Agenda

1. sliding window maximum
2. Rotten Oranges
3. Asteroid Collision

Sliding Window Maximum

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

(0 1 2 3 4 5 6 7)

value of each window max = { 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 = max(max, arr[j]);
    ans.add(max);
}
```

TC : O(N*K)
SC : O(1)

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

0 1 2 3 4 5 6 7

K = 3

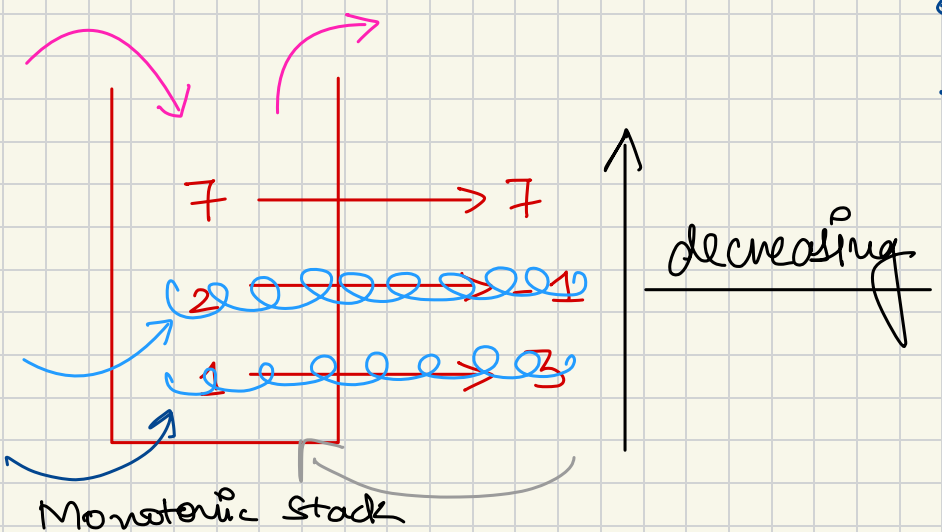
next greater element on right

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

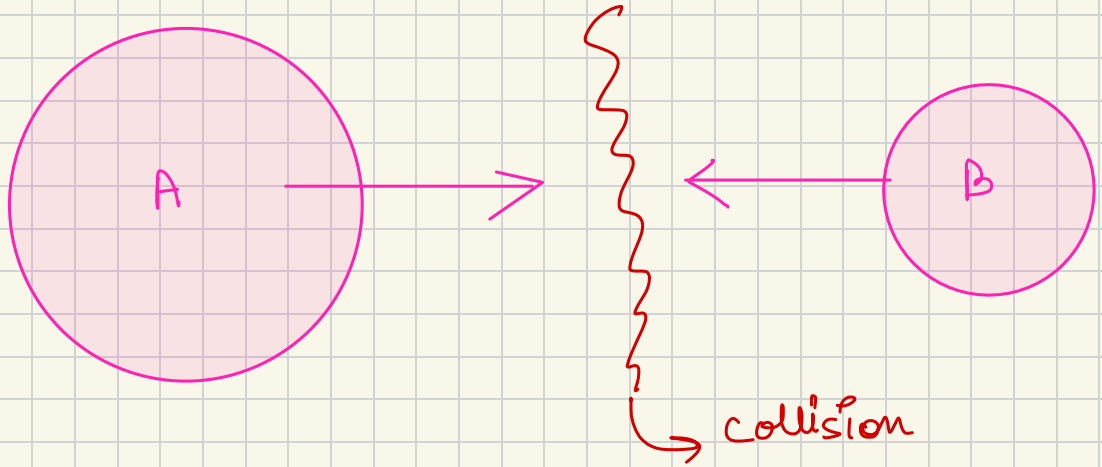
ei = x

si = x - k + 1

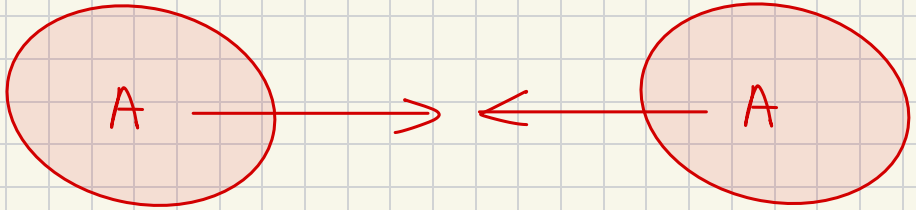
Range



Asteroid Collision.



