Lecture :- Stacks 2

Agenda

- Nearest smaller element
- Largest rectangle in histogram.
- (Max - min) for all subarrays

Qu! Given A[], find the index of nearest smaller element on left for all i index in A[].

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

Element	Nearest smaller element	Index of nearest smaller element
8	—	-1
2	—	-1
4	2	1
9	4	2
7	4	2
5	4	2
3	2	1
10	3	6

Bruteforce :- TC: $O(n^2)$
SC: $O(1)$

Quiz If A =

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

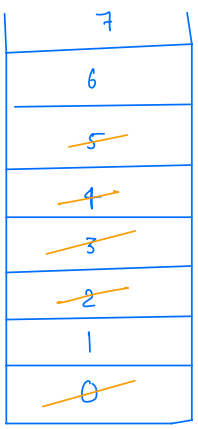
for any index > 5, can index 0 become nearest smaller element on left ?

a) Yes
b) No ✓

Brute force code h/w

Optimised Approach

	0	1	2	3	4	5	6	7
ip ⇒	8	2	4	9	7	5	3	10
result ⇒	-1	-1	1	2	2	2	1	6




```
while( !stack.isEmpty() &&
arr[i] <= arr[stack.peek()] ) {

        stack.pop();
}
if( arr[i] > arr[stack.peek()] )
        ans = top idx of stack
```

Pseudocode

```
for(i=0; i<n; i++) {
    while( !stack.isEmpty() &&
           arr[i] <= arr[stack.peek()] ) {
        stack.pop();
    }
    if( !stack.isEmpty()) {
        result[i] = stack.peek();
    } else{
        result[i] = -1;
    }
    stack.push(i);
}
```

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

Related Questions

Q. for all i, find nearest smaller element or equal element to left.

```
while( ! stack.isEmpty() &&
        arr[i] < arr[stack.peek()] ) {
    stack.pop();
}
```

Qu. for all i, find nearest greater element on left.

```
while( ! stack.isEmpty() &&
        arr[i] >= arr[stack.peek()] ) {
    stack.pop();
}
```

Qu. for all i, find nearest greater or equal element on left.

```
while( ! stack.isEmpty() &&
        arr[i] > arr[stack.peek()] ) {
    stack.pop();
}
```

Qu. for all i, find nearest smaller element on right.

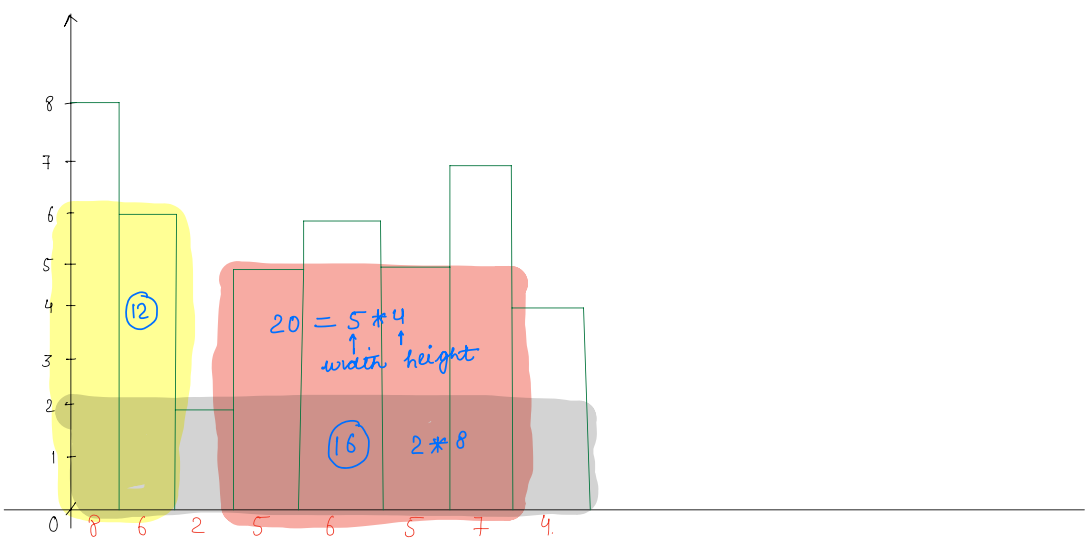
```
for(i=n-1 ; i>=0; i--) {
        while( ! stack.isEmpty() &&
                arr[i] <= arr[ stack.peek() ] ) {
            stack.pop();
        }
        if ( ! stack.isEmpty() ) {
            result[i] = stack.peek();
        } else{
            result[i] = -1;
        }
        stack.push(i);
}
```

Qu Given A[i] = height of ith bar.

width of each bar = 1

find the area of largest rectangle formed by continuous bars.

