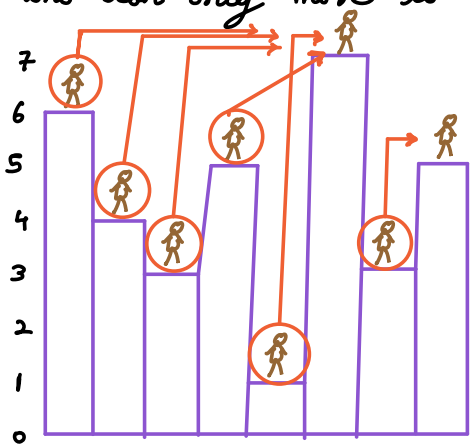
Q → Given N buildings with height of each building. There is a person standing on top of each building who can only move to building on right. Find max high a person can go from his/her current position.



$maxR[i] = max(A[i], maxR[i+1])$

$ans[i] = maxR[i] - A[i]$

I/P → A = [6 4 3 5 1 7 3 5]

maxR = [7 7 7 7 7 7 5 5] ← SC = O(N) → carry forward SC = O(1)

✓ Ans = [1 3 4 2 6 0 2 0] ✓

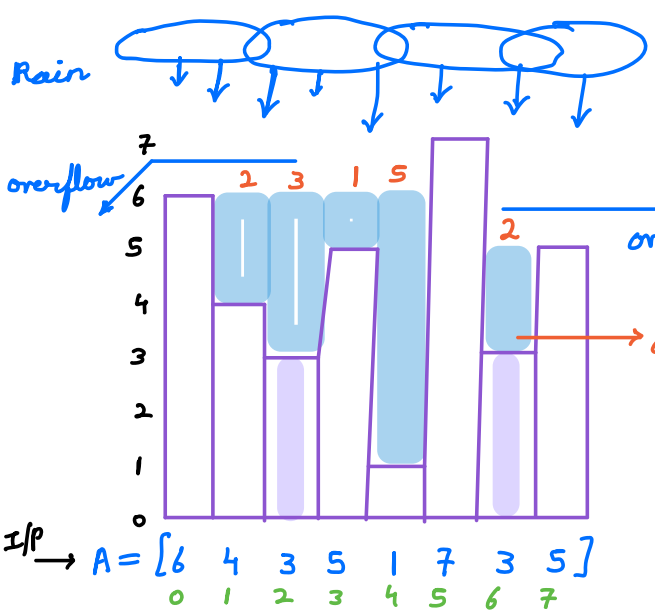
TC = O(N)

SC = O(1)

```
int maxR = A[N-1]
ans[N-1] = 0
for i → (N-2) to 0   // R → L
    maxR = max(maxR, A[i])
    ans[i] = maxR - A[i]
```

i/p → Algo → o/p ✓ ✓

Q → Given N buildings with height of each building. Find the rain water trapped above all buildings.



I/P → A = [6 4 3 5 1 7 3 5]

0 1 2 3 4 5 6 7

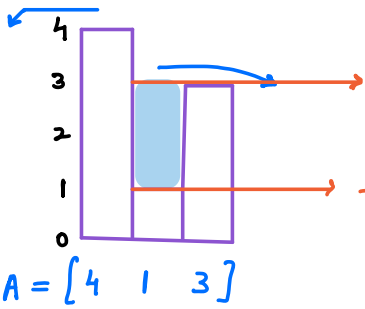
Area = base * height

base ∀ buildings = 1

area = b * h = 1 * 2 = 2

total = 2 + 3 + 1 + 5 + 2

= 13 (Ans)

