

# "Hello Everyone!"

## ① Distribute Candies

$$A = [1, 5, 2, 1]$$

④  $\xrightarrow{\text{Right}}$

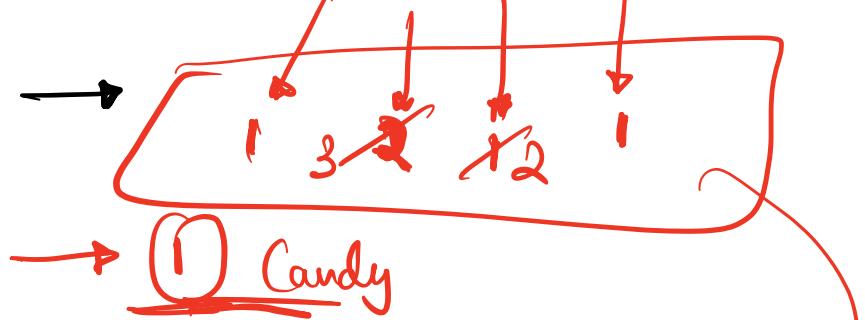
$$\underline{A(i)} > \underline{A(i+1)}$$
$$\Leftrightarrow \underline{c(i)} > \underline{c(i+1)}$$

$$\underline{A(i)} > \underline{A(i-1)}$$
$$\Leftrightarrow \underline{c(i)} > \underline{c(i-1)}$$

