C++
ARRAY-3

Lecture- 14

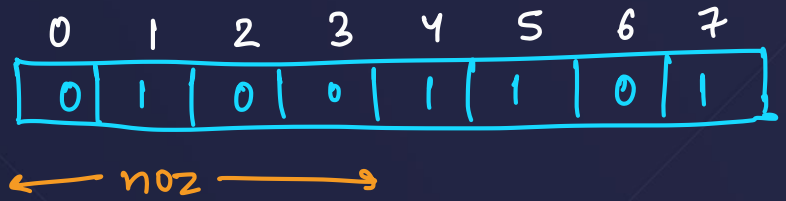
Raghav Garg

Today's checklist

1) Problem solving on Arrays

Ques : Sort the array of 0's and 1's .

int n;



M-I : Two Pass Method

int noz = 0; 1 2 3 4

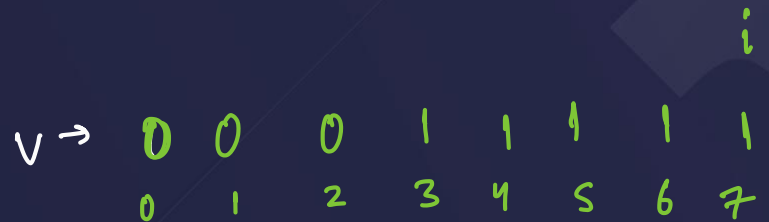
int noo = 0; 1 2 3 4

0 → (noz - 1)th ⇒ 0

nozth ⇒ 1

Ques : Sort the array of 0's and 1's .

```
void sort01(vector<int>& v){
    int n = v.size();
    int noo = 0;
    int noz = 0;
    for(int i=0;i<n;i++){
        if(v[i]==0) noz++;
        else noo++;
    }
    // filling elements
    for(int i=0;i<n;i++){
        if(i<noz) v[i] = 0;
        else v[i] = 1;
    }
}
```



n = 8

noo = ~~0~~ 1 2 3 4 5

noz = ~~0~~ 1 2 3

Total time taken → 2n

Ques : Sort the array of 0's and 1's.

int * ptr ✗

M-2 : * Two Pointers * → two variables

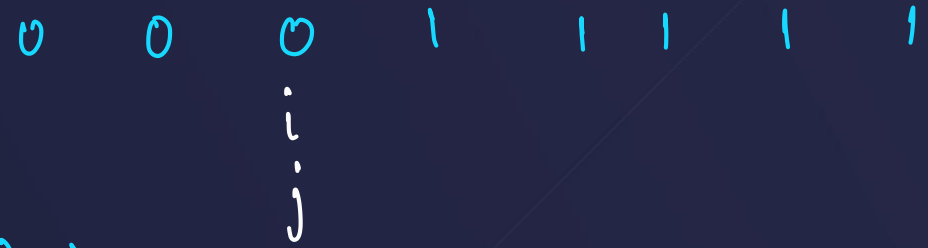
0	1	2	3	4	5	6	7
0	0	0	0	1	1	1	1
			j	i			

int i = 0
int j = n-1

Hint → 0's are to the front
1's are to the back
swapping

```
while (i<j){
    if(arr[j] == 1) j--;
    if(arr[i]==0) i++;
    if(arr[i]==1 && arr[j]==0)
        swap( );
}
    i++;
    j--;
```

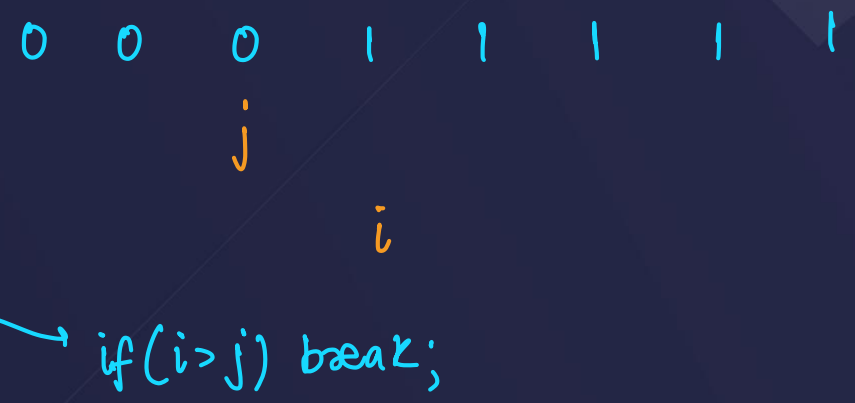
Ques : Sort the array of 0's and 1's .



