Day 9

17 November 2022 20:32

$A_1 =$ | 1 | 2 | 3 | next permutation

| 1 | 2 | 3 | np

1 3 2 ✓

2 1 3 ☆

2 3 1 np

3 1 2 ☆

3 2 1 ☆

| 1 | 2 | 3 | 4 |

A_1 | 4 | 3 | 2 | 1 |

next_per

| 1 | 2 | 3 | 4 |

| 1 | 2 | 3 | → [1, 3, 2]

| 3 | 2 | 1 |

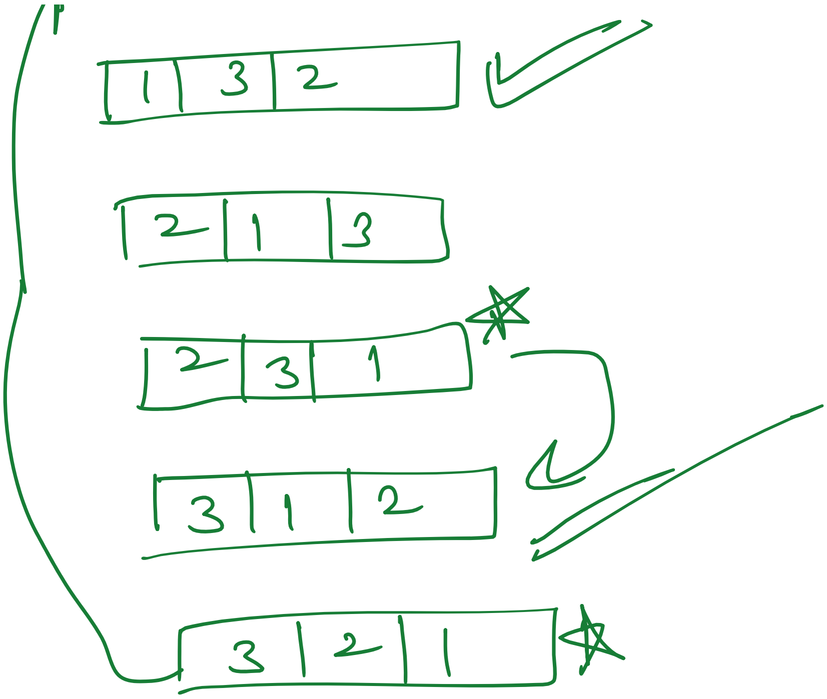
| 1 | 1 | 5 | next greater arrangement

| 1 | 5 | 1 | np

A - | 1 | 2 | 3 | next permutation

| 1 | 2 | 3 | ☆

| 1 | 3 | 2 | ✓



1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

NEXT PERMUTATION

A_1 = | 1 | 2 | 3 | → If this is given array

The next permutation will be 1 3 2

1 2 3

1 3 2

2 1 3

2 3 1

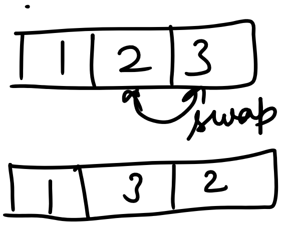
3 1 2

3 2 1

The next permutation if given array was | 2 | 3 | 1 |

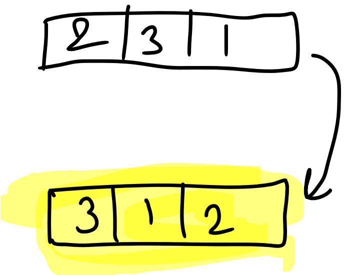
So How we can find the next permutation.

Case 1



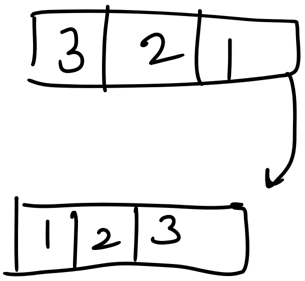
This will be next permutation.