→ Min no. of candies

sum

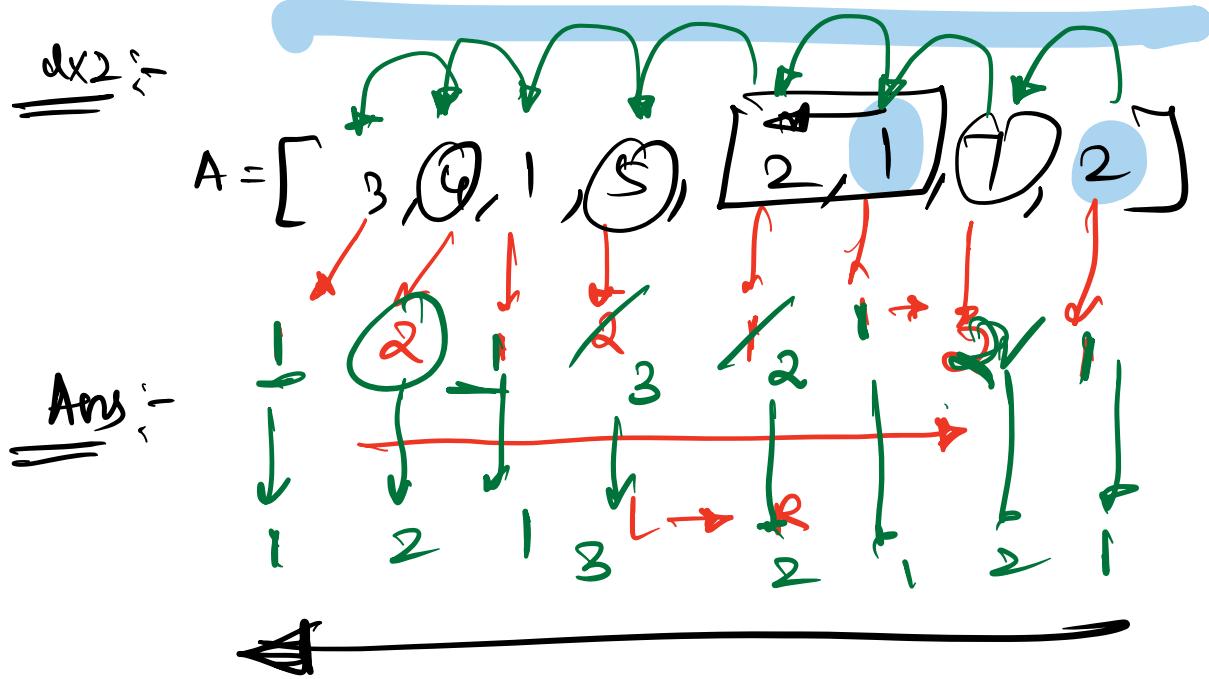
Ex1:  $A = [1, 5, 2, 1]$



at least 1

$\rightarrow \min$

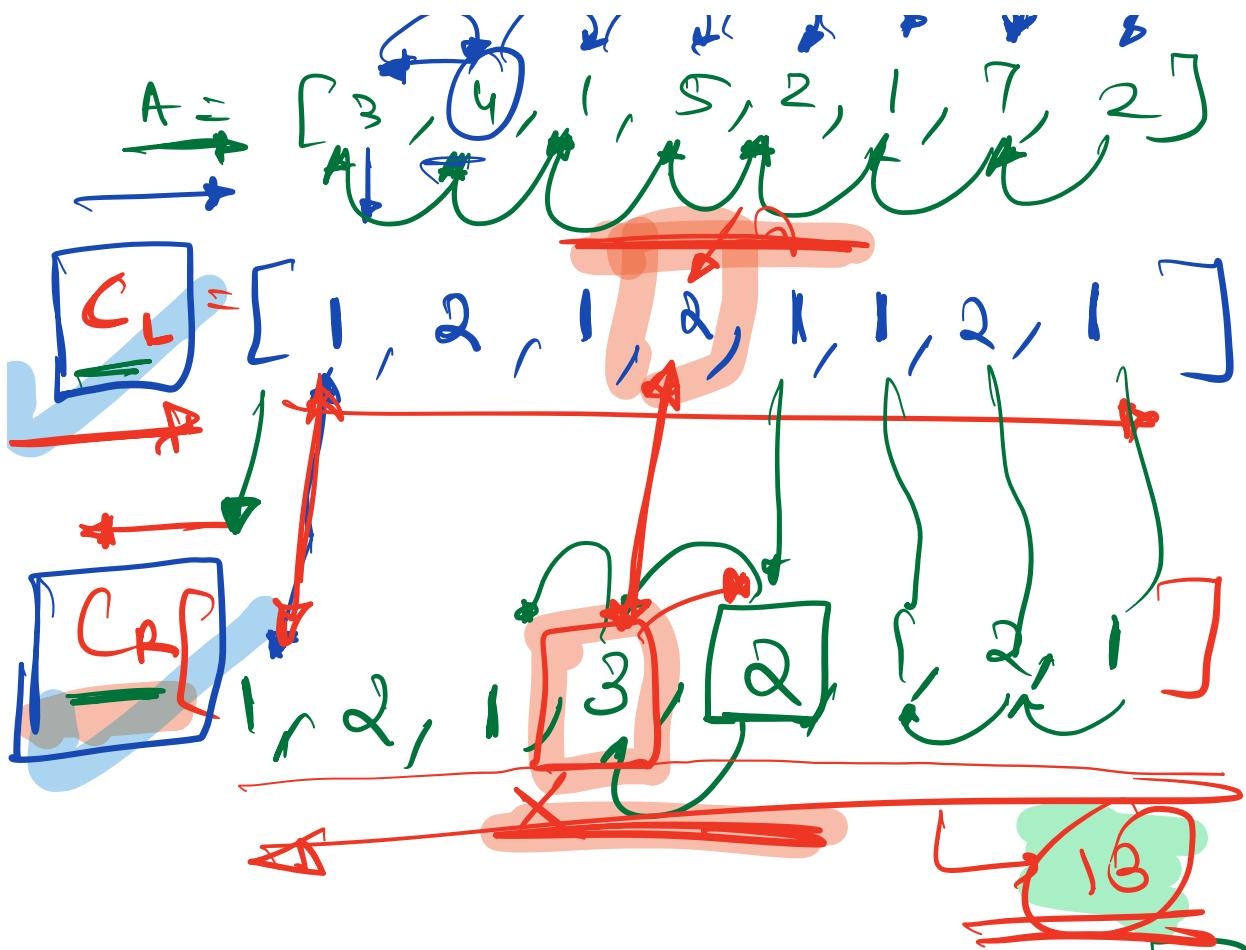
Ans:  $1 + 3 + 2 + 1 \rightarrow 7$



$$\text{Ans} = \underbrace{1 + 2 + 1}_{1} + \underbrace{3 + 2}_{1} + \underbrace{1 + 2 + 1}_{1}$$

$\text{Ans}$   $\rightarrow$  (13)





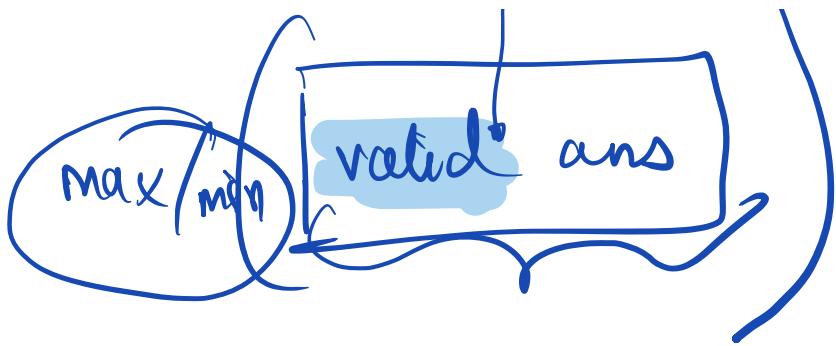
$$C = [1, 2, 1, 3, 2, 1, 2, 1]$$

~~$C = \min(C_L, C_R)$~~

$\checkmark \max(C_L, C_R)$

Correctness

min max



Pseudocode :-

C[0] = 1

for i := 1 to n - 1

L ↗ R.

A hand-drawn diagram consisting of three elements. At the top left is a blue squiggle mark. At the top center is another blue squiggle mark. A long, straight blue horizontal line extends from the right towards the left. A blue arrow points to the left along this line, starting from its leftmost end.

