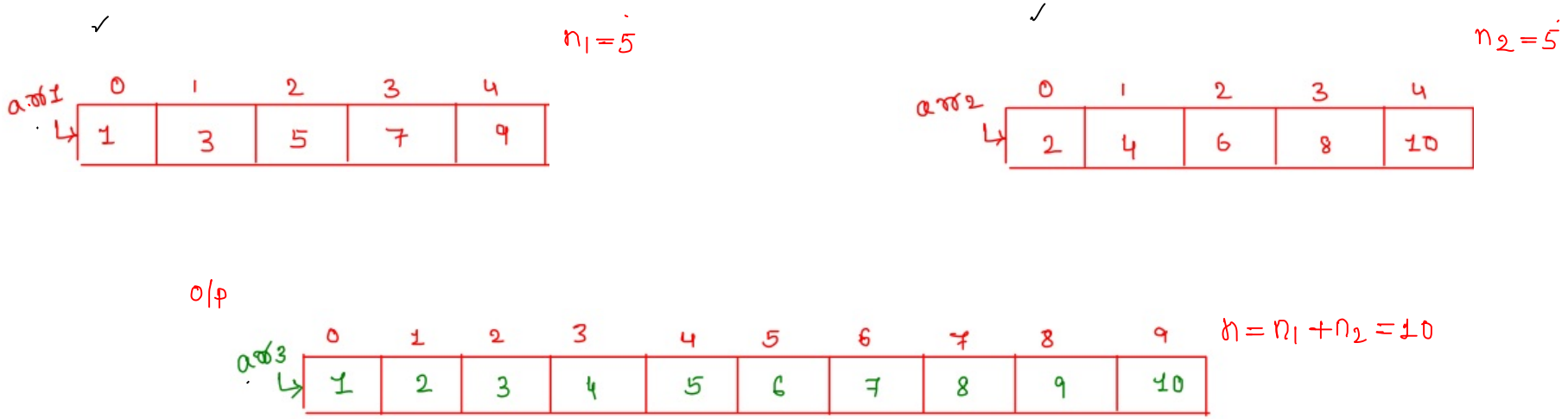
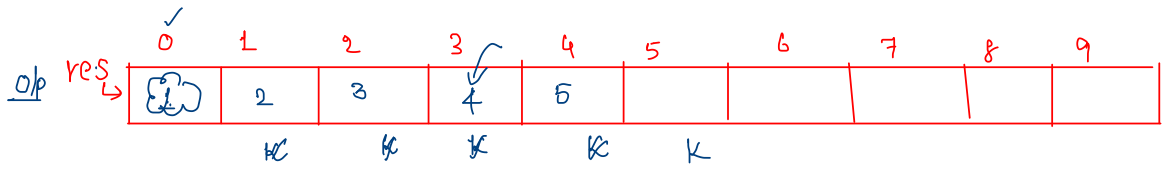
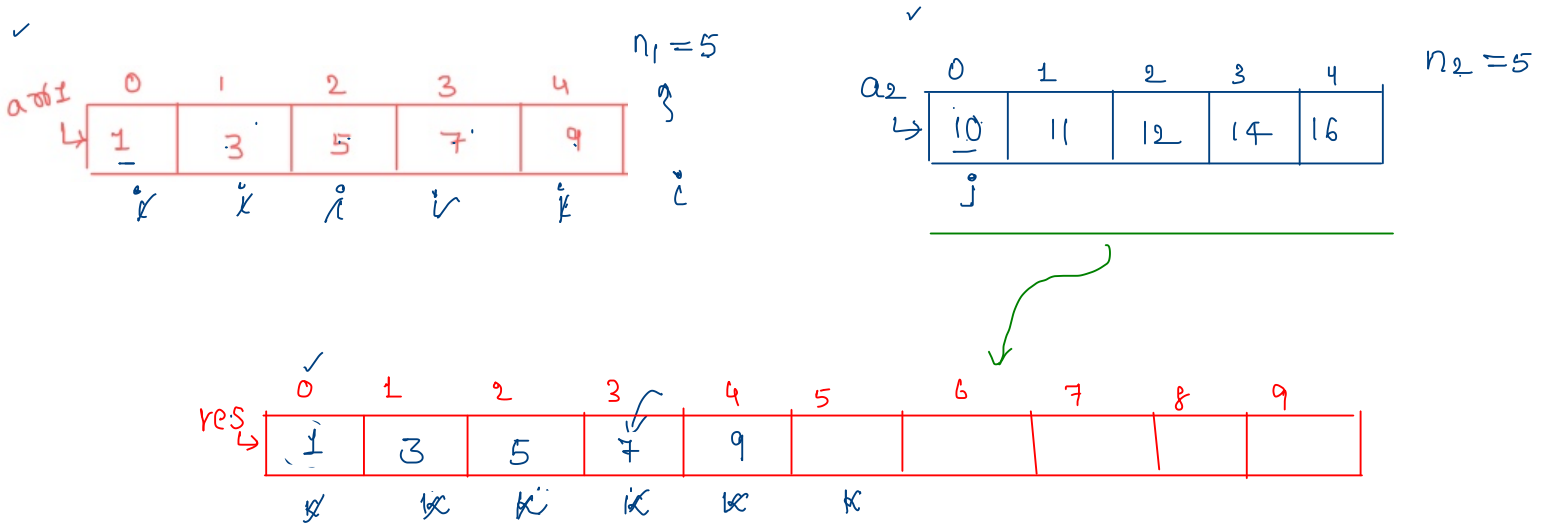
Two Pointer [Model-2 : Same Direction]

