PAGE NO. icon
DATE : / /

Stack

1. Nearest greater to left
2. Nearest greater to right
3. Nearest smaller to left
4. Nearest smaller to right

(1) Nearest Greater to right

	0	1	2	3	4	5	6
arr[] :	1	3	0	0	1	2	4
o/p :	3	4	1	1	2	4	-1

Brute force

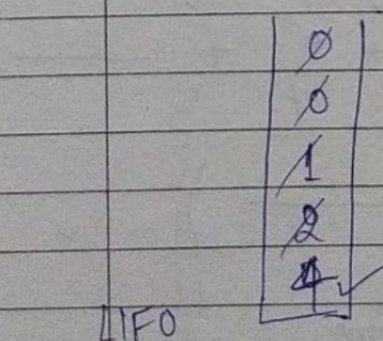
```
for (int i = 0; i < n-1; i++) {
    for (int j = i+1; j < n; j++) {
```

j depend on i → $O(n^2)$ → iske koi marks nhi milne wale

Matlab stack ka question

You really need to optimize

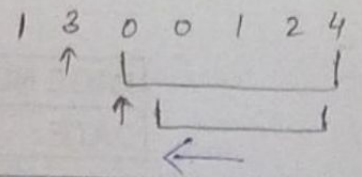
1 3 0 0 1 2 4



stack (not empty) X → s.top()

stack (empty) ✓ → -1

top() → jab tak stack ka top greater nhi ho jata



PAGE NO. icon
DATE : / /

1	3	2	4
i↑	i↑	i↑	i↑

List list

list : | -1 | 4 | 4 | 3 |

↓ reverse list

↓ o/p

$O(n^2) \rightarrow O(n)$

Code

* Stack empty → list.add(-1)

s.top() > arr[i] → s.top()

s.top() <= arr[i] → Pop
- stack empty
- s.top() greather than arr[i]

Code :-

```
List<Integer> list = new ArrayList<>();
Stack<Integer> st = new Stack<>();
for(int i=0; i<size; i++)        ③ ↓ I
                                 ② I
for(int i = size-1; i>=0; i--){
    if(st.size()==0){
        list.add(-1);
    }
    else if(st.size()>0 && st.peek() > arr[i]){   < ⑧ III
        list.add(st.peek());
```

1 3 2 4 ←

PAGE NO. icon
DATE : / /

```
else if( st.size() > 0 && st.peek() <= arr[i] ){   >= (3) III
    while (st.size() > 0 && st.peek() <= arr[i] ){   >= (3) III
        st.pop();
    }
    if (st.size() == 0) {
        list.add(-1);
    }
    else {
        list.add (st.peek() );
    }
}
st.push( arr[i] );
}
```

(2) No need to reverse (3) II

// also head return the REVERSED LIST (1) II

(2) Nearest greater ~~than~~ left

Left to Right

Change (1) Loop (2) I ⇒ left to right

(2) Don't reverse the list (2) II

PAGE NO.
DATE : / /

(3) Nearest smaller to Left

arr[] : | 4 | 5 | 2 | 10 | 8 |

i i i i i

o/p → -1 4 -1 2 2

```
for( int i =0 ; i<n ; i++)
    for( int j = i-1 ; j>0 ; j--)
```

j dep on i ⇒ STACK

Changes
① Less than ③ III
② Reverse X ③ II
③ Left Traverse ③ I

(4) Nearest Smaller to Right

arr[] : | 4 | 5 | 2 | 10 | 8 |

i i i i i

o/p 2 2 -1 8 -1

```
for( int i =0 ; i< n ; i++)
    for (j= i+1 ; j < n ; j++)
```

↳ j dep on i ⇒ STACK

Changes
① Loop : Right to left
② Smaller
③ Reverse

PAGE NO. icon
DATE : / /

Stock Span Problem

	0	1	2	3	4	5	6
arr :	100	80	60	70	60	75	85

↑ Day 6

consecutive smaller or equal

arr[]: 100 80 60 70 60 75 85

o/p ⇒ 1 1 1 2 1 4 6

For 75 ⇒ 60, 70, 60 are smaller & counting itself (75)

3 + 1 = (4)

Nearest greater to left (kahan stop kr rhe h? jab greater element mil rha hai on left)

