timetable

Tuesday, 21 June 2022 11:06 AM

5-7	bike ,exercise
7-9	Dsa
10:30-11:20	DSA
11:30-1:30	Dsa
1:30-2:30	lunch
3-4	sleep
4-6	Tree
6-7	Free
7-8	Book Read /SM listen
9-11	Dsa
Last 1	important question video

Array Sheet

Wednesday, 15 June 2022 1:46 PM

Day start	**Revision => Solve Previous night read Question => Daily Solve Atleast 4-5 Question**			
Formula	**Read Twice Question =>Think Thrice Solution=> Code one times**			

1. Slinding Window
2. Two Pointer Approach
3. Kadane Algo
4. Dutch National Flag Algo
5. Binary Search Tree

Vectors

Wednesday, 15 June 2022 10:59 AM

vector<int> g1;

From <https://www.geeksforgeeks.org/vector-in-cpp-stl/>

g1.push_back(i);

From <https://www.geeksforgeeks.org/vector-in-cpp-stl/>

1. push_back() – It push the elements into a vector from the back
2. pop_back() – It is used to pop or remove elements from a vector from the back.
3. sort(v.begin(), v.end());
4. sort(res.begin(),res.end(),greater<int>());
5. Res.empty()- for check empty vector
6. set<int>s---------- s.insert(b[i]);
7. vector<int>res(n,10):- Create a vector of size n with value 10
8. unordered_map<int,int>m;-
   The time complexity of map operations is O(log n) while for unordered_map, it is O(1) on average.
   It mainly used to find occuernce of element.

```
if (m.find(key) == m.end())
        cout << key << " not found\n";
    else
        cout << "Found " << key << endl;
```

9. __builtin_popcount(b) :-for calculate number of 1 in binary terms for ex 4=100 it return 1
10. . vector<pair<int,int>>v;
    for(int i=0;i<n;i++)
    {
    v.push_back({nums[i],i});
    }.
11. res.insert(res.end(),row.begin(),row.end());
    For insert row in vector ex row vector={3,4} res={1,2} ==>res{1,2,3,4}

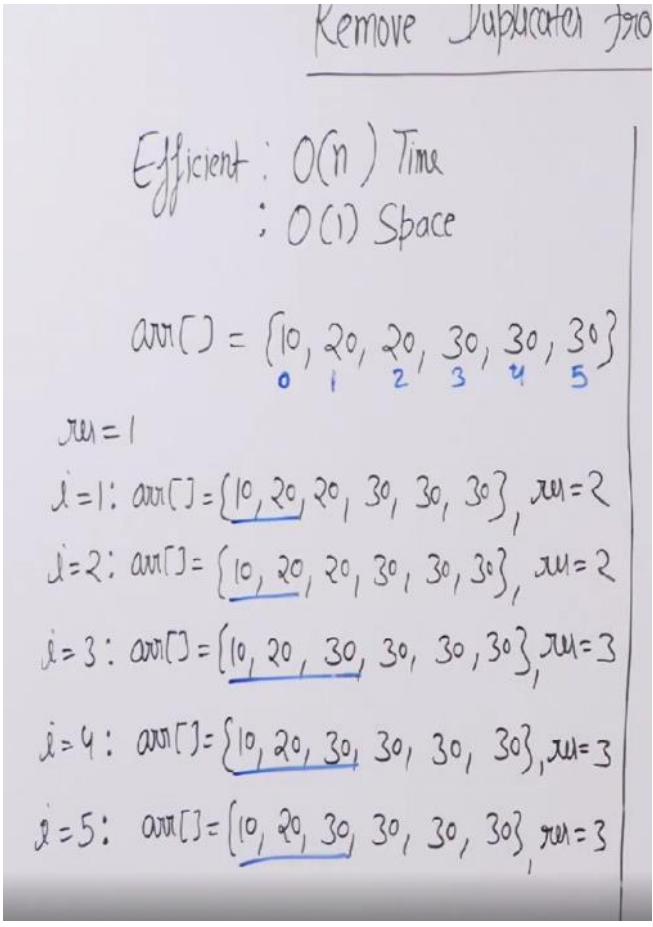
Remove duplicate elements from sorted Array

Wednesday, 15 June 2022 11:00 AM

Input:
N = 3
Array = {1, 2, 2}
Output: 2

From <https://practice.geeksforgeeks.org/problems/remove-duplicate-elements-from-sorted-array/1#>

```
int remove_duplicate(int arr[],int n){

   int res=1;
   for(int i=1;i<n;i++)
   {
      if(arr[i]!=arr[res-1]){
         arr[res]=arr[i];
         res++;

      }
   }
   return res;
   }
```