Case 2



This will be next permutation

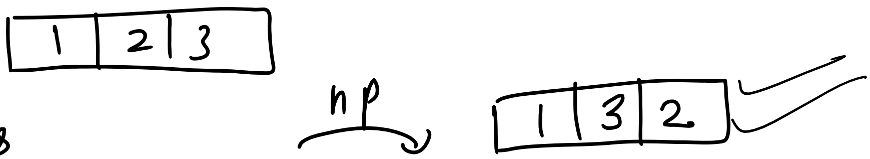
Case 3



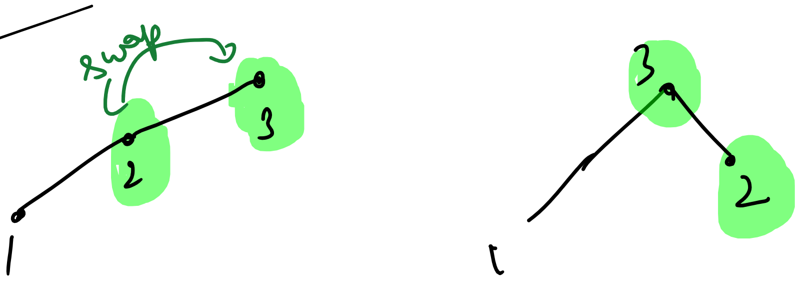
next permutation

Lets think it graphically what is happening

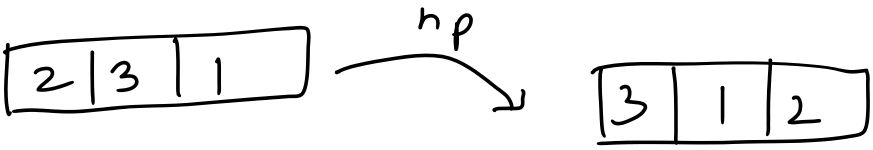
Case 1

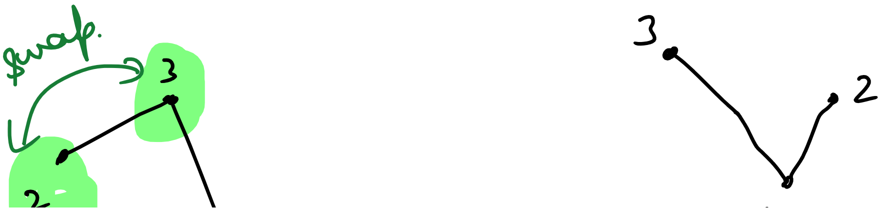


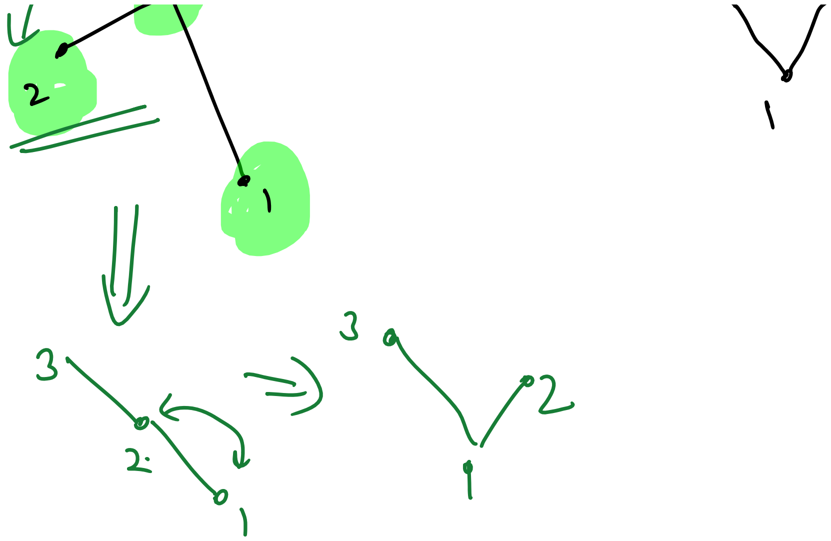
*Focus on changing elements



Case 2

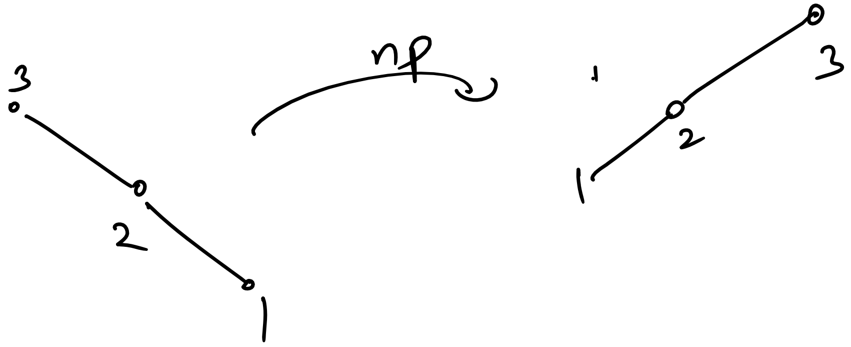






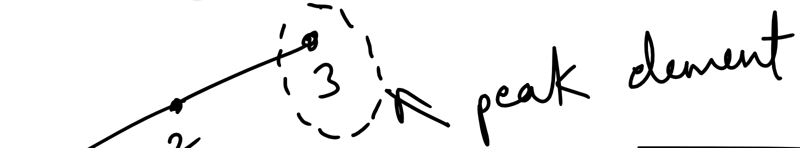
Case 3

3	2	1



★ Try to think कि ये क्या रहा है ||

1	2	3



np

1	3	2

5	7	3	2	1

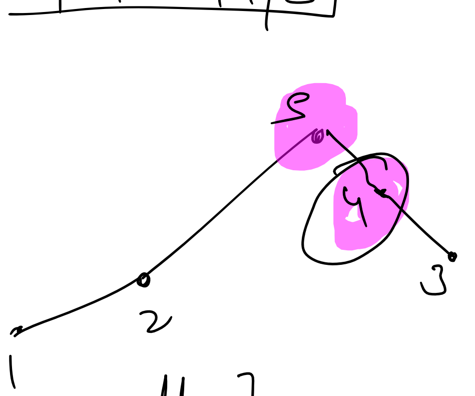
↓ np

7	5	1	2	3

next permutation का मतलब है कि इससे just greater जो बन सकता है Based upon given arrangement।

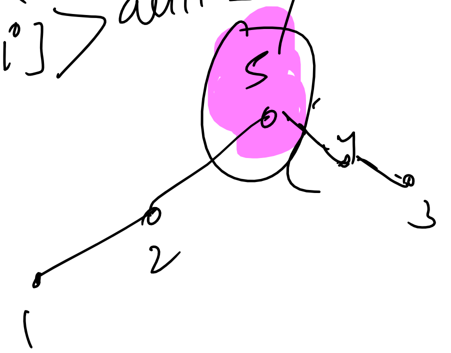
except for the case जब हम greatest permutation पर हों ||

1	2	5	4	3



जो right side से सबसे बड़ी value होगी जिससे हम peak बना रहे हैं। उसके साथ कुछ करना है ||

[peak basically अगर arr[i] > arr[i-1]] → peak है

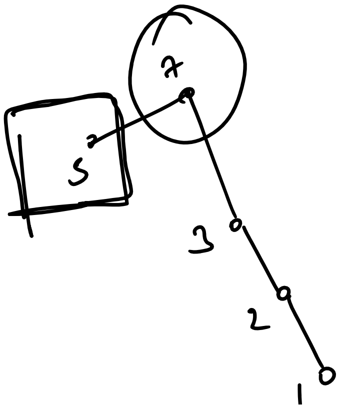


np ⟶

1	3	2	4	5

5	7	3	2	1

peak



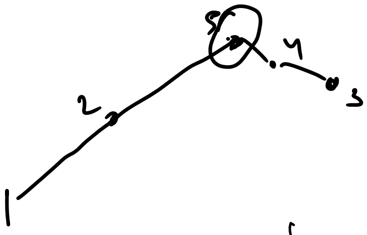
⟹

Answer

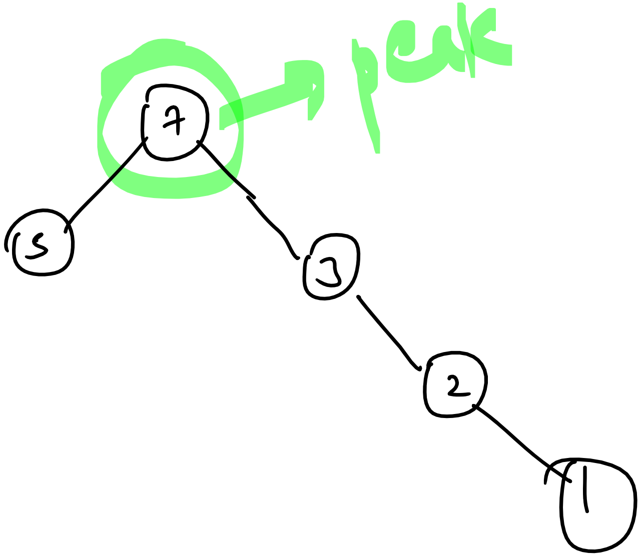
7	1	2	3	5

1°

1	2	5	4	3

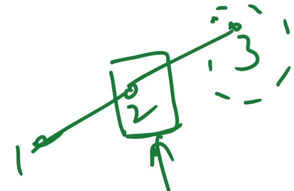


★★ right से पहला peak मिलेगा तो उसका कुछ use करो ||

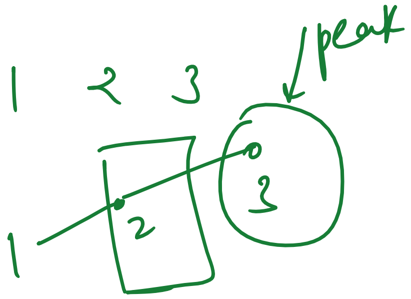


अब ?

