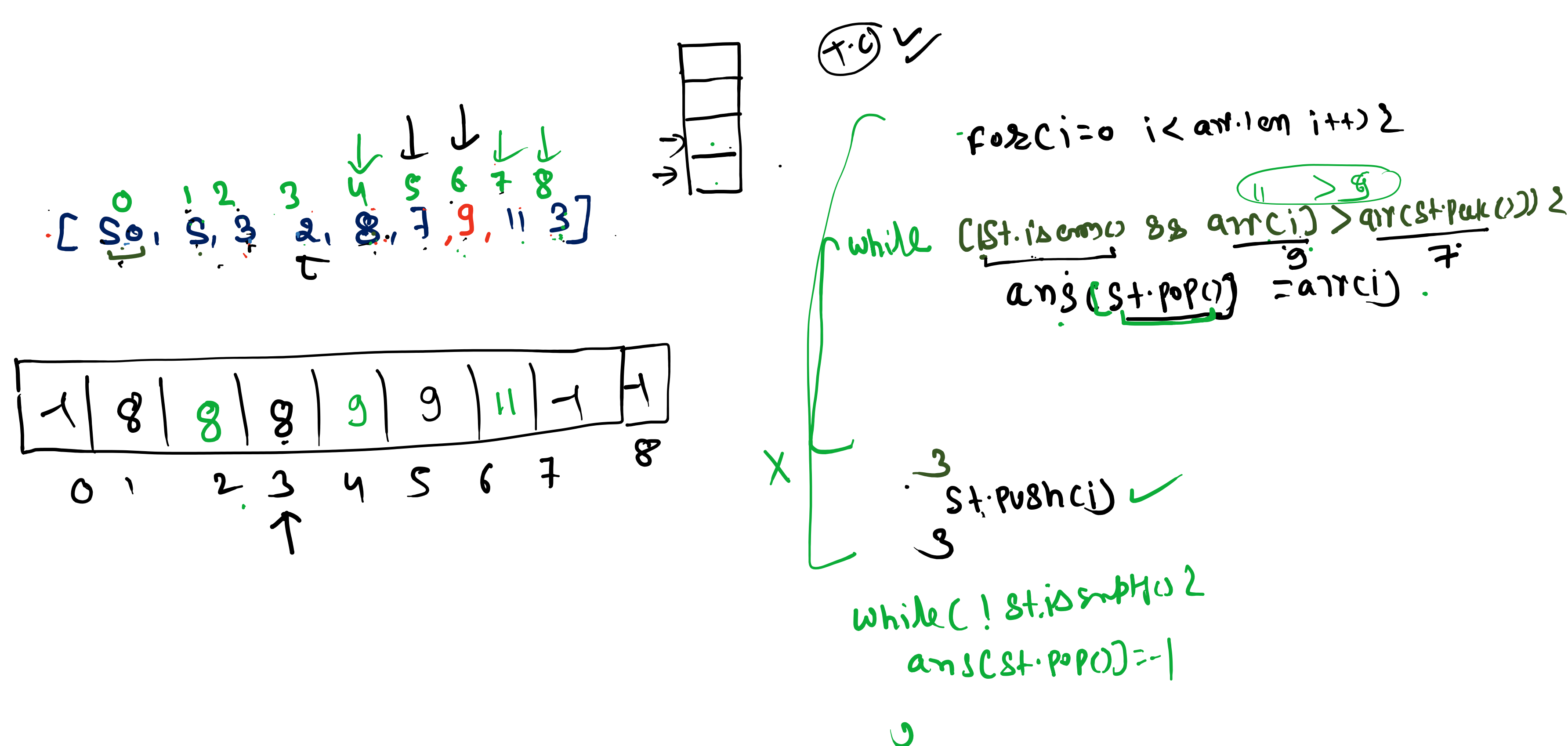
