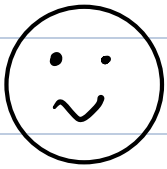
Array : One Dimensional

Welcome to Advance DSA 1 Module



Hello Everyone

Q) Given an integer array A of size N. find maximum subarray sum out of all subarrays.

Eg.

0	1	2	3	4	5	6
-2	3	4	-1	5	-10	7

Ans = 11

Eg.

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

Ans = 18

Quiz!:

0	1	2	3	4
4	5	2	1	6

Ans = 18

Quiz 2: {-4, -3, -6, -9, -2}

Ans = -2

Brute force:

$O(N^2)$

→ Consider all subarray.

→ Iterate over each subarray to calculate sum and compare with maximum answer.

$O(N)$

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

Using Prefix sum → TC: $O(N^2)$ SC: $O(N)$

A[0]

Ans = ~~-3~~ ~~X~~ ~~7~~

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

j

18

Sum = ~~0 4 10~~ 18

```
ans = A[0];
for (i=0; i<n; i++)
{   sum = 0
    for(j=i; j<n; j++)
    {
        sum = sum + A[j];
        if (sum > ans) { ans = sum;}
    }
}
return ans;
```

TC: $O(N^2)$

SC: $O(1)$

Optimisation: Possible cases.

Case 1: All elements are positive.

{4, 2, 1, 6, 7} sum of entire array.

Case 2: All elements are negative.

{-4, -2, -1, -6, -7} → max element.

Case 3: Some positive in between negatives.

{-3, -2, -6, 4, 3, 6, 9, -5, -3, -1, -6}

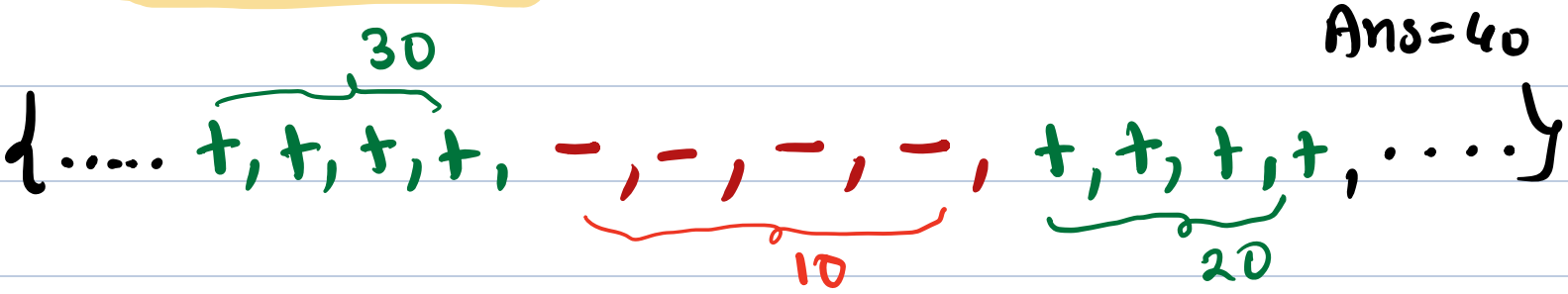
Case 4: Positive and negative on either side.