8	6	2	5	6	5	7	4



Brute force:

```
function findMaxRectangleArea( hist[] ) :
    maxArea = 0;
    for( i = 0 ; i<n; i++) {
        minHeight = arr[i];
        for(j = i; j<n; j++) {
            minHeight = min(minHeight, arr[j]);
            width = j - i + 1;
            area = minHeight * width;
            maxArea = max(maxArea, area);
        }
    }
}
```

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

Approach 2

	0	1	2	3	4	5	6	7
	8	6	2	5	6	5	7	4
nearest smallest on left	-1	-1	-1	2	3	2	5	2
nearest smallest on right	1	2	8	7	5	7	7	8

4 * (8 - 2 - 1) = 20

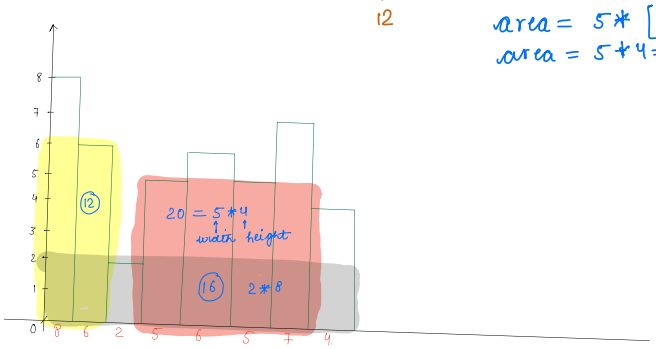
7 * (7 - 5 - 1)
7

5 * (7 - 2 - 1)
5 * 4 = 20

6 * (5 - 3 - 1)
6 * (1) = 6

6 * (2 - (-1) - 1)
6 * (3 - 1)
6 * 2
12

valid idx = (2, 7)
area = 5 * [7 - 2 - 1]
area = 5 * 4 = 20



Pseudocode

```
int largestRectangleInHistogram(A[]) {
    int[] left = nearestSmallerLeft(A);  —— O(n). O(n)
    int[] right = nearestSmallerRight(A);  —— O(n). O(n)
    int maxArea = 0;
    for(i=0; i<n; i++) {  —— O(n)
        int height = A[i];
        int width = right[i] - left[i] - 1;
        maxArea = max(maxArea, height * width);
    }
    return maxArea;
}
```

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

Break: 8:26 – 8:36

Qu Given A[] with distinct integers, for all subarrays find (max − min) and return its sum as answer

0	1	2
2	5	3

subarray	Max	Min	Max − Min	
[2]	2	2	0	
[5]	5	5	0	
[3]	3	3	0	
[2 5]	5	2	3	5 − 2
[5 3]	5	3	2	5 − 3
[2 5 3]	5	2	3	5 − 2
			\+	
			8 Ans	5−2 + 5−3 + 5−2
				3*5 − 2*2 − 1*3

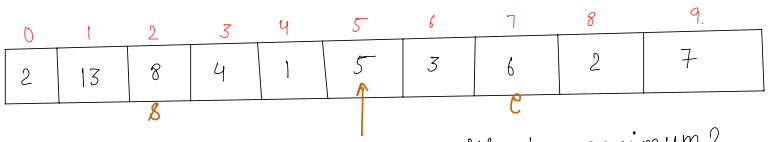
Brute force

∀ subarrays —
find max & min
calc. max − min = diff
ans += diff;

return ans

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

Approach



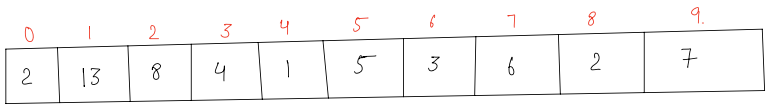
Qu In how many subarrays, 5 will be maximum?

Nearest greater el on left = 2nd idx.

" " " " right = 7th idx

∴ ith idx is max = left * right
= (i-s) * (e-i)
= [5-2] * [7-5]
= 3 * 2 = 6

4 1 5
1 5
5
5 3
1 5 3
4 1 5 3

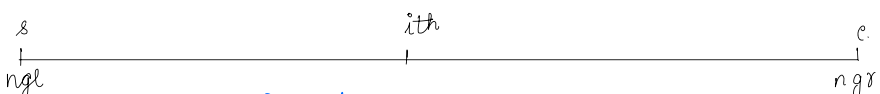


Qu In how many subarrays, 6 will be maximum?

Nearest greater el on left = 2nd idx

" " " " right = 9th idx

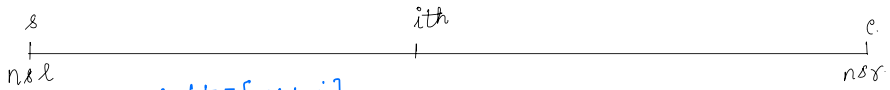
Generalisation



left = [s+1, i] = i - (s+1) + 1
= i - s - 1 + 1 = i - s.

right = [i, e-1] = e - 1 - i + 1 = e - i

∴ ith idx is max = left * right
= (i-s) * (e-i)



left = [s+1, i]

right = [i, e-1]

ith idx is min ⇒ left * right

⇒ (i-s) * (e-i)

0	1	2	3	4	5	6	7	8	9
2	13	8	4	1	5	3	6	2	7

Qu In how many subarrays, 5 will be minimum?

Nearest smallest el on left = 4th idx. [5]

" " " " right = 6th idx

Contribution of arr[i] in final ans?

arr[i] * ## max − arr[i] * ## min

arr[i] * (## max − ## min).

Pseudocode

```
int findsumofMaxAndMinDiff(A[]) {
    nsl[];
    nsR[];
    nGl[];
    nGR[];
    ans = 0;
    for(i=0; i<n; i++) {
        noofMax = (i - nGl[i]) * (nGR[i] - i);
        noofMin = (i - nsl[i]) * (nsR[i] - i);
        ans += arr[i] * (noofMax - noofMin);
    }
    return ans;
}
```

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

Thankyou :)