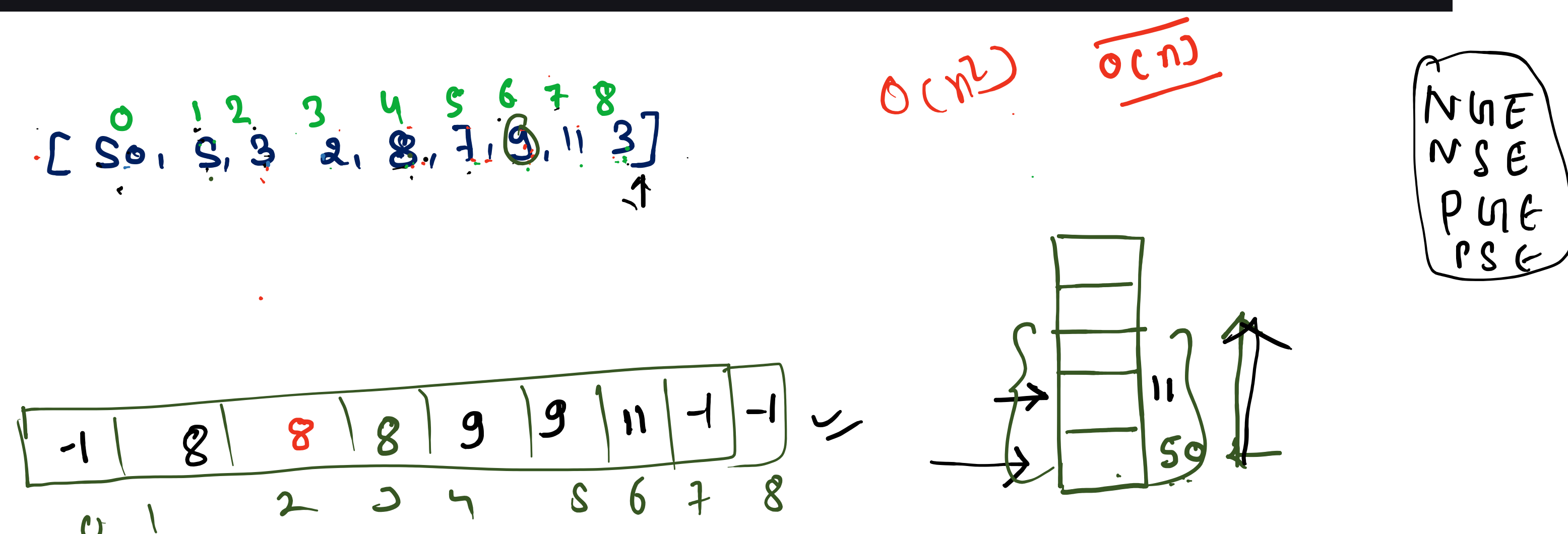