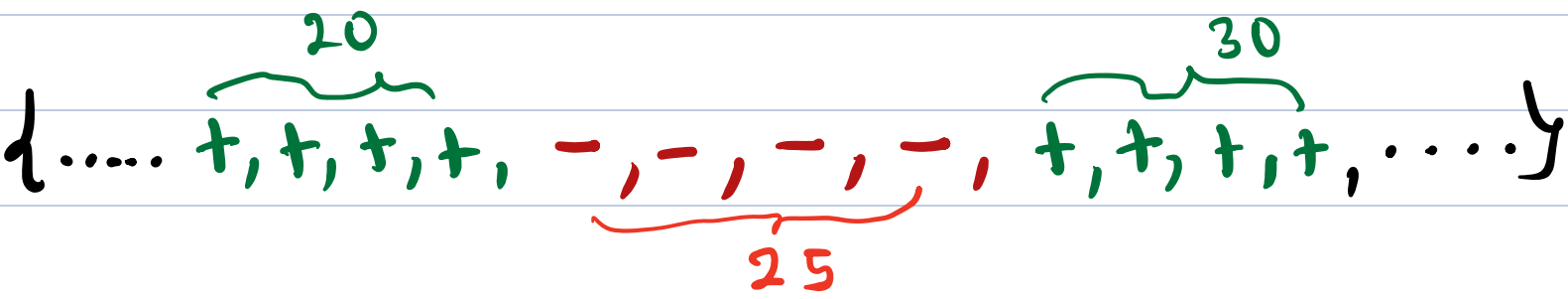
{ -3, -2, -6, 5, 3, 4, 6 }

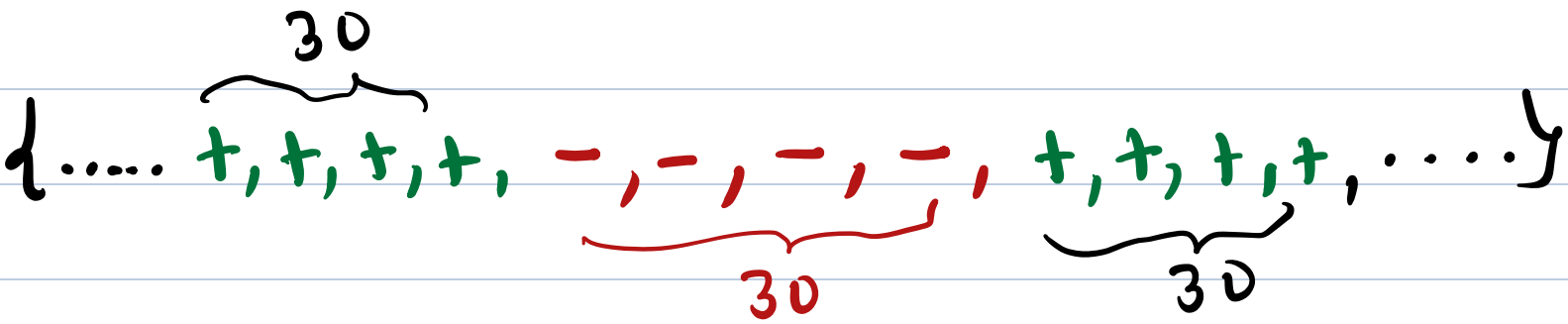
Kadane's

Generic Case:

Ans = 40







Eg. {-2, 3, 4, -1, 5, -10, 7} (i)

ans = ~~-2~~ ~~3~~ ~~7~~ 11

sum = ~~-2~~ ~~0~~ ~~3~~ ~~7~~ ~~6~~ ~~11~~ ~~4~~ ~~8~~

Eg. {-20, 10, -20, -12, 6, 5, -3, 8, -2} (i)

currSum = ~~0~~ ~~-20~~ ~~0~~ ~~10~~ ~~-10~~ ~~0~~ ~~-12~~ ~~0~~ ~~6~~ ~~11~~ ~~8~~ ~~16~~ 14

maxSum = ~~-20~~ ~~10~~ ~~11~~ 16

(i)

{-3 -4 -5}

currSum = ~~-3~~ 0

Ans = -3

Ques 3:

0	1	2	3	4	5	6
-2	3	4	-1	5	-10	7
						i

currSum = ~~0 -2 0 3 7 6 11~~ 8

maxSum = ~~-2 3 7~~ 11

Code:

```
ans = A[0];   sum = 0
for (i = 0 ; i < n ; i++)
{
    sum = sum + A[i];
    if (sum > ans)
    {
        ans = sum;
    }
    if (sum < 0)
    {
        sum = 0;
    }
}
return ans;
```

TC: O(n)
SC: O(1)

Q) Given an integer array A, where every element is 0. return the final array after performing multiple queries.

Query(i, x): Add x to all elements from index i.

Queries $A = \{0, 0, 0, 0, 0, 0, 0\}$ (indices 0 1 2 3 4 5 6)

(1, 3) +3 +3 +3 +3 +3 +3

(4, -2) -2 -2 -2

(3, 1) +1 +1 +1 +1

$A = \{0, 3, 3, 4, 2, 2, 2\}$

Quiz 4

A = { 0, 0, 0, 0, 0 }
0 1 2 3 4

Queries

(1,3) +3 +3 +3 +3

(0,2) +2 +2 +2 +2 +2

(4,1) +1

{ 2, 5, 5, 5, 6 }

Brute force: For each query, traverse the array to update the value.

$$TC: O(N * Q)$$

Optimization:

	0	1	2	3	4
A =	0	0	0	0	0

Queries
(1, 3) +3
(0, 2) +2
(4, 1) +1

A = {2, 3, 0, 0, 1}

PF = {2, 5, 5, 5, 6}

⇐ Ans

It contains queries

Code:

0	3
1	2
2	-4
0	2

```
for (i=0; i<Q; i++)
{
    index = B[i][0];
    value = B[i][1];
    A[index] = A[index] + value;
}
for(i=1; i<N; i++)
{
    A[i] = A[i-1] + A[i];
}
return A;
```

O(Q)

O(N)

TC: O(N+Q)

SC: O(1)

A {∅, ∅, ∅, 0, 0}

3 2 -4

5

PF: {5, 7, 3, 3, 3}

Variation:

Query $\longrightarrow$ (i, j, x)

Add element x to all the indices between i to j.