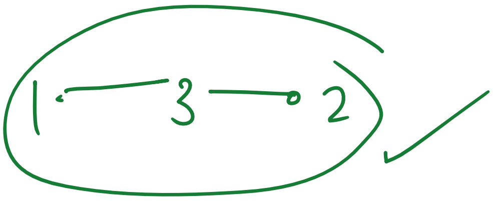
1 2 3 ⟹



मतलब इस Basically peak के left वाले से है क्योंकि जो next permutation बनेगी वो इस digit से बड़ी value जो होगी right side से वो होगी ||



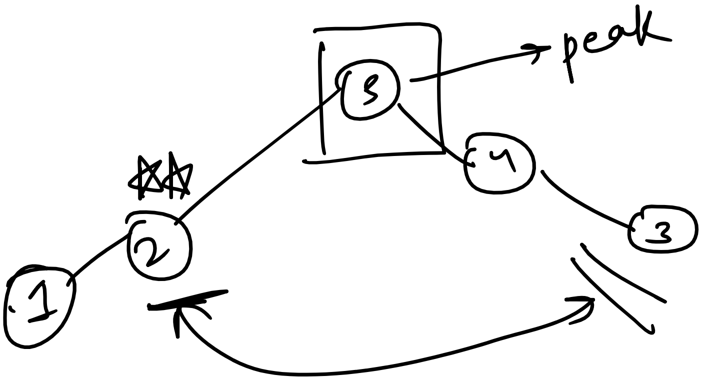
2 से बडी value right side से is 3 तो sweep करना था ||



1	3	2

Ex2

1	2	5	4	3



1 3 5 4 2 (Is it our answer) X

1 3 2 4 5] Answer.

* तो next step of Algorithm is to sort from peak_index to end of the array.

$$1 \quad 3 \quad \underbrace{5 \quad 4 \quad 2}_{\text{sort}}$$

$$\underbrace{1 \quad 3 \quad 2 \quad 4 \quad 5}$$

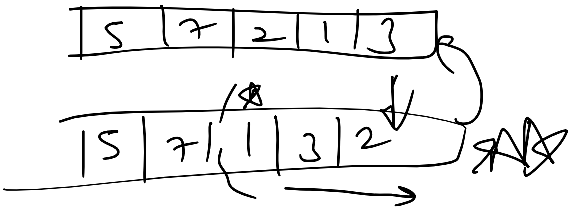
Answer

Steps followed till Now

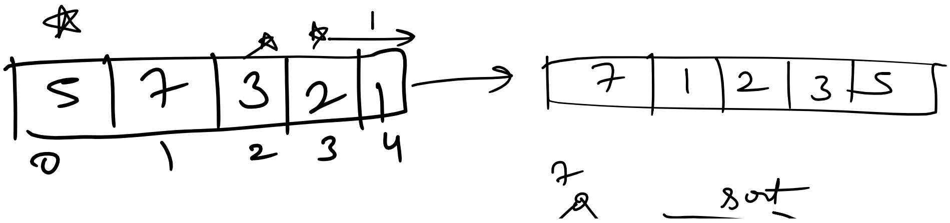
1. Find peak element index.
   (arr[i] > arr[i-1])

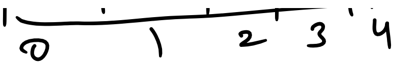
2. Basically हो peak_index -1 वाली value से बड़ी value जो right side से मिलेगी उसको swap करना है ||
   such that
   swap(Arr[peak_index -1] , Arr[rsbv])

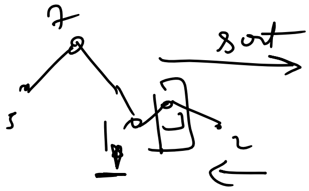
3. Sort करना है peak-index से लेके Array end तक.