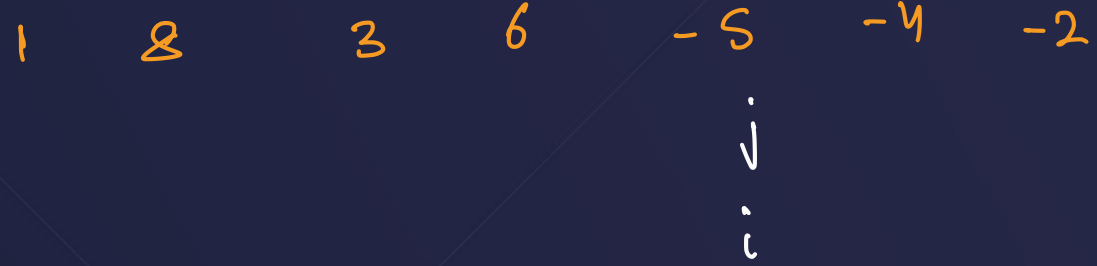

```
if (arr[j]==1) j--;
if (arr[i]==0 ) i++;
if(arr[i] ==1 && arr[j] ==0){
    arr[i] = 0;
    arr[j] = 1;
    i++;
    j--;
```

Ques : Sort the array of 0's and 1's .

```
void sort01m2(vector<int>& v){
    int n = v.size();
    int i = 0;
    int j = n-1;
    while(i<j){
        if(v[i]==0) i++;
        if(v[j]==1) j--;
        if(v[i]==1 && v[j]==0){
            v[i] = 0;
            v[j] = 1;
            i++;
            j--;
        }
    }
}
```



Ques : Move all negative numbers to beginning and positive to end with constant extra space. (Classwork)

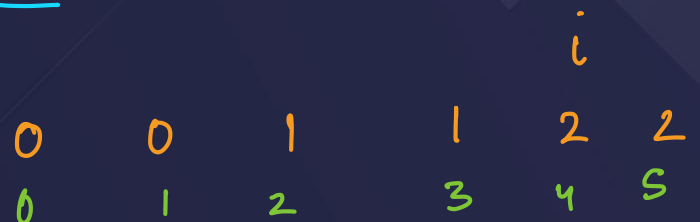


Ques : Sort the array of 0's , 1's and 2's . (LeetCode 75)

Dutch Flag Algorithm

M-I

```
// fill
// [2,0,2,1,1,0]
for(int i=0;i<n;i++){
    if(i<noz) nums[i] = 0;
    else if(i<(noz+noo)) nums[i] = 1;
    else nums[i] = 2;
}
return;
```



noz = 2

noo = 2

notw = 2

Ques : Sort the array of 0's , 1's and 2's .

M-2 : 3 pointer algorithm (dutch flag algo)

lo mid hi

hint → ✓ 0 to lo-1 → 0 lo to mid-1 → 1

✓ hi+1 to n-1 → 2

mid → khelna → if

Ques : Sort the array of 0's , 1's and 2's .

		lo		hi		
0	0	1	1	2	2	
				mid		

```
int lo = 0
int mid = 0
int hi = n-1
```

1) `if (nums[mid] == 2)`
   `swap (mid , hi)`
   `hi--;`

2) `if (nums[mid] == 0)`
   `swap (mid, lo)`
   `lo++`
   `mid++`

3) `if (nums[mid] == 1)`
   `mid++;`

```
while (mid <= high){
    3 conditions
}
```

