# Maximum subarray sum

06 August 2023 12:56

arr = [2, -5, [7, -6, 5, 4], -10]

ans = 10

generate all possible subarrays

↑

① 2 loops

~~$O(N^2)$~~ $O(N)$

$O(1)$

arr = [3 4 -6 8 -3 1]

ans = 9

Kadane's algorithm ②

| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| arr | 2, | -5, | 7, | -6, | 5, | 4, | -10 |

maxSum = ~~INT_MIN~~ ~~2~~ ~~7~~ 10

currSum = ~~0~~ ~~2~~ ~~-3~~ ~~0~~ ~~7~~ ~~1~~ ~~6~~ ~~10~~ 0

+ve +ve

100 100

= 200

+ve -ve

100 -50

= 50

+ve -ve

50 -100

= -50

0 ⇒

If currSum is negative, make it zero.

$O(N)$ - Time

$O(1)$ - Space

max = -∞    INT_MIN

min = ∞    INT_MAX

arr : [ -4  -2  -6  -1 ]

currSum = ~~0~~ ~~-4~~ ~~0~~ ~~-2~~ ~~0~~ ~~-6~~ ~~0~~ -1

maxSum = ~~INT_MIN~~ ~~-4~~ ~~-2~~ -1

arr = [ -7  -9  -1 ]

curr = ~~0~~ ~~-7~~ ~~0~~ ~~-9~~ ~~0~~ -1

max = ~~INT_MIN~~ ~~-7~~ -1

Max product of 3 numbers

06 August 2023 13:41

strictly 3 numbers

arr = [ 1 5 0 3 4 ]

ans = $5 \times 3 \times 4 = 60$

arr = [ -2 0 4 -3 1 ]

ans = 24 ($-2 \times 4 \times -3$)

arr = [ -1 -5 0 -3 ]

ans = 0 ( )

arr = [ -3 10 2 -8 4 -6 ]

ans = 480 ($10 \times -8 \times -6$)

1. 3 for loops. $O(N^3)$ $O(1)$
2. sort $O(N \log N)$ $O(1)$
3. 5 variables $O(N)$ $O(1)$ → single pass

-8 -6 -3 2 4 10

op1 = max1 X max2 X max3 ($2 \times 4 \times 10$) = 8

op2 = min1 X min2 X max1 ($-8 \times -6 \times 10$) = 480 op1 $(-8 \times -8 \times 10) = 480$

op2 = min1 × min2 × max1

ans = max (op1, op2) = $\boxed{480}$

−5   −4   −3   −2   −1

$(-5 \times -4 \times -1) = -20$

$(-3 \times -2 \times -1) = \boxed{-6}$

max1 = ~~−∞~~ ~~−3~~ 10

max2 = ~~−∞~~ ~~−3~~ ~~2~~ 4

max3 = ~~−∞~~ ~~−3~~ 2

min1 = ~~∞~~ ~~−3~~ −8

min2 = ~~∞~~ ~~10~~ ~~2~~ ~~−3~~ −6

| ~~i~~ | ~~i~~ | ~~i~~ | ~~i~~ | ~~i~~ | i |
|---|---|---|---|---|---|
| −3 | 10 | 2 | −8 | 4 | −6 |

$\longrightarrow$ $\boxed{\begin{array}{c} O(N) \\ O(1) \end{array}}$

op1 = max1 × max2 × max3 = $(10 \times 4 \times 2) = 80$

op2 = min1 × min2 × max1 = $(-8 \times -6 \times 10) = 480$

ans = max (op1, op2)

# Second largest number in an array

06 August 2023 13:56

~~i~~ ~~i~~ ~~i~~ ~~i~~ ~~i~~ ~~i~~ i

arr : [ -3, 0, 7, -4, -8, 2, 1 ]