	0	1	2	3	4	5	6
A = {	0,	0,	0,	0,	0,	0,	0 }

Queries

(1, 3, 2) +2 +2 +2

(2, 5, 3) +3 +3 +3 +3

(5, 6, -1) -1 -1

= { 0, 2, 5, 5, 3, 2, -1 }

Break: 10.20

Quiz 5: A = {0, 0, 0, 0, 0, 0, 0, 0}

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

A = {-1, 2, 6, 2, 5, 2, 3, 0}

	0	1	2	3	4	5	6	7
A =	0	0	0	0	0	0	0	0
		+3	+3	+3	+3	+3	+3	+3
						-3	-3	-3

i j x

1 4 3

```
A[i] = A[i] + value;
A[j+1] = A[j+1] - value;
```

B →

1	4	3

0	3	-1
2	2	4

Code:

```
for(i=0; i<Q; i++)
{
    int S1 = B[i][0];
    int E1 = B[i][1];
    int value = B[i][2];
    A[S1] = A[S1] + value;
    A[E1+1] = A[E1+1] - value;
}
for(i=1; i<N; i++)
{
    A[i] = A[i-1] + A[i];
}
return A;
```

O(Q) — for the Q loop

E1+1<N

O(N) — for the N loop

TC: O(N+Q)

SC: O(1)