~~C~~ R  $C_R[n-1] = 1$   
for  $i := n-2 \rightarrow 1$

{

if  $A[i] > A[i+1]$

{

$$C_R[i] = C_R[i+1] + 1$$

{

else

$$C_R[i] = 1$$

~~Sum = 0~~  
for  $i := 0 \rightarrow n-1$

{

$$C[i] = \max(C_L[i], C_R[i])$$



$$\text{Sum} = \text{Sum} + C[i]$$

{}

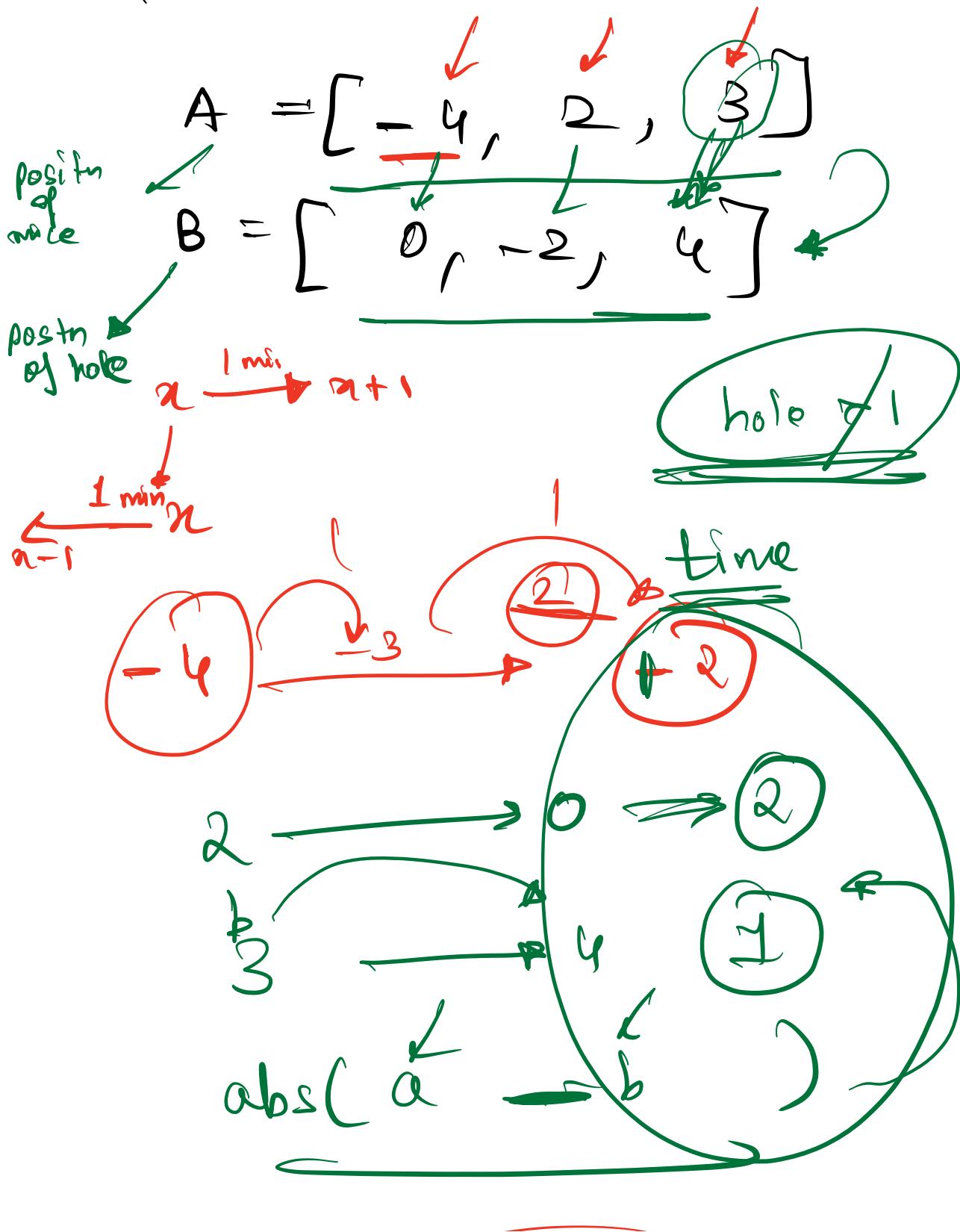