NOTE: at time of collision, smaller one gets destroyed and bigger one stays unaffected



NOTE: if two asteroid of same size collide, Both will get destroyed.

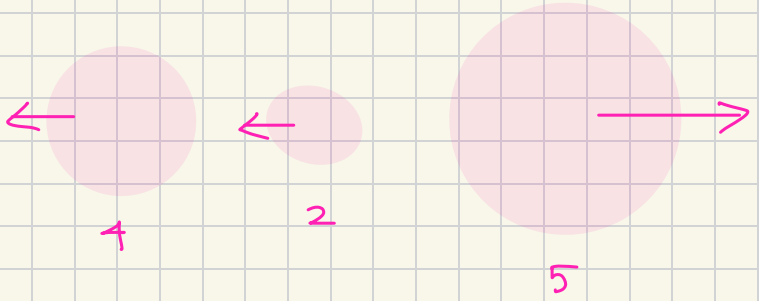
int[] = { 1, 2, 3, -4, -2, 5, -3 }

(+)ve → moving towards right

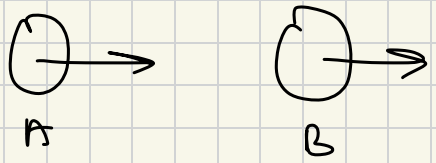
(-)ve → moving towards left

abs(val) → size of the asteroid

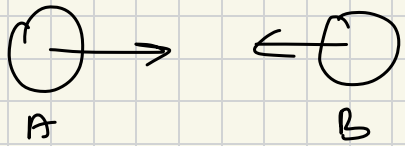
{-4, -2, 5}



