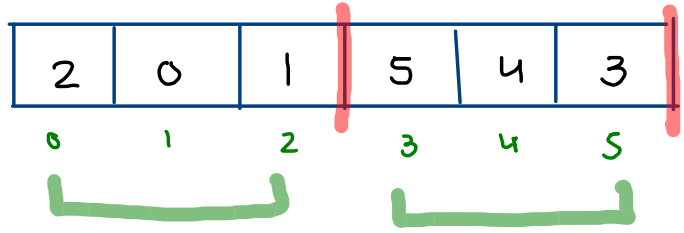
769. Max Chunks To Make Sorted

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 -> 0 to 5



maxR

~~2~~

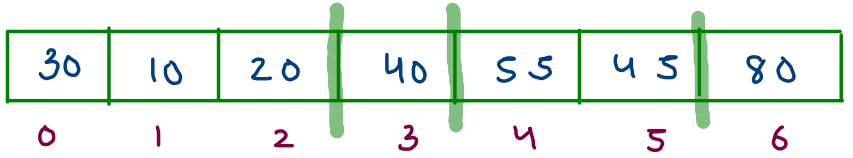
5

chunks = ~~0~~

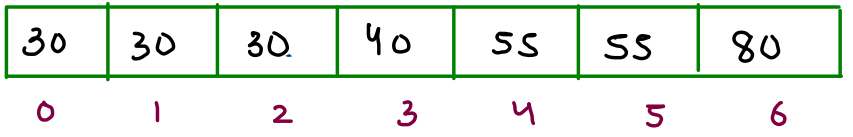
~~1~~

2

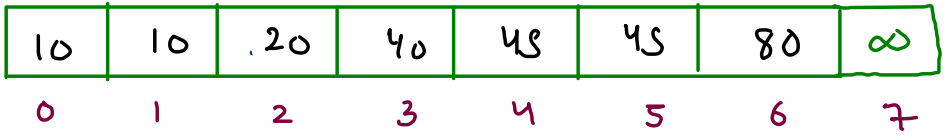
768. Max Chunks To Make Sorted II



lmax



rmin

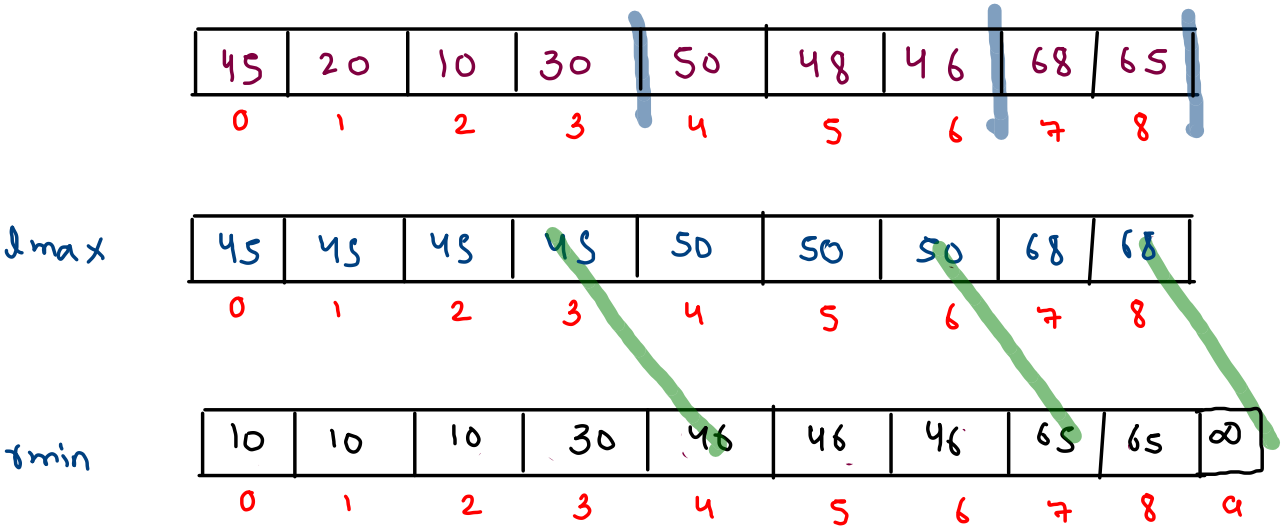


T : O(n)

S : O(n)

```
lmax[i] < rmax[i+1] {
    chunks++;
}
```

```
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++;
    }
}
```



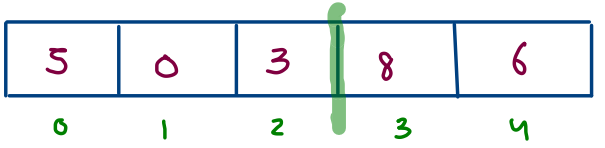
915. Partition Array into Disjoint Intervals

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.

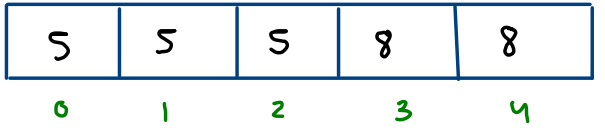
: [5,0,3,8,6]

arr :

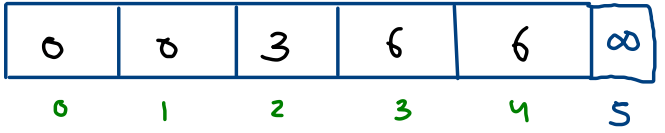


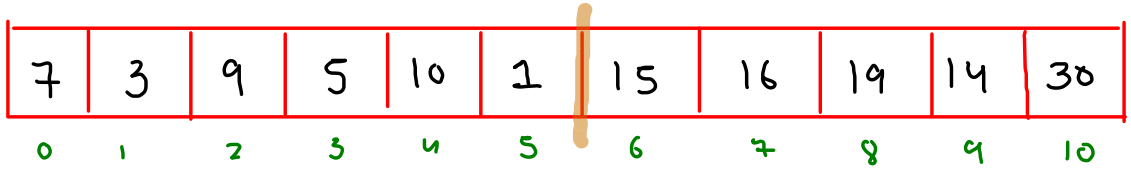

```
if( dmax[i] ≤ rmin[i+1] ){
    return i; //len -> i+1
}
```

dmax



rmin





dmax = 7

pmax = 7

pa = -1

```
if( arr[i]  > dmax ) {
        dmax = arr[i];
}
else if (arr[i] < pmax ) {
        pa = i;
        pmax = dmax;
}
```

① . travel left to right, dmax.

② . potential answer -> pa