$$TC: (n+n+n)$$

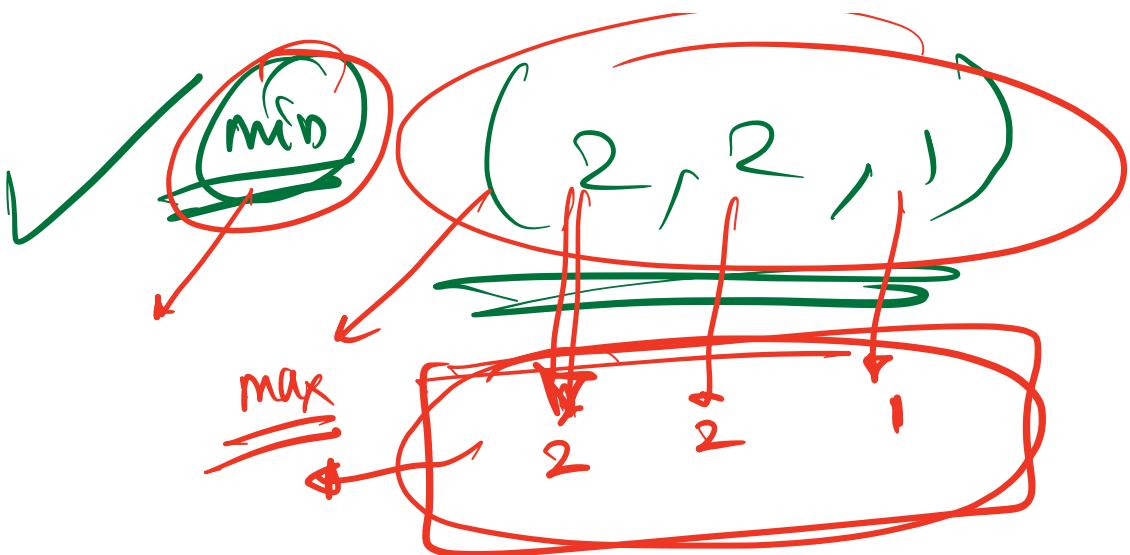
$$\boxed{TC \rightarrow O(n)}$$

$$\alpha, c_R, C \rightarrow n$$

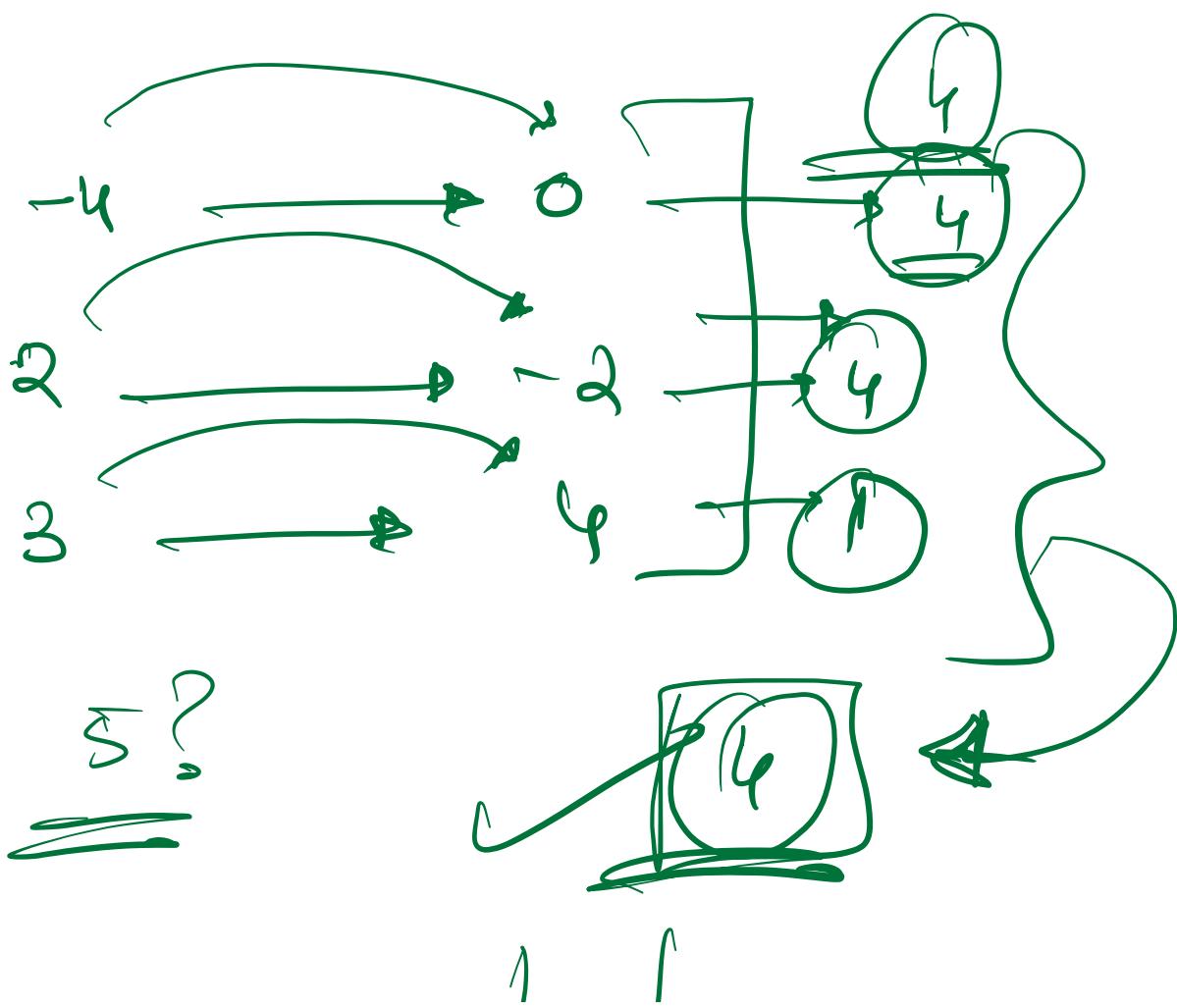
$$\xrightarrow{\quad} SC: \boxed{SC: O(n)}$$

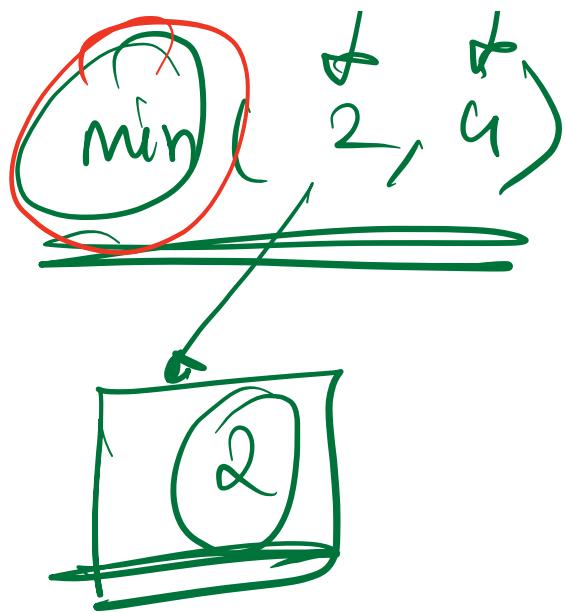
## ② Mice and Holes :





