Stacks 2

Monday, 8 November 2021 8:15 PM

2 + 3 * 4
↳ prefix / postfix

---

① when do I push
② " " I pop
③ what do the values in the stack signify

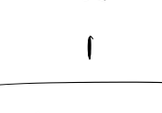
Nearest Smallest Element

given arr_N integers — strictly / equal works as well (works)
for each index find the nearest smaller element on the left (X including self)

0	1	2	3	4	5	6	7	8	9	10	11
10	5	6	−4	2	2	0	1	2	3	12	15
NA	NA	5	NA	−4	−4	−4	0	1	2	3	12

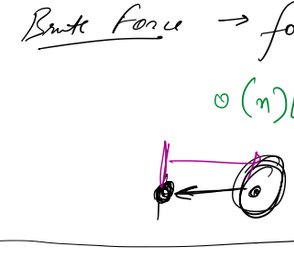
1	1	1	1	1
NA	NA	NA	NA	NA
5	4	3	2	1
NA	NA	NA	NA	NA
1	2	3	4	5
NA	1	2	3	4

Brute Force → for each element
O(n) [iterate on the left till you find a smaller element]
TC O(n²)
SC O(1)

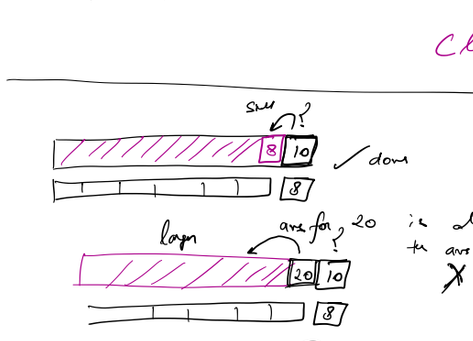


---

1 2 3 4
NA 1 2 3



prefix min
smallest on left ✓
closest X



100	80	60	70	60	75	85
NA	NA	NA	60	NA	60	75

60 75 85

Array — dynamic ✓✓✓ — at index / at index
> Stack pop/push/is-empty

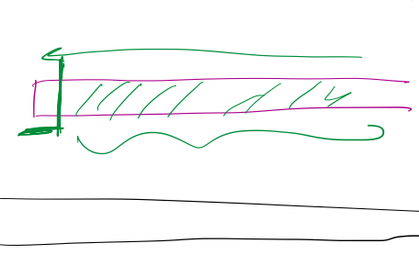
```
stack = []
ans = []
for x ← Ar:   O(n)
    while (!stack.isEmpty() && stack.top() >= A[i])   O(n)
        stack.pop()
    if stack.isEmpty()
        print (NA)            ans[i] = NA
    else
        print( stack.top() )  ans[i] = st.top()
    stack.push(x)
```

TC = O(n)
SC = O(n)

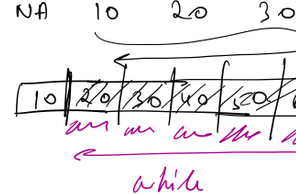
O(n²) X

Ar[i] → pushed 1 time (n)
pop 0/1 time ≤ n

max no of operation ≤ 2n

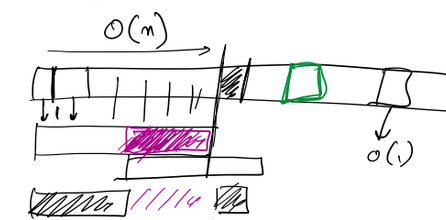


10	20	30	40	50	60	20	30	40
NA	10	20	30	40	50			



while O(n)

O(2n) stack operations



push → always push
pop → smaller smaller on the right
→ O(n)

(n − x) * (n − y) < n² = O(n²)
< O(n²) ≠ O(n)
$n^{1.5}$ n log n

---

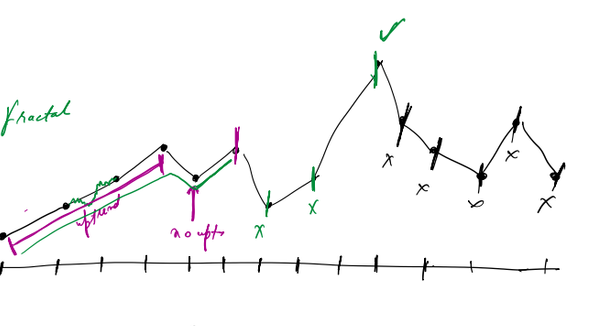
nearest smaller on left → O(n)
on right → O(n)
larger on left → O(n)
larger on right → O(n)