	0	1	2	3	4	5	6
	100	80	60	70	60	75	85
	-1	100	80	80	70	80	100

index

1

5 − 1 = (4) → o/p

arr ka index

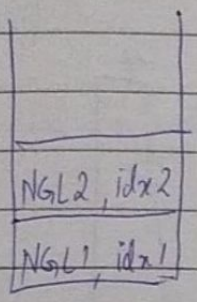
nearest greater to left ka index

75
Pair
~~(100, -1)~~
(100, -1)
(80, 0)
(60, 1)
(70, 3)
(60, 1)
(75, 1)
(85, 0)

(NGL, index)

PAGE NO. icon
DATE : / /

Stack < pair < Integer, Integer > > st = new Stack<>();



(NGL element, index)

Changes :-

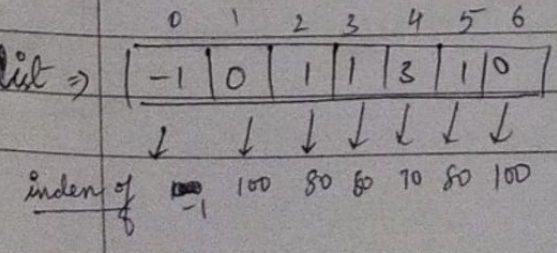
① List.add(st.top().second);

② st.push({arr[i], i});

③ st.top() → st.top().first

list |-1|0|1|1|3|1|0|

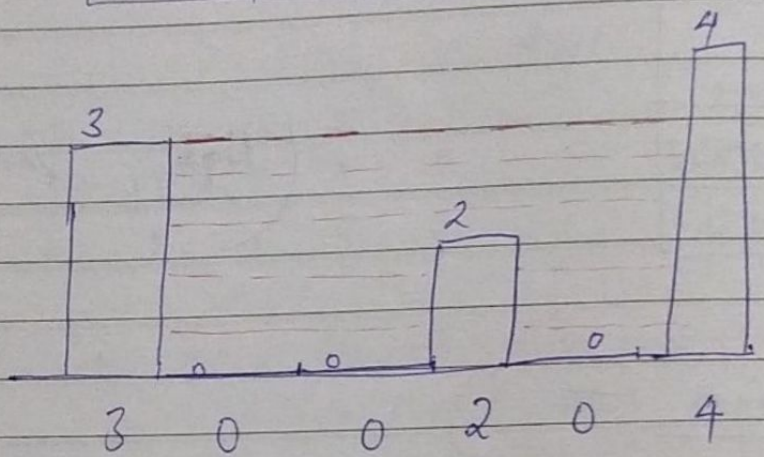
④ At end : for(int i=0; i<V.size(); i++){
V[i] = i - V[i];
}
return V;



0-(-1) = 1
1-0 = 1
2-1 = 1
3-1 = 2
4-3 = 1
5-1 = 4
6-0 = 6
} o/p

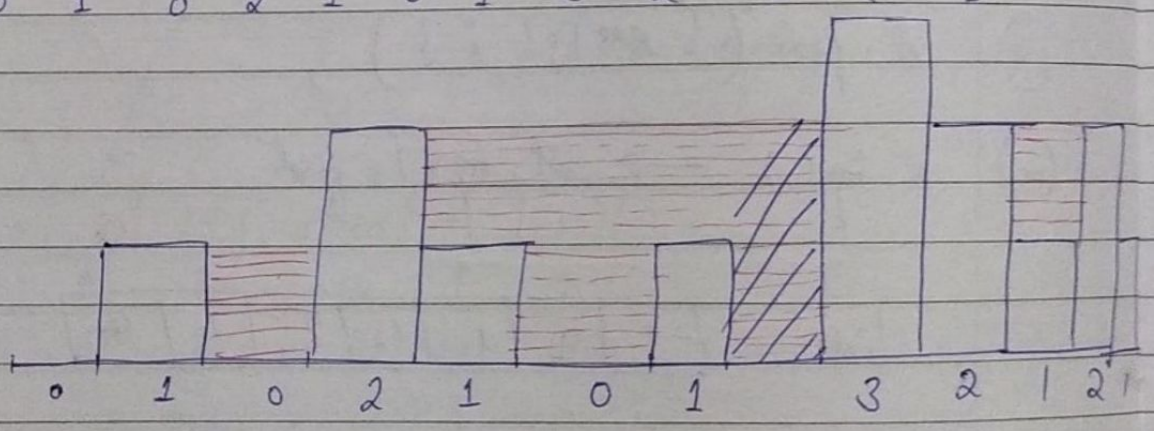
Rainwater Trapping Problem

Eg arr: | 3 | 0 | 0 | 2 | 0 | 4 |



o/p :
3+3+1+3
= 10

Eg. 0 1 0 2 1 0 1 3 2 1 2 1



o/p : 1 + 1 + 2 + 1 + 1 = 6

Idea : Har building ke liye

uski left array ka max

uski right array ka max

MIN

PAGE NO. icon
DATE : / /

@ any :