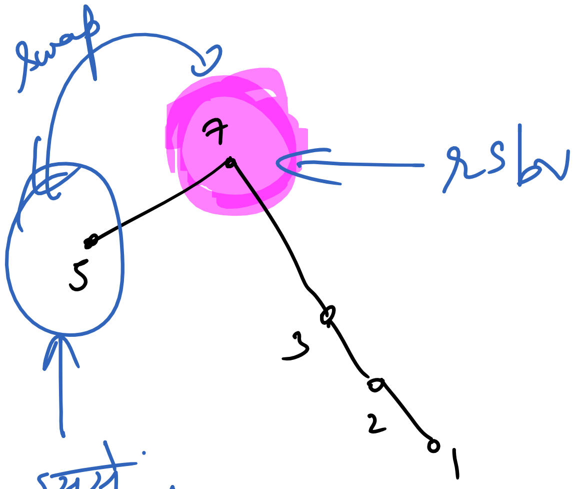


Ex









7	5	3	2	1

sort this part

इससे बड़ी value जो right side से मिलेगी

Step 3 peak_index से Array_size तक sort कर दो

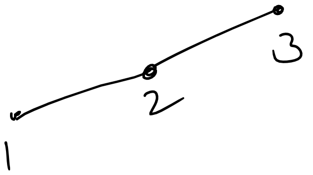
7	1	2	3	5

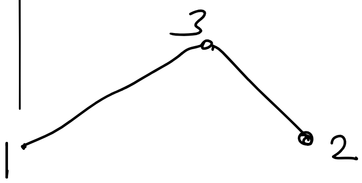
Answer

1	2	3

np

1	3	2

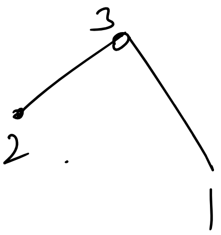


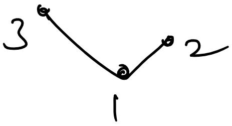


2	3	1

np

3	1	2





5	7	3	2	1

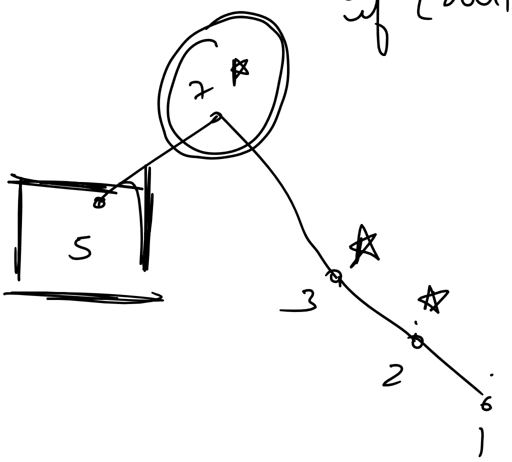
$\xrightarrow{np}$

7	1	2	3	5

5 7 3 2 1

7 1 2 3 5

if (arr[i] > arr[i-1]) → peak element



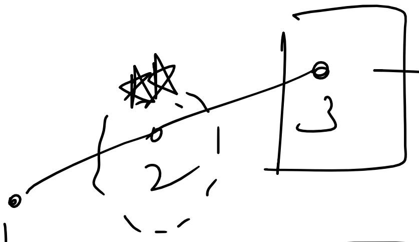
left

☆ peak element

☆ peak element ~~to~~ side

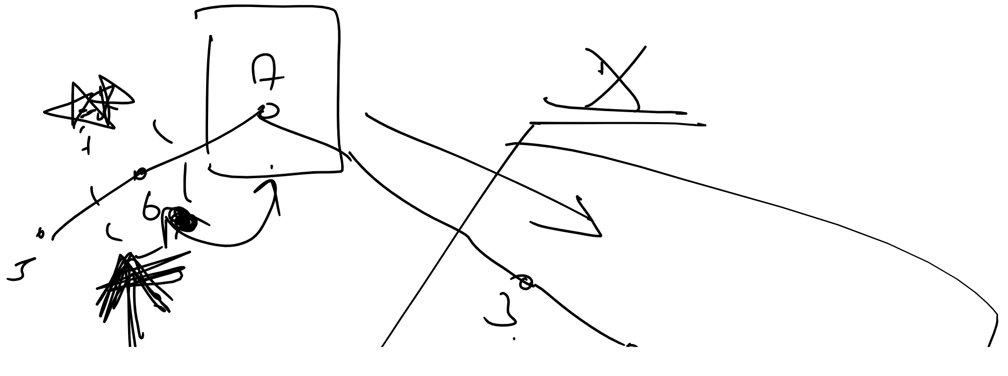
1	2	3

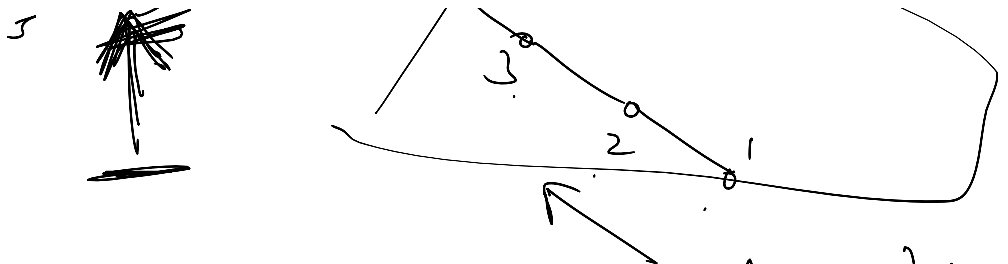
right side



→ peak

5	6	7	3	2	1





☆ $A_1[p\text{-}index - 1]$ ⟶ ⟵ if ($A_1[j] > A_1[p\text{-}index - 1]$)

5	6	7	3	2	1

np ⟶

5	7	1	2	3	6

5	7	6	3	2	1

5 7 1 2 3 6

pindex sorted

5	7	1	2	3	6

Algorithm

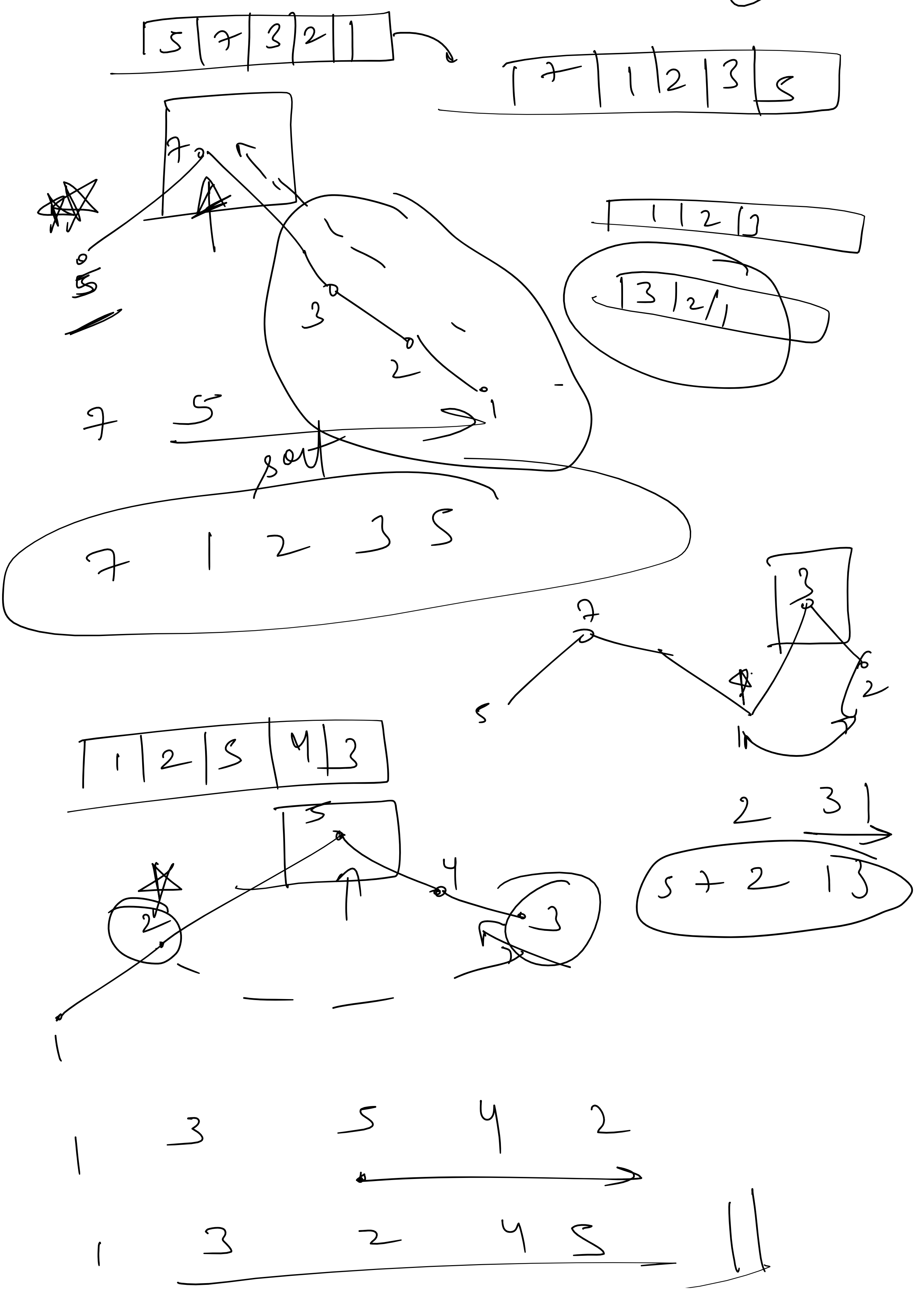
(1) peak_index ⟶ right_side X

(2) A[peak_index −1] ⟵ right traversal

A[j]