9, 8

ans = 2

max1 = ~~$-\infty$~~ ~~-3~~ ~~0~~ 7

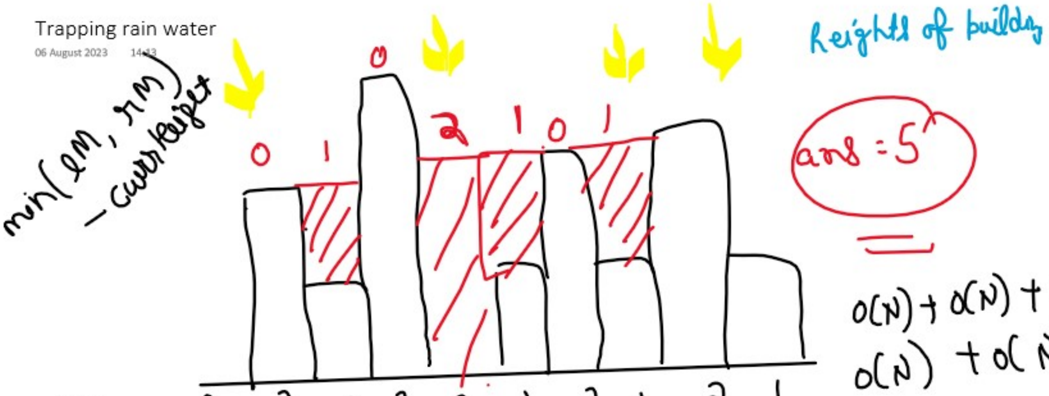
max2 = ~~$-\infty$~~ ~~-3~~ ~~0~~ 2

O(N log N) → sort

O(N) → 2 var

Trapping rain water

06 August 2023 14:13



ans = 5

Time Complexity

↓

$O(N) + O(N) + O(N) = 3N = O(N)$

$O(N) + O(N) = 2N = O(N)$

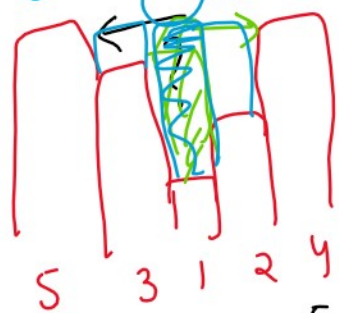
↑

Space complexity

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| arr :- | 0 | 2 | 1 | 3 | 0 | 1 | 2 | 1 | 2 | 1 | | |
| leftMax :- | 0 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | | |
| rightMax | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 1 | | |
| variable ~~water~~ :- | 0 | 0 | 1 | 0 | 2 | 1 | 0 | 1 | 0 | 0 | = | $\boxed{5}$ |

if left and right neighbouring buildings are of greater height.

0 1 $\boxed{3}$ 2 0 = 6 units



leftMax = 5 } min
rightMax = 4

$\boxed{4} - \boxed{1} = 3$ units

↓

curr height

↓ ↓

$$\boxed{\text{water} = \min(\text{leftMax}, \text{rightMax}) - \text{currHeight};}$$

$$= \min(5, 4) - 1 = 4 - 1 = \boxed{3}$$

3 approaches.

✗ ① Two loops. Indiviual find l and r.
$O(N^2) \rightarrow O(1)$ $O(N)$

✓ ② store leftMax and rightMax in array.
O(N) O(N)

② store leftMax and rightMax in array.

O(N) , O(N)

H.W.

③ Two pointers.

O(N), O(1)

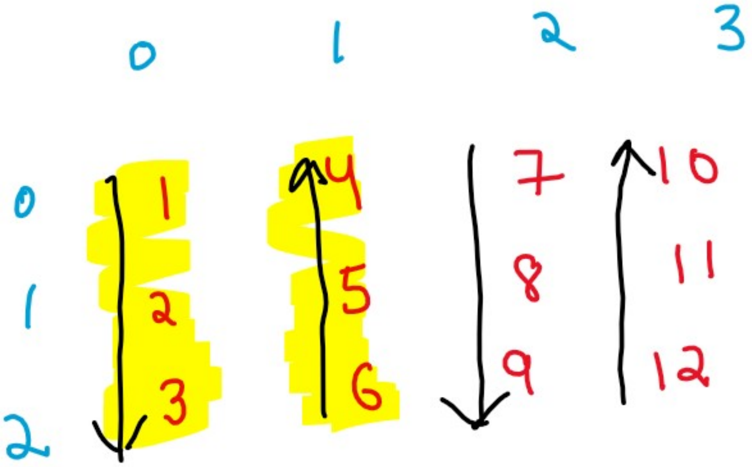
# Wave Print

06 August 2023 15:03

2D arrays

int matrix $\overset{\text{rows}}{[3]}\ \overset{\text{columns}}{[3]}$;

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1^{0,0} & 2 & 3^{0,2} \\ 1 & 4 & 5^{1,1} & 6 \\ 2 & 7^{2,0} & 8 & 9 \end{array}$$



$\underbrace{1, 2, 3,}\ \underbrace{6, 5, 4,}\ \underbrace{7, 8, 9,}\ \underbrace{12, 11, 10}$

rows = 3 (N)
cols = 4 (m)
(N×m)

```
for(col = 0 to m)
    for (row = 0 to n)
    for( row = n-1 to 0)
```

Col is even → Top down → for (row = 0 to n)

col is odd → Bottom up → for( row = n-1 to 0)

time → $O(N \times m) = O(N^2)$

space → $O(1)$

# Rotate matrix

06 August 2023 15:17

→ rotate the matrix 90° in clockwise direction.