Given an array, print the Next Greater Element (NGE) for every element. The Next Greater Element for an element x is the first greater element on the right side of x in array. Elements for which no greater element exist, consider next greater element as -1.

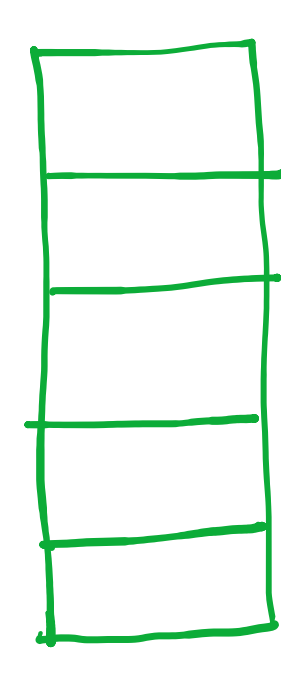
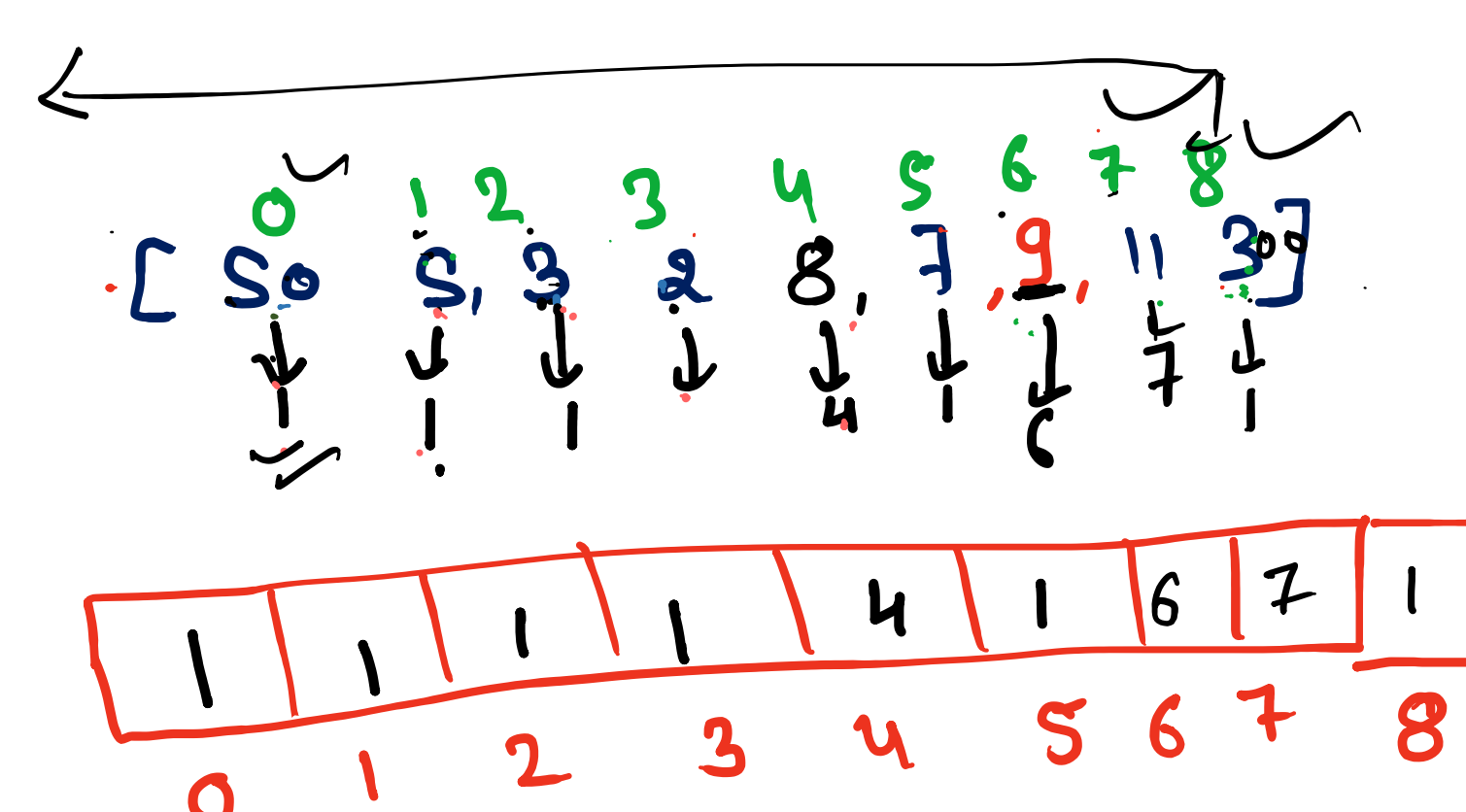
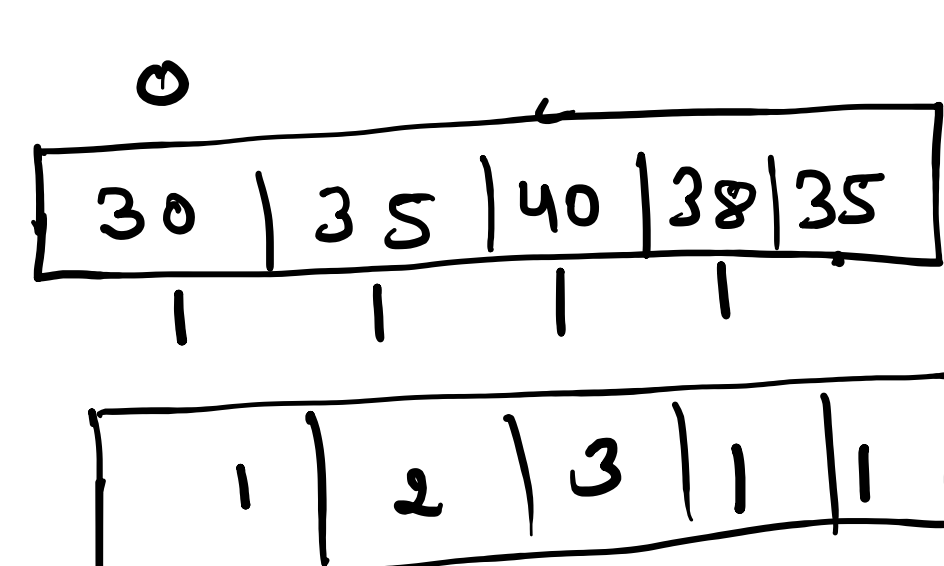
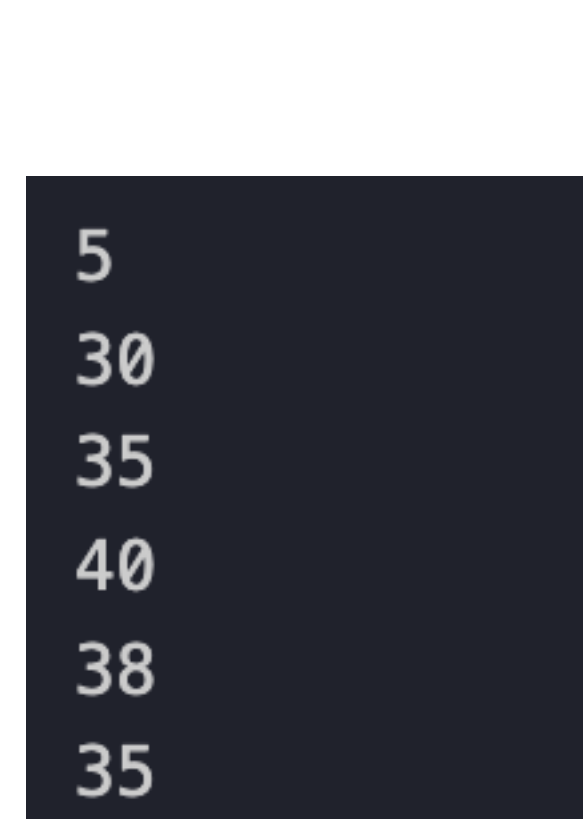


```
public static void NGE(int[] arr) {
    Stack<Integer> st = new Stack<>();
    int[] ans = new int[arr.length];
    for (int i = 0; i < arr.length; i++) {
        while (!st.isEmpty() && arr[i] > arr[st.peek()]) {
            ans[st.pop()] = arr[i];
        }
        st.push(i);
    }
    while (!st.isEmpty()) {
        ans[st.pop()] = -1;
    }
    for (int i = 0; i < ans.length; i++) {
        System.out.println(arr[i] + " " + ans[i]);
    }
}
```