~~☆~~      Correctness       $\rightarrow$  min





Logic: → every mice gets assigned to the closest hole

unoccupied ↓  
that I have handled

greedy ✓

$$A = [-4, 2, 3]$$

$$B = [0, -2, 4]$$

$$\Rightarrow \text{sort } A = [-4, 2, 3]$$

$$\Rightarrow \text{sort } B = [-2, 0, 4]$$

$s = 1$

Proof

## Proof of Correctness

① let  $i_1 < i_2$  be the pos of 2 mile

$\underline{i_1 i_2}$

② let  $j_1 < j_2$  be the pos of holes

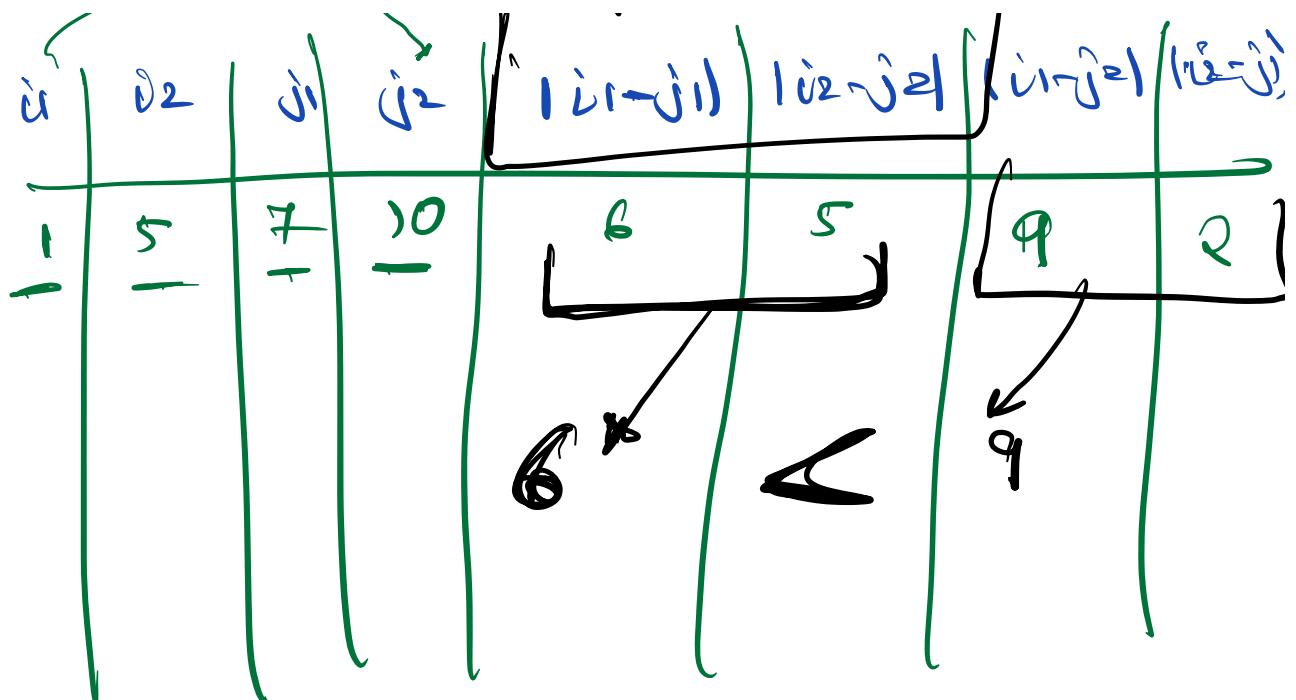
$$\max(|i_1 - j_1|, |i_2 - j_2|)$$

~~ans~~

$$\max(|i_1 - j_2|, |i_2 - j_1|)$$

where  $|a - b|$  represents absolute diff. of  $(a - b)$





$j_1 < j_1 < j_2 < i_2$

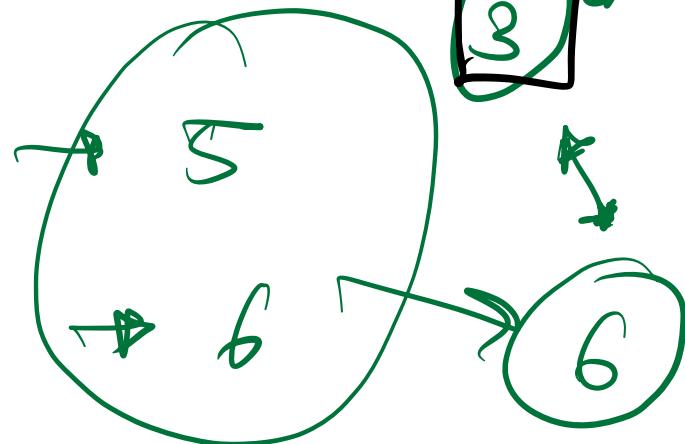
$i_1 \rightarrow 2$   
 $i_2 \rightarrow 10$   
 $j_1 \rightarrow 4$   
 $j_2 \rightarrow 7$

$i_1 - j_1$   
 $i_2 - j_2$



$i_1 - j_1 - j_2$

$j_1 - i_2$



**TODO**

## Pseudocode:

```
solve(A, B)
{
    n → no. of ele
    mx = +10
    sort(A)
    sort(B)
    for i: 0 → n-1
    {
        mx = max(mx, abs(a(i) - b(i)))
    }
}
```

time

$i_1 < i_2$

$j_1 < j_2$

$$TC: O(n \log n)$$

SC: Depends on sorting Algo

10:22 → 10:32

# Magician and chocolate

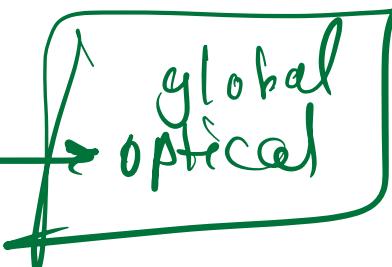
$$A = 3 \rightarrow \underline{\underline{3}}$$