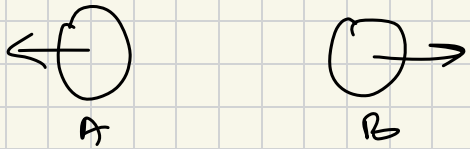
Cases



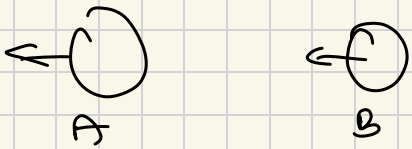
No collision



Collision



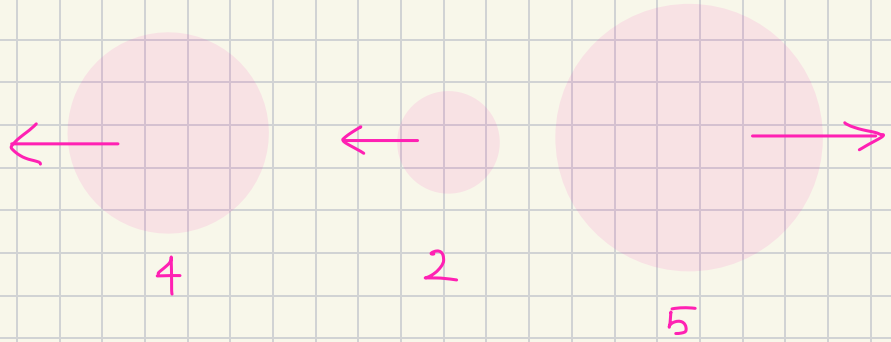
No Collision



No Collision

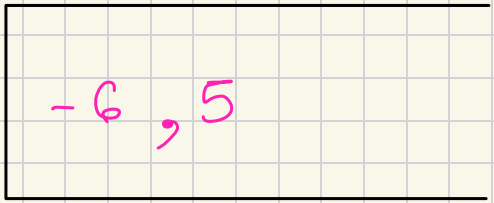
int[] = { 1, 2, 3, -4, -2, 5, -3 }

Stable universe



ans = { -4, -2, 5 }

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

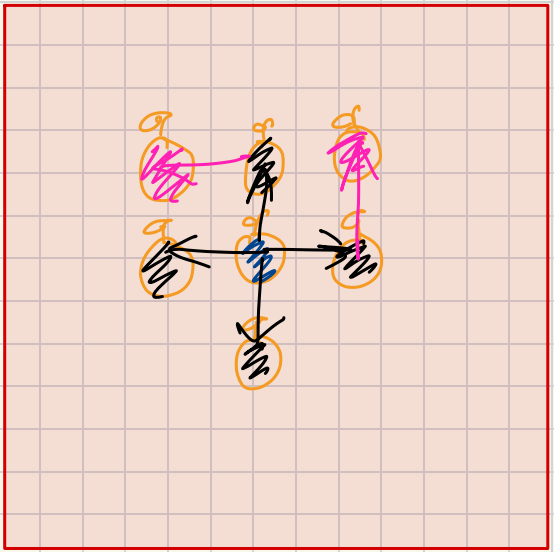


Stable universe

{ -6 , 5 }