(3) sort(A[peak_index], last) //

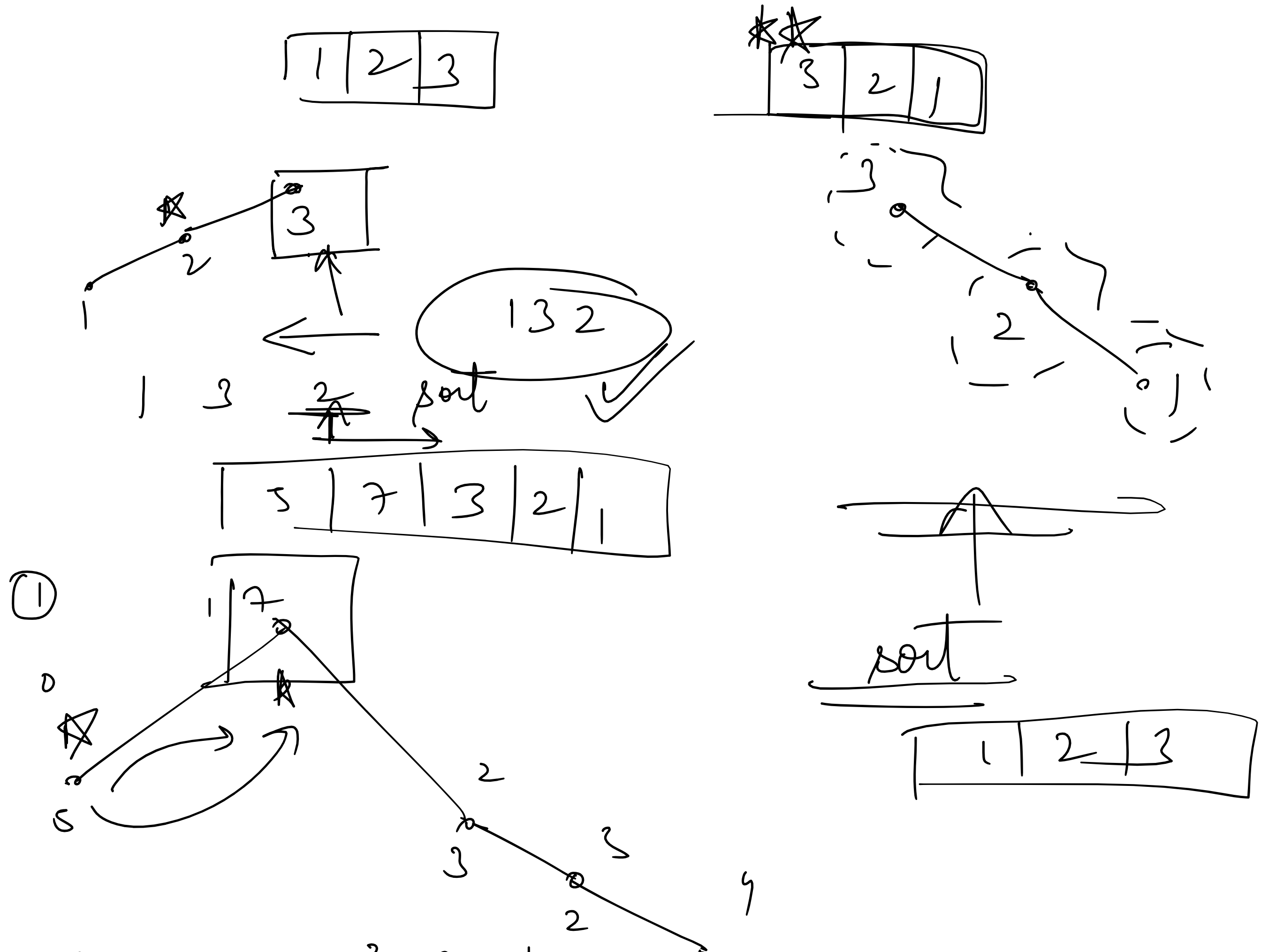
(3) sort(A, peak_index,)



1 3 2 4 5 11

5 7 3 2 1

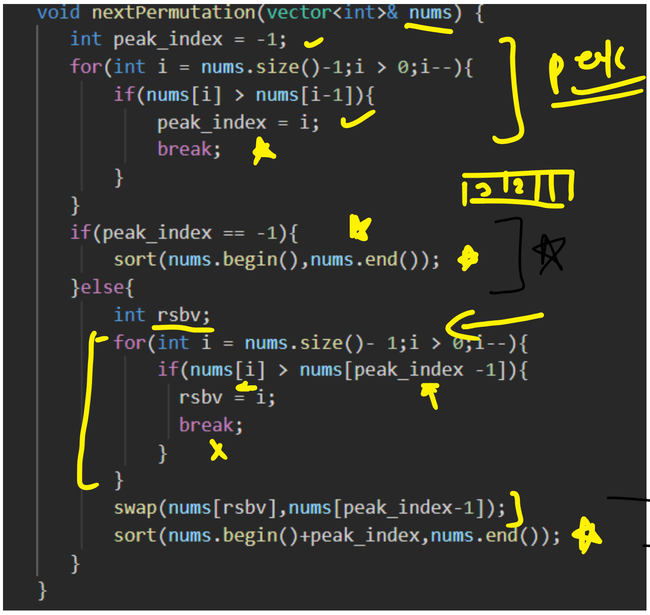
sort



7 5 3 2 1

7 1 2 3 5 Answer

Fn (aleen))



```
void nextPermutation(vector<int>& nums) {
    int peak_index = -1;
    for(int i = nums.size()-1;i > 0;i--){
        if(nums[i] > nums[i-1]){
            peak_index = i;
            break;
        }
    }
    if(peak_index == -1){
        sort(nums.begin(),nums.end());
    }else{
        int rsbv;
        for(int i = nums.size()- 1;i > 0;i--){
            if(nums[i] > nums[peak_index -1]){
                rsbv = i;
                break;
            }
        }
        swap(nums[rsbv],nums[peak_index-1]);
        sort(nums.begin()+peak_index,nums.end());
    }
}
```

T.C-(nlogn)

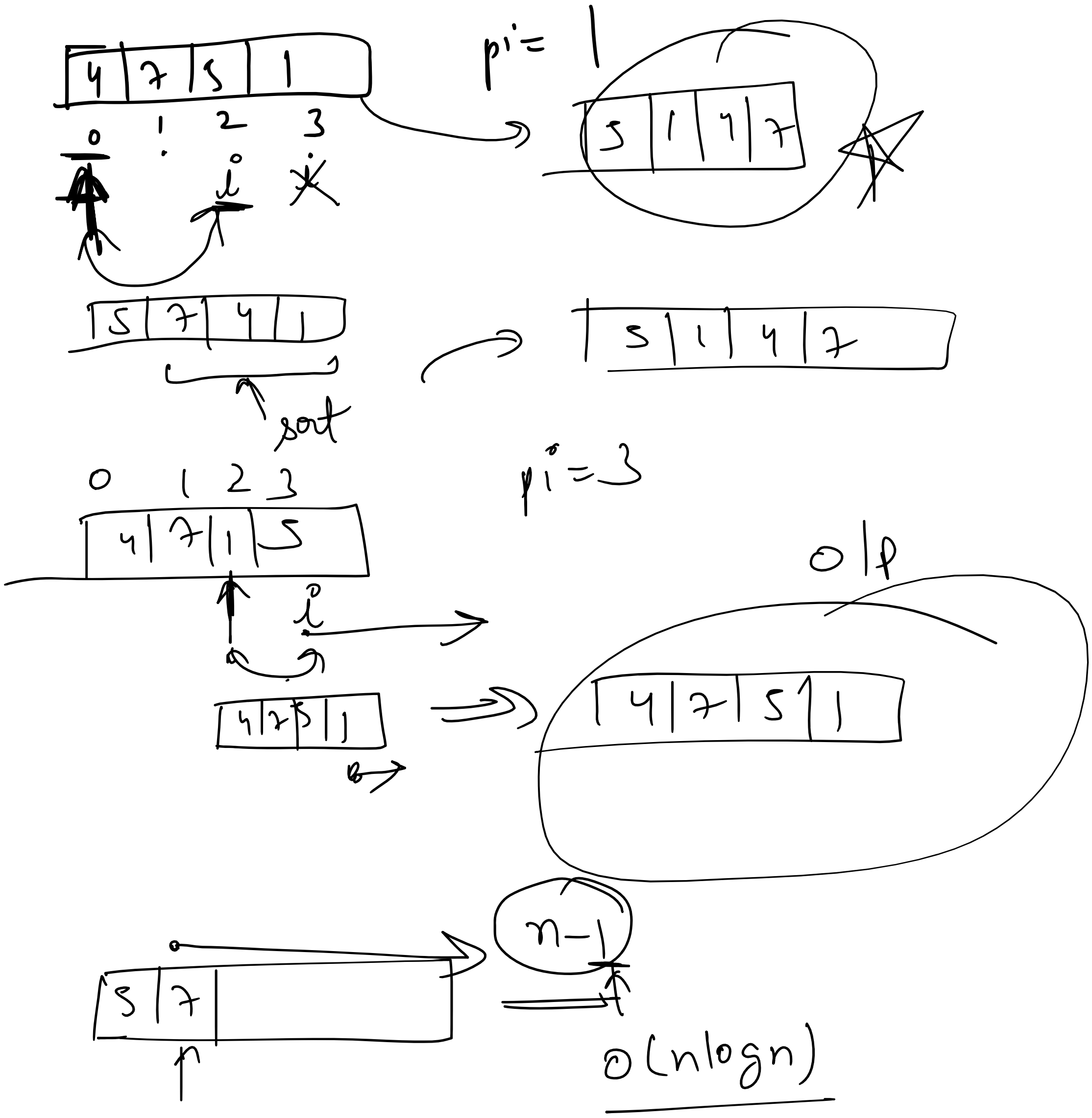
S.C-O(1)