Q 2 2 2 4

i=0, SE=1, E1=4, val=3

0 1 2 3 4

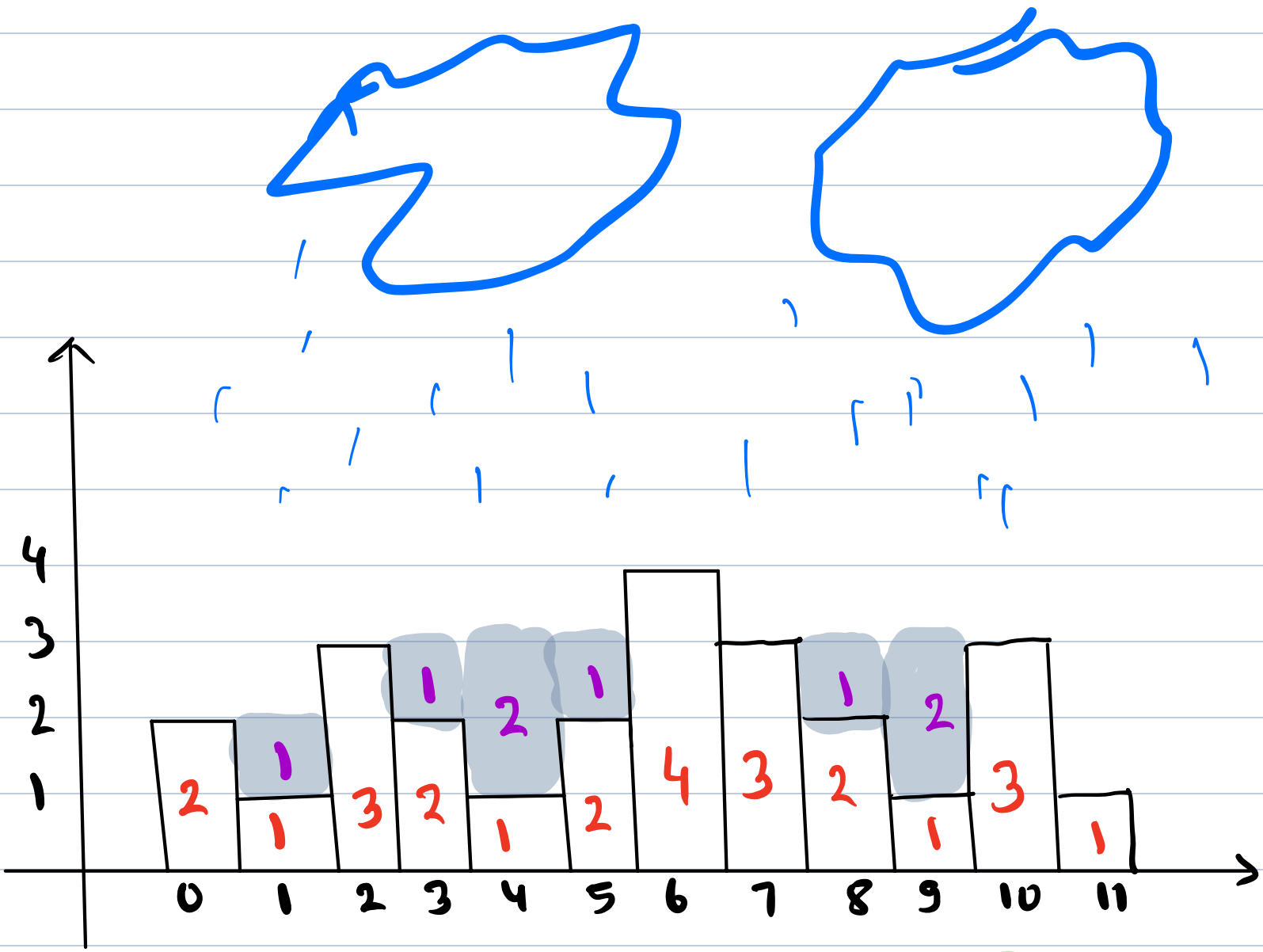
{0, 0, 0, 0, 0}

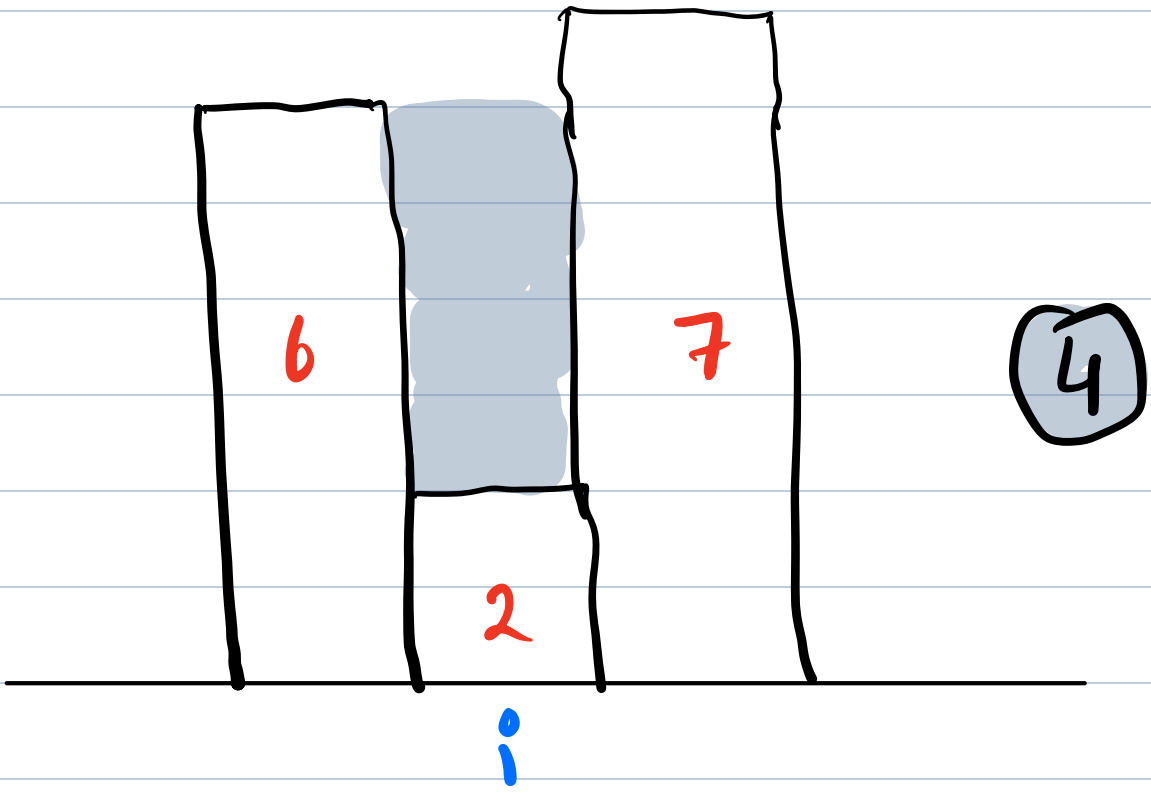
-1 3 4 -4 +1

