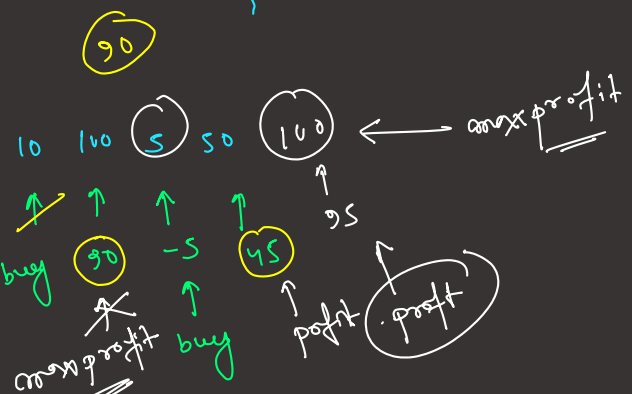
else {
buying price = prices[i];
}
}

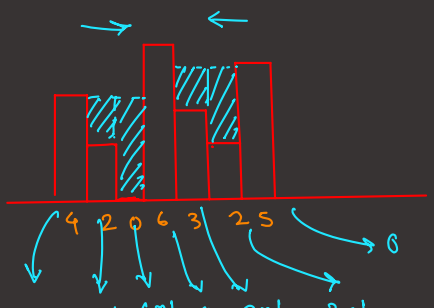


Day-3

8 Trapping Rainwater

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining

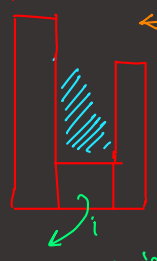
height = [4, 2, 0, 6, 3, 2, 5]



tw = 0 | 2-1 = 2 | 4-1 = 4 | 0 | 2-1 = 2 | 3-1 = 3

∴ Total tw = 11

left boundary →
← right boundary



wl = min(lb, rb) − height[i]

tw = wl × 1

if (n > 2 && not asc && not desc) → true → print code

O(n)

asc: temp = False
arr[i-1] < arr[i] → temp = true → Break

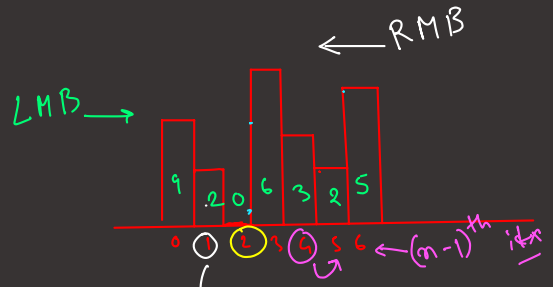
H.W

n > 2

(i) tw = 0

(ii)

if height[] is sorted in asc or desc order → (iii)



LMB = | 4 | 4 | 4 | 6 | 6 | 6 | 6 | → Left Max Boundary

0 1 2 3 4 5 6

RMB = | 6 | 6 | 6 | 6 | 5 | 5 | 5 | → Right Max Boundary.

0 1 2 3 4 5 6

Trapped water at idx i = $\min(LMB[i], RMB[i]) - height[i]$

(Water level)

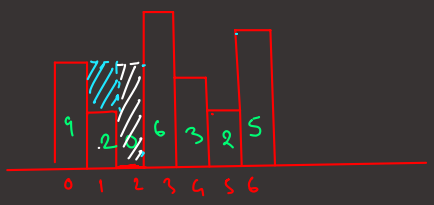
Total water = $\sum tw$

$LMB[i] = \max(LMB[i-1], height[i])$ if $i \neq 0$

for $i=0$, $LMB[0] = height[0]$

$RMB[i] = \max(RMB[i+1], height[i])$ if $i \neq n-1$

if $i = n-1$, $RMB[i] = height[i]$



$tw[0] = \min(LMB[0], RMB[0]) - height[0]$

$= \min(4, 6) - 4 = 4 - 4 = 0$

$tw[1] = \min(4, 6) - 2 = 4 - 2 = 2$

$tw[2] = \min(4, 6) - 0 = 4 - 0 = 4$

⋮

Sorting

① **Bubble Sort** : Large elements come to the end of array by swapping with the adjacent element.

