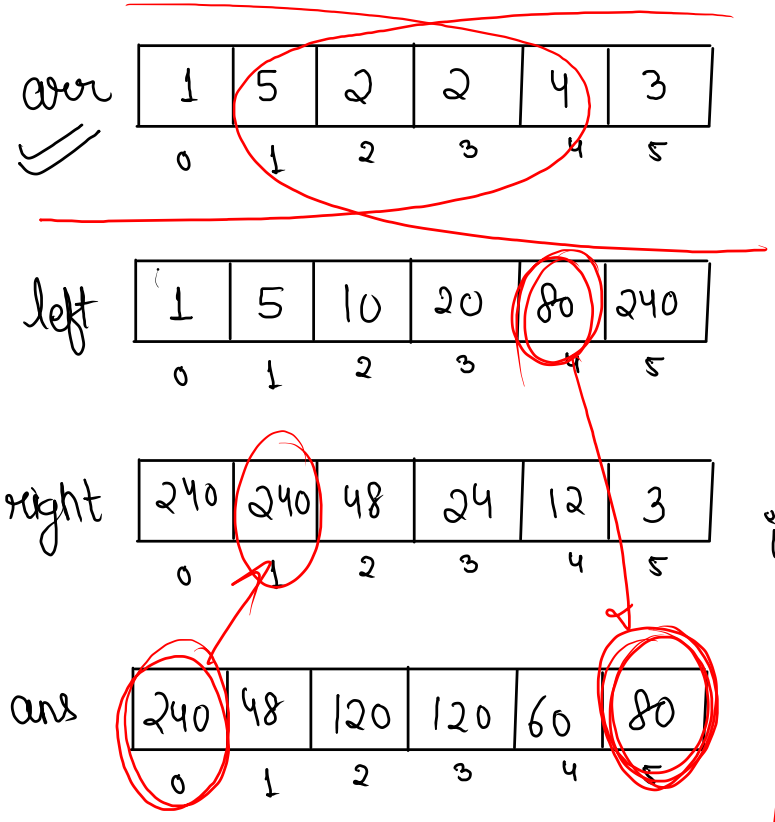
Ques Product of Elements Except Itself



prod = 1 x 5 x 0 x 2 x 4 x 3

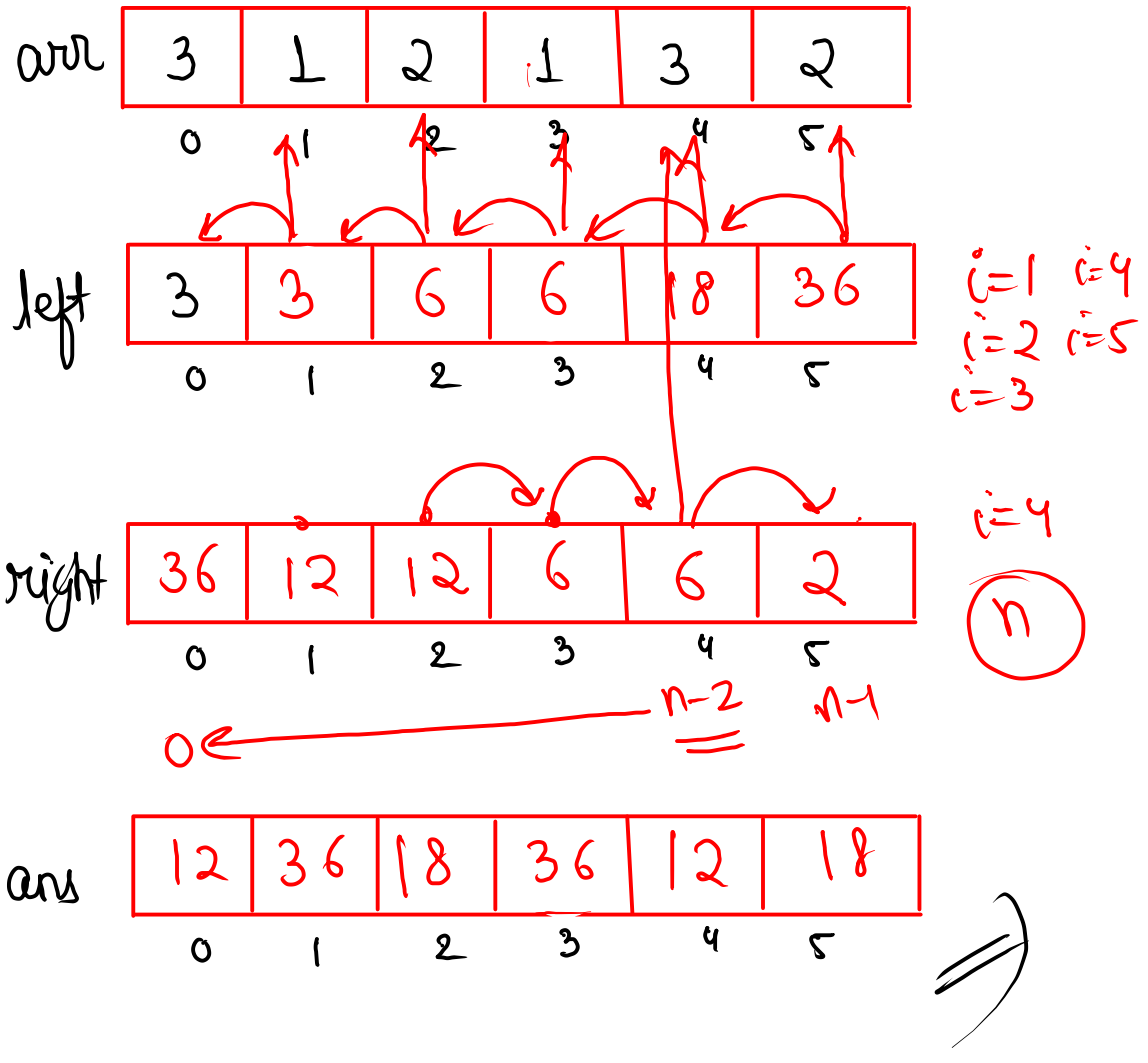
indicate :- product of itself and all left elements

indicate :- product of itself and all elements

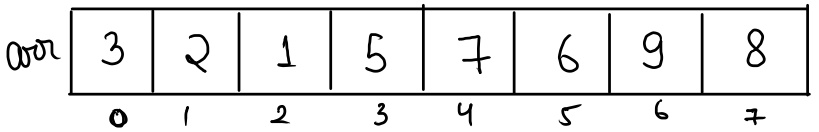
Steps

1) Create left array which indicates prod. of all left elements including itself
2) Create right array which indicates prod. of all right elements including itself
3) answer for each index i, is prod. of element at i-1 in left array and i+1 element in right array.

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    int[] ans = productExceptItself(arr, n);
    for (int i = 0; i < n; i++) {
        System.out.println(ans[i]);
    }
}

public static int[] productExceptItself(int[] arr, int n) {
    // step 1
    int[] left = new int[n];
    left[0] = arr[0];
    for (int i = 1; i < n; i++) {
        left[i] = left[i - 1] * arr[i];
    }

    // step 2
    int[] right = new int[n];
    right[n - 1] = arr[n - 1];
    for (int i = n - 2; i >= 0; i--) {
        right[i] = right[i + 1] * arr[i];
    }