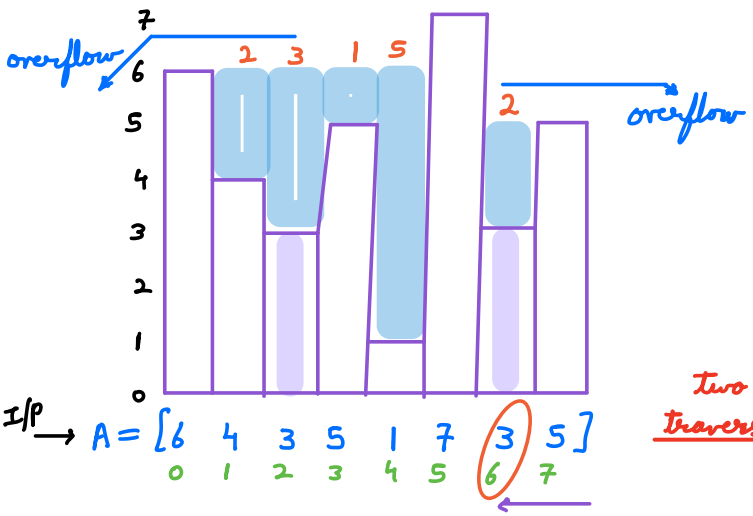


3 = min (4, 3) ← min (maxR[i], maxL[i]) ✓

> 3 - 1 = 2 ✓

1 = A[i] ✓ ←

A = [4 1 3]



I/P → A = [6 4 3 5 1 7 3 5]
0 1 2 3 4 5 6 7

TC = O(N) ✓

SC = O(N)

SC = O(1) ? ✗

two traversal { max from right L ← R
max from left L → R

maxR = [7 7 7 7 7 7 5 5] → maxR[i] = max (A[i], maxR[i+1])

maxL = [6 6 6 6 6 7 7 7] → maxL[i] = max (A[i], maxL[i-1])

water [0 2 3 1 5 0 2 0] → min (maxL[i], maxR[i]) − A[i]

Ans = 2+3+1+5+2 = 13 ✓

A = [6 4 3 5 1 7 3 5] ✓
0 1 2 3 4 5 6 7

i i j

6-4 = 2

5-5 = 0

5-3 = 2

0

7-7 = 0

maxL = 6

maxR = ~~5~~ 7

min (maxR[i], maxL[i])

A = [1 4 8 5 1 7 3 1]
0 1 2 3 4 5 6 7
i i i j j j j j

0 0 0 2 6 0 0 0

maxL = ~~1 4~~ 8

max R = ~~1 3~~ 7

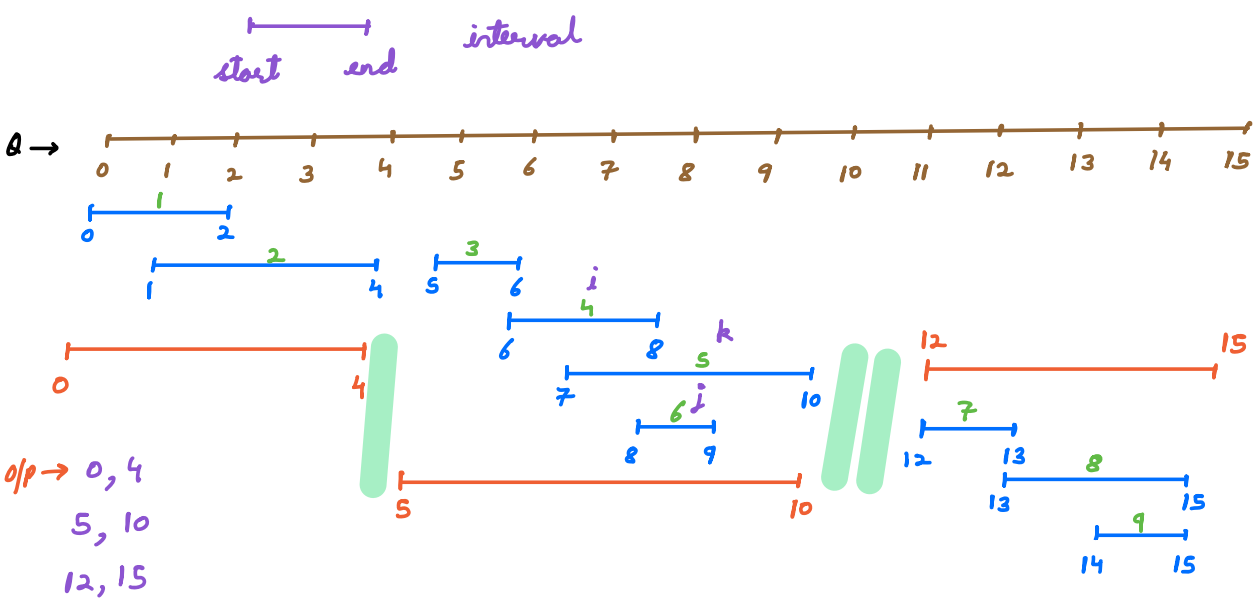
SC = O(1) ✓

```
maxL = A[0]   = 6 8
maxR = A[N-1] = 5 7 8
i = 0        j = N-1
while (i <= j) {
    if (A[i] <= A[j]) {
        ans += min(maxL, maxR) - A[i]
        i += 1
        maxL = max(maxL, A[i])
    } else {
        ans += min(maxL, maxR) - A[j]
        j -= 1
        maxR = max(maxR, A[j])
    }
}
return ans
```