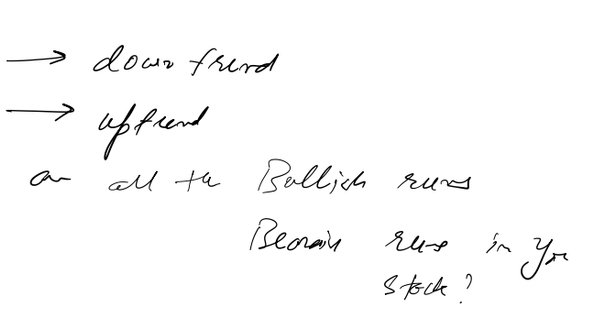
Ar → each index i find
smaller left
≤ right
≥ left
≥ right
O(n) time
O(n) space

Bearish → down trend
Bullish → uptrend

what are all the Bullish runs / Bearish runs in your stock?

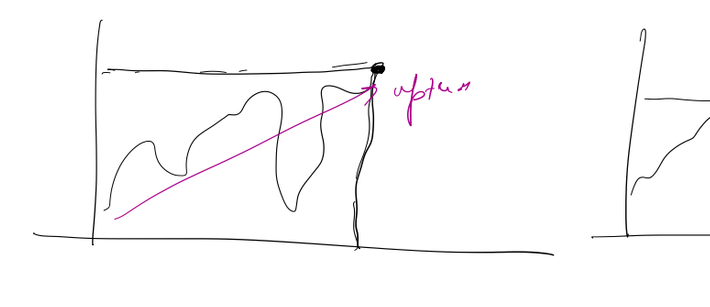
given any day
if all the days before it are all ≤ current price
✓ uptrend



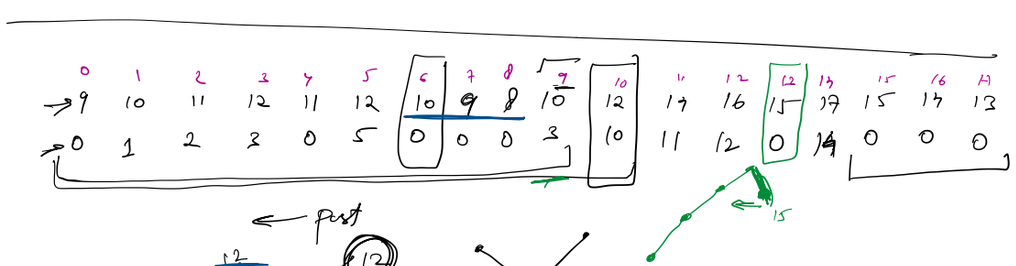


given N stock prices one for each day
for each day → identify for how long the market been in an uptrend

i → longest cons. seq of elements ending at i such that all elements of the seq ≤ A[i]



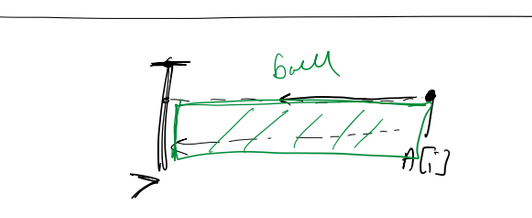
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
→9 10 11 12 11 12 10 9 8 10 12 14 16 15 17 15 14 13 15
→0 1 2 3 0 5 0 0 0 3 10 11 12 0 14 0 0 0 3



algo trading bot
↳ hypothesis → market is in bull → buy
P(up | bull for the past 10 days) = 0.7
buy order

---

each A[i] closest larger on the left



TC — O(n)
SC — O(n)

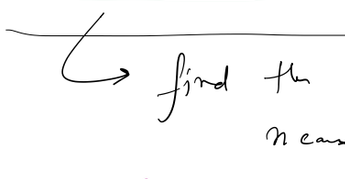
↳ find the nearest larger on left
nearest smaller on left