    // step 3
    int[] ans = new int[n];
    ans[0] = right[1];
    ans[n - 1] = left[n - 2];
    for (int i = 1; i < n - 1; i++) {
        ans[i] = left[i - 1] * right[i + 1];
    }

    return ans;
}
```



HW_Second Largest in array 2



logic

↳ keep maintaing largest and second largest at all time

int max = $-\infty$

int second_max = $-\infty$

i=0, (3) second_max = ~~$-\infty$~~ $-\infty$
max = ~~$-\infty$~~ 3

---

i=1, (2) max = 3
second_max = ~~$-\infty$~~ 2

---

i=2, (1) max = 3
sec_max = 2

---

i=3, (5) sec_max = ~~2~~ 3
max = ~~3~~ 5

---

i=4, (7) sec_max = 5
max = 7

---

i=5 (6), max = 7
s_max = ~~8~~ 6

curr = 6
max = 7
s_max = 5

---

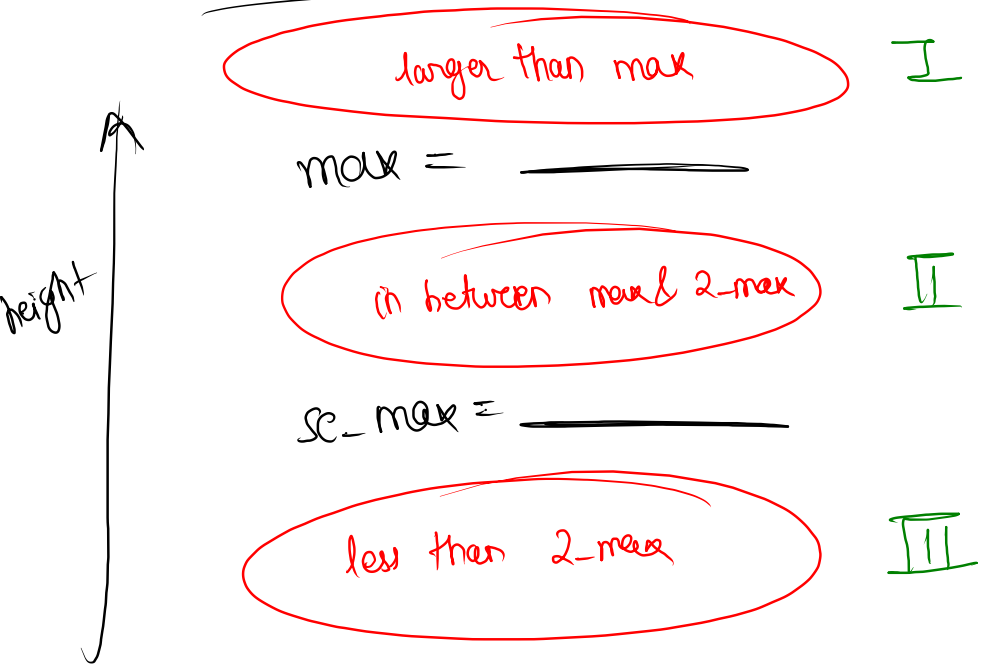
i=6 (9), s_max = ~~6~~ 7
max = 9

---

i=7 (8), max = 9
s_max = ~~7~~ 8

Sysо(s_max)

3 possibilities in this question



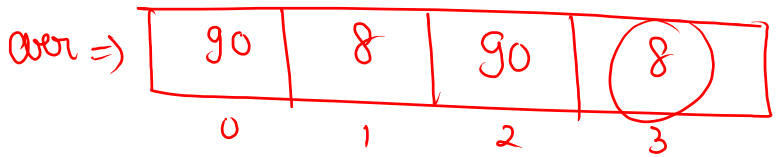

```java
public static void secondLargest(int[] arr, int n) {
    int max = Integer.MIN_VALUE;
    int s_max = Integer.MIN_VALUE;

    for (int i = 0; i < n; i++) {
        int curr = arr[i];
        if (curr > max) {
            s_max = max;
            max = curr;
        } else if ( curr != max && curr > s_max ) {
            s_max = curr;
        }
    }
    System.out.println(s_max);
}
```

i=0, s_max = ~~-∞~~ -∞, max = ~~-∞~~ 90

i=1, max = 90, s_max = 8

i=2, max = 90, s_max = 8

i=3, max = 90, s_max = 8



max = -∞

s_max = -∞

fact :-
only array & string can have ∞ no. of variations
all other can't