Ques : Merge two sorted arrays . (LeetCode - 88)

arr 1

arr 2

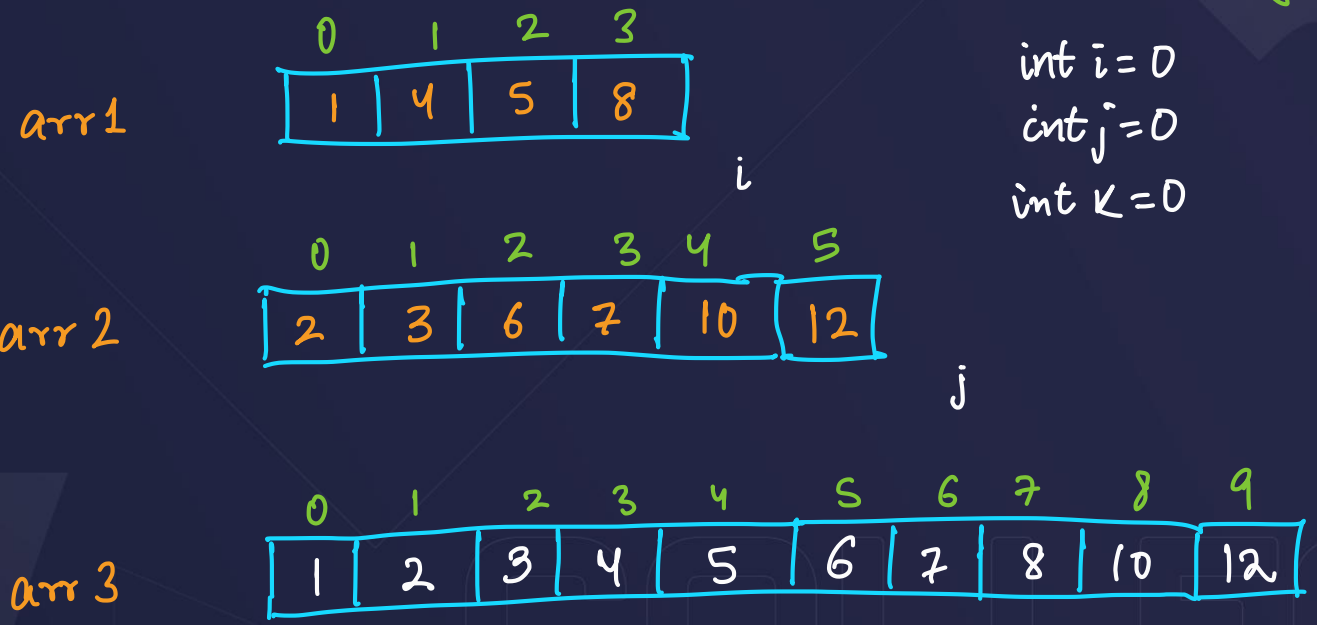
arr 3

#Hint :- 3 pointers → i, j, K

Ques : Merge two sorted arrays .

int arr3[m+n];

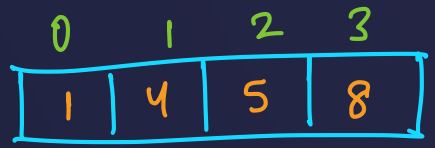
int i = 0
int j = 0
int k = 0



Ques : Merge two sorted arrays .

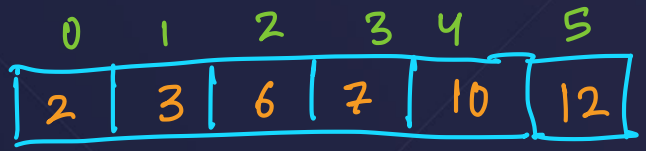
```
int i = m-1
int j = n-1
int k = m+n-1
```