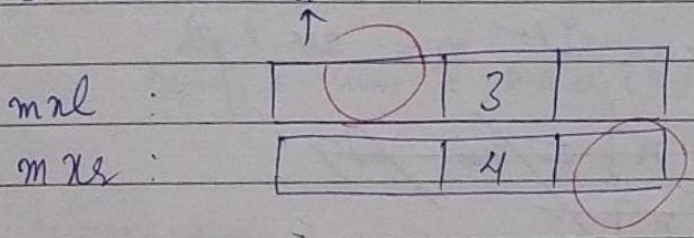
water[i] = Min (max L, max R) − arr[i]

water[i] = min(3, 4) − 2

arr[i] = 2

= 3 − 2 = 1 ✱✱

3 0 0 2 4



∑ water[i] = min (mxl [i], mxs [i]) − arr [i]

Q// How to find mxl & mxs ?

Leftse bada

(bada) mxl :

~~mxs :~~

0 1 0 2 1 0 1 3 2 1 2 1

0 1 1 2 2 2 2 3 3 3 3 3

mxs :

0 1 0 2 1 0 1 3 2 1 2 1

3 3 3 3 3 3 3 3 2 2 2 1

Right se bada

	3	0	0	2	0	4
mxl:	3	3	3	3	3	4
mxr:	4	4	4	4	4	4
min:						
→	3	3	3	3	3	4
−	3	0	0	2	0	4
Σ	0 + 3 + 3 + 1 + 3 + 0				=	10

o/p Ans

Code : int n = arr.length;

~~int mxl[n] = new int;~~
~~int mxr[n];~~

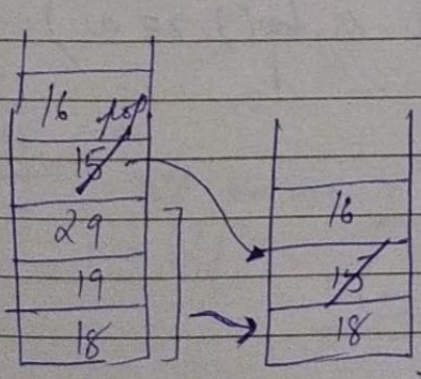
```
int[] mxl = new int[n];
int[] mxr = new int[n];
mxl[0] = arr[0];
for(int i=1; i<n; i++){
    mxl[i] = max(mxl[i-1], arr[i]);
}
mxr[n-1] = arr[n-1];
for(int i = n-2; i>=0; i--){
    mxr[i] = max(mxr[i+1], arr[i]);
}
```

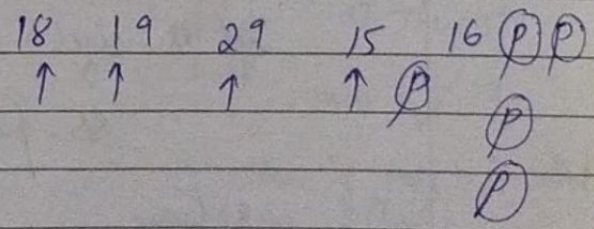
```
int water[] = new int[n];
for(int i=0; i<n; i++){
    water[i] = min(mxl[i], mxr[i]) - arr[i];
}

int sum = 0;

for(int i=0; i<n; i++){
    sum = sum + water[i];
}
return sum;
```

