

Arrays 1

Content

- Find maximum subarray sum.
- Perform multiple queries.
- Rainwater trapping.

- Rules →
- ① Private chat for answers
 - ② Use question
 - ③ Be patient.

Maximum Subarray Sum

continuous part of Arr

Given $\text{int}[] A$. Find max subarray sum of all subarrays

$A = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ -2 & 3 & 4 & -1 & 5 & -10 & 7 \end{matrix}$ $\text{ans} = 11$

$A = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ -3 & 4 & 6 & 8 & -10 & 2 & 7 \end{matrix}$ $\text{ans} = 18$

Quiz

$A = \begin{matrix} 4 & 5 & 2 & 1 & 6 \end{matrix}$ $\text{ans} = 18$

Observation — $\forall_i A_i > 0$ entire array sum = ans.

$A = \begin{matrix} -4 & -3 & -6 & -9 & -2 \end{matrix}$

Observation — $\forall_i A_i < 0$ pick the largest no.

Bruteforce

calculate sum of all subarrays and check max

$\text{ans} = -\infty$ // INT_MIN

```
for i → 0 to N-1 { // start index of sub
  for j → i to N-1 { // end index of sub
    sum = 0
    for k → i to j {
```

```

    sum += A[k]
    ans = max(ans, sum)
}
print(ans)

```

TC: $O(N^3)$
SC: $O(1)$

can be optimised via
Prefix sum.

TC: $O(N^2)$
SC: $O(N)$

Carry forward \longrightarrow calculate & use immediately.

```

ans = -∞
for i  $\longrightarrow$  0 to N-1 {
    sum = 0 // carry forward
    for j  $\longrightarrow$  i to N-1 {
        sum += A[j]
        ans = max(ans, sum)
    }
}
print(ans)

```

TC: $O(N^2)$
SC: $O(1)$

Example:
 i: -1 10 -2
 j: -1 10 -2
 sum = 0 -2
 ans = ~~0~~ ~~-1~~ ~~9~~
 10

Kadane's

Case 1 > all no. are +ve

$A = 1 \ 2 \ 3$

+ve +ve +ve

Case 2 > all no. are -ve

$A = -1 \ -2 \ -3$

pick the largest

Case 3 > -ve -ve -ve +ve +ve +ve -ve -ve -ve

Case 4 > Generic case

→ take as in increased overall sum
 $\cancel{7} \ -1 \ 1 \ 2 \ 3$

→ don't take
 $\cancel{7} \ -10 \ 1 \ 2 \ 3$

↓

	0	1	2	3	4	5	6	7	8
	-2	3	4	-1	5	-10	7	-9	12

sum	0	-2 ⁰	3	7	6	11	1	8	-1 ¹⁰	12
-----	---	----------------------------	---	---	---	----	---	---	-----------------------------	----

ans	-∞	-2	3	7	7	11	11	11		12
-----	----	----	---	---	---	----	----	----	--	----

Pseudocode

TC : $O(N)$

SC : $O(1)$

ans = $-\infty$

sum = 0

for $i \rightarrow 0$ to $N-1$ {

 sum += $A[i]$

 ans = max(ans, sum)

 if (sum < 0) {

 sum = 0

 }

}

return ans

$A =$

	0	1	2	3
	-3	-5	-2	-6
sum = 0	-3 ⁰	-5 ⁰	-2 ⁰	-6 ⁰
ans = -∞	-3	-3	-2	-2

		0	1	2	3	4	5	6	7	8
$A =$		6	-3	9	-15	3	12	0	-24	3
sum	0	6	3	12	-3 ⁰	3	15	15	-9 ⁰	3
ans	$-\infty$	6	6	12	12	12	15	15	15	15

-2 3 4 -1 5 -10 7

ans = $-\infty$

sum = 0, start, L, R

final ans

for i \rightarrow 0 to N-1 {

sum += A[i]

if (sum > ans) {

ans = sum

L = start

R = i

}

if (sum < 0) {

sum = 0

start = i+1

}

}

// Print the array b/w L to R

Perform multiple queries from i to last index

Given an $\text{int}[]$ A where every element is 0

Return the final array after performing multiple queries

Query (i, x) \longrightarrow Add x to all no. from i to $N-1$

	0	1	2	3	4	5	6
$A[]$	0	0	0	0	0	0	0
$i \quad x$							
Query (1, 3)		+3	+3	+3	+3	+3	+3
Query (4, 2)					+2	+2	+2
Query (3, 1)				+1	+1	+1	+1
	0	3	3	4	6	6	6