A = [6 4 8 5 1 7 0 5]
0 1 2 3 4 5 6 7
i j
0 2 0 2 6 0 5 0 ✓

TC = O(N)
SC = O(1) ✓

12:45 PM



Given a sorted list of intervals (sorted based on start time) merge all overlapping intervals & print the final merged intervals in sorted order.

if start is same then wrt end time.

I/P → S = [. . .]
E = [. . .]

S[i] —— E[i]
one interval

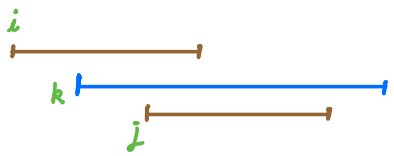
Observations → 1) If i^{th} interval overlap with j^{th} interval.

$i < k < j$

interval in b/w ⇒ k will also overlap with i & j.

2) If i^{th} interval overlap with j^{th} interval. → $S[j] <= E[i]$

$i < j$



3) Check & merge adjacent intervals.

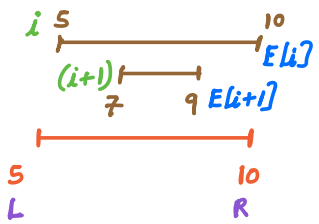
$i \rightarrow S[i] — E[i]$

$(i+1) \rightarrow S[i+1] — E[i+1]$

$S[i] <= S[i+1]$

Start (L) = S[i] ✓

End (R) = max(E[i], E[i+1]) ✓



L = S[0] R = E[0]

for i → 1 to (N-1)

if (S[i] <= R) ← ✓

R = max(R, E[i])

else

print (L, R)

L = S[i] R = E[i]

✓ → print (L, R)

	0	1	2	3	4	5	6
S = [	1	2	4	9	10	15	16]
E = [	5	4	7	10	12	17	20]
	i	i	i	i	i	i	i

L = ~~1~~ ~~9~~ 15

R = ~~5~~ ~~7~~ ~~10~~ ~~12~~ ~~17~~ 20

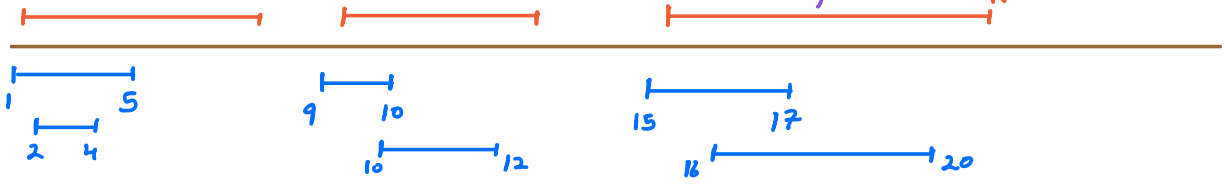
o/p → 1, 7

9, 12

15, 20

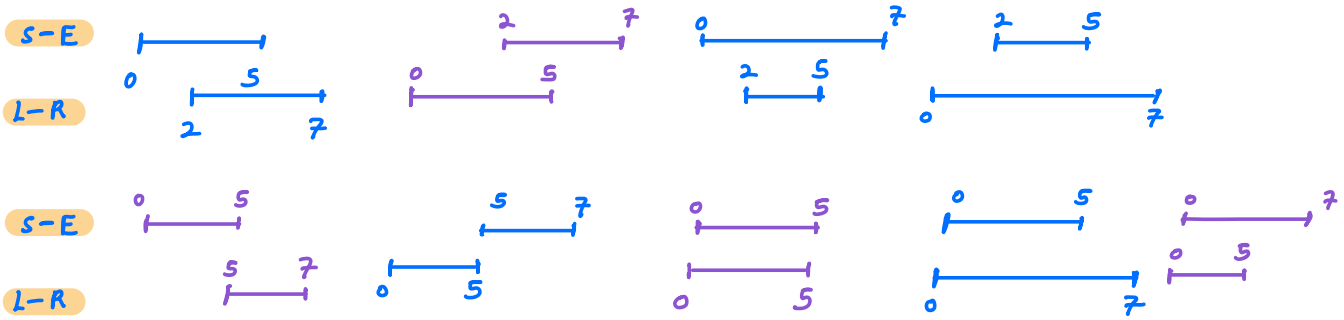
TC = O(N)