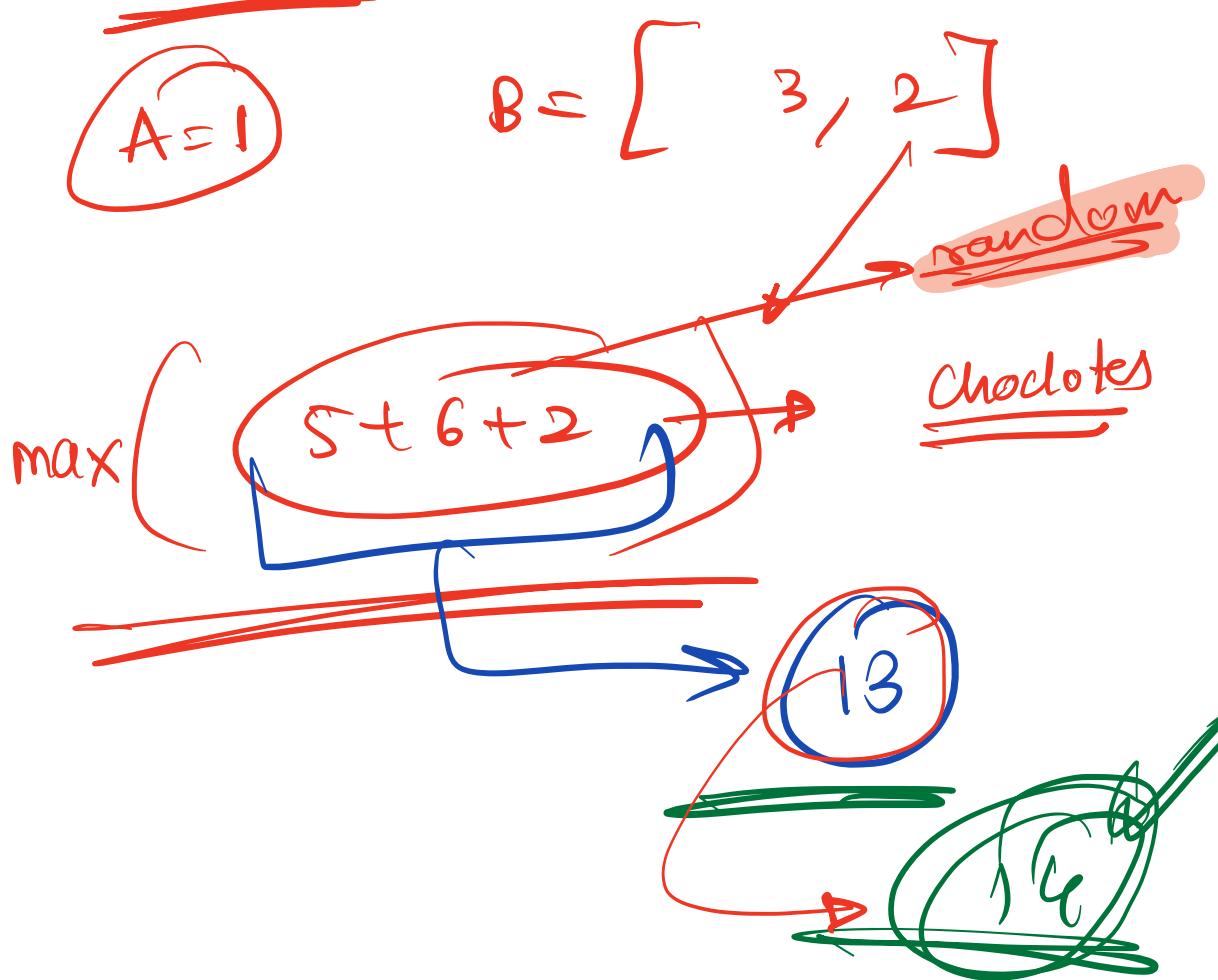
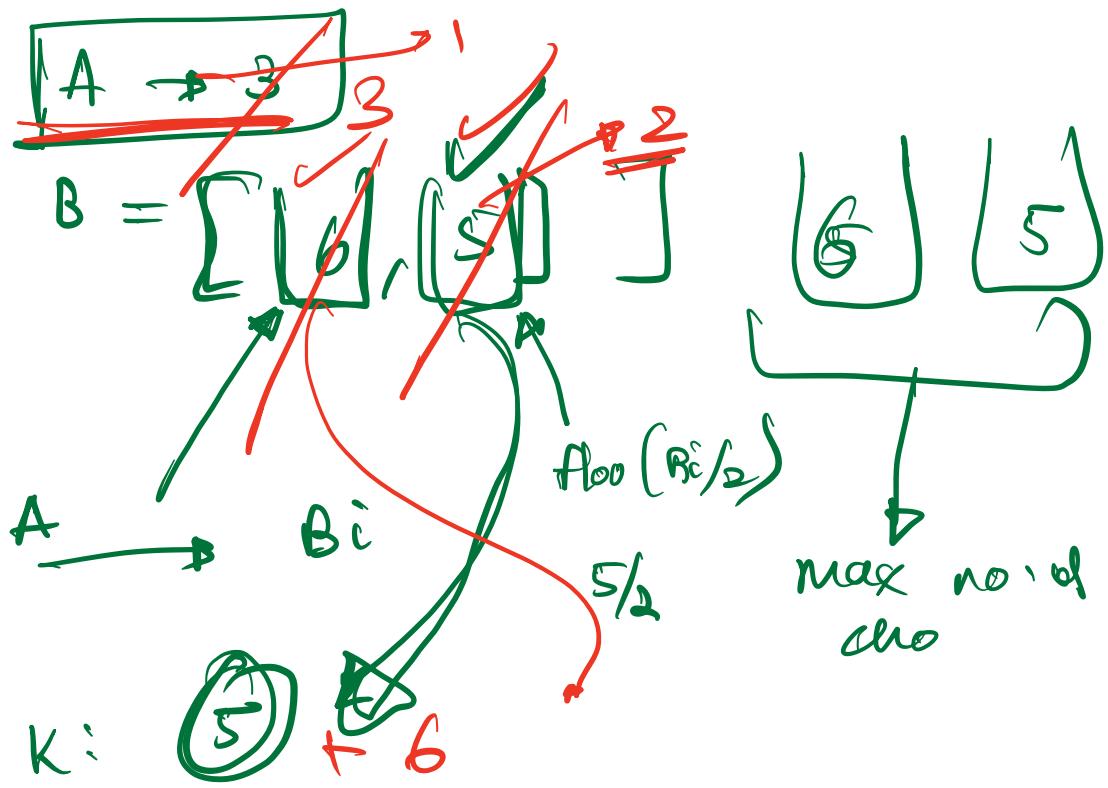
$$B = [6, 5]$$



