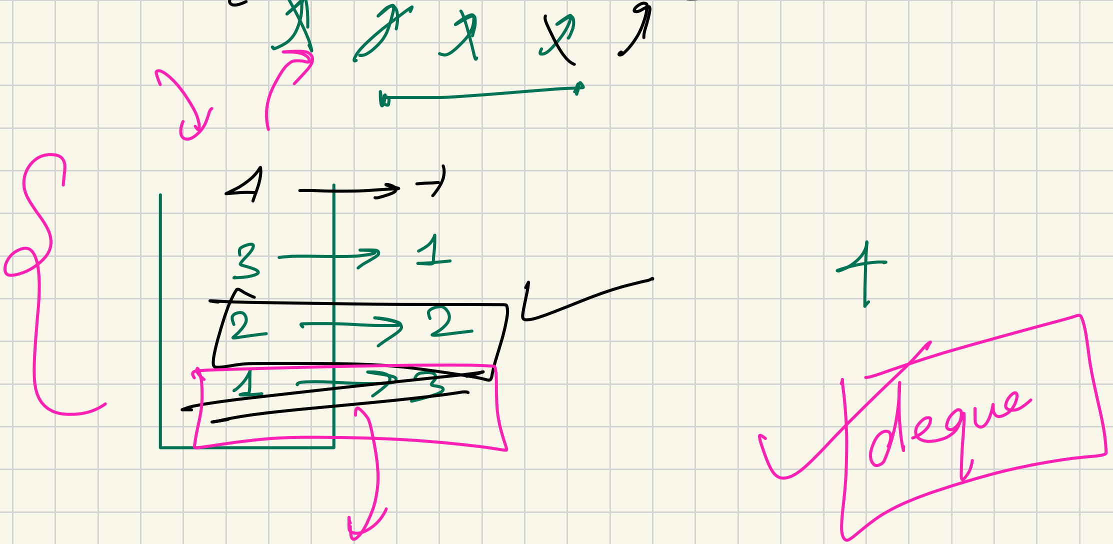
# sliding window maximum.

$$\begin{array}{cccccccccc} & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ arr & = & \{\ 1, & 3, & -1, & -3, & 5, & 3, & 6, & 7\ \} \end{array}$$

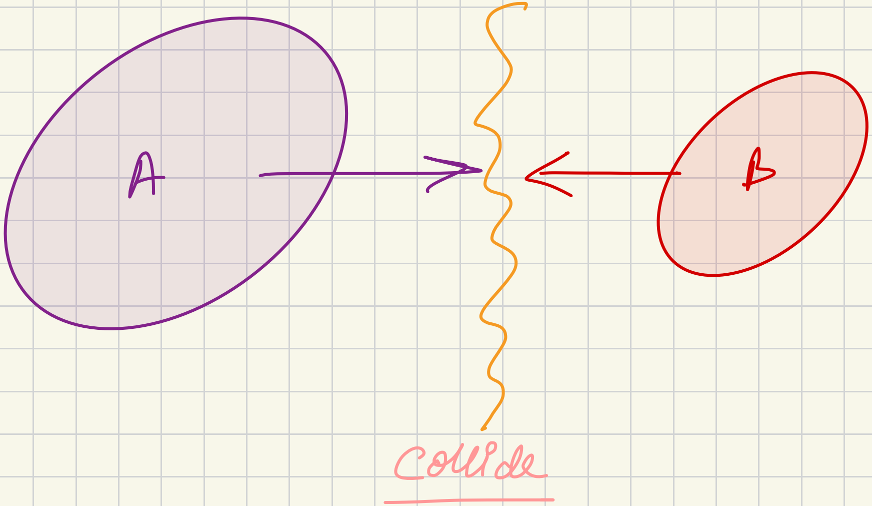


$$\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \{4, & 3, & 2, & 1, & -1\} \end{array}$$

$k = 3$



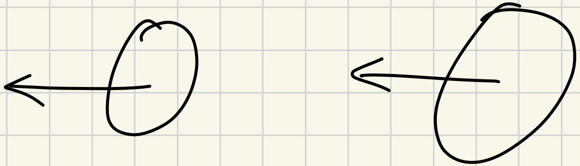
# Asteroid Collision



NOTE: Smaller one will get destroyed, and bigger remains unaffected

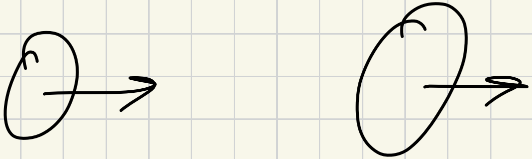
if same size asteroid collids, both will get destroyed