$n+1$ time, n time
 $1 \rightarrow 2$, $2n$ time

The stock span problem is a financial problem where we have a series of N daily price quotes for a stock and we need to calculate span of stock's price for all N days. You are given an array of length N, where i^{th} element of array denotes the price of a stock on i^{th} . Find the span of stock's price on i^{th} day, for every $1 \leq i \leq N$.

A span of a stock's price on a given day, i , is the maximum number of consecutive days before the $(i+1)^{\text{th}}$ day, for which stock's price on these days is less than or equal to that on the i^{th} day.

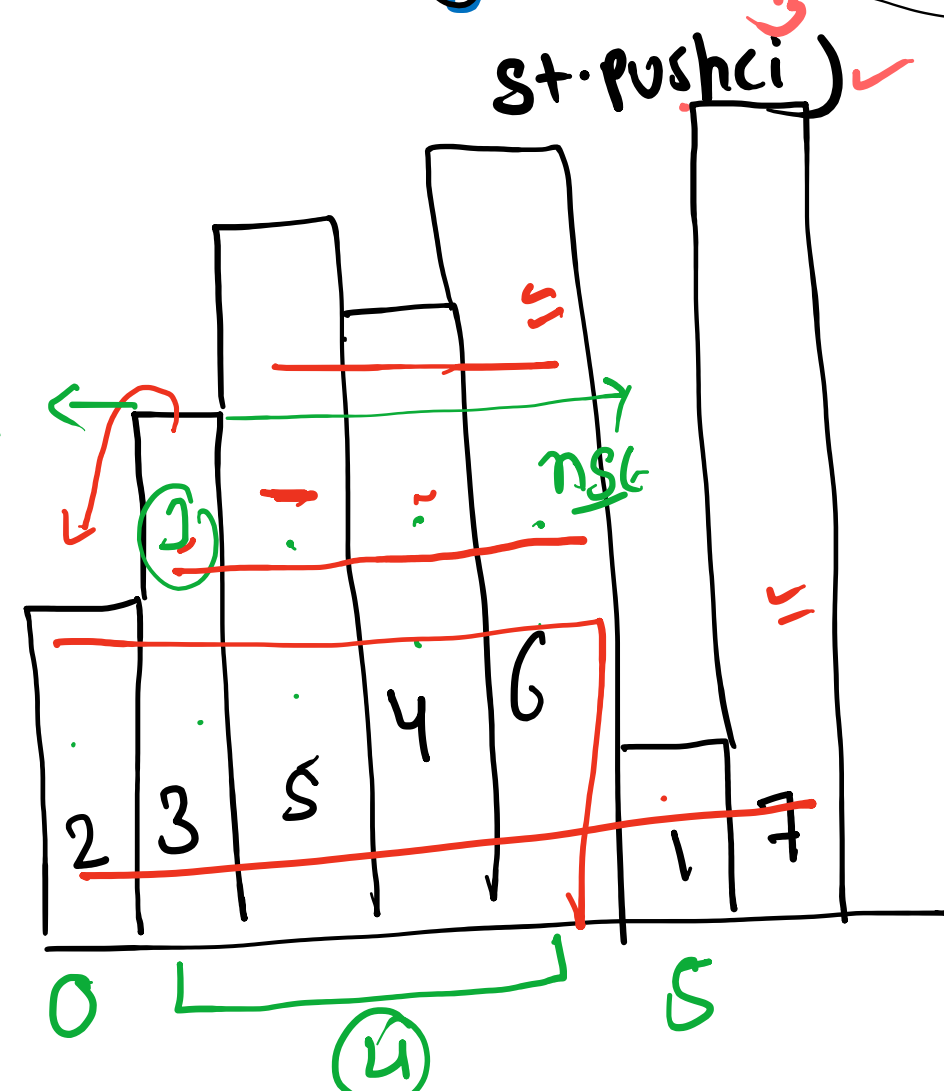


$for(i=0; i < arr.len; i++)$
 while (list.isEmpty() || $arr[i] > arr[st + end(i)]$)
 $st = pop()$

```

3 // swap
if (i < j) {
    ans[i] = i + 1;
    ans[j] = j + 1;
}
}