1) Merge Two Sorted Arrays






```
function fun(arr1,arr2,n1,n2)
{
        i=0,j=0,k=0
        res=[]*(n1+n2)
        while(i<n1 and j<n2)
        {
                if(arr1[i]<arr2[j])
                {
                        res[k]=arr1[i];
                        i++;
                        k++
                }
                else
                {
                        res[k]=arr2[j];
                        j++;
                        k++;
                }
        }


}
```




```
while(j<n2)
{
        res[k]=arr2[j]
        j++
        k++
}

while(i<n1)
{
        res[k]=arr1[i]
        i++
        k++
}
```

→ O(n) T.C

→ O(n) S.C

```
function fun(arr1,arr2,n1,n2)
{
        i=0,j=0,k=0
        res=[]*(n1+n2)        → ? O(n)
        while(i<n1 and j<n2)
        {
                if(arr1[i]<arr2[j])
                {
                        res[k]=arr1[i];
                        i++;
                        k++
                }
                else
                {
                        res[k]=arr2[j];
                        j++;
                        k++;
                }
        }
```

```
while(j<n2)
{
        res[k]=arr2[j]
        j++
        k++
}
```

Rev

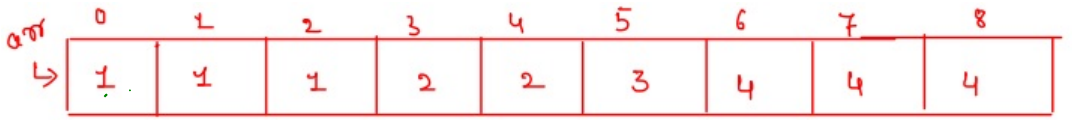
```
while(i<n1)
{
        res[k]=arr1[i]
        i++
        k++
}
```