Bruteforce \longrightarrow

For every query

Add x to all no. from i to $N-1$

TC: $O(Q * N)$

SC: $O(1)$

	0	1	2	3	4	5	6
$A[]$	0	0	0	0	0	0	0
$i \quad x$							
Query (1, 3)		+3					
Query (4, 2)					+2		
Query (3, 1)				+1			
	0	3	0	1	2	0	0

Prefix sum

0 3 3 4 6 6 6

Pseudocode

Queries $[][]$

$\begin{matrix} i & x \\ \{0, 1\} \end{matrix}$

for $q \longrightarrow 0$ to $Q-1$ {

$i = \text{Queries}[q][0]$

$x = \text{Queries}[q][1]$

$A[i] += x$

TC: $O(Q+N)$

SC: $O(1)$

for ($i \longrightarrow 1$ to $N-1$) {

$A[i] = A[i] + A[i-1]$

print(A)

Perform multiple queries from i to j index

Given an $\text{int}[]$ A where every element is 0

Return the final array after performing multiple queries

Query (i, j, x) \longrightarrow Add x to all no. from i to j

	0	1	2	3	4	5	6
$A[]$	0	0	0	0	0	0	0
i							
j							
x							
Query (1, 3, 2)		+2	+2	+2			
Query (2, 5, 3)			+3	+3	+3	+3	
Query (5, 6, -1)						-1	-1
	0	2	5	5	3	2	-1

	0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0
i								
j								
x								
1 4 3		3	3	3	3			
0 5 -1	-1	-1	-1	-1	-1			
2 2 4			4					
4 6 3					3	3	3	
	-1	2	6	2	5	3	3	0

Bruteforce

For each query \longrightarrow

Add x from index i to j

TC : $O(QN)$

SC : $O(1)$

0	1	2	3	4	5
0	0	0	0	0	0

Query (1, 3, 2)



+2	+2	+2	+2	+2
			-2	-2

0	2	2	2	0	0
---	---	---	---	---	---

Query (1, 2)

Query (4, -2)

0	1	2	3	4	5
0	0	0	0	0	0

+2

-2

0	2	0	0	-2	0
---	---	---	---	----	---

ps



0	2	2	2	0	0
---	---	---	---	---	---

Pseudocode

Queries [][][] { i, j, x }

```

for q → 0 to Q-1 {
    i = Queries[q][0]
    j = Queries[q][1]
    x = Queries[q][2]
    A[i] += x
    if (j+1 < N) A[j+1] -= x
}

