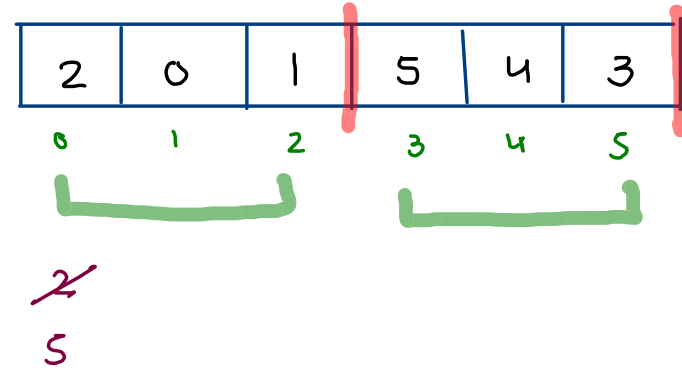


## 769. Max Chunks To Make Sorted

You are given an integer array `arr` of length `n` that represents a permutation of the integers in the range `[0, n - 1]`.

```
for(int i=0; i < arr.length;i++) {  
    maxR = Math.max(maxR,arr[i]);  
  
    if(maxR == i) {  
        chunks++;  
    }  
}
```

ele  $\rightarrow$  0 to 5



chunks = ~~0~~  
~~2~~  
2

## 768. Max Chunks To Make Sorted II

30	10	20	40	55	45	80
0	1	2	3	4	5	6

$lmax$

30	30	30	40	55	55	80
0	1	2	3	4	5	6

$rmin$

10	10	20	40	45	45	80	$\infty$
0	1	2	3	4	5	6	7

$T : O(n)$

$S : O(n)$

$lmax[i] < rmin[i+1] \{$

$chunks++;$

$\}$

left  $\swarrow$   $\searrow$  right  
0 1 0  
i

```

lmax[0] = arr[0];
for(int i=1; i < arr.length;i++) {
    lmax[i] = Math.max(lmax[i-1],arr[i]);
}

rmin[arr.length] = Integer.MAX_VALUE;
for(int i=arr.length-1; i >= 0;i--) {
    rmin[i] = Math.min(rmin[i+1],arr[i]);
}

int chunks = 0;
for(int i=0; i < arr.length;i++) {
    if(lmax[i] <= rmin[i+1]) {
        chunks++;
    }
}

```

45	20	10	30	50	48	46	68	65
0	1	2	3	4	5	6	7	8

*lmax*

45	45	45	45	50	50	50	68	68
0	1	2	3	4	5	6	7	8

*rmin*

10	10	10	30	46	46	46	65	65	∞
0	1	2	3	4	5	6	7	8	9

## 915. Partition Array into Disjoint Intervals

Given an integer array `nums`, partition it into two (contiguous) subarrays `left` and `right` so that:

- Every element in `left` is less than or equal to every element in `right`.
- `left` and `right` are non-empty.
- `left` has the smallest possible size.

$if (dmax[i] \leq rmin[i+1]) \{$

$\text{return } i; \text{ // len} \rightarrow i+1$

$\}$

`[5,0,3,8,6]`

$array;$

5	0	3	8	6
0	1	2	3	4

$dmax$

5	5	5	8	8
0	1	2	3	4

$rmin$

0	0	3	6	6	$\infty$
0	1	2	3	4	5

7	3	9	5	10	1	15	16	19	14	30
0	1	2	3	4	5	6	7	8	9	10

$lmax = 7$

$rmax = 7$

$pa = -1$

if (arr[i] > lmax) {

    lmax = arr[i];  
}

else if (arr[i] < rmax) {

    pa = i;  
    rmax = lmax;  
}

①. travel left to right,  
lmax.

②. potential answer  $\rightarrow pa$