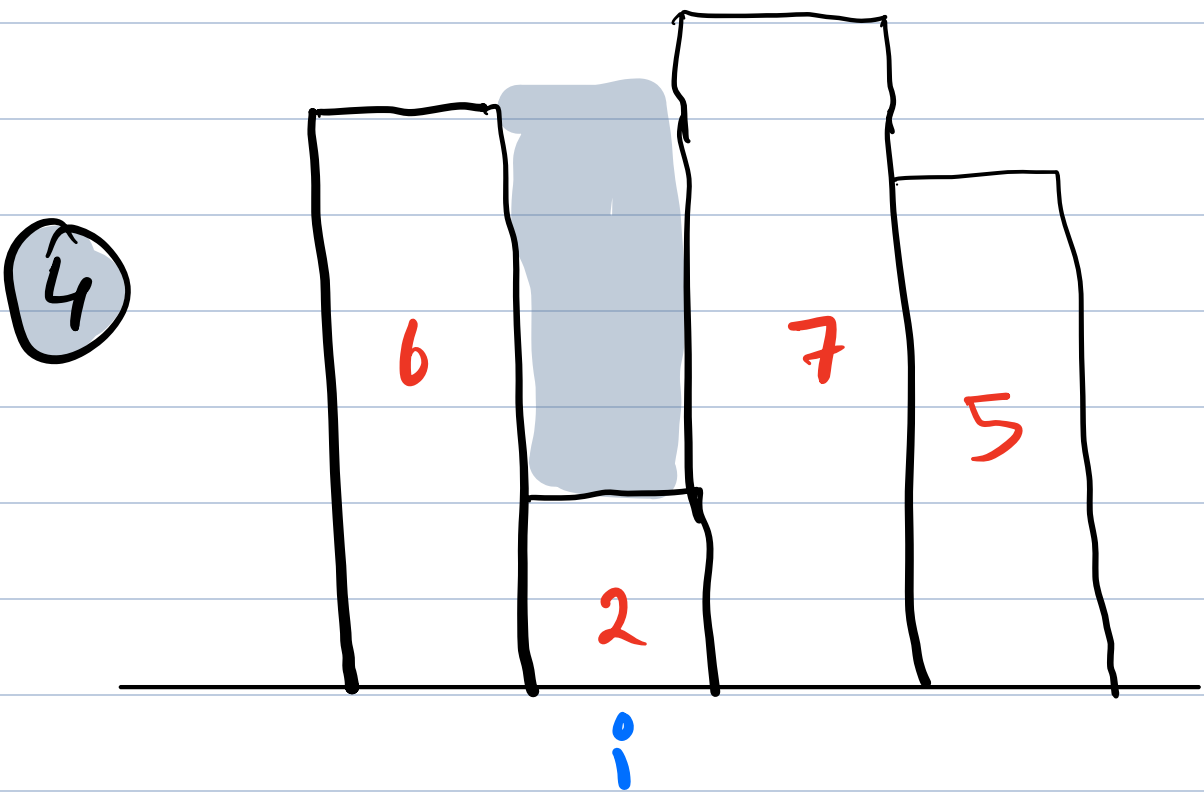
{-1, 2, 6, 2, 3}

Q) Given N buildings with height of each building. find the rain water trapped between the buildings.

