



Agenda

- ① Sliding window maximum
- ② Asteroid collision
- ③ Rotten Oranges

Sliding window maximum 0

int[] arr = { 1, 3, -1, -3, 5, 3, 8, 12 }

int K = 3

maxWin[] = { 3, 3, 5, 5, 8, 12 } → ans

Brute force 0

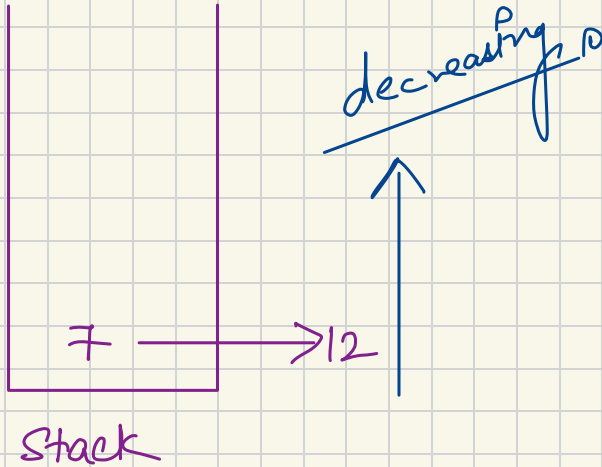
TC: $O(N \times K)$
SC: $O(1)$

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

int[] arr = { 0, 1, 2, 3, 4, 5, 6, 7 }
 { 1, 3, -1, -3, 5, 3, 8, 12 }
 (Each element in the array is crossed out with a diagonal line.)