$$\underline{\underline{t=1}} \rightarrow$$

local opt

  
global  
optimal

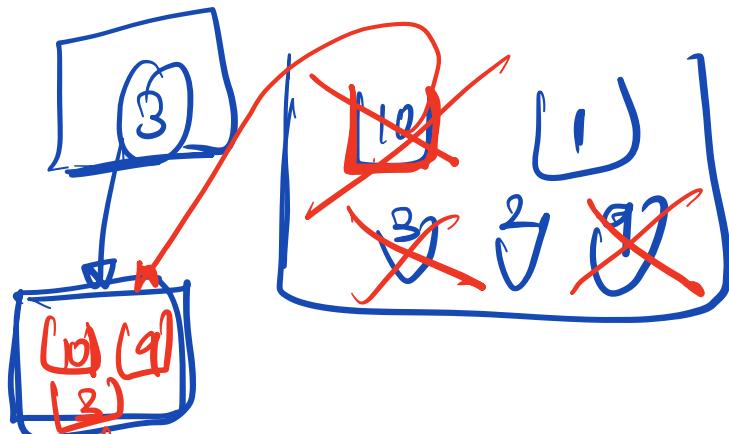


Task → pick max chocolate

each →

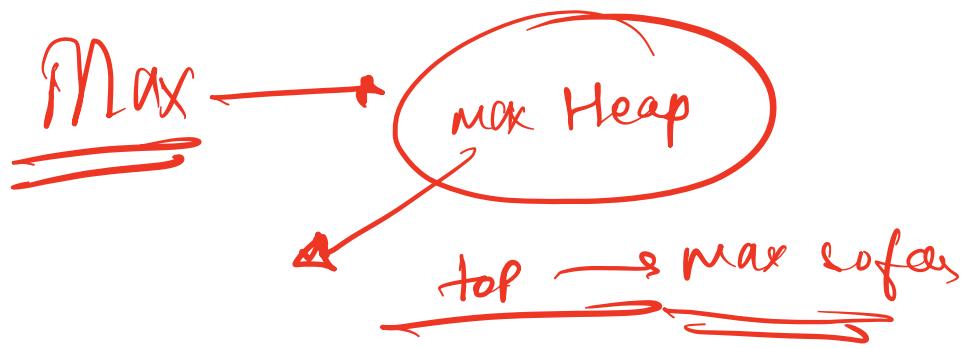
pick max cho at  
curr step

ans

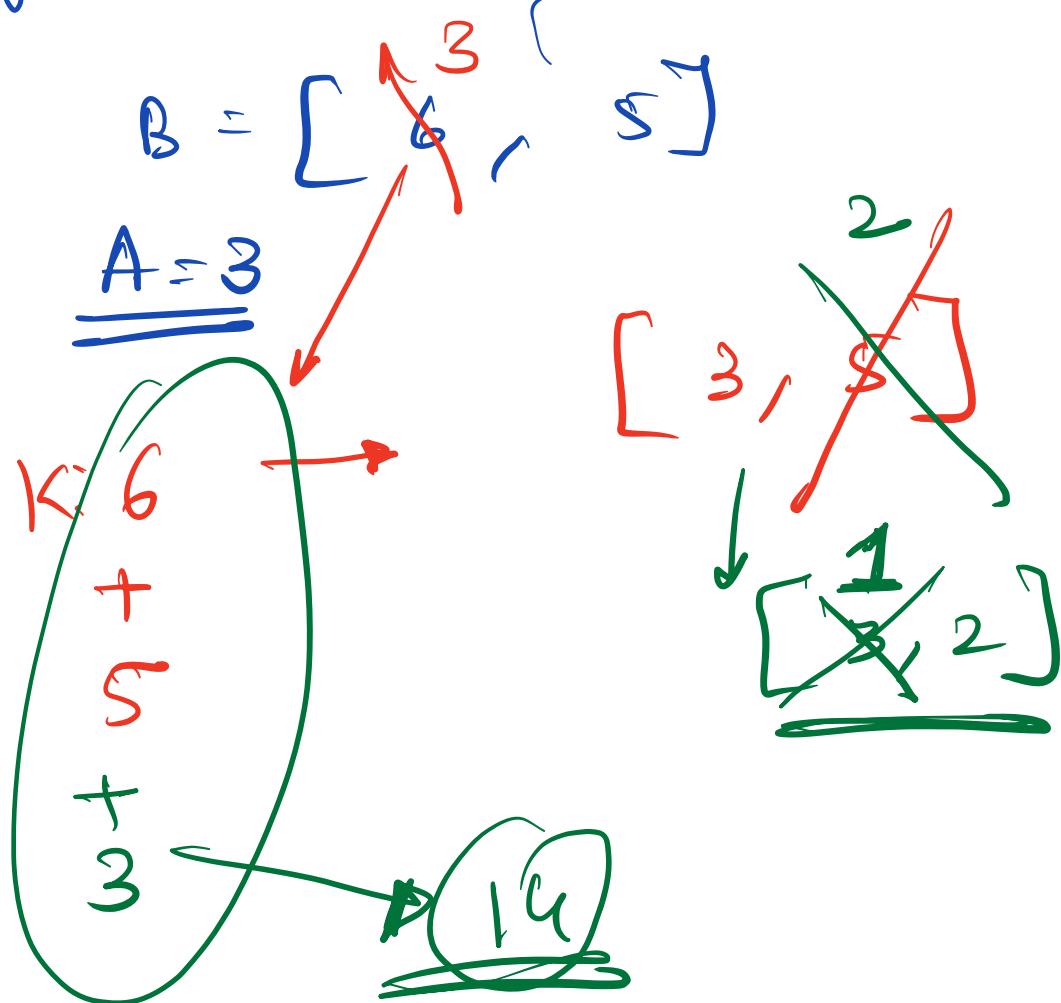


each step → max avail at moment

22



Greedy Approach



Max heap

Imp :-

each time you pick  
 $B_i$  out, insert  $flo(B_i/2)$

back into heap

~~Logic :- Maintain a Max  
heap~~

↳ heap gives max  
among all beige chocolates  
at top

② Simply pick top of  
heap and add to ans.

③

insert floor ( $\beta^c/s$ )

Back to heap.

do this A  
times

Pseudocode

## Pseudo code

solve( A, BL )

{  
① Create max heap → mh

for i : 0 → n - 1