j ←

i/p always ✓

6) Remove Duplicates from Sorted array :-

Ex:-

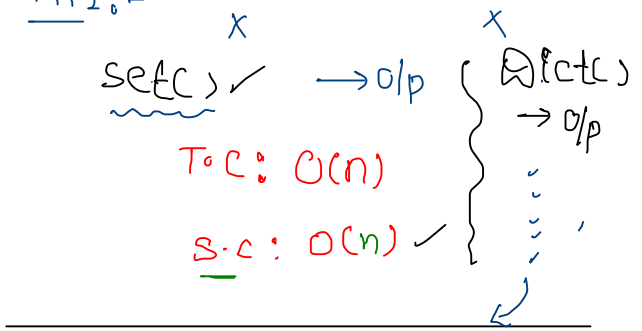


n = 9

o/p

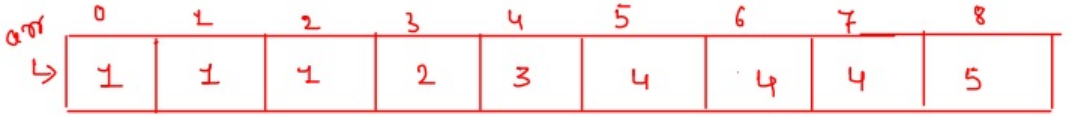
→ 1, 2, 3, 4

AP1 :-

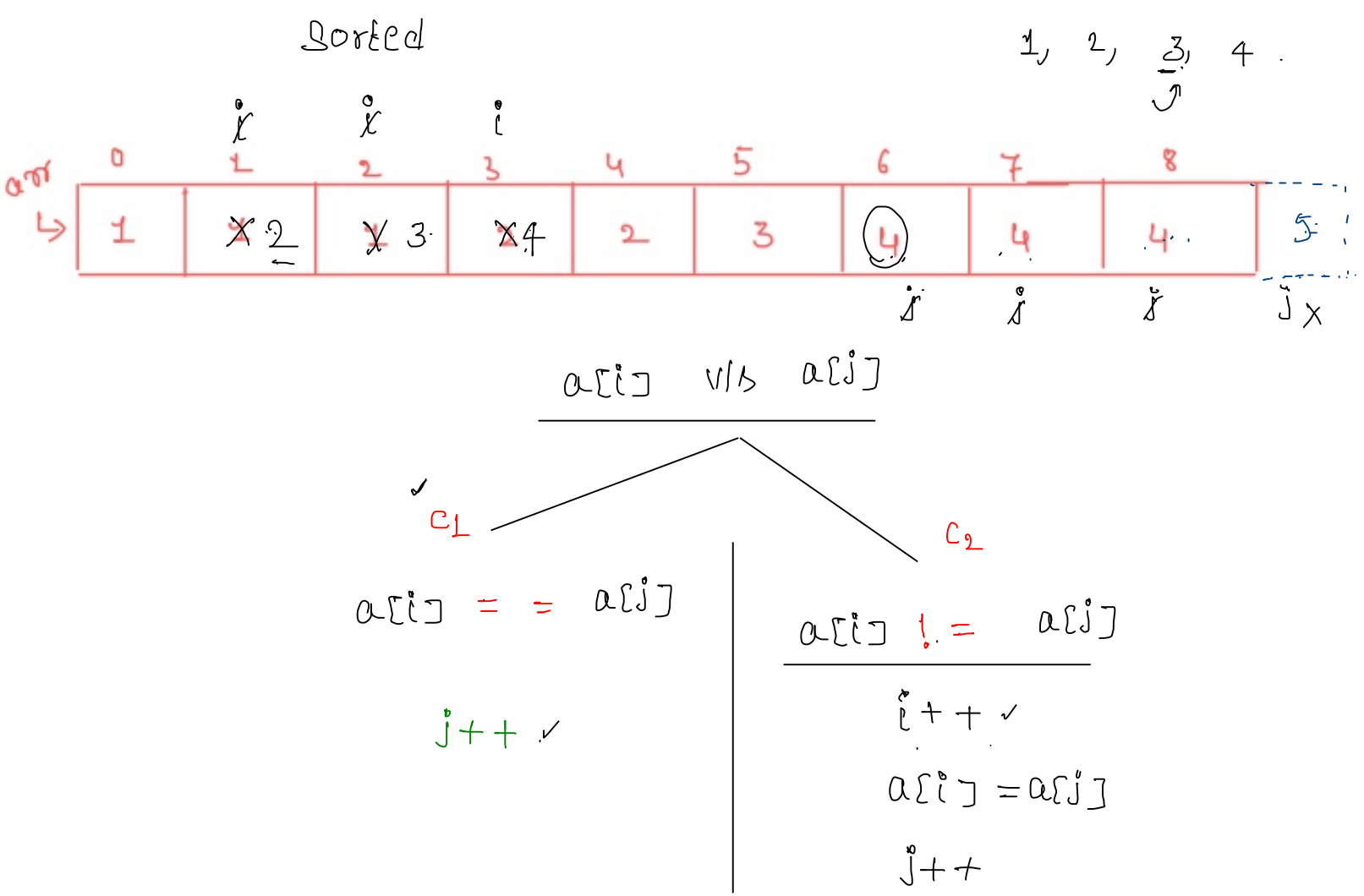


AP2 :-

Ex:



→ 1, 2, 3, 4, 5



a[i] v/s a[j]

c1

a[i] == a[j]

j++

c2

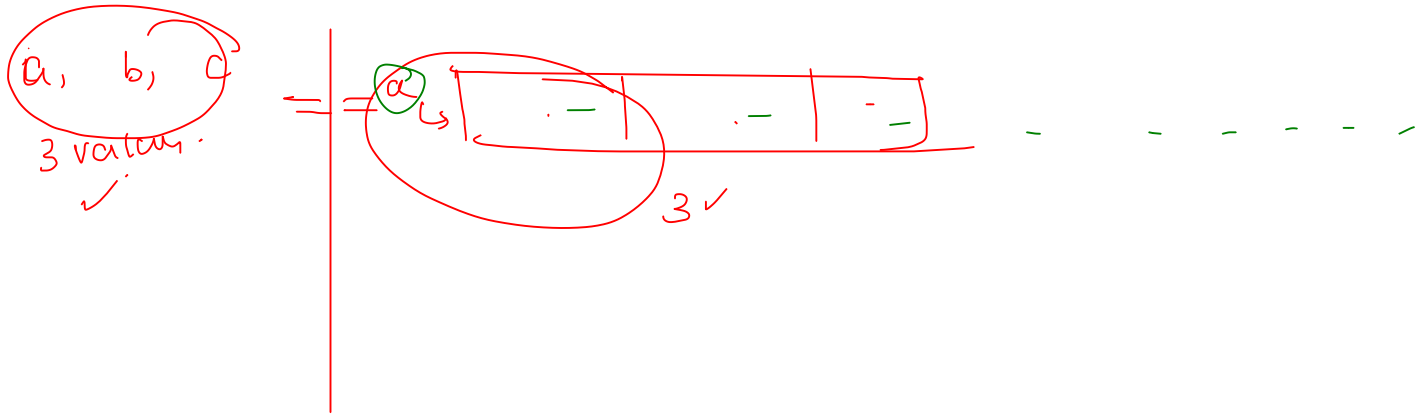
a[i] != a[j]

i++

a[i] = a[j]

j++

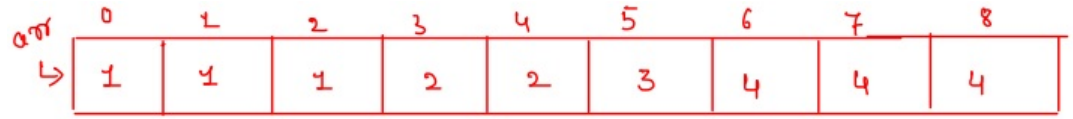
Uniqueue elements if I want to print
start=0
end=i