int K = 3

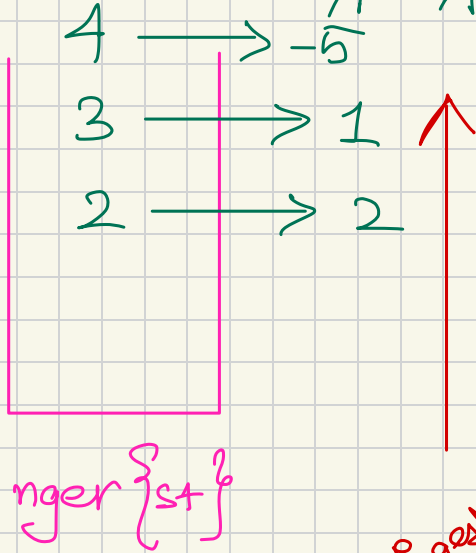
{ 3, 3, 5, 5, 8, 12 }



int[] arr = { 4, 3, 2, 1, -5 } k = 3

0 1 2 3 4

↑ ↑ ↑ ↑ ↑



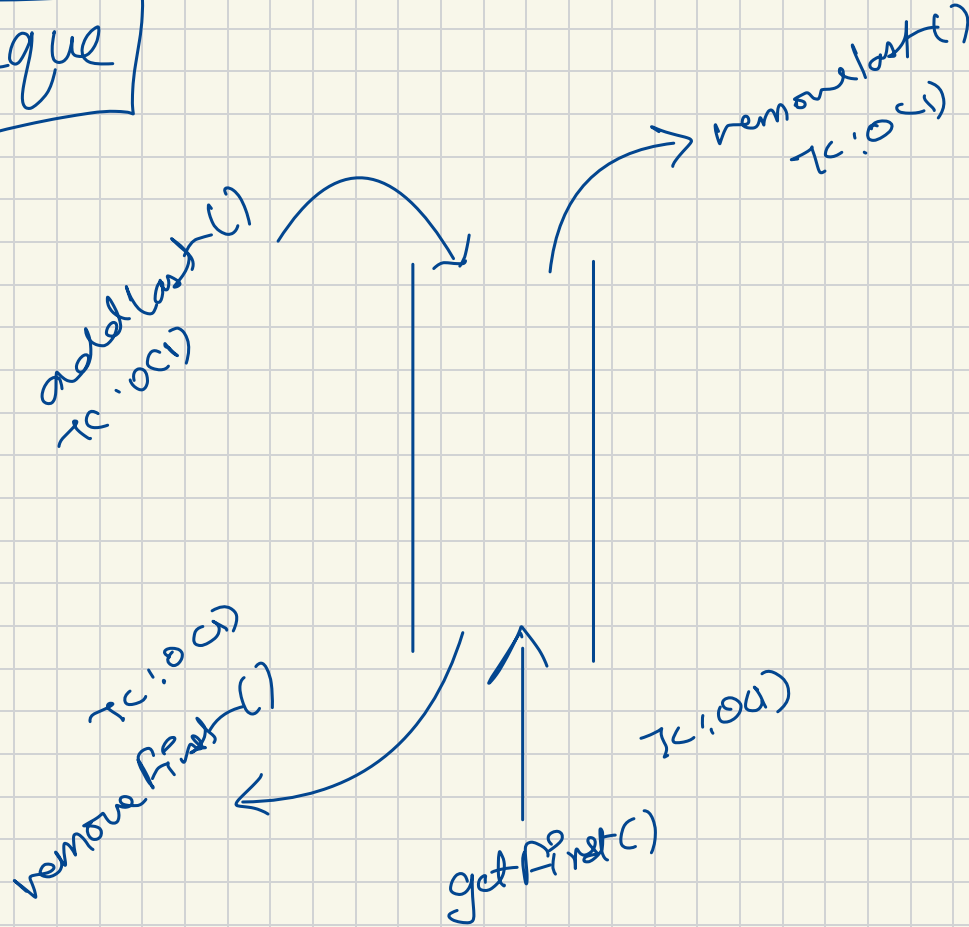
Window 1 : 4

Window 2 : 3

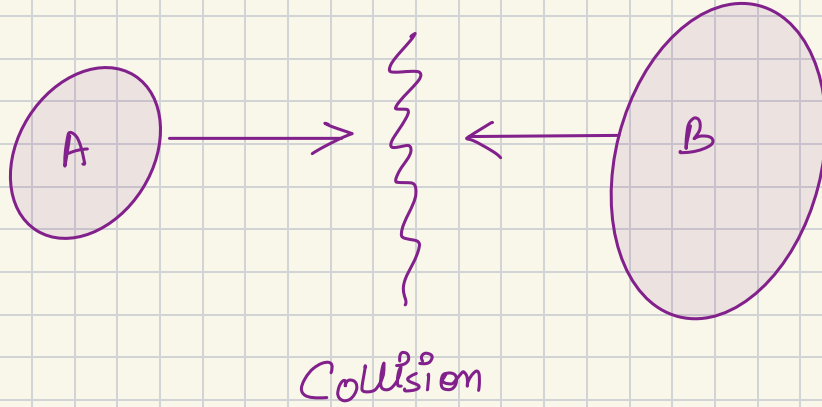
Window 3 : 2

~~biggest to smallest~~
decreasing Order!

deque



Asteroid Collision



NOTE : Smaller one will get destroyed and bigger one will be unaffected

NOTE : two asteroids of same size collide, both will get destroyed.