Min Stack

① push ② minElement()
② pop

stack → supporting stack

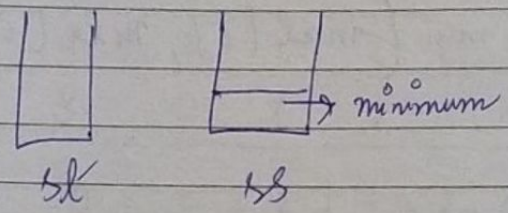




PAGE NO.
DATE : / /

Min Stack Implementation with Extra Space

16 19 29 15 16

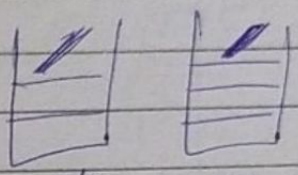


* Single element is always minimum (obviously)

```
Stack<Integer> st = new Stack<>();
Stack<Integer> ss = new Stack<>();

int getMin() {
    if( ss.size() == 0 )
        return -1

    return ss.peek();
}

void push( int a ) {
    st.push(a);
    if( ss.size() == 0 || ss.top() >= a )
        ss.push(a);
    return;
}
```

equality pe dhyaan!!

```
int pop() {
    if (st.size() == 0)
        return -1;
    int ans = st.top();
    st.pop();
    if (ss.peek() == ans)
        ss.pop();
    return ans;
}
```



Imp*** **Min Stack Implementation in O(1) Space**

O(1) ⇐

Variables

arr
Stack
List
} O(n)
↳ containers

If you use n variables

O(1)

Problem kya aaya?

7
5

If greater ele at top → NO issues

min = 5

3
5

→ smaller element at the top

min = ~~5~~ 3

Now if pop() is called on stack

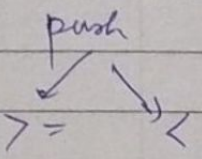
↓

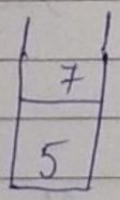
then min = 5

but how to retrieve that old value from min ⇒ NOT POSSIBLE

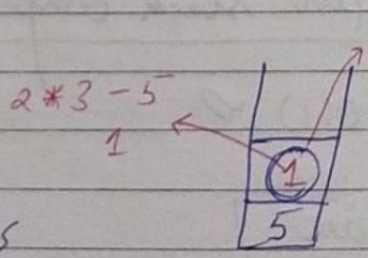
Iska think karne ke liye we store correct value ⇐ Wapas min to 3 se 5 kese IMPOSSIBLE

```
int minElem = Integer.MAX_VALUE;

int getMin() {
    if (s.size() == 0)
        return -1;

    return minElem;
}

void push (int x) {
    if (s.size() == 0) {
        s.push(x);
        minElem = x;
    }
    else {
        if (x >= minElem) {
            s.push(x);
        }
        else if (x < minElem) {
            st.push(2*x - minElem);
            minElem = x
        }
    }
}
```





minElem = 5

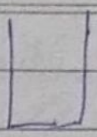


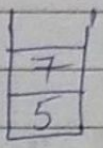
x = 3

minElem = 5

x 3

```
void pop () {
    if (st.size() == 0) {
        return -1;
    }
    else {
        if (s.top() >= minElem)
            s.pop();
        elseif (s.top() < minElem){
            minElem = 2 * minElem -
                s.top();
            s.pop()
        }
    }
}
```

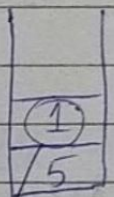
① 

② 

minElem = 5

7 > 5

If so, then it can be safely popped, bcoz minElem = 5 hi rahega

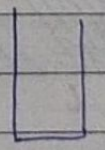
③ 

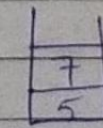
minElem = 3

flag → yeh bata rahaha ki agar 1 ko pop kiya toh minElem change hoga

2ME - y → s.top()

```
int top() {
    if (s.size()==0) {
        return -1;
    }
    else{
        %%
```

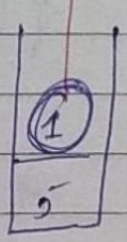
①  return -1

② 

minElem = 5

```
if( st.top() >= minElem){
        return s.top();

}
elseif(s.top() < minElem){
    return minElem;

}
}
```