```

TC: $O(Q+N)$

SC: $O(1)$

Break

```

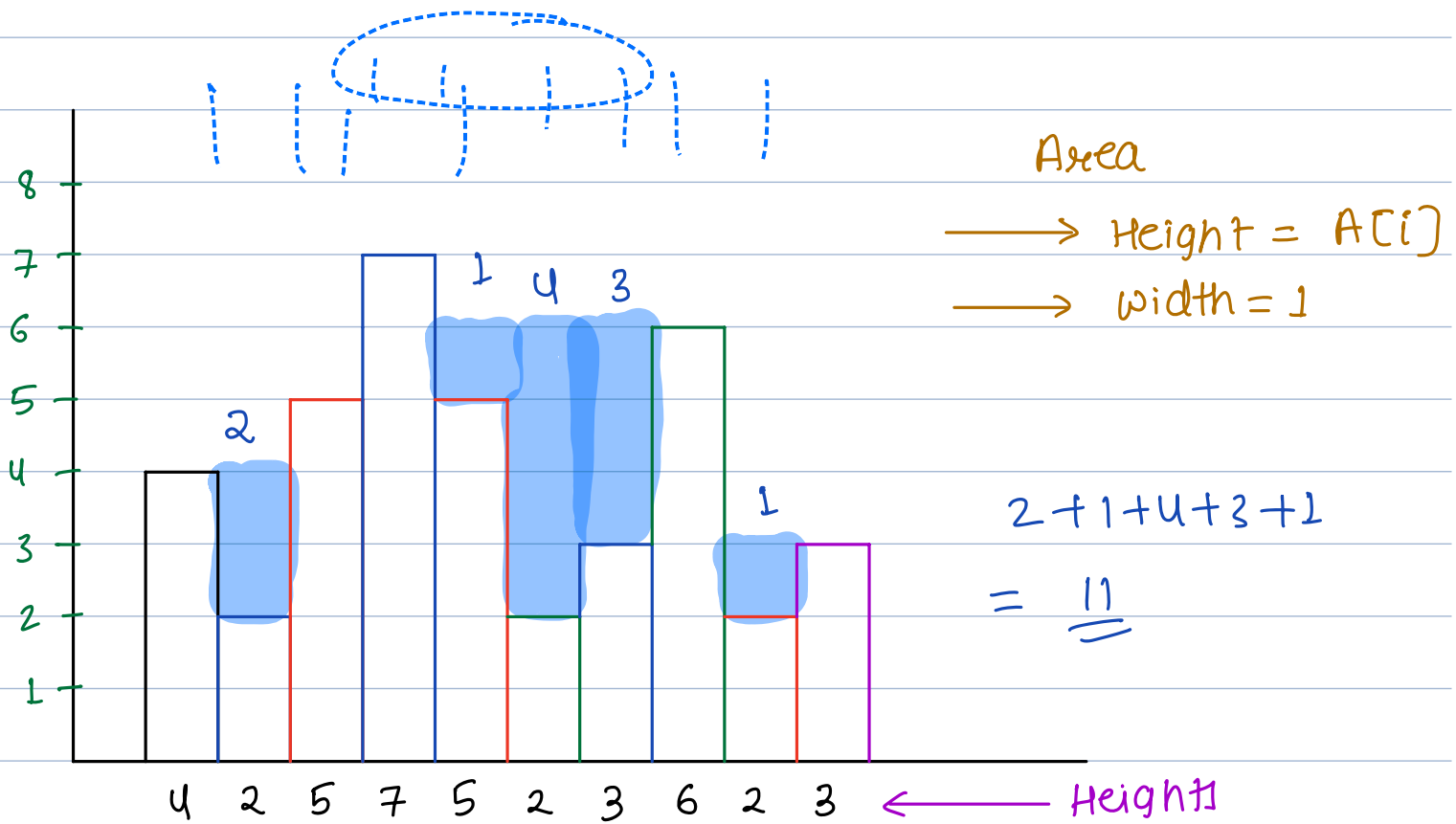
for ( i → 1 to N-1 ) {
    A[i] = A[i] + A[i-1]
}