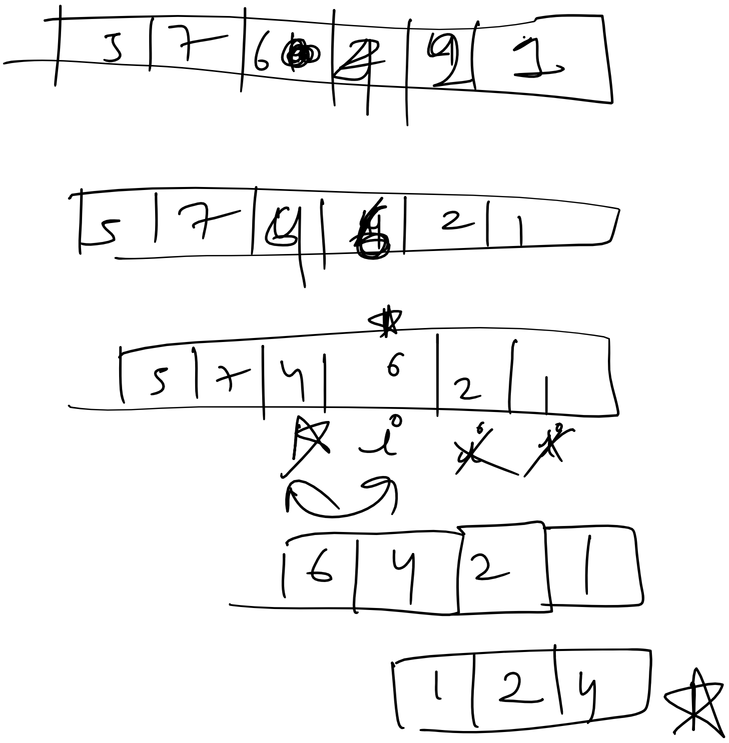
O(n)

~~O(nlogn)~~ O(n)

O(N)

O(nlogn)



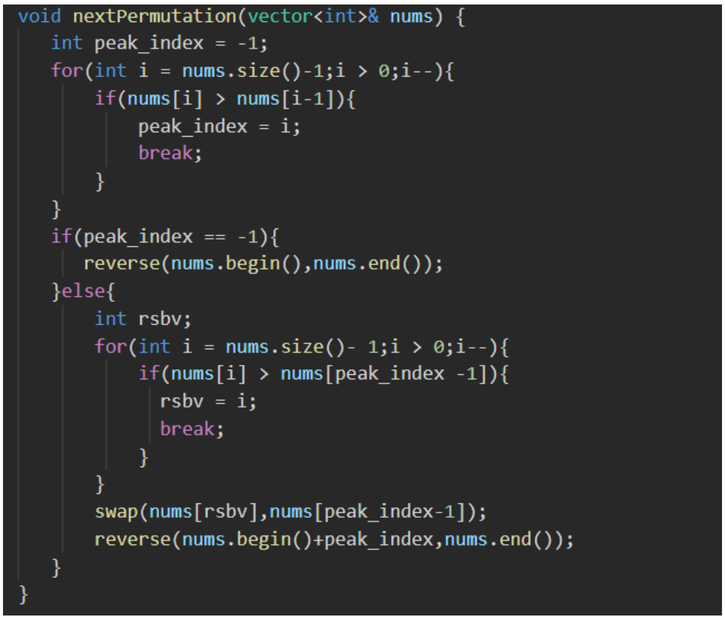


sorting

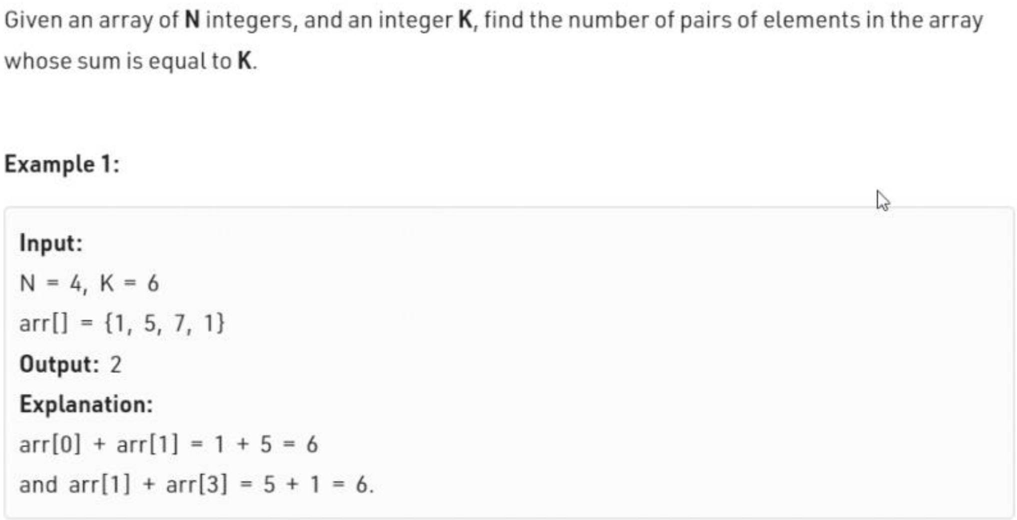
Sort → reverse.

T.C − O(N)

S.C − O(1)



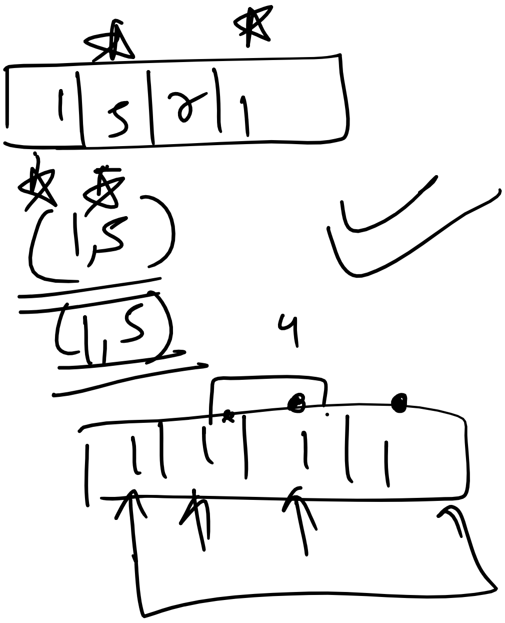
```
void nextPermutation(vector<int>& nums) {
    int peak_index = -1;
    for(int i = nums.size()-1;i > 0;i--){
        if(nums[i] > nums[i-1]){
            peak_index = i;
            break;
        }
    }
    if(peak_index == -1){
       reverse(nums.begin(),nums.end());
    }else{
        int rsbv;
        for(int i = nums.size()- 1;i > 0;i--){
            if(nums[i] > nums[peak_index -1]){
              rsbv = i;
              break;
            }
        }
        swap(nums[rsbv],nums[peak_index-1]);
        reverse(nums.begin()+peak_index,nums.end());
    }
}
```



Given an array of **N** integers, and an integer **K**, find the number of pairs of elements in the array whose sum is equal to **K**.

