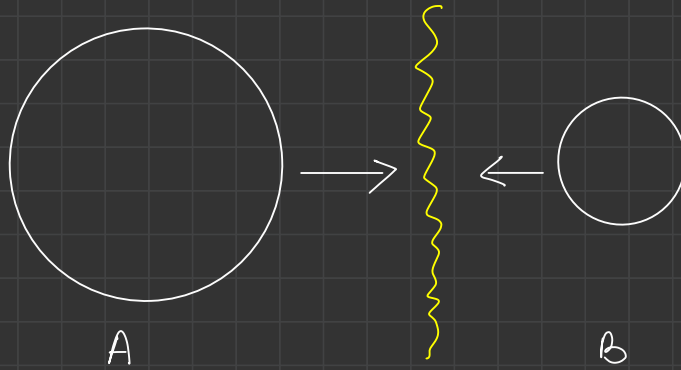
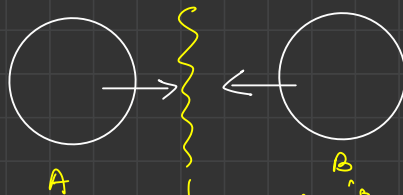




Asteroid collision

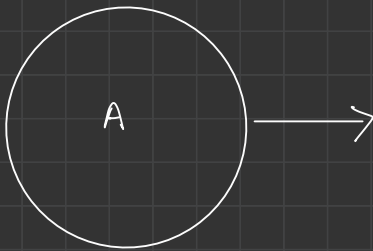


} collision
} smaller one will get destroyed }

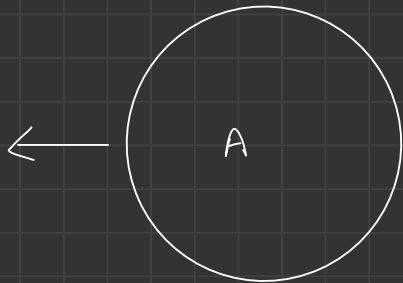


} collision } both will get destroyed }

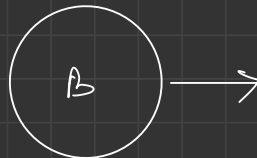
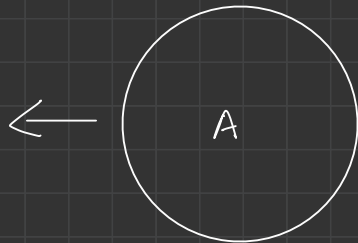
NOTE Every
asteroid is moving
with same
speed.



they will never collide.



No Collision



No Collision

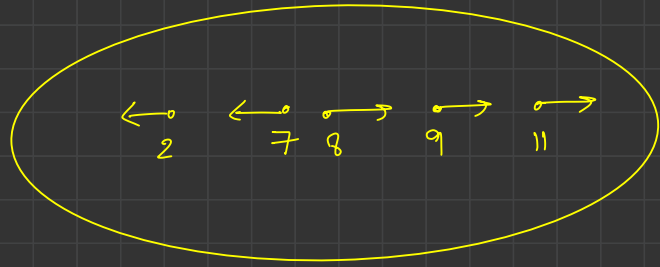
asteroids[]: { -2, 3, 4, -4, 5, -7, 8, 9, -3, -2, 11 }

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

11
9
8
-7
-2

stack

(freely moving asteroids)















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

ans

Rotten Oranges

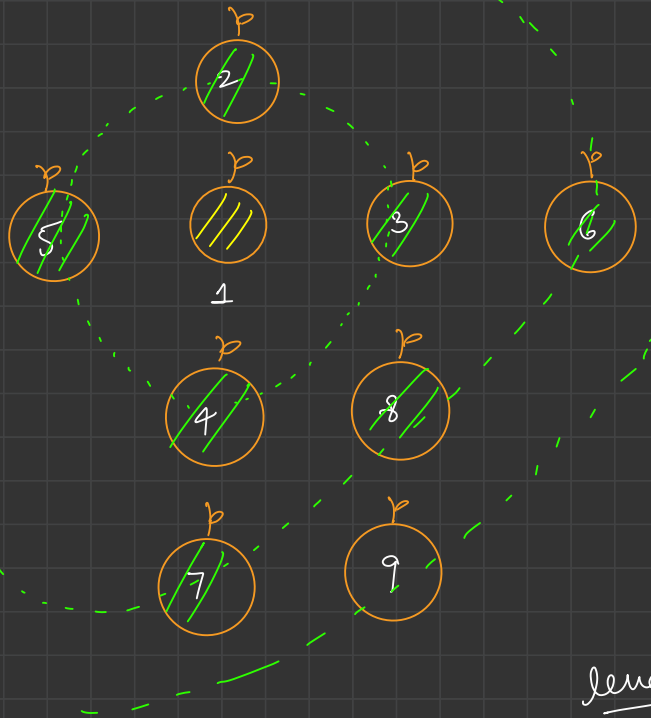
0 → Empty
1 → fresh
2 → rotten

min^m amt of time, in which
all oranges will be rotten up,
if not possible return -1

NOTE: A rotten can rotten fresh orange in
1 unit of time in all 4 dir.

