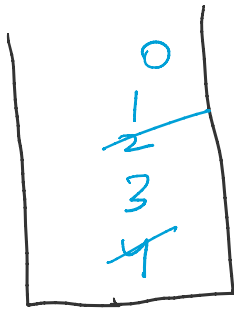


ci = 1

68, 735, 101, 770, 525



ci = 3 2 1 0

temp = 2

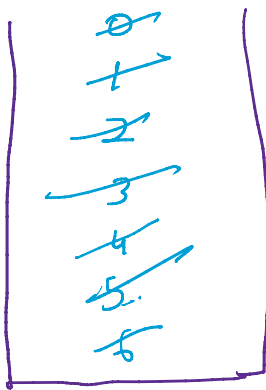
2 - 1 = 1

[1, 2, 3, 4, 5]

[100, 70, 70, 70, 60, 120, 30]
m[1, 1, 2, 3, 1, 6, 1]

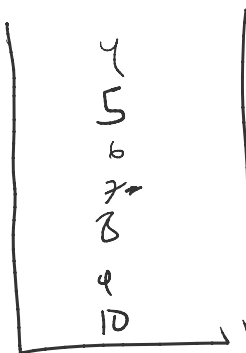
T.C = O(n)

S.C = O(1)

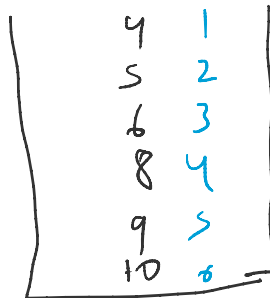


ci = 8 4 3 2 1
t = 6 4

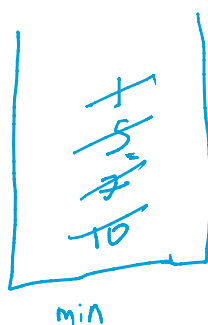
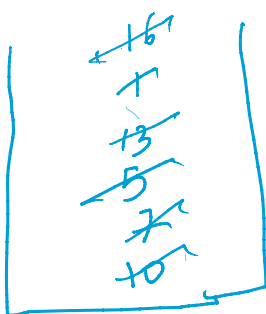
```
public static int[] stockSpan(int[] arr) {
    Stack<Integer> stack = new Stack<>();
    stack.push(arr.length - 1);
    int ci = arr.length - 2;
    int[] res = new int[arr.length];
    while (ci >= 0) {
        while (!stack.isEmpty() && arr[stack.peek()] < arr[ci]) {
            int temp = stack.pop();
            res[temp] = temp - ci;
        }
        stack.push(ci);
        ci--;
    }
    while (!stack.isEmpty()) {
        int curr = stack.pop();
        res[curr] = curr + 1;
    }
    return res;
}
```



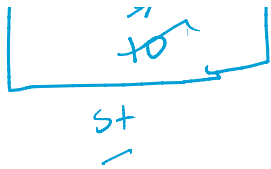
=>



[10, 7, 5, 13, 1, 16]



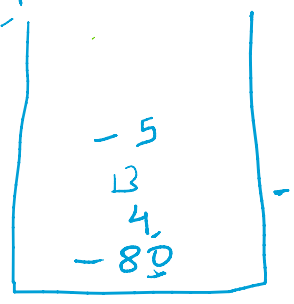
[10, 7, 5, 13, 1, 16]



[10, 7, 5, 13, 1, 16]

min = ~~100~~ ∞ // Integer.MAX_VALUE

$2 \times 1 - 7 = -5$



val = 10 & 13
 if (val < min) {
 push (2 * val - min)
 min = val
 } else push (val)

$push = 2 \times val - min$
 $min = 2 \times val - push$
 $-5 = 2 \times 1 - 7$
 $2 = 2 \times 1 + 5$

if (peek() < min) {
 newMin = 2 * min - pop();
 return min;
}
 return pop();