ye corrupt value hai
isko asli value minE
me stored hai



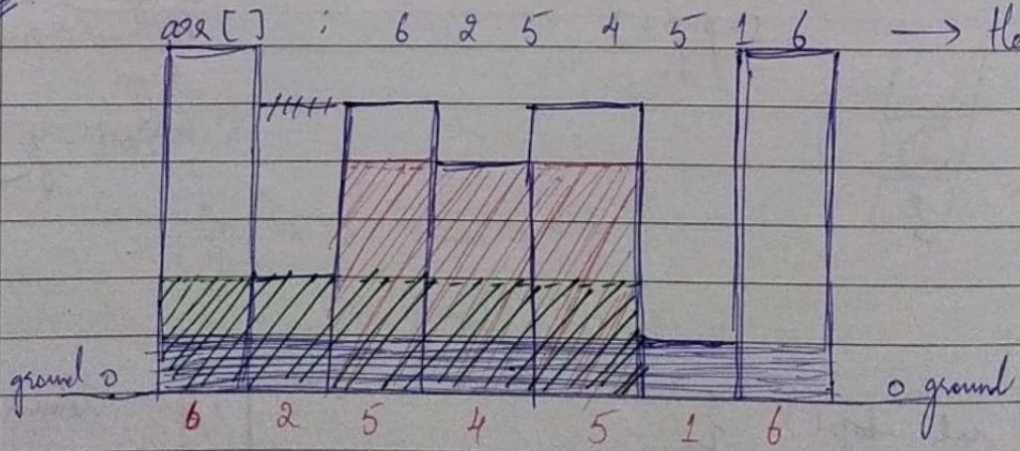
minElem = 3

agar st.top() minElem se bhi chhota hai, that means st.top() flag hai

kyunki MF stack ka hi MF hai

* Famous
* Popularly asked

MAXIMUM AREA HISTOGRAM (MAH)



Red
Area = 3 X 4 = 12

Blue
1 X 7 = 7

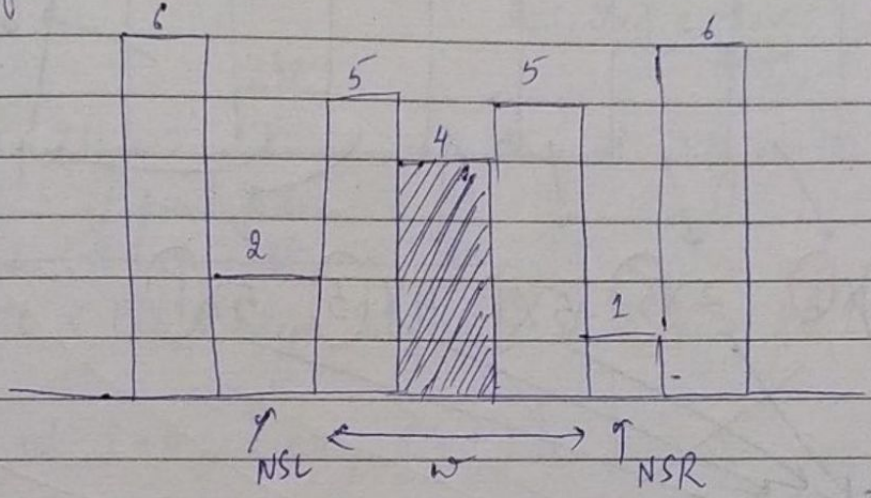
Green
2 X 5 = 10

→ MAXIMUM

Index 4

NSR 5
NSL 1
4 - 1 = 3 width

	0	1	2	3	4	5	6
arr :	6	2	5	4	5	1	6
Index NSR → right :	1	5	3	5	5	~~7~~	~~7~~
Index NSL → left :	-1	-1	1	1	3	-1	5



Area tab tak expand ho sakta hai jab tak building ke left & right me koi chhota ~~element~~ building na mil jaaye