→ square matrix

Time → $O(N^2)$
Space → $O(N^2)$

input

$$\begin{matrix} 1^{0,0} & 2^{0,1} & 3^{0,2} \\ 4^{1,0} & 5^{1,1} & 6^{1,2} \\ 7^{2,0} & 8^{2,1} & 9^{2,2} \end{matrix} \Rightarrow \begin{matrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{matrix}$$

output

transpose ↓ swap($\underline{i}$, $\underline{j}$)

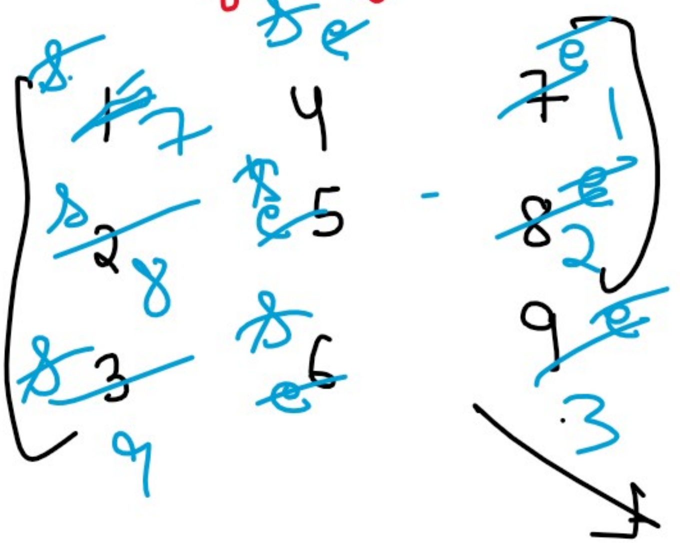
```
for(i=0 to n-1)
    for (j=i+1 to n-1)
```

1. Take transpose
2. Reverse the rows

Time → $O(N^2)$
Space → $O(1)$

→ $j=0$ && $j<i$

→ $j=i+1$ && $j<n$



reverse each row

$$\begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

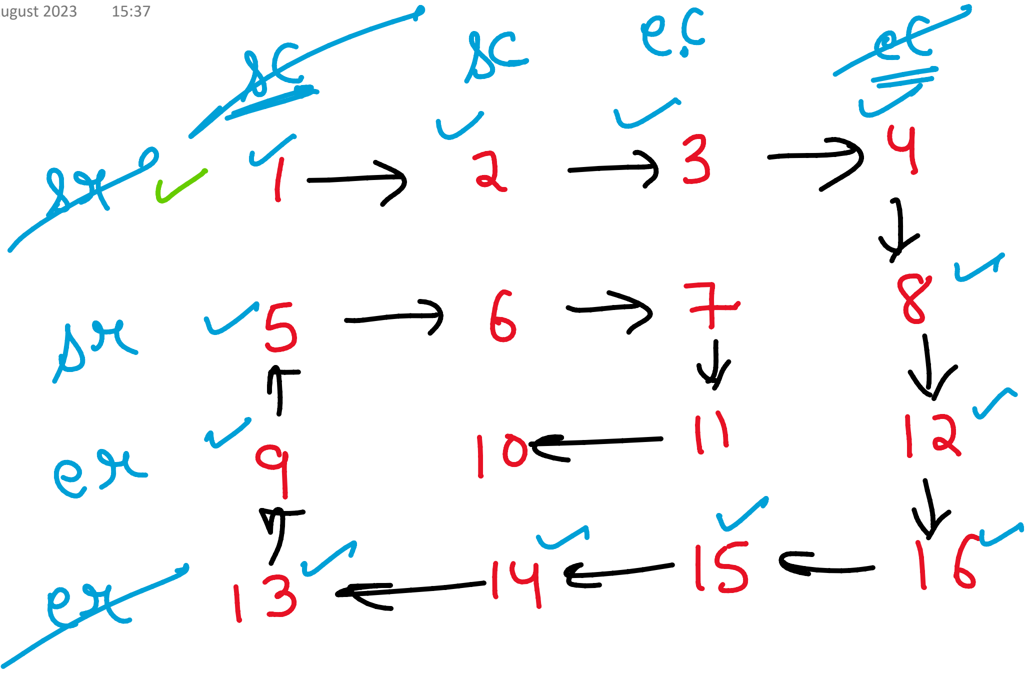
qe

gou

# Spiral Order Print

06 August 2023 15:37

n×m



| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 |

```
for(i = sc to ec)
    sr++
for(i = sr to er)
    ec--
for(i = ec to sc)
    er--
for(i = er to sr)
    sc++
```

1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10

while ( sr<=er or sc<=ec)

N → rows
M → cols

Time → O(N×m)
Space → O(1)

1 row → M cols
N row → N×M

N×M
4×4 = 16

sc ← ec ~~ec~~

1 2 3 4 5

~~sr~~ pr 6 7 8 9 10

sr 11 12 13 14 15

sr > er

7, 8, 9

(8, 7)