asteroids[] = { -3, -4, 5, 3, -3, -4, 6, -9, 10, 12, 9, 6, -10 }

(+)ve : moving right

(-)ve : moving left

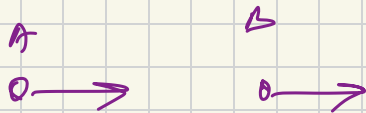
← (3) ← (4) ← (9) (16) → (12) →

—————

↳ stable universe

~~0/1~~
✓ { -3, -4, -9, 10, 12 }

Case 1



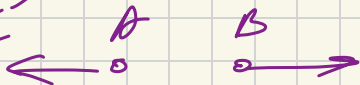
No

Case 2



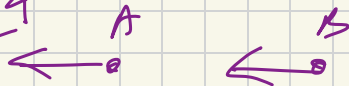
Yes

Case 3



No

Case 4

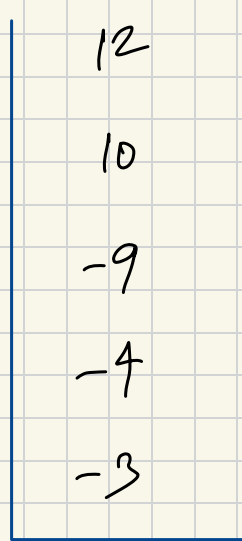


No

asteroids[] = { -3, -4, 5, 3, -3, -4, 6, -9, 10, 12, 9, 6, -10 }

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

No collision
⇓



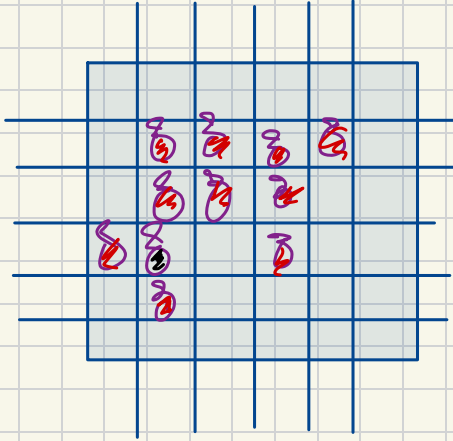
{ -3, -4, -9, 10, 12 } ^{o/p}

TC: O(N) SC: O(N)

{ known universe }

Rotten Oranges

L R F S



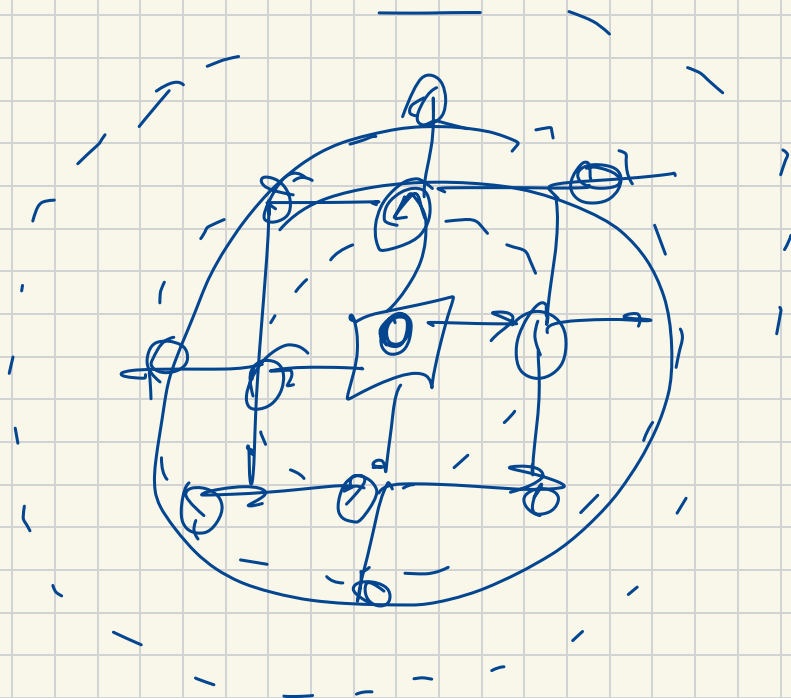
~~1 unit~~

~~2 unit~~

~~3 unit~~

~~4 unit~~

5 unit



	0	1	2	3	4
0	0	1	1	0	0
1	0	1	2	0	0
2	0	0	2	0	0
3	0	0	0	0	1
4	0	1	1	2	1
5	0	1	1	1	1

\checkmark ~~$(2,2)$~~ ~~$(4,3)$~~ ~~$(1,2)$~~ ~~$(4,2)$~~ ~~$(4,1)$~~ $(5,4)$