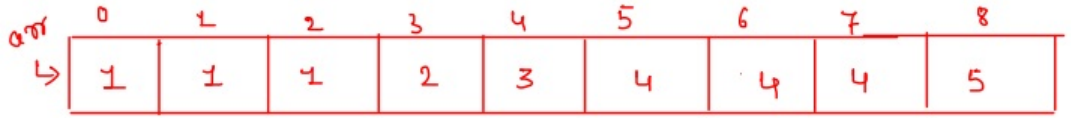

a, b, c
3 values.
a
is
3

HW dry-run

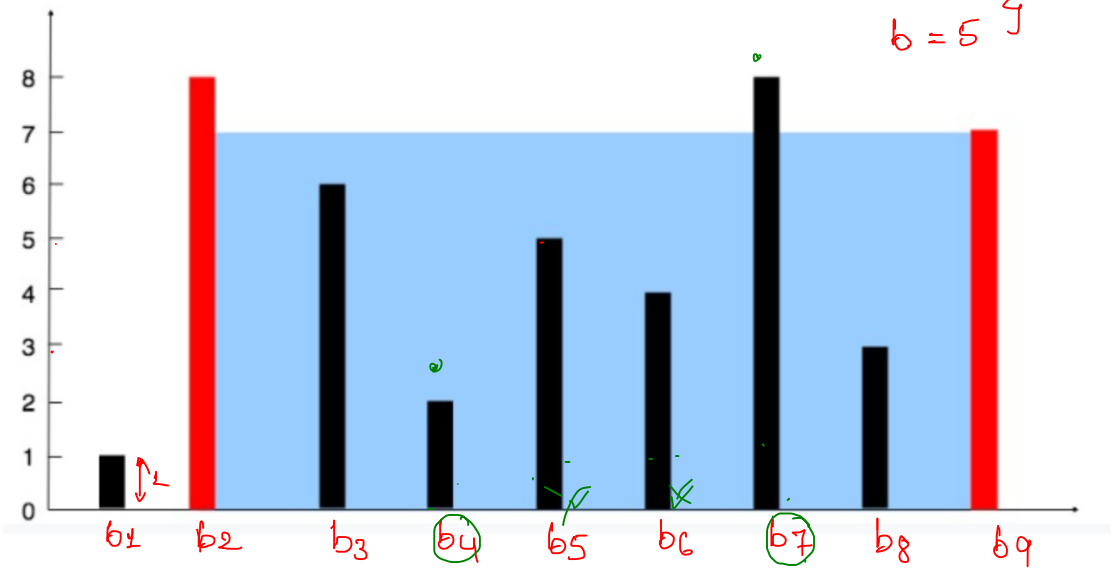
```
function removeDupSortedArray(arr, n)
{
        j=0
        for(i=0;i<=n-2;i++)
        {
            if(arr[i]!=arr[i+1])
            {
                arr[j]=arr[i]
                j++
            }
        }
        arr[j]=arr[n-1]

        for(i=0;i<=j;i++)
        {
            print(arr[i])
        }
}
```

ex:-





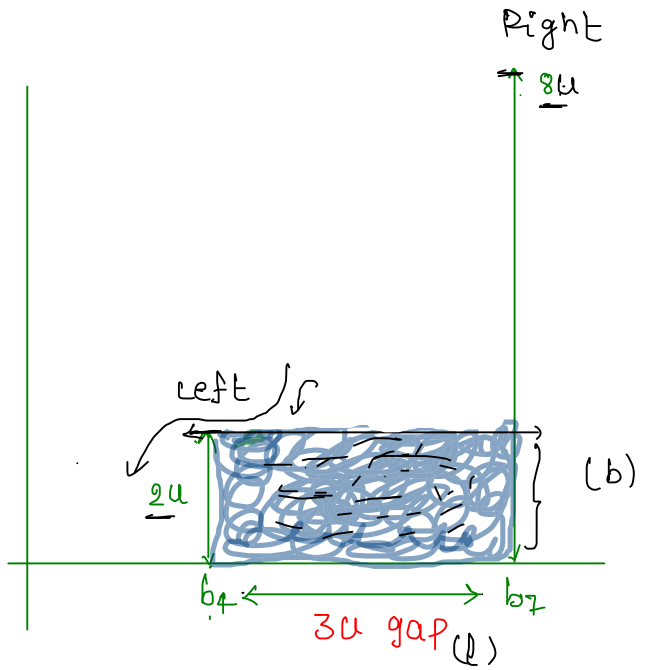
Container With Most water



First testcase

The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

1.water can be stored between any of the two blocks

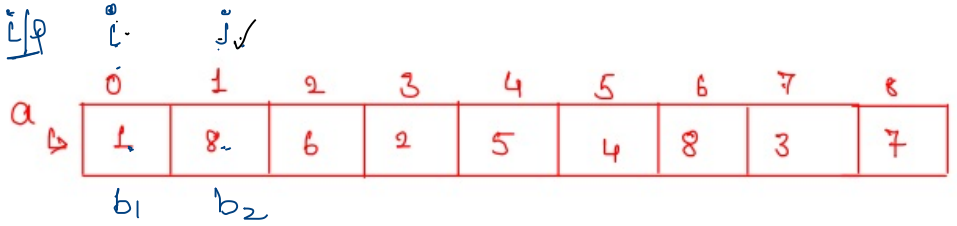


Area = l x b

= 3 x min[2u, 8u]