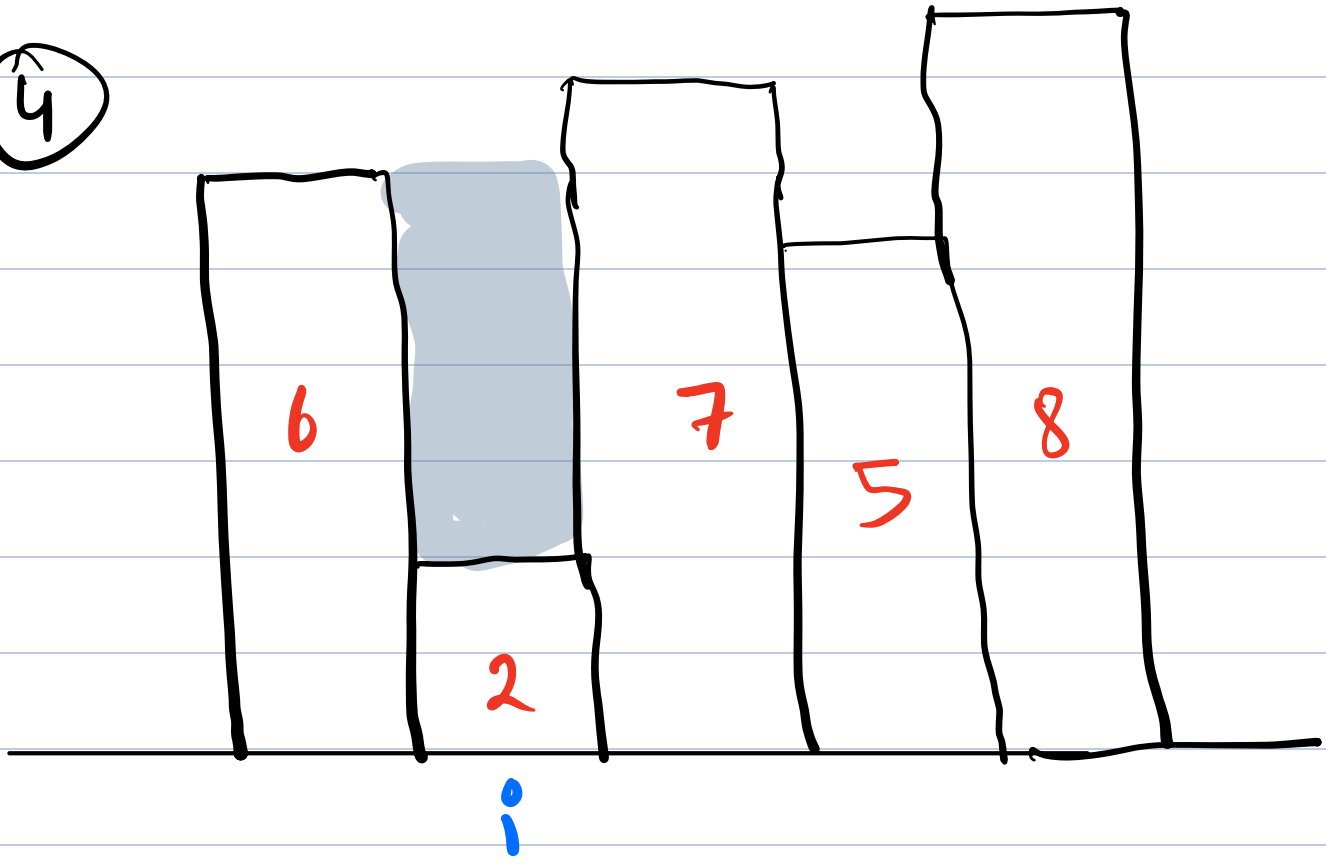
A[] → { 2, 1, 3, 2, 1, 2, 4, 3, 2, 1, 3, 1 }