SC = O(1)

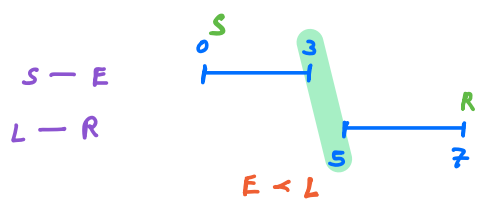


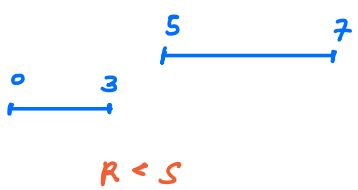
$S[i] <= E[i]$ $S[i] <= S[i+1]$

Check if any 2 interval overlaps → $!(E<L \ || \ R<S)$ ←



check if any 2 intervals do not overlap → $E<L \ || \ R<S$

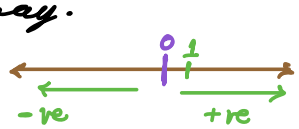




Q → Given an integer array.

Find first missing +ve integer in the array.

1, 2, 3, 4 . . .



A = [10, 1, 3, 2, -6, -5, 7] → 4 (Ans)

A = [3, 8, 6, 2, -1, 0] → 1 (Ans)

A = [1, 2, 3, 4, 5] → 6 (Ans)

min ans = 1 max ans in A[N] = N+1

Bruteforce → ∀ numbers from 1 to N (if 1 to N are present ⇒ Ans = (N+1))

check if it is present in array.

Travel to check TC = $O(N^2)$ SC = $O(1)$

Sort →

TC = O(N log(N)) A = [10, 1, 3, 2, -6, -5, 7]

↳ [-6, -5, 1, 2, 3, 7, 10]

→ x x ✓ ✓ ✓ ↓

4 (Ans)

TC = O(N log N + N)

= O(N log(N))

Visited → A = [10, 1, 3, 2, -6, -5, 7]

∀ numbers from 1 to N → visited array of length N

check if it is preser in array.

vst[i] → element (i+1) is present ✓

index → 0 1 2 3 4 5 6 (0 — (N-1)) (1 — N)

A = [10, 1, 3, 2, -6, -5, 7]

vst = [T T T F F F T]

(i+1) → 1 2 3 4 5 6 7

→ ✓ ✓ ✓ x

Ans = 4

∀i vst[i] = false

if (A[i] >= 1 && A[i] <= N)

vst[A[i] - 1] = true

TC = O(N)

SC = O(N)

SC = O(1) TC = O(N) Modify I/P

A[i] is preesent or not → A[i] is b/w 1 to N

swap index (A[i]-1) with index i.

already correct

(A[i] < 1 or A[i] > N) → ignore

→ x ✓ ✓ ✓ x x ✓

0 1 2 3 4 5 6

A = [10, 1, 3, 2, -6, -5, 7] [1 — 7]

1 10 10

2

[1 2 3 10 -6 -5 7] → A[i] = (i+1) ⇒ visited

→ ✓ ✓ ✓ x

Ans = 4

0 1 2 3 4 5 6

A = [10, 1, 3, 2, 3, -5, 7]

1 2 10

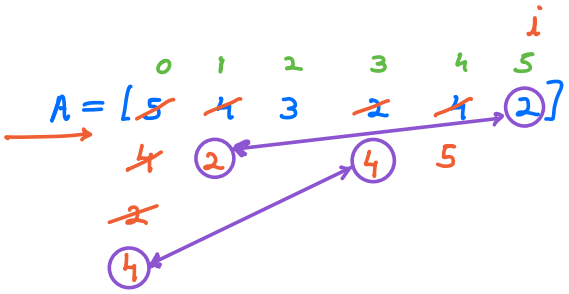
→ ✓ ✓

∀i

1 <= A[i] <= N &&

A[i] != A[A[i] - 1]

swap (i, A[i]-1)



1 N
[1 6]

max # swaps = O(N)

∵ in 1 swap we place one element at correct position.

TC = O(N) SC = O(1)