0	1	2	3	4	5	6	7	8	9	10	11	12
9	10	11	12	11	12	13	14	12	10	12	13	14
0	1	2	3	0	5	6	7	0	0	2	3	12

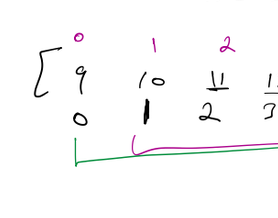
while (stack.top() <= A[i])
pop

12 days

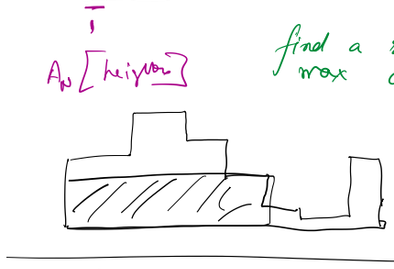
C++ → stack < pair<int, int>> / Point2D
java → stack < List<int> > 2 in size
Python → [(a, b), (x, y)]



find a rectangle of max area!



rectangle with max area.



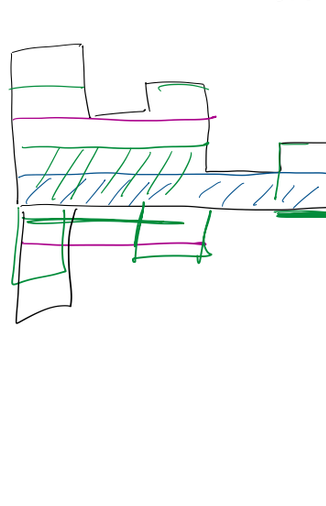
perspective of all rectangles — no confusion
invert the problem

find the area of the rectangle with the max area.

→ for height 1 find the longest base
↳ base #1
h 2 find the lg base
↳ base #2

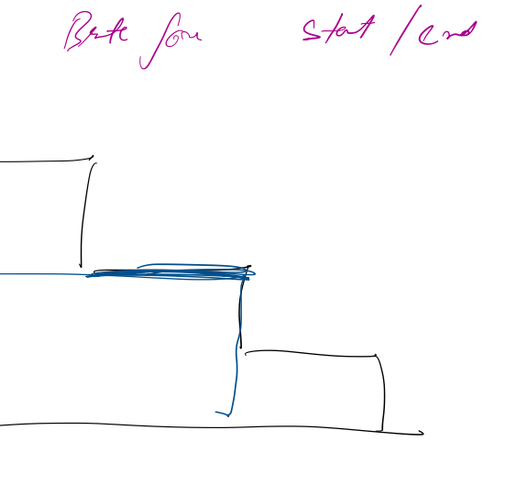
Brute force

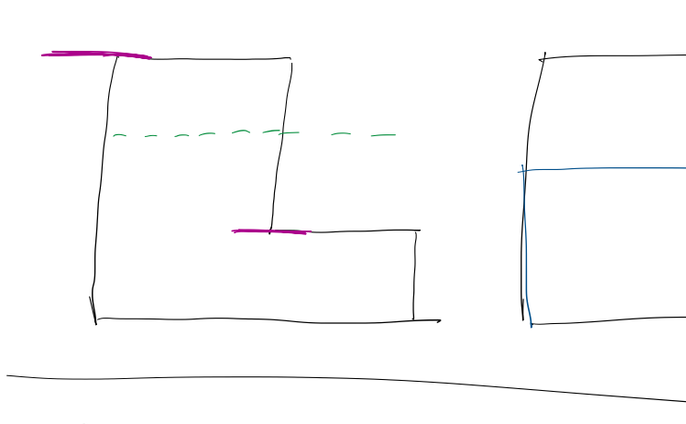
all rectangle → hard
for each height? find the longest base



max
for each height
for all start
for all ends
check if start—end maintain height
valid base
Area : h × b
↑ Σ height