```

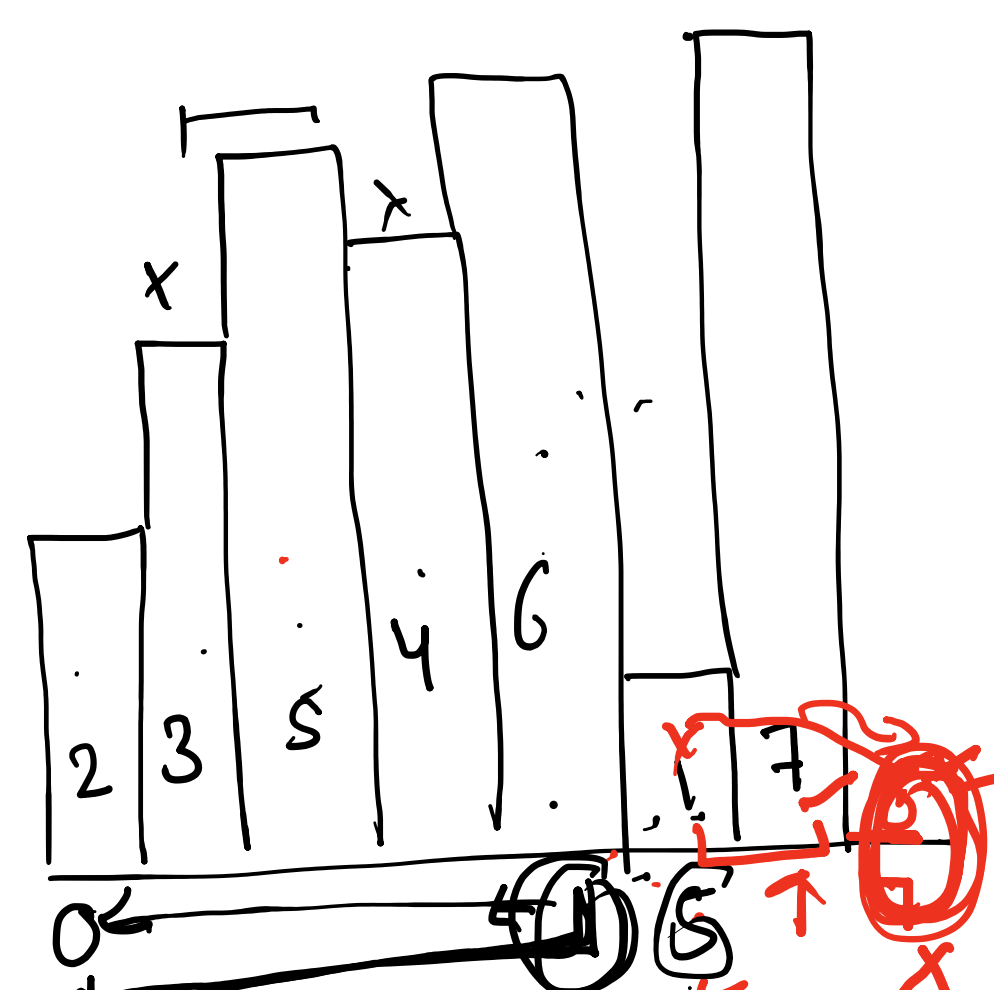
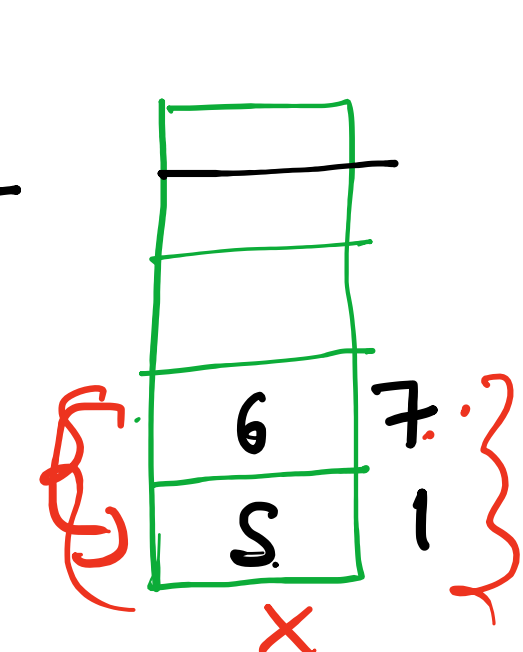
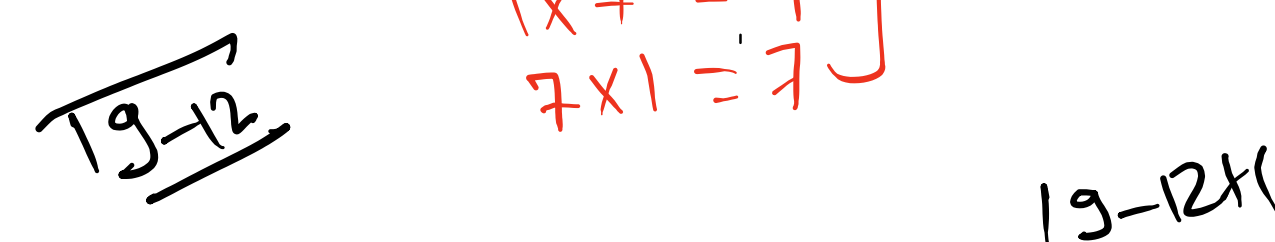
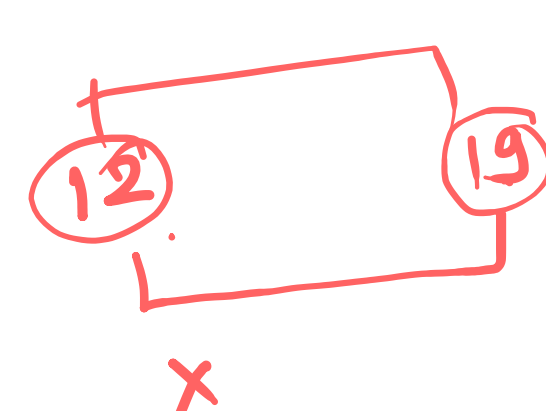


RAIN PST NST

$\begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ f & 2 & 3 & 5 & 4 & 6 & 1 \end{matrix}$

$2 \times 5 = 10$ ✓
 $3 \times 4 = 12$
 $5 \times 1 = 5$
 $4 \times 3 = 12$
 $6 \times 1 = 6$
 $1 \times 7 = 7$
 $7 \times 1 = 7$