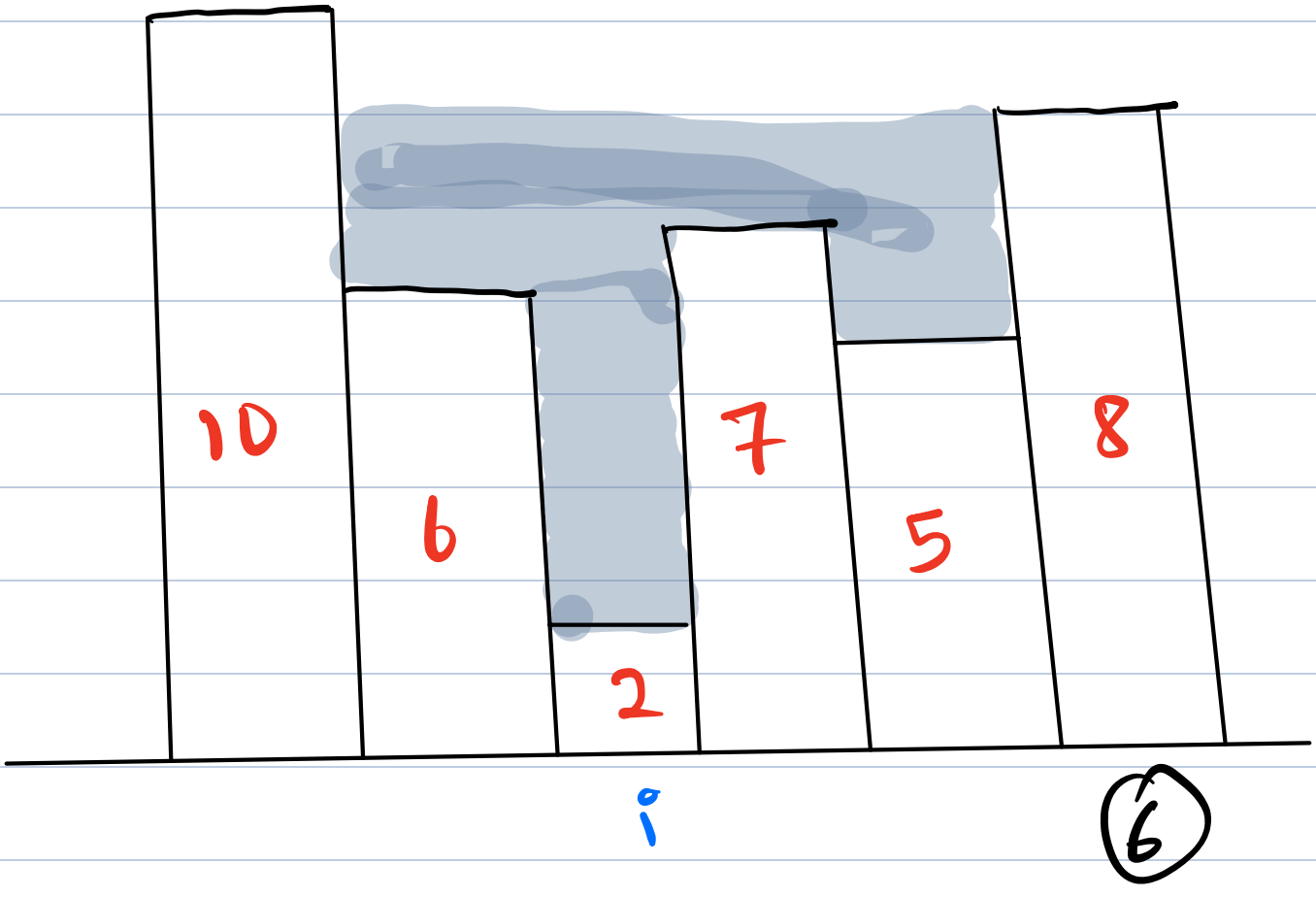


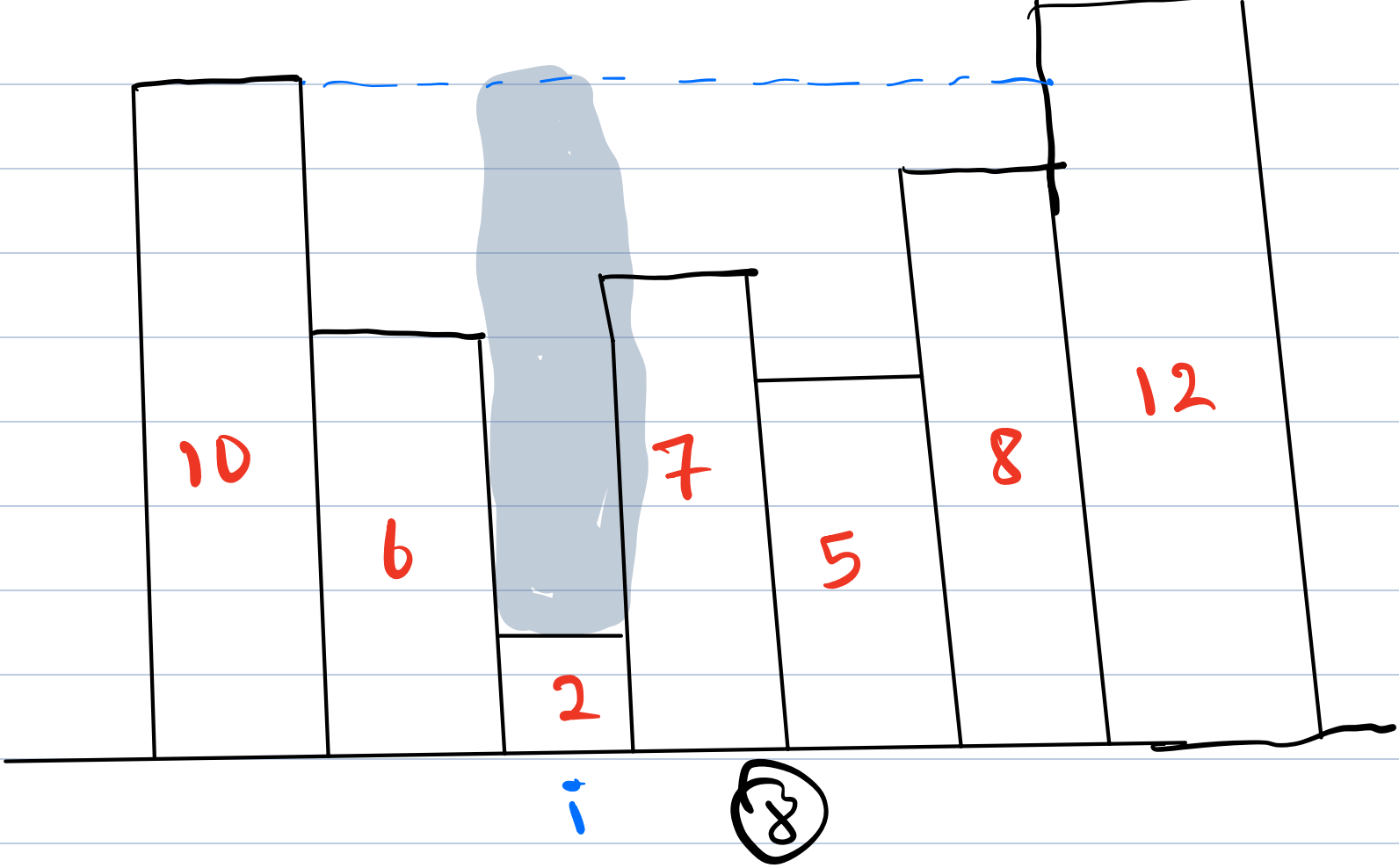

6
7
4
2
i


4
6
7
5
2
i

4





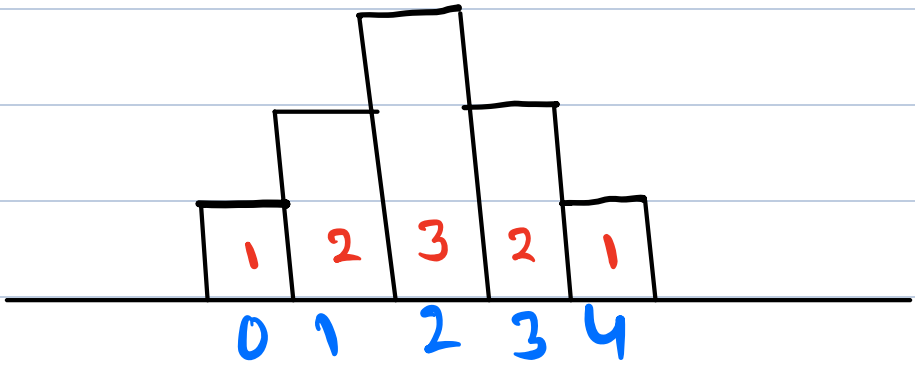


Conclusion: The amount of water that will be accumulated over building of index i =

$$\min(\text{maxHeight}(0, i-1), \text{maxHeight}(i+1, n-1)) - \text{height}(i)$$

Ans = 0

Quiz 6:



maxLeft = 1
maxRight = 3

min = 1

Brute Force:

For every i^{th} building, we need to find max. height on left side and right side.

```
ans=0

for(i=0; i<n; i++)
{
    left=0, right=0
```

```
for(l=0; l<i; l++)
{
    if(A[l]>left)
    {  left = A[l];
    }
}
for(r=i+1; r<N; r++)
{
    if(A[r]>right)
    {  right = A[r];
    }
}
level = min(left, right)
if(level > A[i]
{
    ans = ans + level - A[i];
}
}
return ans;
```

TC: $O(N^2)$

SC: $O(1)$

{2, 1, 3, 2, 5}

i

Ans = ~~0~~ X 2

0 1 2 3 4