A → | 8 | 5 | 7 | 3 | 2 |

i = 0

1st pass

8 5 7 3 2

5 8 7 3 2

5 7 8 3 2

5 7 3 8 2

5 7 3 2 8

result for pass : 1

comp = 4

max swap = 4

j = 0 to n-1

i = 1

2nd Pass

5 7 3 2 8

5 7 3 2 8

5 3 7 2 8

5 3 2 7 8

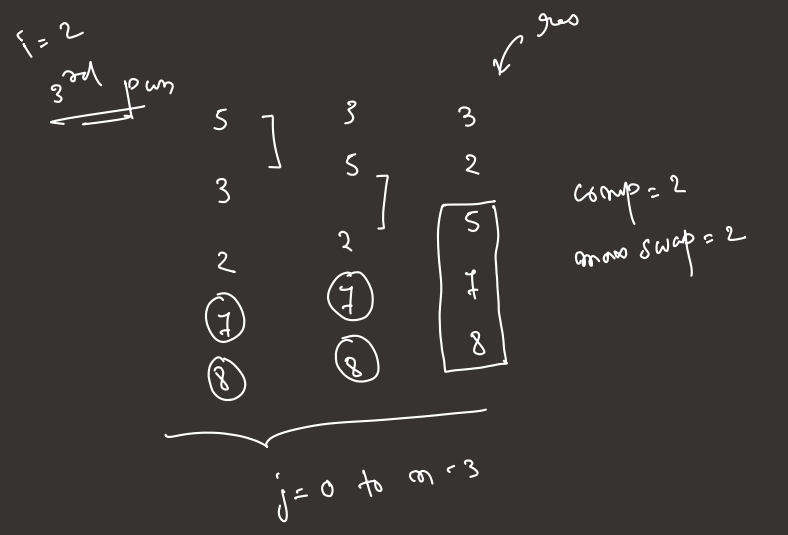
result of 2nd pass

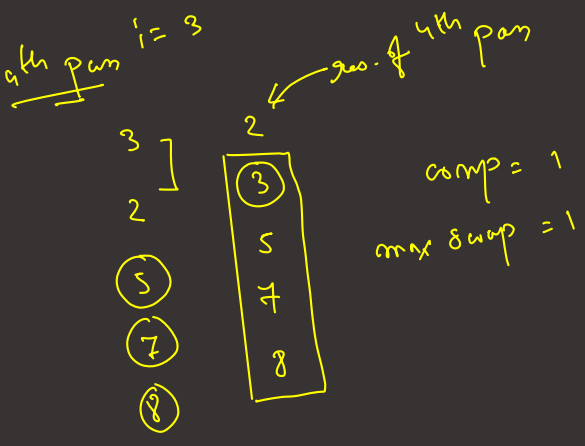
comp = 3

swap = 2

max swap = 3

j = 0 to n-2





$\therefore$ total no of comp $= (n-1) + \cdots + 3 + 2 + 1$

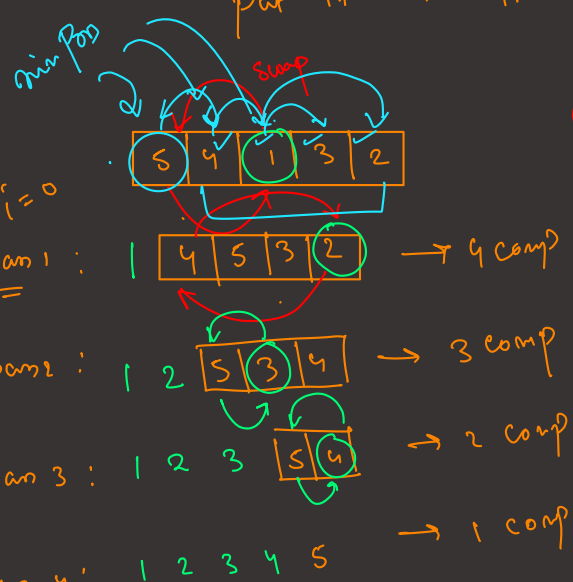
$= \frac{n(n-1)}{2}$ ← max swap

time complexity $= O(n^2)$ $\left[O\left(\frac{n^2}{2} - \frac{n}{2}\right) = O(n^2/2) = O(n^2) \right]$

