

Arrays - II

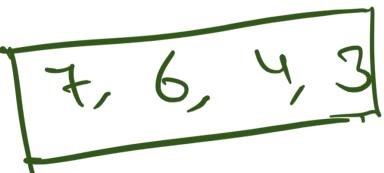


$\text{cur} = \text{a}[i]$

Leaders in an Array

$\text{a}[] = [2, 7, 6, 1, 4, 3]$

leaders: the elements which do not have any larger element on their right side.

Output : 
[7, 6, 4, 3]

$O(n^2)$ \rightarrow $O(N)$

loop1 :

```
for (int i = 0; i < n; i++) {  
    cur = a[i]; boolean isLeader = true;  
    for (int j = i+1; j < n; j++) {  
        if (a[j] > cur) {  
            isLeader = false; break;  
        }  
    }  
    if (isLeader) {  
        print(a[i]);  
    }  
}
```

$$O(N^2)$$

$a[] = [2, 7, 6, 4, 1, 3]$

7, 6, 4, 3

3, 4, 6, 7

Create a Array

A diagram illustrating a linked list structure. It consists of five rectangular boxes arranged horizontally. The first four boxes contain the numbers 3, 4, 6, and 7 respectively. The fifth box is empty. Below the first four boxes, a bracket is drawn under them, with the letter 't' written below it, indicating a subsequence. To the right of the fifth box, an arrow points to it with the letter 'n', indicating the next element.

Maximum Sum Subarray

$a[] = [6, -7, 4, -2, 1, 5, -4]$

Subarray → Sub part of an array.

$[6, -3]$
 $[6, -3, 4], [-3, -4]$

⑦

$$\begin{matrix} 1 \rightarrow 7 \\ 2 \rightarrow 6 \\ 3 \rightarrow 5 \\ \vdots \end{matrix} \rightarrow \frac{1+2+3+\dots+7}{2} = \underline{\underline{28}}$$

$1 \rightarrow 1$

$\frac{n(n+1)}{2} \rightarrow O(N^2)$

$a[] = [6, -7, 4, -2, 1, 5, -4]$

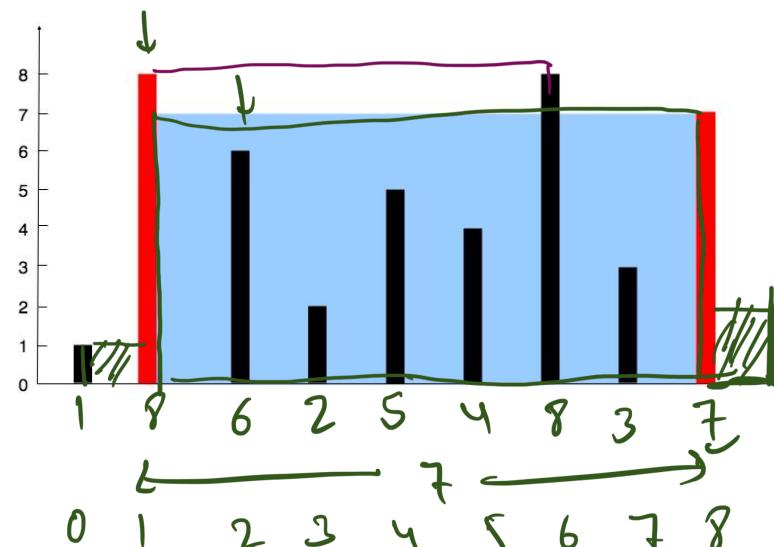
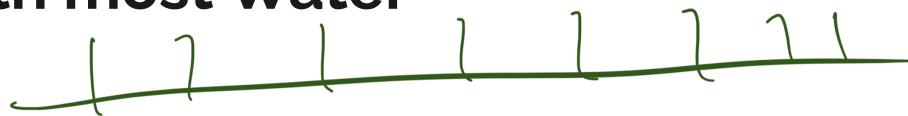
int curSum = ~~0 + 6 → 6 + -7 → -1~~ → ~~0 + 4 → 4 + -2 = 2~~
 int maxSum = ~~-98~~ ~~6~~ 8