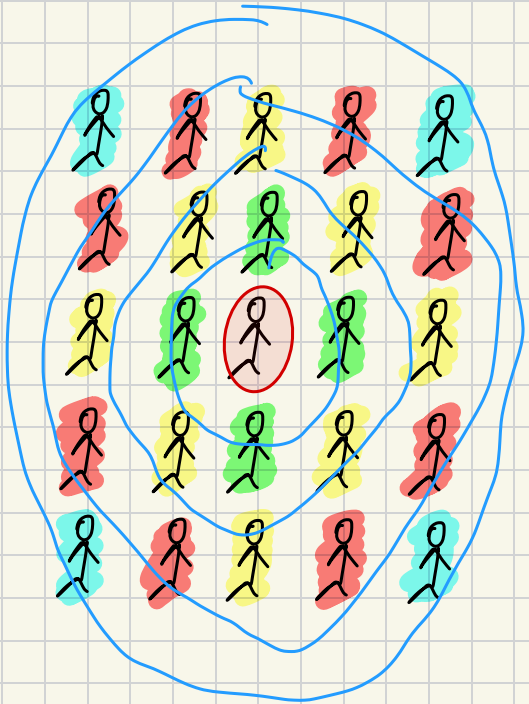
Ans !

Rotten Oranges (BFS)



it took 2 units of time to rotten all the oranges

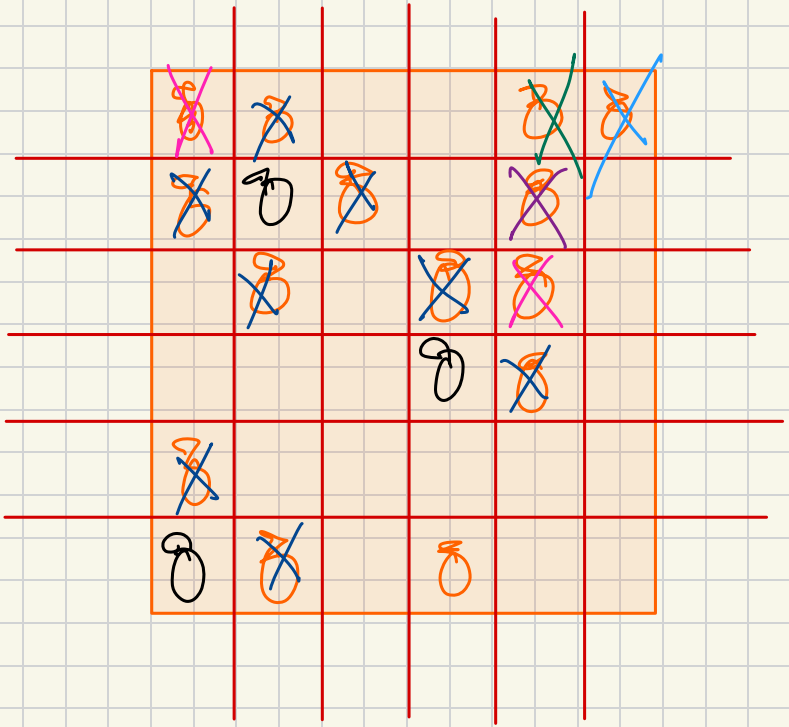
Covid



min time = 4 units

move radially

BFS



time = ~~1~~ unit

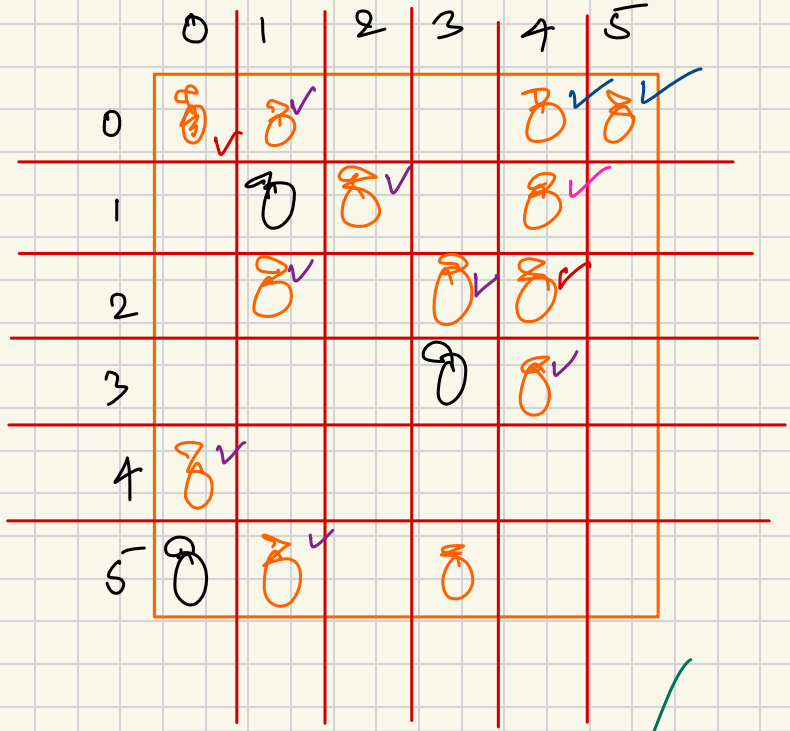
~~2 units~~

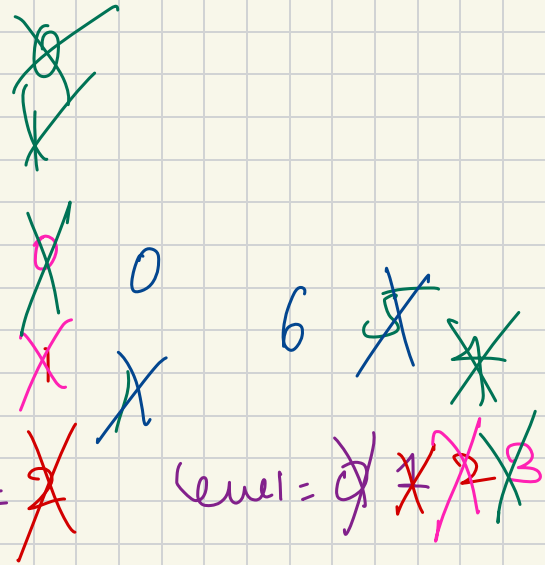
~~3 units~~

~~4 units~~

5 units

(Time = Level - 1)





(1,4) (0,4) (0,5)

H.W → (1) Height problem