Example 1:

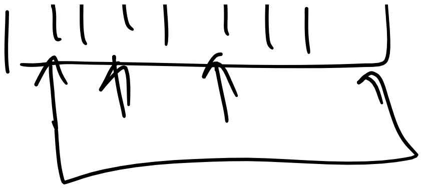
Input:
N = 4, K = 6
arr[] = {1, 5, 7, 1}
Output: 2
Explanation:
arr[0] + arr[1] = 1 + 5 = 6
and arr[1] + arr[3] = 5 + 1 = 6.



K = 6

k = 2

6 pair O/P

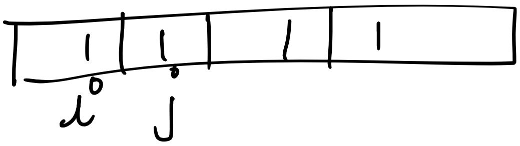


6 pair O/P

}

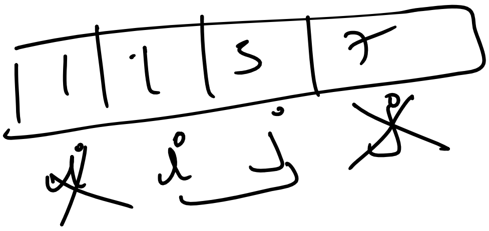
Naive

O(N^2)

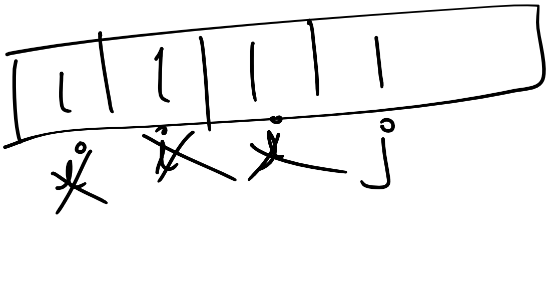


$O(N^2)$

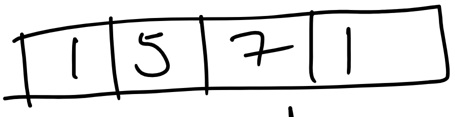
```
for(int i=0; i<n; i++){
    for(int j=i+1; j<n; j++){

    }
}
```