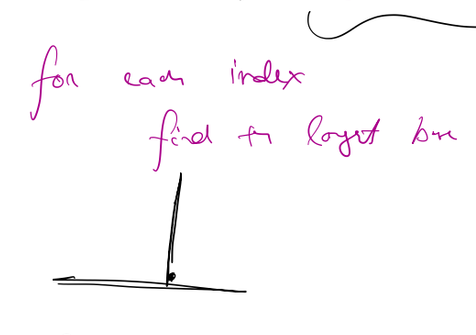
TC — O(n · n²)
SC —
O(n³) TC
O(n³) TC



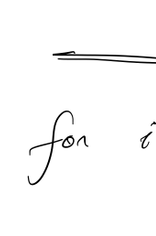


---

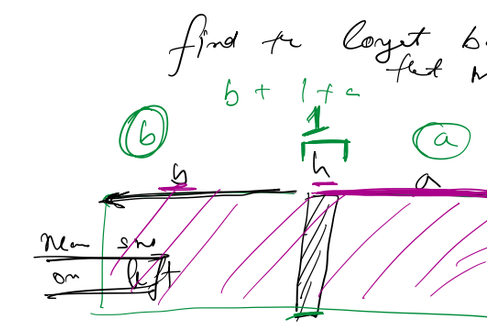
for each height
↳ find the longest base
BF



for each index
find the longest base including this ind

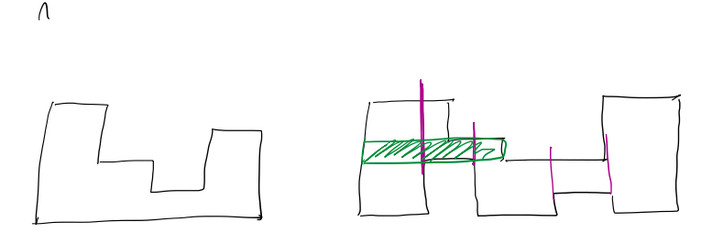


for i ← 0 … n−1
h = A[i]
find the longest base containing i that maintains a height of h
b + 1 + a



(a + b + 1) h

max
i



10:47 → 11:00

---

given A_N → distinct values
find the [max(S) − min(S)] for all subarrays of A_N
v₁ + v₂ + … for all subarrays

A_N = [1 3 2 0]

max − min
[1] → 0
[1 3] → 2
[1 3 2] → 2
[1 3 2 0] → 3

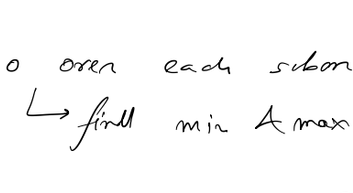
[3] → 0
[3 2] → 1
[3 2 0] → 3

[2] → 0
[2 0] → 2

[0] → 0

0 + 2 + 2 + 3 + 0 + 1 + 3 + 0 + 2 + 0 = 13

Brute force → go over each subarray → O(n²)
↳ find min & max → O(n)
O(1)
O(n³) TC
O(1) SC

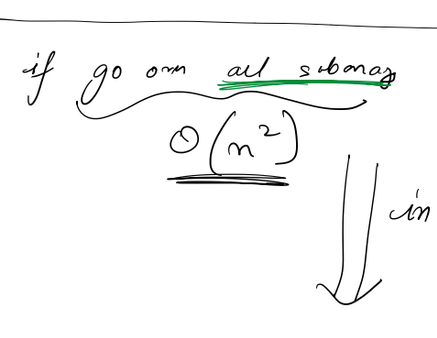


→ O(n²) TC
O(1) space

---

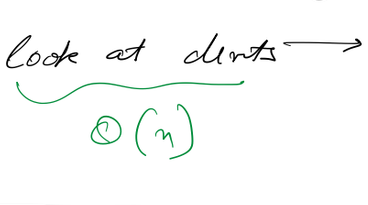
if go over all subarrays & do some computation
O(n²) O(1)
⇓ invert the problem.

look at elements → contribution of this element in the ans
O(n) O(1)