## Case 1



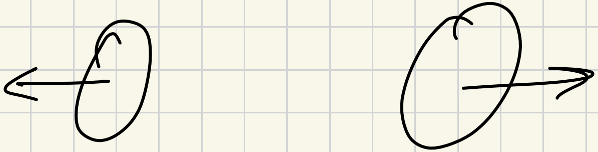
no collision

## Case 2



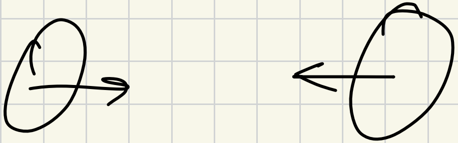
no collision

## Case 3



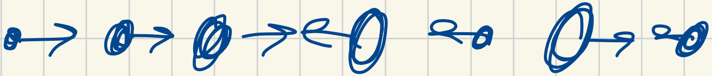
no collision

## Case 4



yes collision

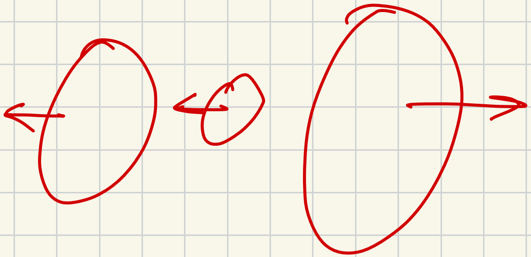
asteroids [ ] = $\{ 1, 2, 3, -4, -2, 5, -3 \}$



(+) ve → moving right

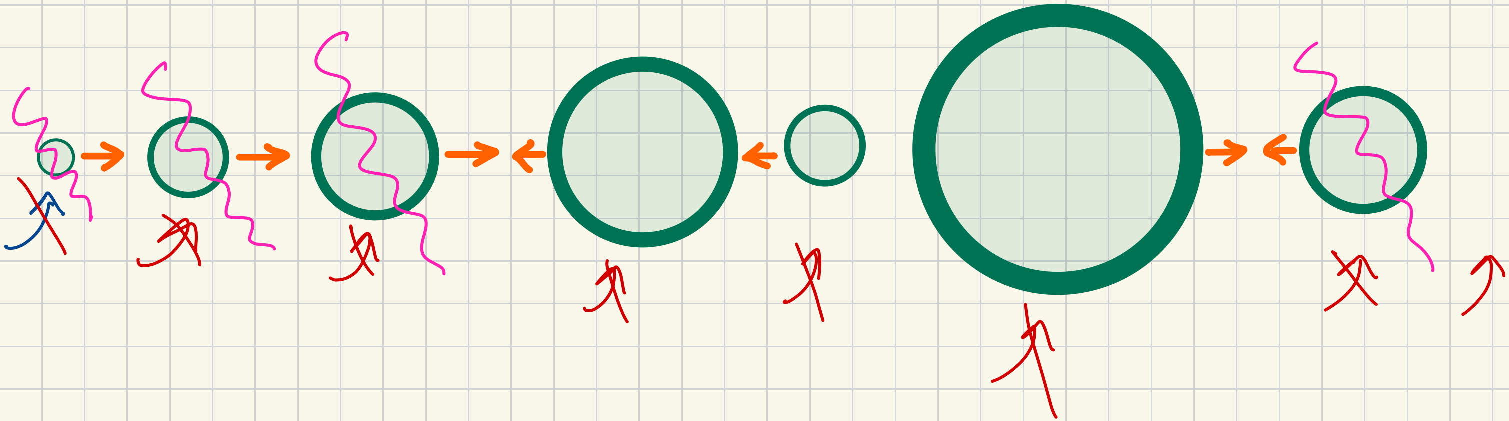
(−) ve → moving left

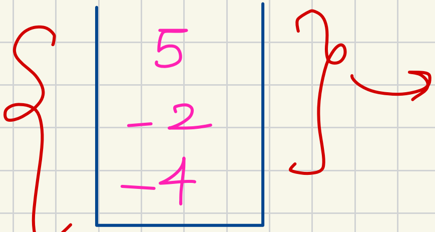
asteroids [] = { 1, 2, 3, -4, -2, 5, -3 }



{ -4, -2, 5 } → ans!