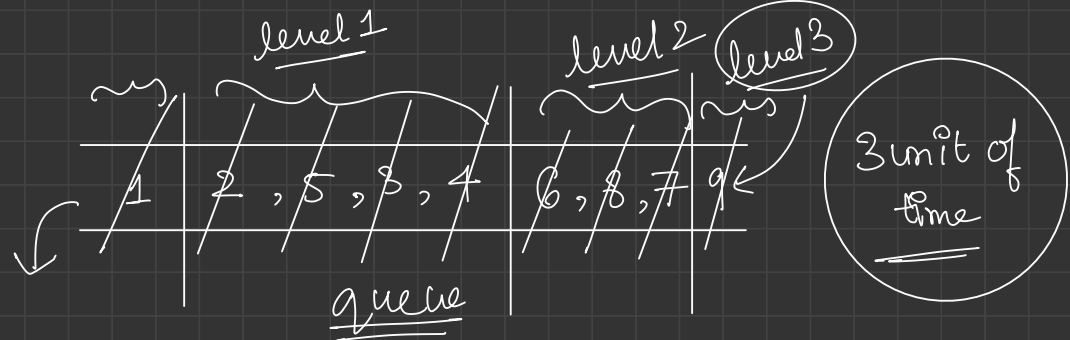
time = ~~1~~ ~~2~~ 3

















BFS ✓
↳ Breadth first Search
↳ min

DFS
↳ depth first search

FIFO



	0	1	2	3	4
0					
1					
2					
3					
4					

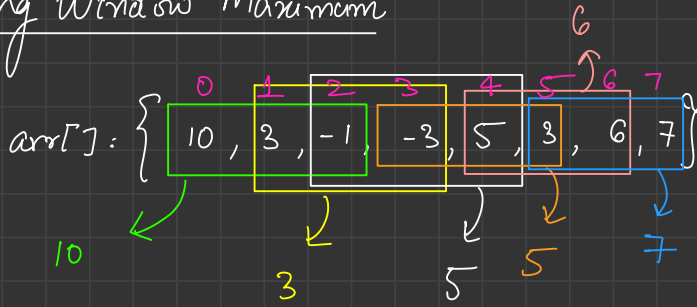
~~(2, 1)~~

level = ~~0~~
~~1~~
~~2~~
~~3~~ ~~4~~ ~~5~~ ~~6~~ 7

size = ~~1~~ 8

Sliding Window Maximum

$n = 8$



k=3

{ 10, 3, 5, 5, 6, 7 } → ans

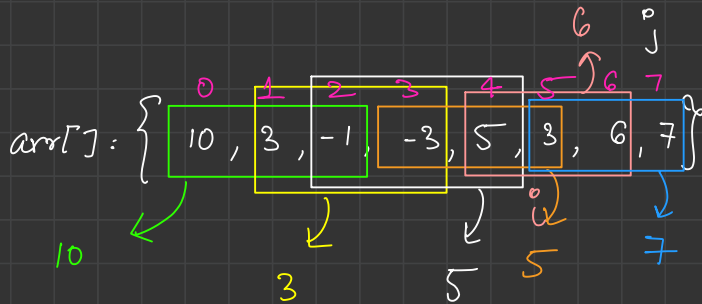
TC: $O(N \cdot K)$
SC: $O(1)$

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

TC: $O(N)$?

$N=8$

$K=3$



ngexil[]: { 8, 4, 4, 4, 6, 6, 7, 8 }

$i=0$ \longrightarrow 10

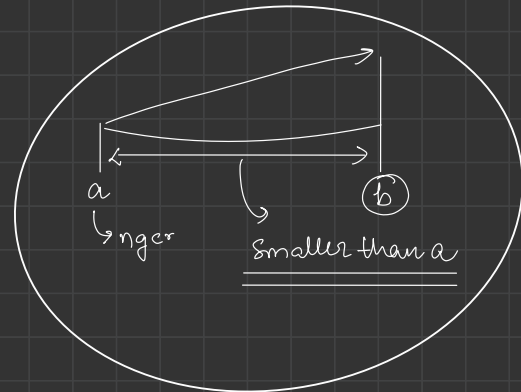
$i=1$ \longrightarrow 3

$i=2$ \longrightarrow 5

$i=3$ \longrightarrow 5

$i=4$ \longrightarrow 6

$i=5$ \longrightarrow 7



$K=3$

```
// N - K + 1 windows can be opened
int[] ans = new int[N - K + 1];

int j = 0;
for (int i = 0; i <= N - K; i++) {
    if (j < i) {
        j = i;
    }

    while (ngeri[j] < i + K) {
        j = ngeri[j];
    }

    ans[i] = arr[j];
}

return ans;
```

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

j

i

$ngeri[]: \{ 8, 4, 4, 4, 6, 6, 7, 8 \}$

$\{ 10, 3, 5, 5, 6, 7 \}$

$TC: O(N)$

$SC: O(N)$

$TC: O(N), SC: O(K)$ ✓