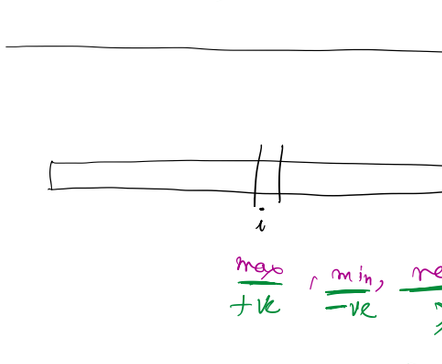


max +ve, min −ve, neither X

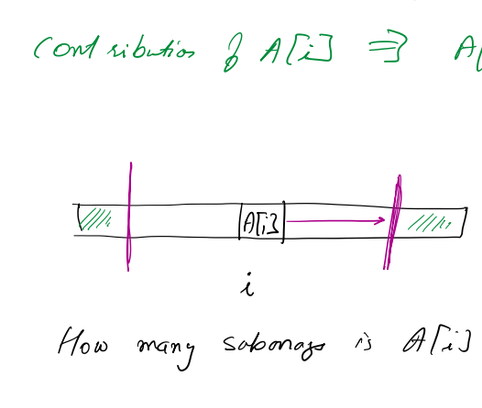
contribution of A[i] ⇒ A[i] * (# of subarray in which A[i] is max − # of sub in which A[i] is min)



How many subarrays is A[i] max in?



# of subarray in which A[i] is max = ?



(i − k) * (j − i)] * A[i]

# of subarr in which A[i] is min] * (−A[i])

TC — O(n)
SC — O(n)

A

value index W → for each i find the nearest smaller on left
index X → " " " " " " on right
index Y → " " " " larger on left
index Z → " " " " larger on right

O(n) space
O(n) time

total = 0
for i ← 0 … n−1
// subarrays in which A[i] is max
start from → Y[i]+1 … i
ends from → Z[i]−1 … i

(i − Y[i]) * (Z[i] − i) * A[i]
(i − W[i]) * (X[i] − i) * (−A[i])

total +=

O(n) time
O(n) space

9
A[i]
< 9
11
nearest larger

max − min for each subarray

A[i] is max in 10
A[i] —
A[i] —
A[i] —
10

A[i] is min 16 subs
A[i]
A[i]
A[i]
16

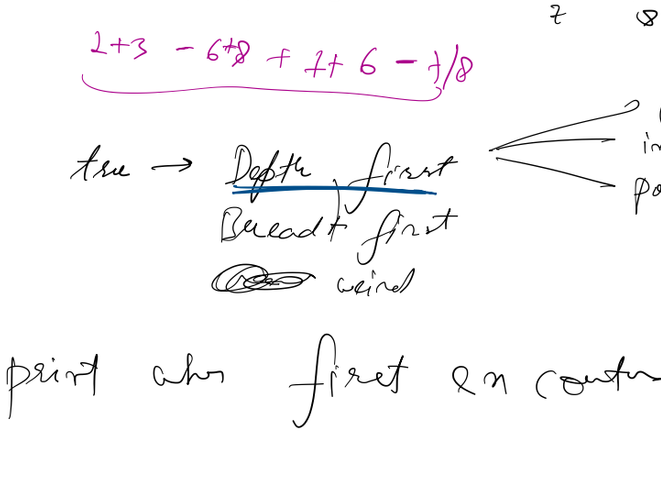
7 8 5 9 4 −1 10 16 8
in how many subarrays is 9 the max?

$$\sum_i (\#\max - \#\min) A[i] = \sum_i A[i] \left[(i - LL) * (LR - i) \atop (i - SL) * (SR - i) \right] \#\text{of subarray}$$

---

any computation → tree
expression

expression
2 + 3 − 6 * 8 + 7 + 6 − 7 / 8   infix notation



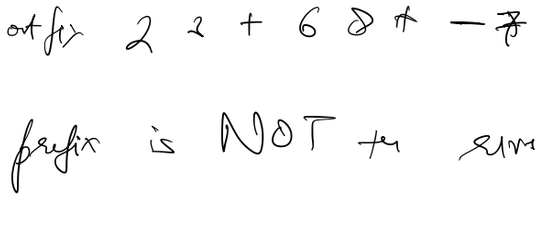
2 + 3 − 6 * 8 + 7 + 6 − 7 / 8

tree → Depth first, Breadth first, Level
prefix, infix, postfix

print when first encounter → prefix

root
left
right

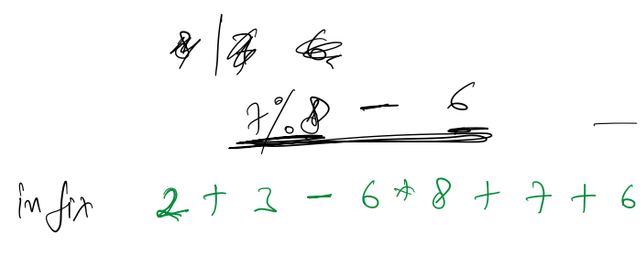
root, left, right → prefix   preorder
left, root, right → infix   in order
left, right, root → postfix   post order