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





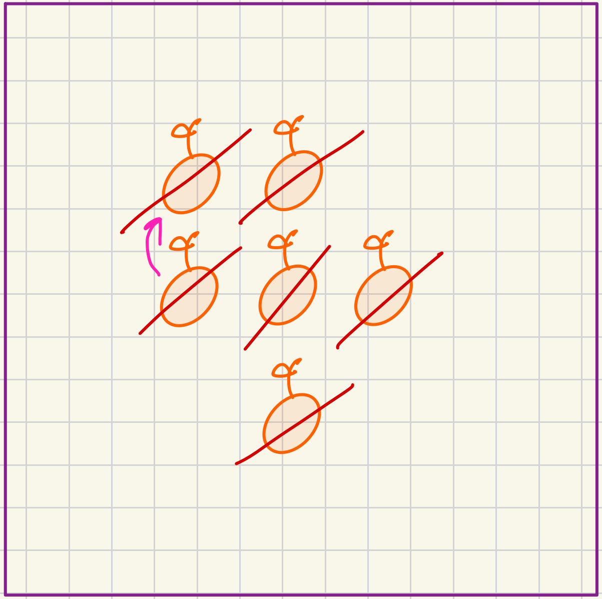
stack

neutral people in universe .

(known universe)

# Rotten Oranges :

at each unit of time a rotten orange will rotten fresh oranges at all 4 dir, at 1 unit of dist.