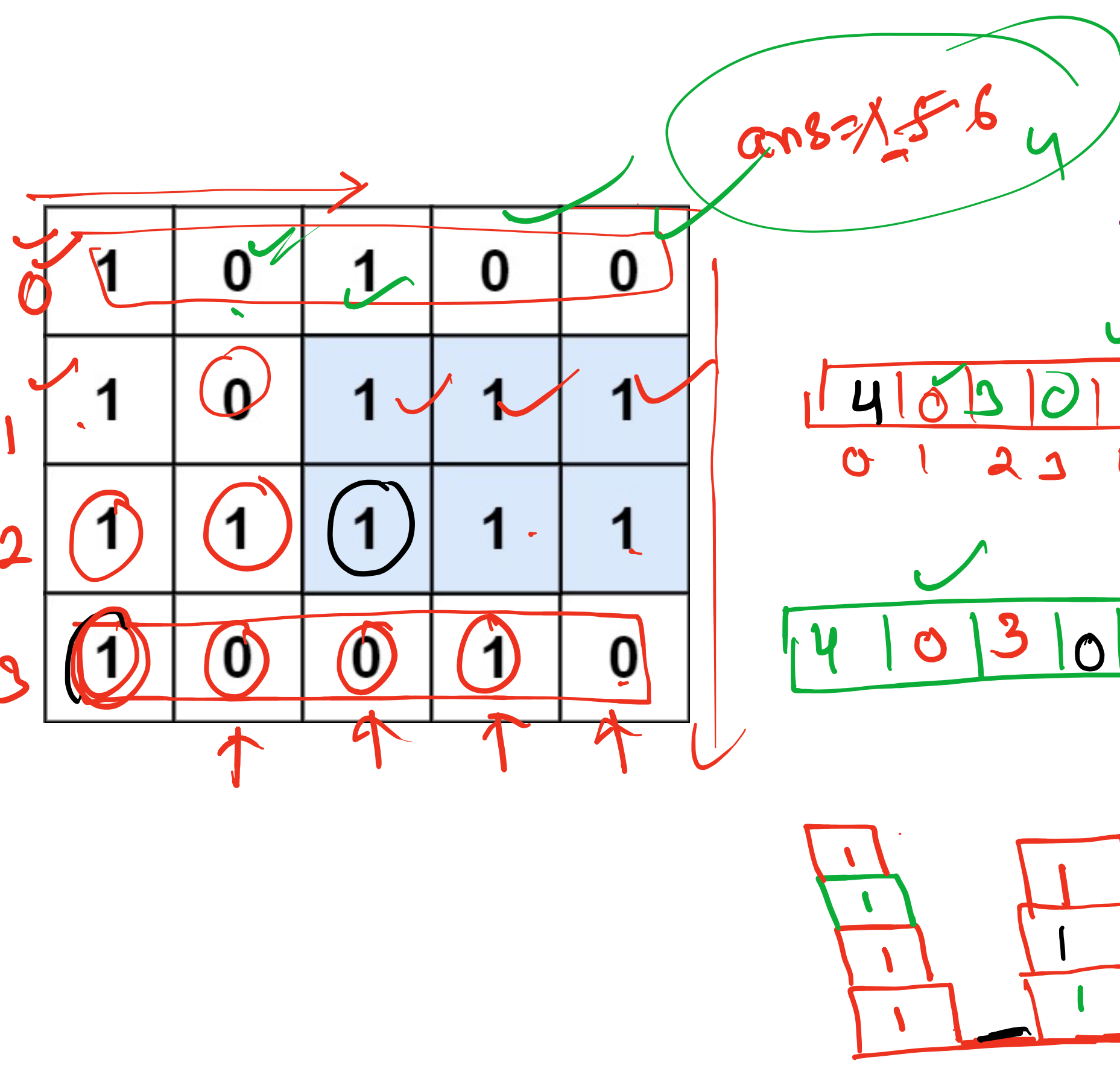
(12) Ans



```

for (i = 0; i < arr1.length; i++) {
    while (!st.isEmpty() && arr1[i] < arr1[st.peek()]) {
        h = arr1[st.pop()];
        R = i;
        if (st.isEmpty()) {
            L = start + peek();
        } else {
            L = Math.max(L, h * (R - L + 1));
        }
        h = Math.max(h, h * R);
    }
    st.push(i);
}

```



```
for (int i = matrix.length - 1; i >= 0; i--) {
    for (int j = 0; j < matrix[0].length; j++) {
        if (matrix[i][j] == '1') {
            arr[j]++;
        } else {
            arr[j] = 0;
        }
    }
}
```