```

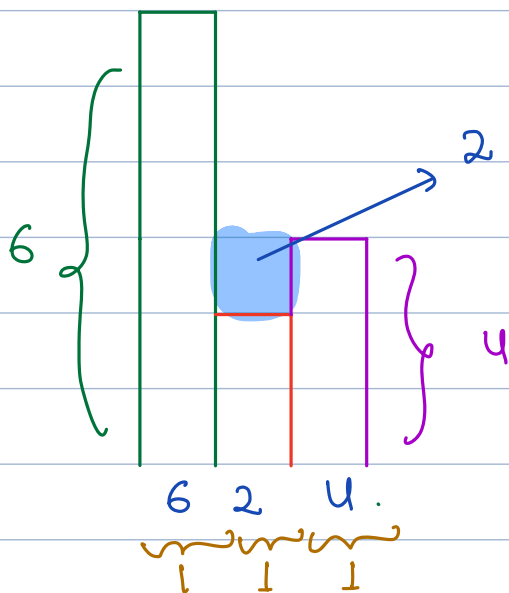
22:38

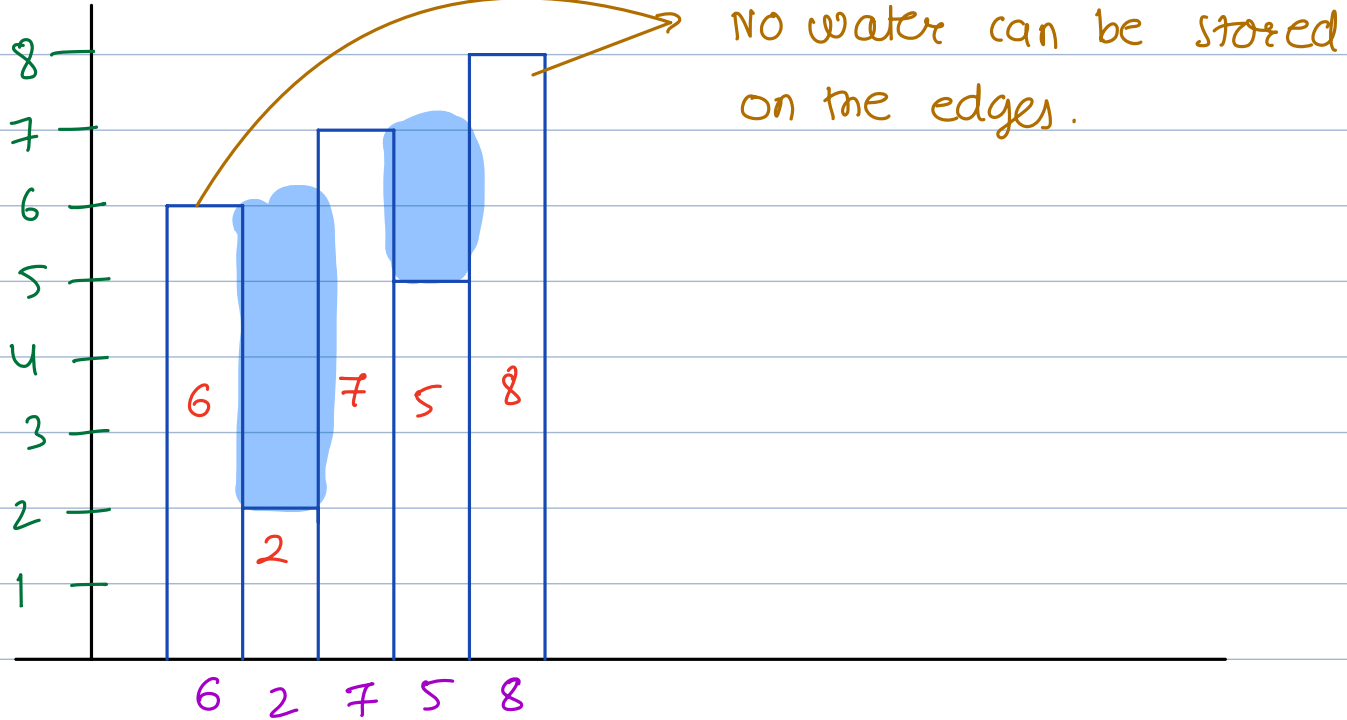
print (A)