```
for (int i = 0 ; i < n-1 ; i++)
    for (int j = 0 ; j < n-1-i ; j++)
        // start of code.
```

↓ comp or swap

Selection Sort — pick the smallest (from unsorted part), put it at the beginning.



① check for the min position of unsorted array.

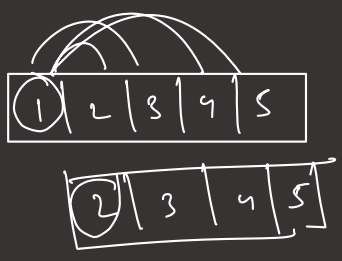
② else [1st index] ↔ else [min position]

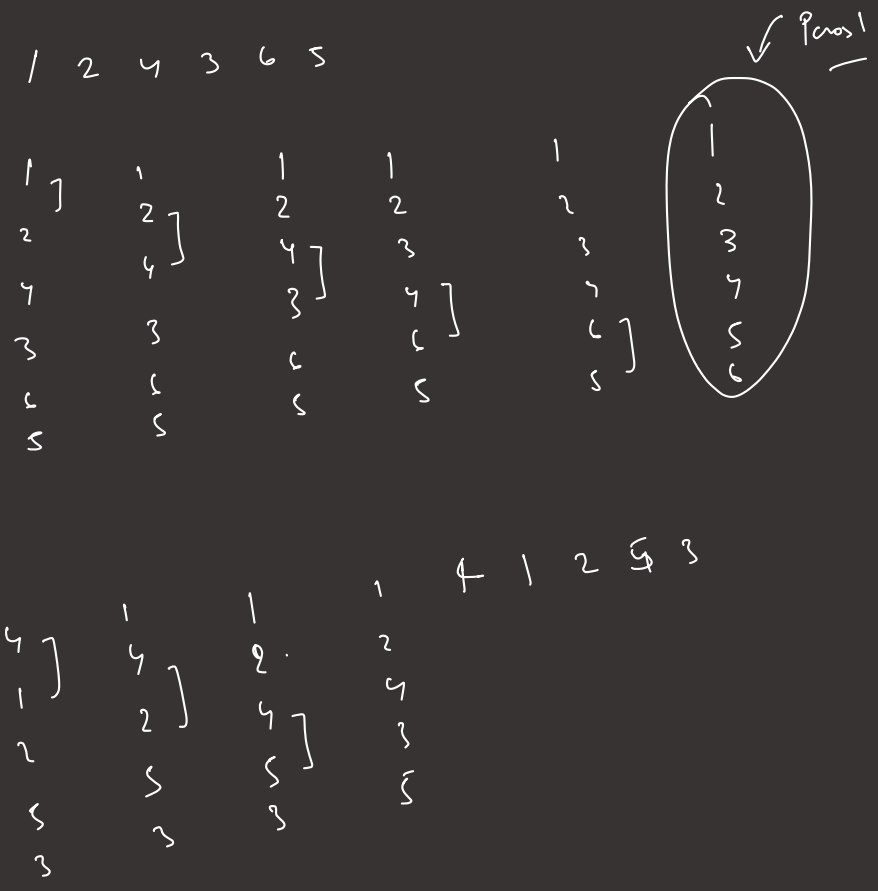
total no. of swap $= n-1$

$= O(n)$

total comp $= \frac{n(n-1)}{2} = O(n^2)$ ← time Complexity

① Selection Sort is not adaptive




Pivot