{2, 1, 3, 2, 5}

Leftmax {0, 2, 2, 3, 3}

Rightmax {5, 5, 5, 5, 0}

Optimisation: We can store the leftmax and rightmax using carry forward approach.

LMax[i] → max ele from 0 to i-1.

RMax[i] → max ele from i+1 to N-1.

```
ans=0
int[] LMax[N];
int[] RMax[N];

for(i=1; i<N; i++)
{
LMax[i]=max(lmax[i-1], A[i-1]);
}
```

A {2, 3, 5, 4, 1}

L {0, 2, 3, 5, 5}

```
for(i = N-2; i >= 0; i--)
{
    Rmax[i] = max(Rmax[i+1],
                  A[i+1]);
}
```

i

A {2, 3, 5, 4, 1}

{5, 5, 4, 1, 0}

```
for (i = 1; i <= N-2; i++)
{
    water = min(Lmax[i], Rmax[i])
            - A[i];

if(water > 0)
```

```
        }
    }
    return ans;
```

	0	1	2	3	4 (i)
A	2	3	5	4	1
L	0	2	3	5	5
R	5	5	4	1	0

ans = 0

i = 1, L = 2, R = 5, M = 2

water = 2 - 3

= -1

i = 2 L = 3, R = 4 M = 3

water = 3 − 5

= −2

i = 3 L = 5, R = 1, M = 1

water = 1 − 4

= −3

TC: O(N)

SC: O(N)