Q> Rain water Trapping

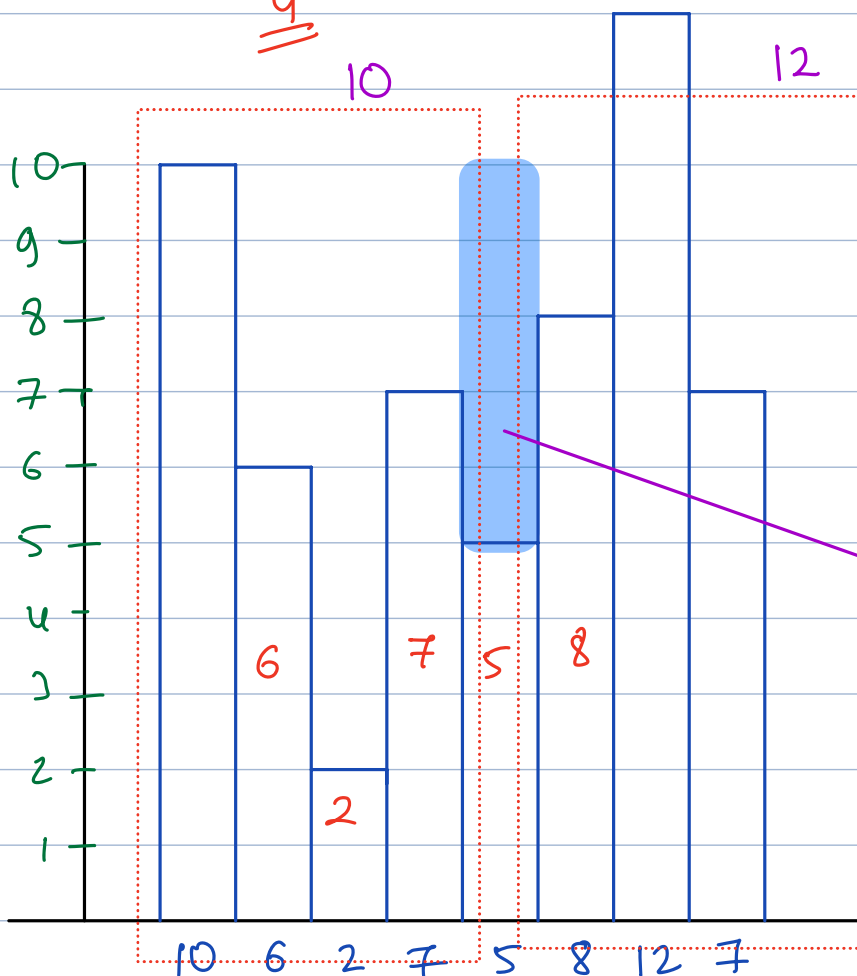


Given $A[N]$ each value represents building height
 calculate the total water accumulated coz of rain





$$\begin{aligned}
 & \underbrace{6 \quad 8} \\
 & 6 - 2 \\
 & \underline{\underline{4}}
 \end{aligned}$$



Till height 10
I can fill the water

So above building 5
water stored

$$\longrightarrow 10 - 5 = \underline{\underline{5}}$$

Bruteforce

water = 0

TC $O(N^2)$

SC $O(1)$

for $i \rightarrow 0$ to $N-1$ {

leftMax = $-\infty$ // 0 will work too.

for $j \rightarrow i$ to 0 {

leftMax = max(leftMax, A[j])
}

rightMax = $-\infty$ // 0 will work too.

for $j \rightarrow i$ to $N-1$ {

rightMax = max(rightMax, A[j])
}

waterHeight = min(leftMax, rightMax)

area = waterHeight - A[i]

water += area

}

print(water)

	↓		
	6	2	4
leftMax	6	6	6
rightMax	6	4	4
waterHeight	6	4	4
area	6-6	2	4-4
water	0	2	<u>2</u>

	4	2	5	7	5	2	3	6	2	3	
left max	4	4	5	7	7	7	7	7	7	7	→ prefix Max
right max	7	7	7	7	6	6	6	6	3	3	→ suffix Max
water Height	4	4	5	7	6	6	6	6	3	3	
area	0	2	0	0	1	4	3	0	1	0	
water	0	2	2	2	3	7	10	10	11	11	

prefix $R \rightarrow L$ = suffix

Pseudocode

```

prefixMax = new int[N]
leftMax = -∞
for i → 0 to N-1 {
    leftMax = max(leftMax, A[i])
    prefixMax[i] = leftMax
}

```

```

suffixMax = new int[N]
rightMax = -∞
for i → N-1 to 0 {
    rightMax = max(rightMax, A[i])
    suffixMax[i] = rightMax
}

```

water = 0

for $i \rightarrow 0$ to $N-1$ {

TC: $O(N)$
SC: $O(N)$

 leftMax = prefixMax[i]
 rightMax = suffixMax[i]
 waterHeight = $\min(\text{leftMax}, \text{rightMax})$
 area = waterHeight - A[i]
 water += area
}

doubt session { attendance is optional
during doubt session }

3 7 4 5 6 1 mloc = 1

left max

right max

water height

area

7 6 5 4 3 2 1

water

100 -1 5

sum 100 99 104

ans - 100 100 104