m arr 1



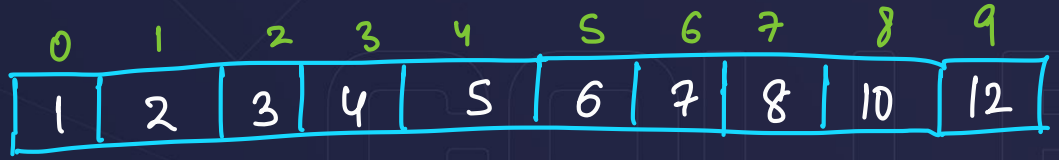
i

n arr 2



j

m+n arr 3



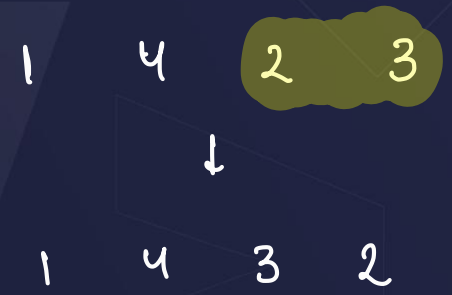
k

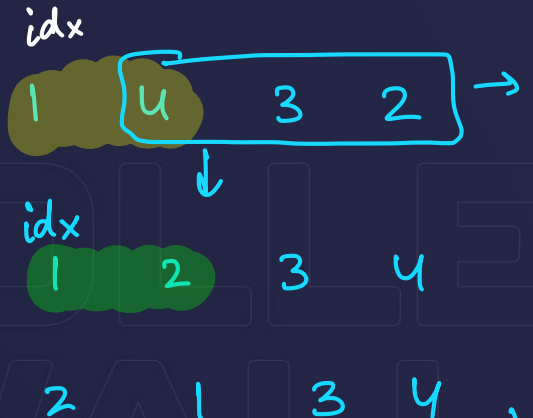
Ques : Find the next permutations of Array .

Note :- If not possible then print the sorted order in ascending order. (LeetCode - 31)

1 , 2, 3

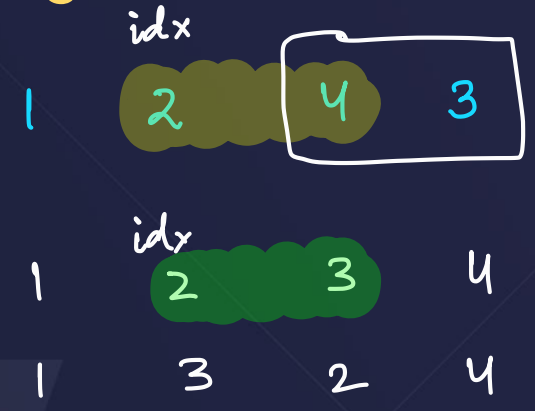
1 , 2, 3 1 ,3, 2 2 , 1, 3 2, 3, 1 3, 1, 2 3, 2, 1





Ques : Find the next permutations of Array .

Note :- If not possible then print the sorted order in ascending order.



pivot idx → int idx = -1;

```
for (int i = n-2; i >= 0; i--){
    if(arr[i] < arr[i+1] ){
        idx = i;
        break;
    }
}
```

3

3

Step-1 → Find pivot idx

Step-2 → idx+1 to end → reverse

Step-3 → idx+1 to end → sort → find just greater number ka idx

Step-4 → Swapping idx, j

Ques : Find the next permutations of Array .

Note :- If not possible then print the sorted order in ascending order.

4 3 2 1 → 1 2 3 4

```
if (idx == -1) {
    reverse
}
```

3

```
i, j → reverse(nums.begin() + i, nums.begin() + j + 1);
```

Ques : Find the next permutations of Array .

...hen print the sorted order in

```cpp
int n = nums.size();
// 1) finding pivot index
int idx = -1;
for(int i = n-2;i>=0;i--){
    if(nums[i]<nums[i+1]){
        idx = i;
        break;
    }
}
if(idx==-1){ // if array is already greatest
    reverse(nums.begin(),nums.end());
    return;
}
// 2) sorting/reverse after pivot
reverse(nums.begin()+idx+1, nums.end());
// 3) swapping idx and idx+1
int temp = nums[idx];
nums[idx] = nums[idx+1];
nums[idx+1] = temp;
return;
```