$4 \leftarrow 1 + 8 \in 5 + 3$

```

int cur = 0
int max = -∞
loop () {
  cur += a[i];
  if (cur < 0) cur = 0
  if (cur > max) max = cur
}
max
  
```

Container with most Water

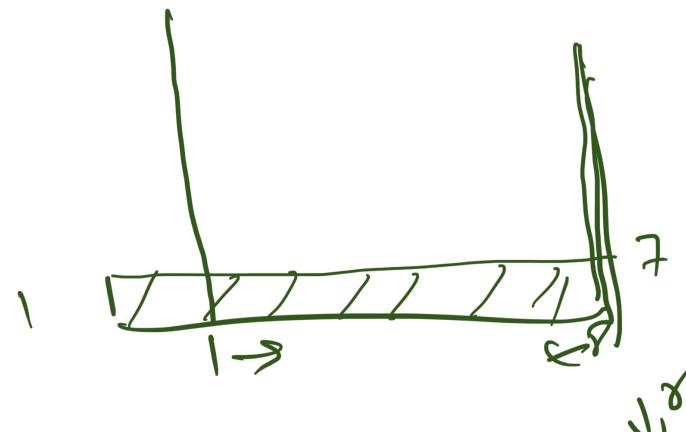
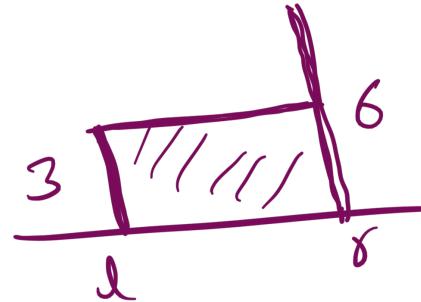


$O(N^2)$.

$$8 \times 5 = 40$$

$O(N^2)$

```
↑  
for ( i → ) {  
    int left = a[i];  
    for ( j = i+1 → ) {  
        int right = a[j];  
        int dist = j - i;  
        area = min(left, right) * dist  
        maxArea = max ( maxArea, area );  
    }  
}
```