L → R
T → D

prefix + − + 2 3 * 6 8 + 7 − 6 % 7 8
infix 2 + 3 − 6 * 8 + 7 + 6 − 7 % 8 → rarely used
postfix 2 3 + 6 8 * − 7 6 7 8 % − + +

prefix is NOT the rev of postfix



prefix + − + 2 3 * 6 8 + 7 − 6 % 7 8

7 % 8 − 6

infix 2 + 3 − 6 * 8 + 7 + 6 − 7 % 8

Binary operators a + b a − b a * b a % b
Unary → !a
ternary →
graph →

postfix 2 3 + 6 8 * − 7 6 7 8 % − + + ⊙
operand operators

2 + 3 = 5
5 − 48 = −43
6 * 8 = 48
48 − 5
7 / 8
2 + 3

Array of string
["2", "3", "−", "48", "+", …]

---

2 + 3 − 6 * 8 + 7 + 6 − 7 % 8

infix
2 + 3 − (6 + 7) * 8 − 3 / (7 + 4)

→ prefix
postfix → evaluate it

infix → postfix

(1 + 2) * 3 / (4 − 5) + 6 − 1   infix
2 1 + 3 4 5 − / * 6 1 − +   postfix

1 stack
(→ push
) → start popping & apply operators & remove the (
operator (1) → pop from the operator stack till we find operator of higher precedence
operand (2) → push

(1 + 2) * 3 / (4 − 5) + 6 − 1

S1 | 3 3
S2 | * /

1 + 2 = 3

less precedence than +

x * y − z

S1 | xy
S2 | −

---

2 * (3 / 6 + 1) − 2 + 3 / 4 + (3 + 2 / (2 * 3 + 2)) = −3.75

precedence rule
(), /, *, +, −

BODMAS
Bracket of Division Mult + Sub

operand
operator

0.5 + 1 = 1.5
2 * 1.5 = 3
3 / 4 0.75
2.75
2 / 2 = 1
6 − 2 = 4
1 * 3 = 3
3 + 3 = 6
− 2.75

precedence
associativity
2 + 3 + 5
(2 + 3) + 5

2 / 3 / 5 ⇒ $\frac{2/3}{5}$ ≠ $\frac{2}{3/5}$
(2 / 3) / 5

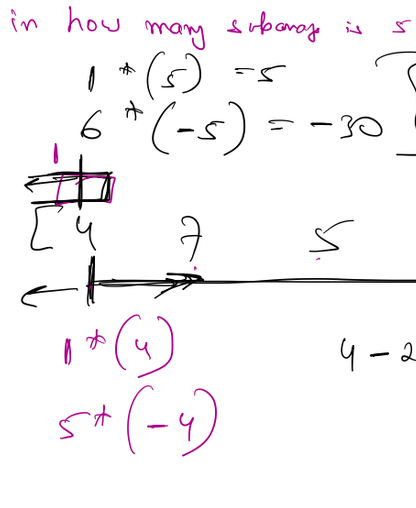
5
eval
2 + 3 = 5

postfix 2 3 +

1 + 2 * (1)
1 + 2 * (1) = 3
2 * 1 = 2
1 + 2 = 3

precedence, associativity, fixity — lexing, parsing, compiling

left to right
right to left



longest area rect

[4 7 5 8 9]   6

4 ① 7 5 ① 8 9
47 ① 75 ① 58 ① 89
475 ② 758 ② 589 ②
4758 ③ 7589 ③
47589 ④

in how many subarrays is 5 the max?
1 * (5) = 5
6 * (−5) = −30
−25

[4 7 5 8 9]
1 * (4)
5 * (−4)
4 − 20 = −16

[4 7 5 8 9]

for each subarray → find max a − find min b

for each element
A[i] → max

max min
A[i] —
A[i] —
# max ≈ * A[i]
− A[i] * min

Σ A[i] (# of max − # of min)