idx

2 3 1

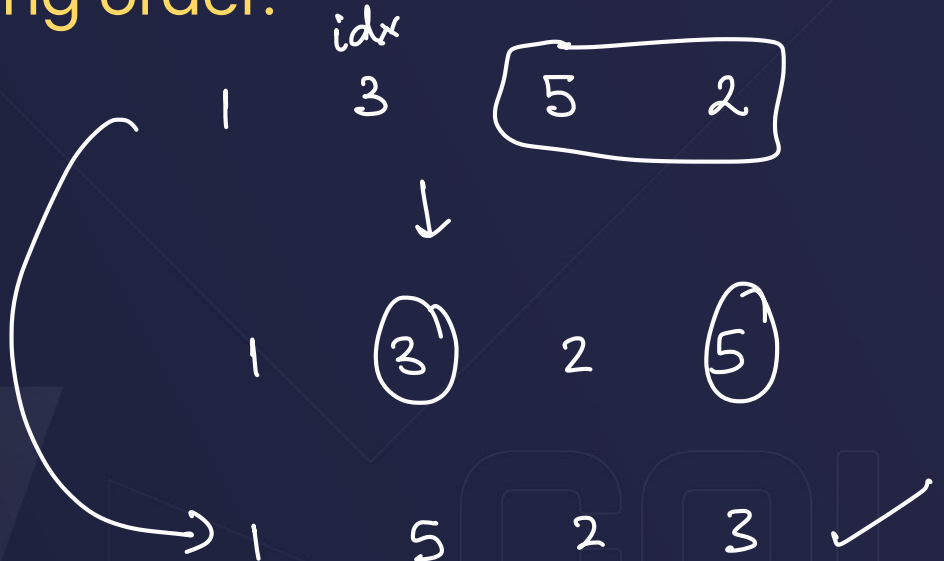
0 1 2

idx = 1 0

2 1 3

Ques : Find the next permutations of Array .

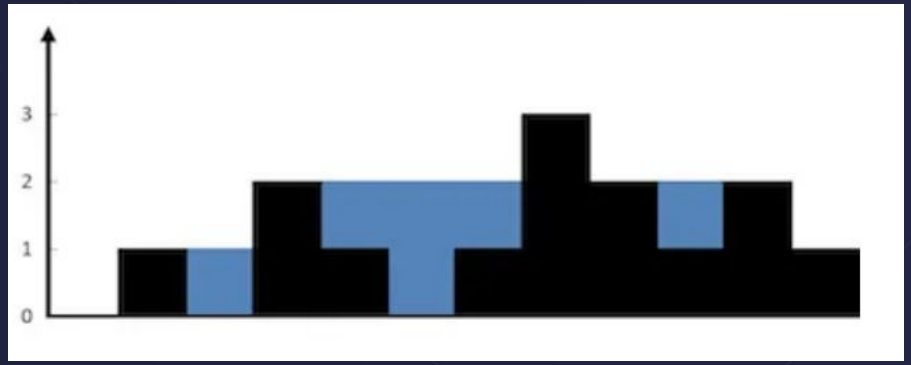
Note :- If not possible then print the sorted order in ascending order.



Ques : Trapping Rain Water

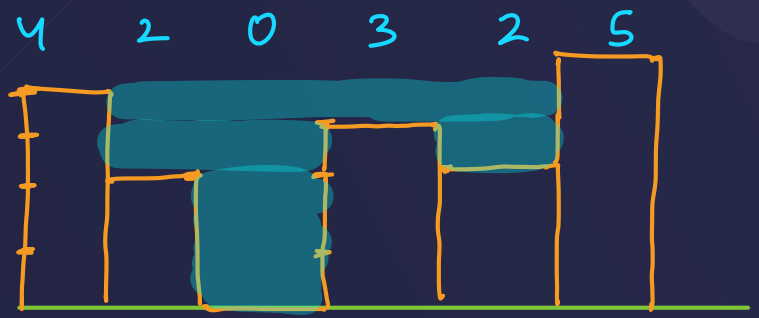
(LeetCode - 42) (Hard)