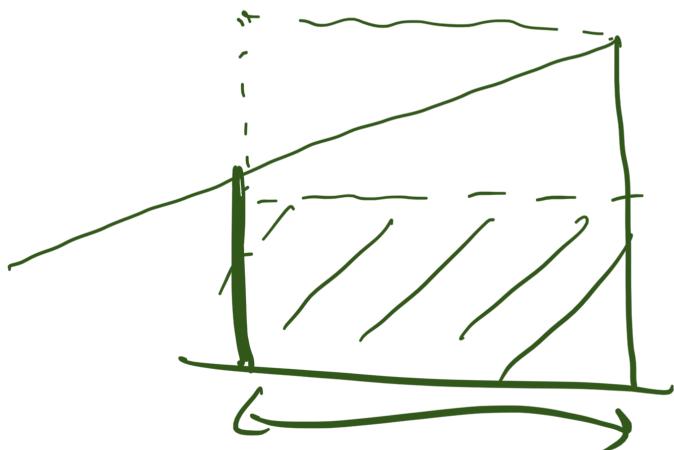
Two pointer approach

$a[] = (1, 8, 6, 2, 4, 5, 8, 3, 7)$

area = ~~$8 * 1 = 8$~~ $7 * 7 = 49$

max Area = ~~$8 * 8 = 64$~~ 49

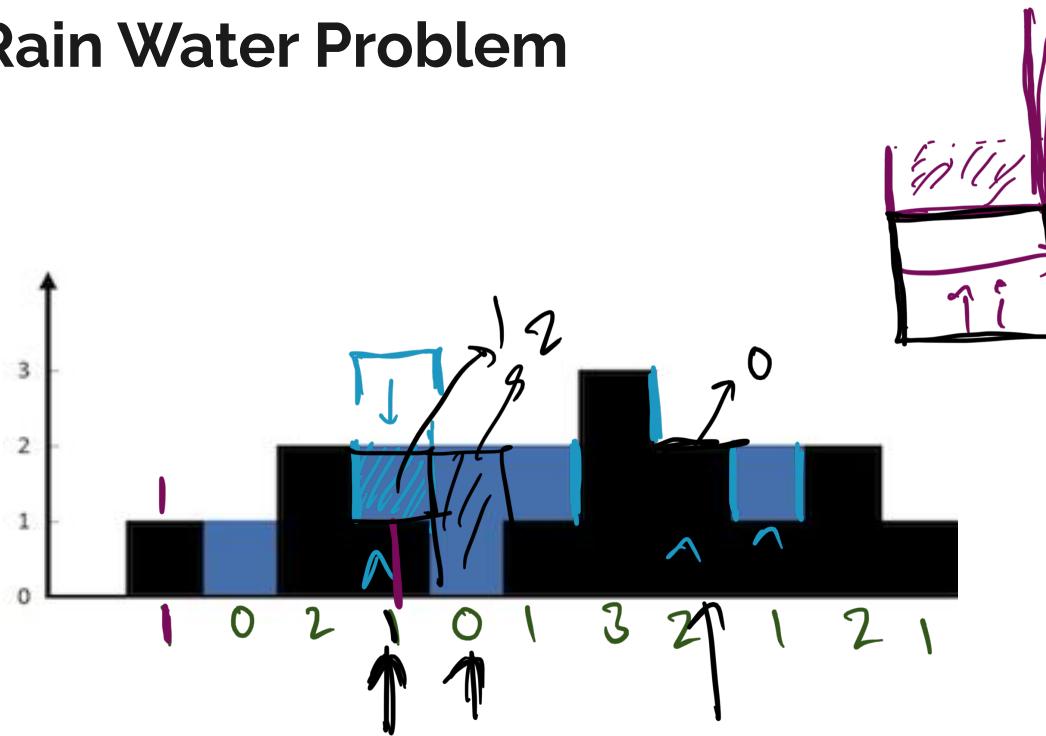
$3 * 6 = 18$



Assignment:

Largest Area in a Histogram

Trapping Rain Water Problem



| | | | | | | | | | | | |
|---------------|---|---|---|---|---|---|---|-----|---|---|---|
| $\lambda[] =$ | 1 | 1 | 2 | 2 | 2 | 2 | 3 | (3) | 3 | 3 | 3 |
| $\gamma[] =$ | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 1 |

$$\min(\lambda[i], \gamma[i]) - a_{\uparrow}[i]$$

$\min(2, 3) - 0$
 $2 - 0 = 2$

$\min(2, 3) - 2$
 $2 - 2 = 0$

$\min(2, 3) - 1$ = ①



```

dgt    d[0] = a[0];
for (int i = 1; i < n; i++) {
    d[i] = Math.max(a[i], d[i-1]);
}

```

Practice Problems

1. Print frequency of all the elements in a sorted Array.
2. In an Array of all 0s & 1s, find the longest length of all consecutive 1s.

Input: 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1

Output: 4

3. Move all 0s to the end of the given Array.

Input: 8, 0, 1, 3, 0, 0, 5

Output: 8, 1, 3, 5, 0, 0, 0

4. Trapping Rain water Problem in O(1) space complexity.
5. Minimum Sum Subarray Problem ✓

$O(N) \rightarrow O(1)$ ✗

Practice Problems

6. Print the elements in the maximum sum subarray . ✓

7. Stocks Buy & Sell problem

<https://leetcode.com/problems/best-time-to-buy-and-sell-stock/>

<https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/>

II link of
Kadens subanay