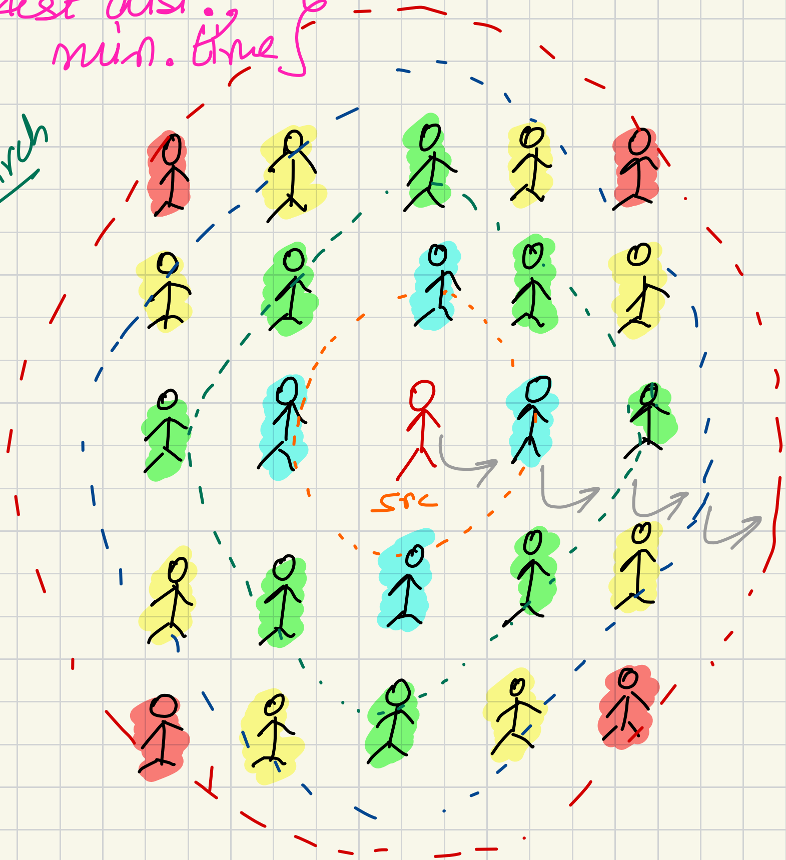
time = 2 units }

BFS

Breadth First Search

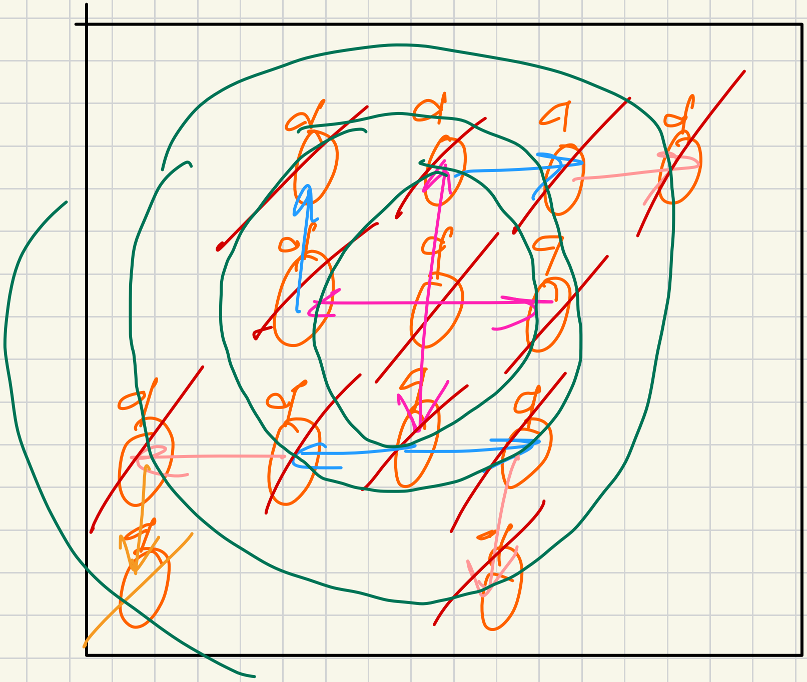
Queues

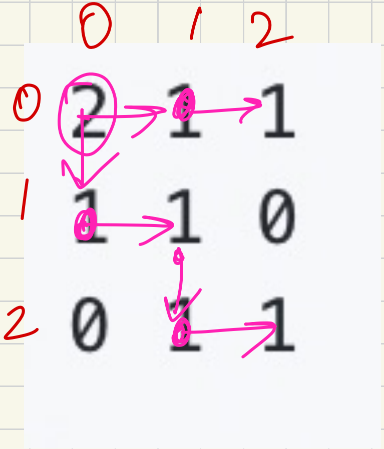
shortest dist.,
min. time



fastest way of
spreding is
radialy

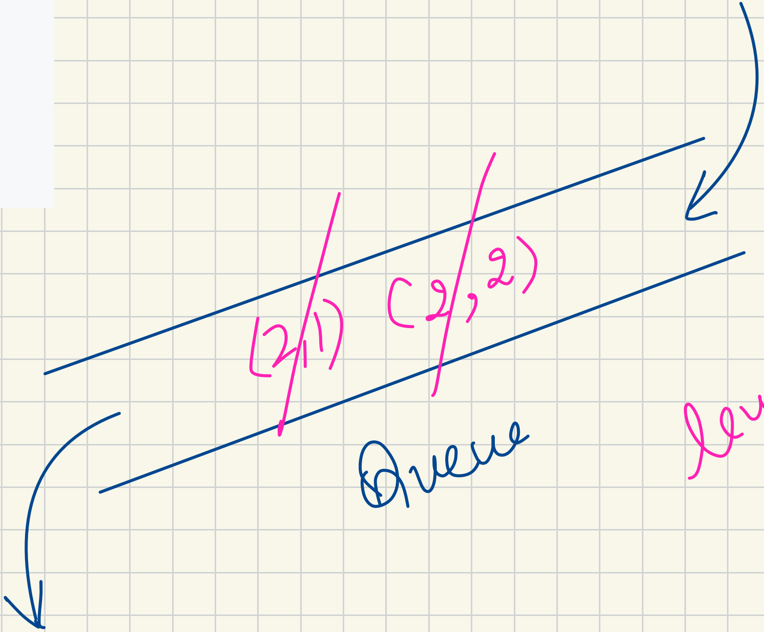
level = 4



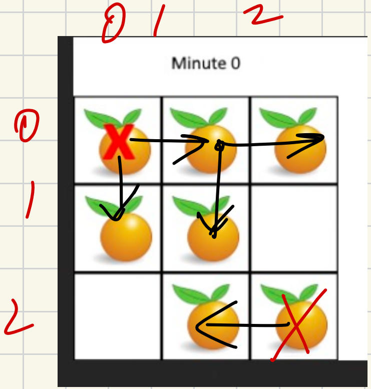


$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 2 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 2 & 0 & 1 & 1 \end{array}$$

2 → rotten
1 → fresh
0 → empty



level = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4



time = level-1

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

level = ~~0~~ ~~1~~ ~~2~~ 3