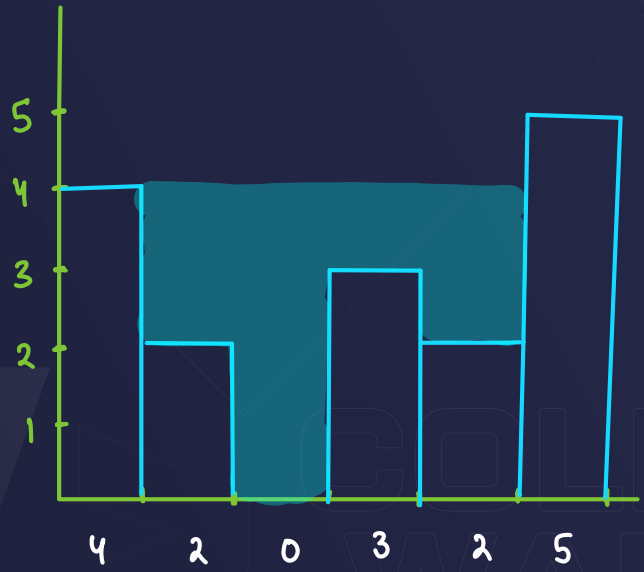
Famous



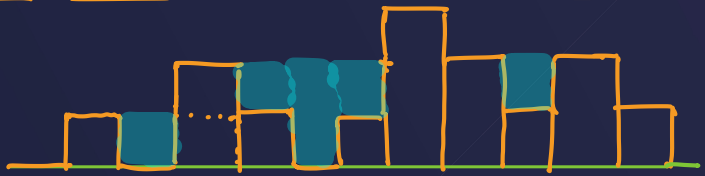
[0 1 0 2 1 0 1 3 2 1 2 1]

water → ?




5
4
3
2
1
4 2 0 3 2 5

Prev. Greatest Element & Next Greatest Element



heights [0 1 0 2 1 0 1 3 2 1 2 1]

if heights[i] < mini[i]

arr [-1 0 1 1 2 2 2 2 3 3 3 3]

brr [3 3 3 3 3 3 3 2 2 2 1 -1]

crr [-1 [0 1 1 2 2 2 2 2 2 1] [-1]

Prev. Greatest Element Array

arr[0] = -1

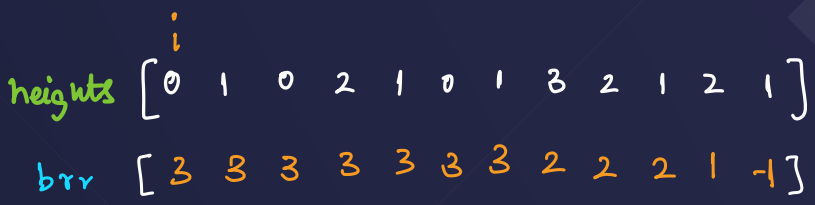
heights [0 1 0 2 1 0 1 3 2 1 2 1] (i)

arr [-1 0 1 1 2 2 2 2 3 3 3 3]

max = 0 1 2 3

```
for(int i = 1; i< 12 ; i++){
    arr[i] = max;
    if(max < heights[i]) max = heights[i];
}
```

Next Greatest Element Array :




```
max = heights[n-1];
max = 1 2 3
```

```
for (int i= n-2; i>=0 ; i--) {
    brr[i] = max;
    if(max < heights[i]) max= heights[i];
}
```

Trapping Rain Water...continued

```
int n = height.size();
// prev greatest element array
int prev[n];
prev[0] = -1;
int max = height[0];
for(int i=1;i<n;i++){
    prev[i] = max;
    if(max<height[i]) max = height[i];
}
// next greatest element array
int next[n];
next[n-1] = -1;
max = height[n-1];
for(int i=n-2;i>=0;i--){
    prev[i] = max;
    if(max<height[i]) max = height[i];
}
```

h → 0 1 0 2 1 0 1 3 2 1 2 1

p → -1 0 1 1 2 2 2 2 3 3 3 3

n → -1

max = 0 / 1

Trapping Rain Water...continued

height [0 1 0 2 1 0 1 3 2 1 2 1]

prev [-1 0 1 1 2 2 2 ~~2~~ ~~3~~ ~~3~~ 3 ~~3~~]

2 2 2 1 -1

max = ~~1~~ ~~2~~ 3

THANK YOU