width = 5 - 1 = 4

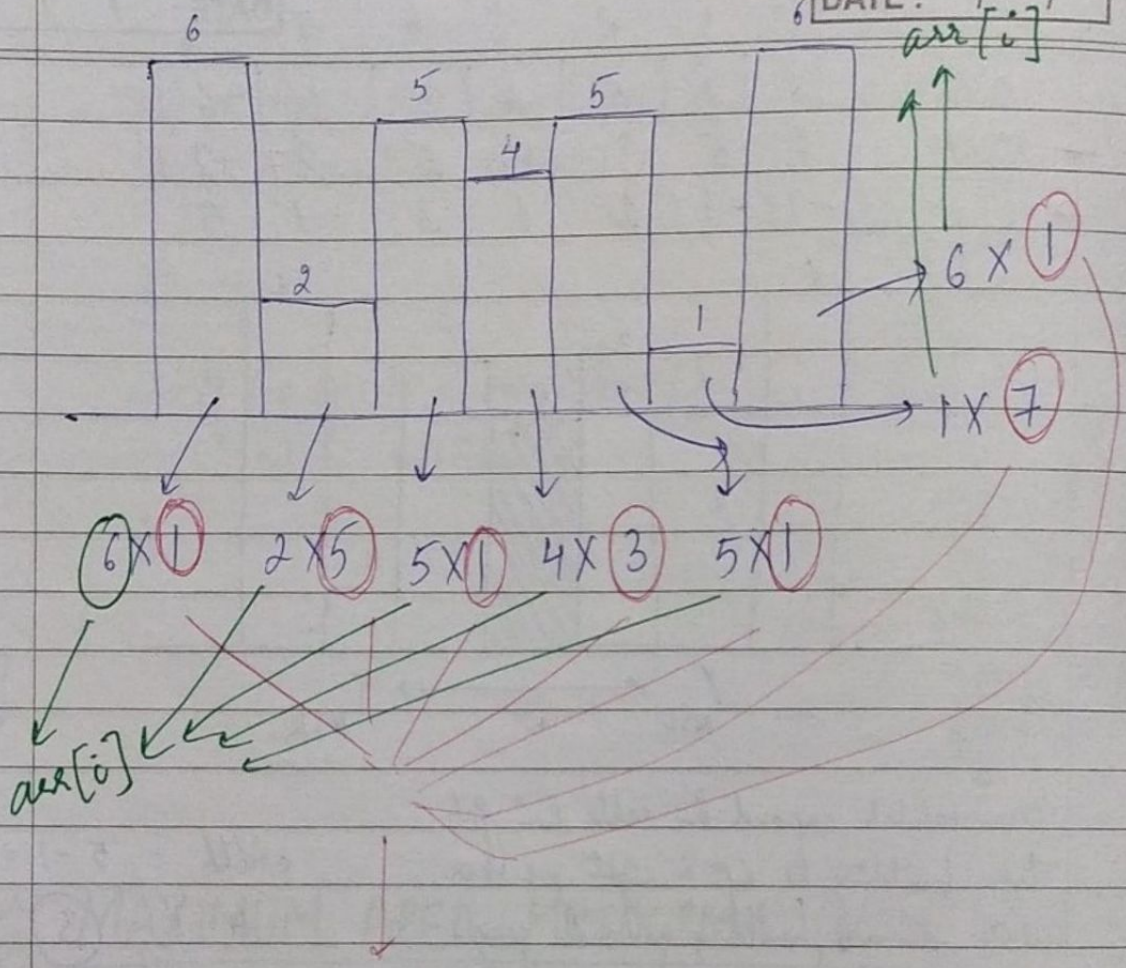
4 - 1 = (3)

right - left - 1 ⇒ width

width[i] = right[i] - left[i] - 1

↓ iss formula se

	0	1	2	3	4	5	6
width :	1	5	1	3	1	7	1



WIDTH ARRAY ELEMENTS

arr[] :	6	2	5	4	5	1	6
width[] :	1	5	1	3	1	7	1

Area [i] = arr [i] * width [i] ;

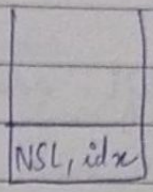
Area [] :	6	10	5	12	5	7	6

Find max^m => ANSWER

PAGE NO. icon
DATE : / /

Code :-

left → NSL Index



pair

left

```
List<Integer> list = new ArrayList<Integer>();
int pseudoIndex = -1
Stack<Integer> st = new
Stack<Pair<Integer, Integer>> st = new Stack<>();

for(int i=0; i<n; i++){
  if(st.size()==0){
    left list.add(-1); → pseudoIndex
  }
  else if(st.size()>0 && st.peek().first < arr[i]){
      left.add(st.peek().second);
  }
  else if(st.size()>0 && st.peek().first >= arr[i]){
    while(st.size()>0 && st.peek().first >= arr[i]){
        st.pop();
    }
    if(st.size()==0){
        left.add(-1);
    }
```

PAGE NO. icon
DATE : / /

```
else {
    left.add(st.pop().second);
}
}
st.push(arr[i],i);
}
return left;
```

3rd by → right → NSR index

```
int pseudoIndex = 7;(arr.length+1)

for(int i=0; i<size; i++){
    width[i] = right[i] - left[i] - 1;
}
for(int i=0; i<size; i++){
    area[i] = arr[i] * width[i];
}
```

Find max in area[i]