mh. insert( a(i) )



while( A ← )

↑ C

chocolats += hm. top()

~~hm. pop()~~

magical

hm. insert( hm. top() )

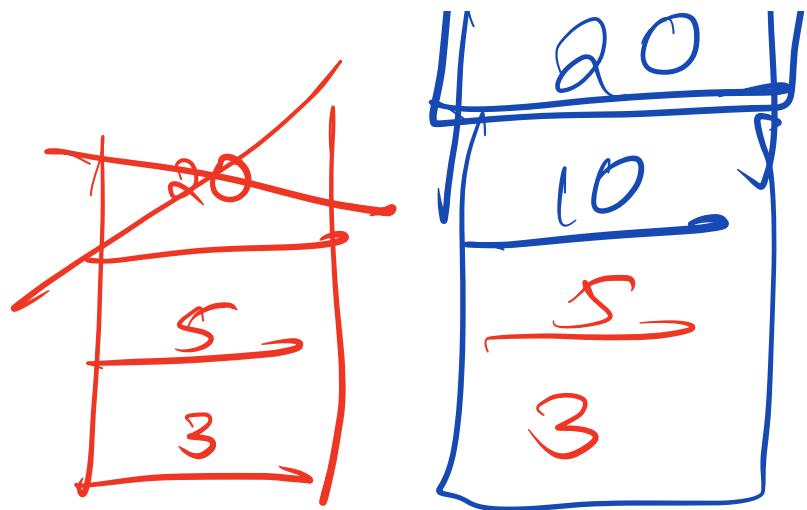
C

D

hm. pop()

}

F → H



max → mix

$$mx/2$$

A diagram illustrating a rotating system. On the left, a blue curved arrow indicates clockwise rotation. To the right, there are two green elliptical orbits. The inner orbit has an arrow pointing to the right and is labeled  $mx$ . The outer orbit has an arrow pointing to the left and is labeled  $mx/2$ .

while(A--)

$$c = \text{hm} \cdot \text{top}(c)$$

choco + = c

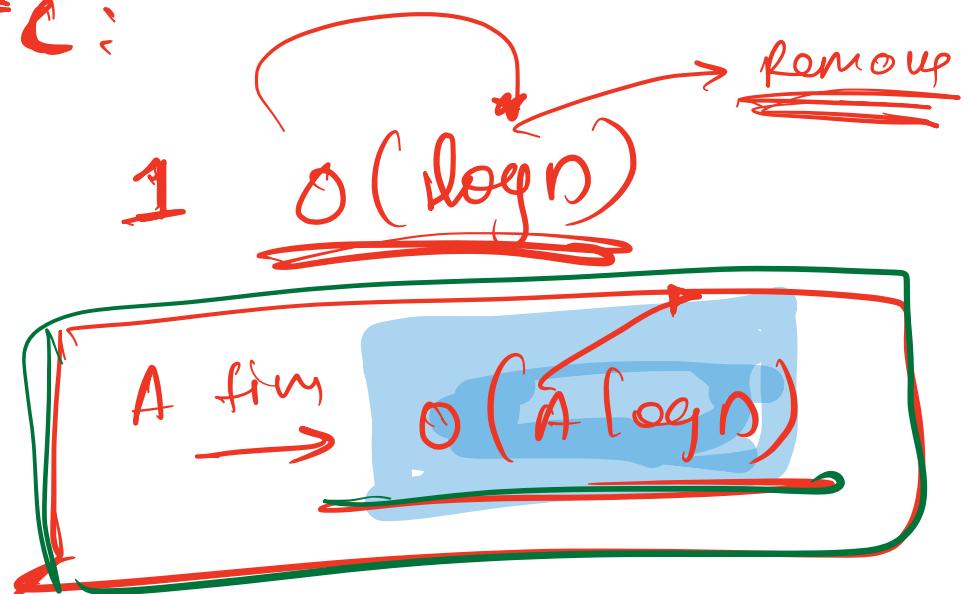
Jan Boncik

(hm · popc)

hm.pop()  
hm.insert(4)

return choco

TC:



~~$\mathcal{O}(n \log n)$~~