2



1 2 3



☆ unordered_map

Dictionary

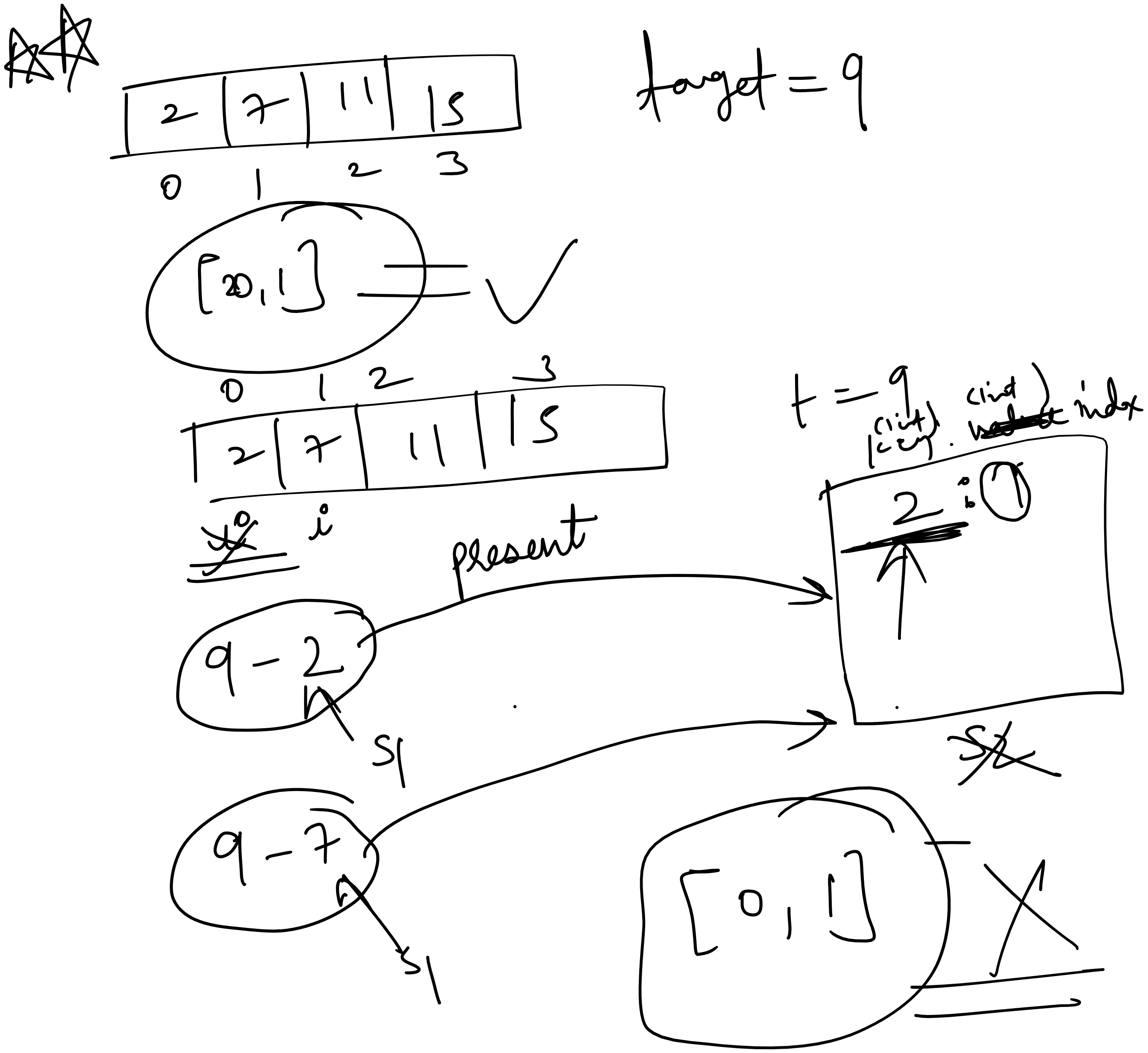
Hashmap

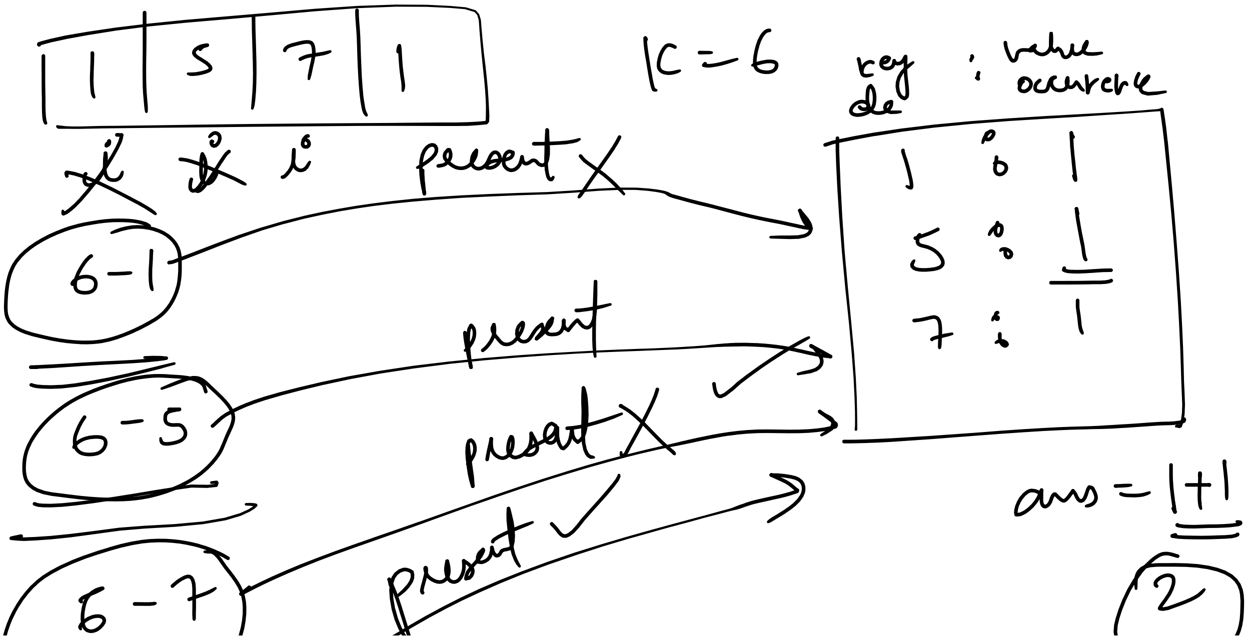
C++

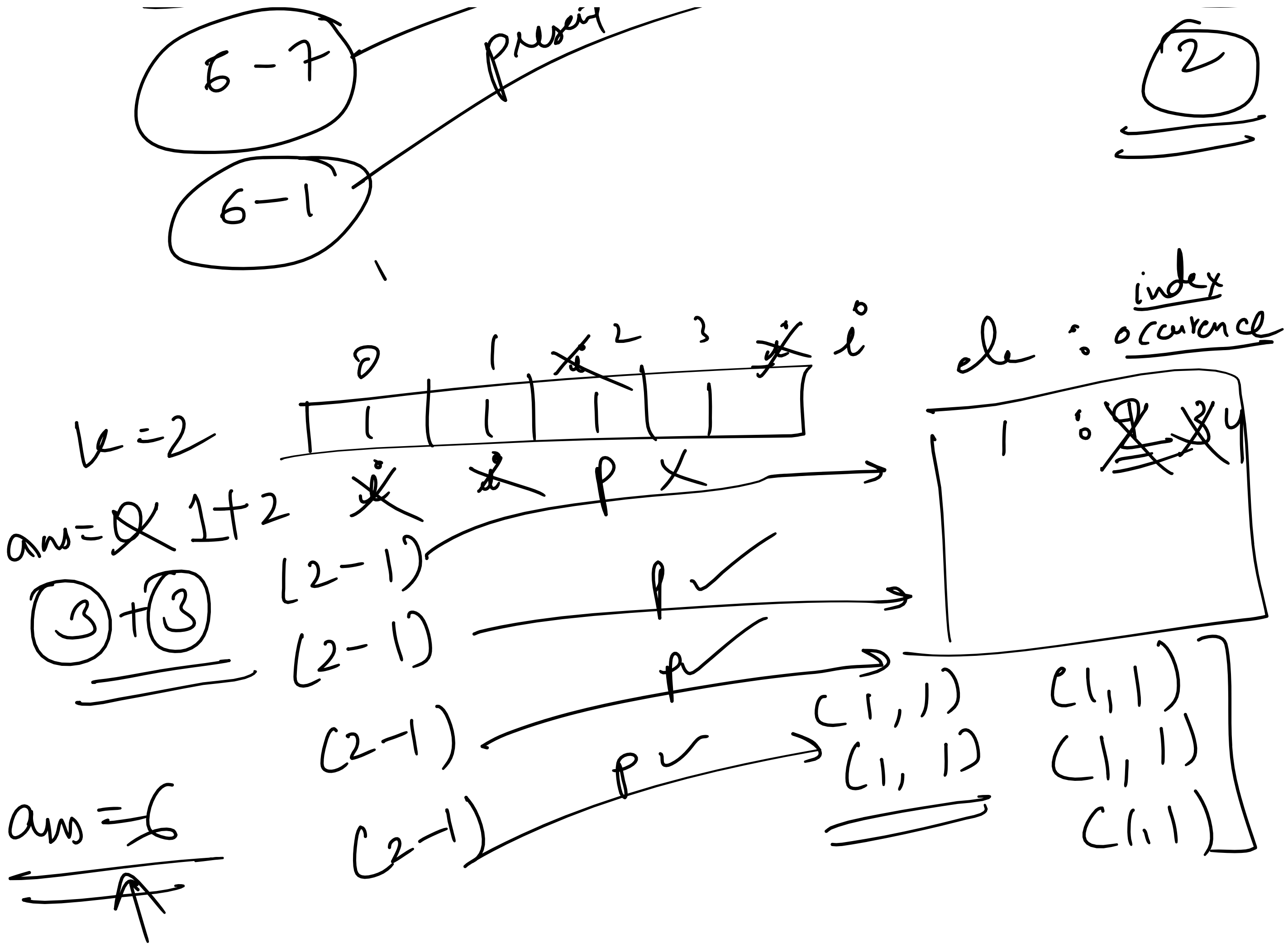
python

Java

Hashmap Java

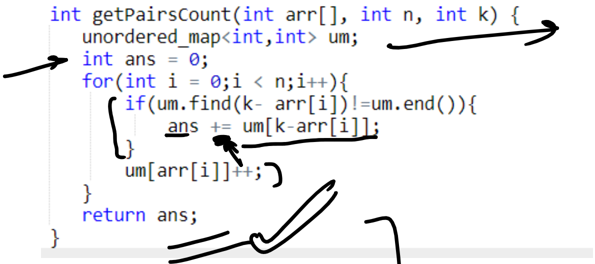






subarray with 0 sum

max length subarray with 0 sum



```
int getPairsCount(int arr[], int n, int k) {
    unordered_map<int,int> um;
    int ans = 0;
    for(int i = 0;i < n;i++){
        if(um.find(k- arr[i])!=um.end()){
            ans += um[k-arr[i]];
        }
        um[arr[i]]++;
    }
    return ans;
}
```

T.C - O(N)

S.C - O(N)

Simple Code