2u

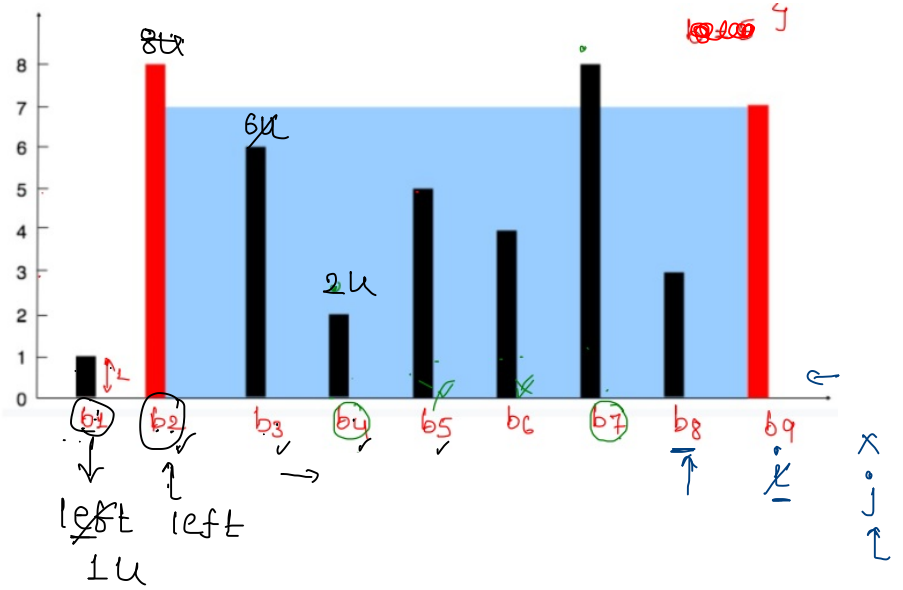
= 6



dep → un-rolling

```
function maxWater(arr,n)   → O(n^2)
{
        maxArea=0
        for(i=0;i<n-1;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        area=(j-i)*min(arr[i],arr[j])
                        maxArea=max(maxArea,area)
                }
        }
        return maxArea
}
```

2ptr



2 X L = 2u

7x7 = 490

n = 9

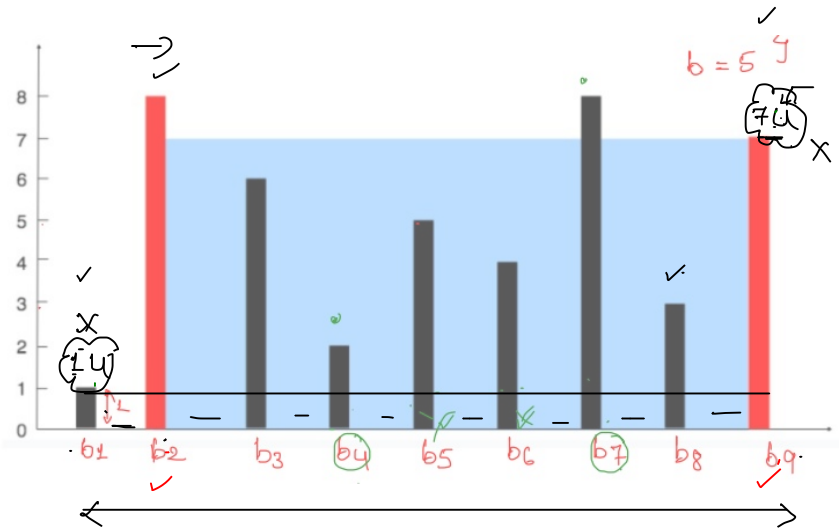
→ max Area

	0	1	2	3	4	5	6	7	8
a	1	8	6	2	5	4	8	3	

l r r

7 x 1 = 7u

area = 8u
49



8 x 1 = 8u

left < right
l++

left > right
r--

left right
1u 7u

2pointer →

2ptr

```
def solve(arr):
    n = len(arr)
    l = 0
    r = n - 1
    ans = 0

    while l < r:
        area = min(arr[l], arr[r]) * (r - l)
        ans = max(ans, area)
        if arr[l] > arr[r]:
            r -= 1
        else:
            l += 1

    return ans
```

leave right

T.C

O(n) ← O(n²)

S.C

O(1) → inplace