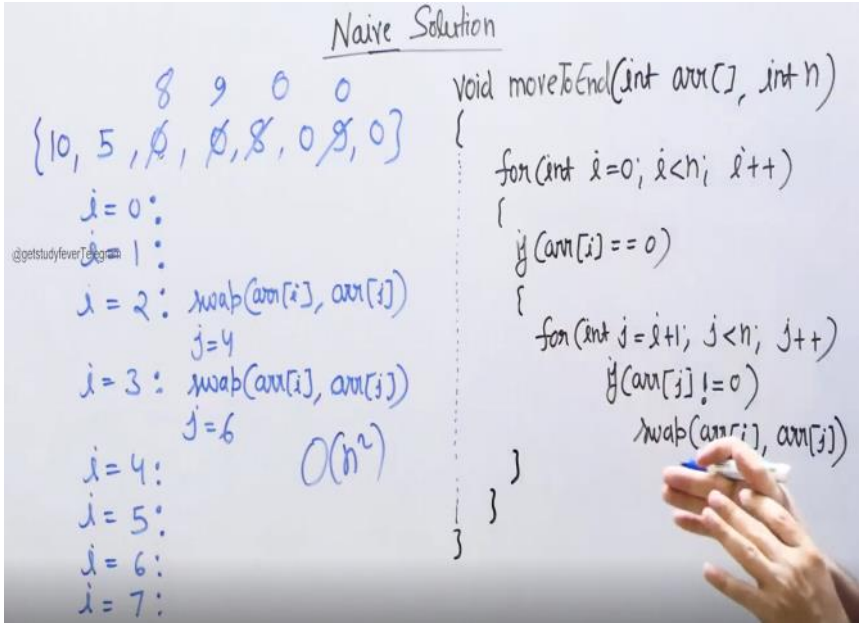
Move all zeroes to end of array

Wednesday, 15 June 2022 12:32 PM

Input:
N = 4
Arr[] = {0, 0, 0, 4}
Output: 4 0 0 0
Explanation: 4 is the only non-zero element and it gets moved to the left.

From <https://practice.geeksforgeeks.org/problems/move-all-zeroes-to-end-of-array0751/1#>

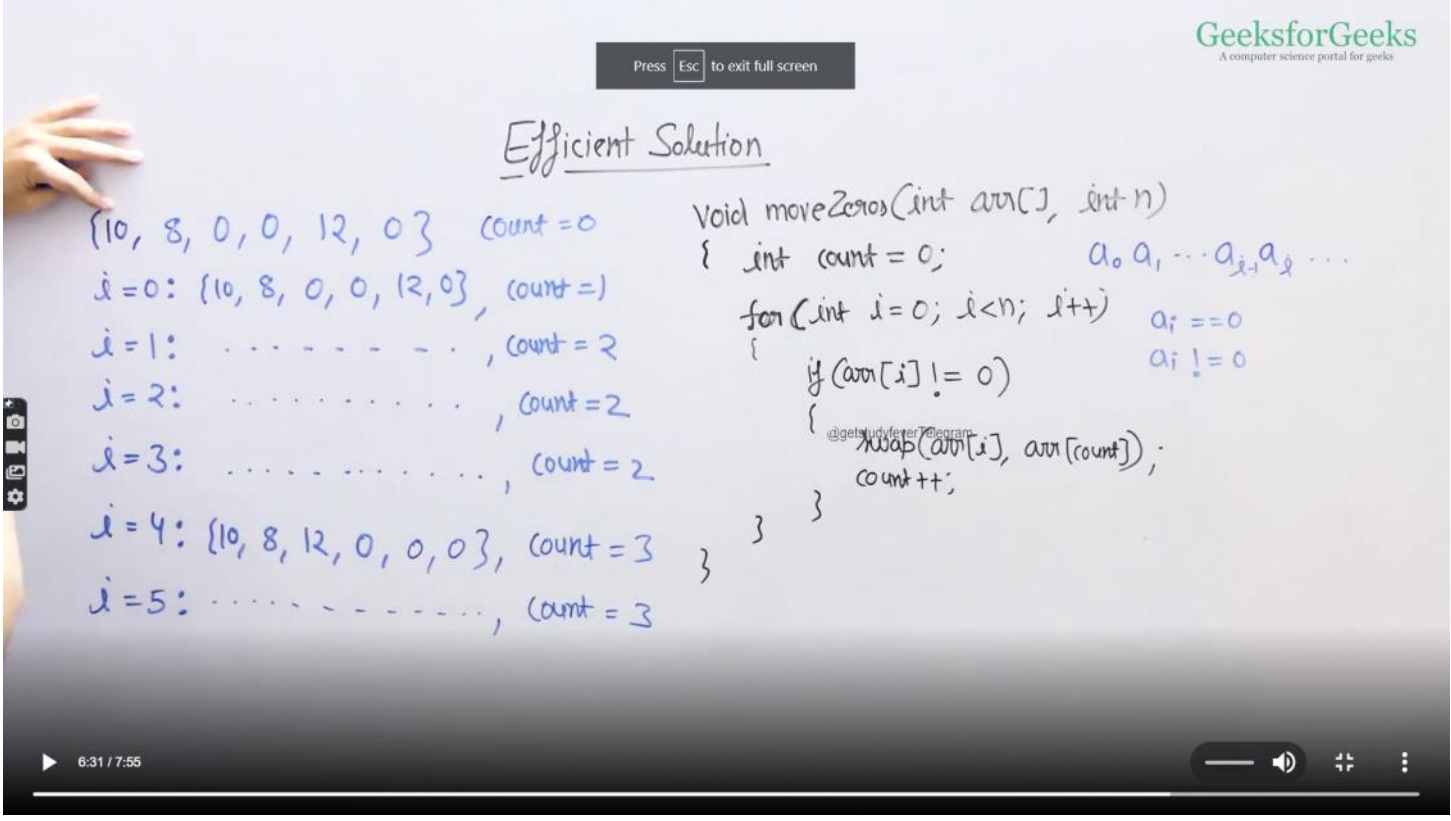
Method1



Method2

```
void pushZerosToEnd(int arr[], int n) {
    // code here
    int count=0;
    for(int i=0;i<n;i++){
      if(arr[i]!=0)
        {
          std::swap(arr[i],arr[count]);
          count++;
        }
    }

  }
```

Time com:O(n)



Left Rotate an element by d times

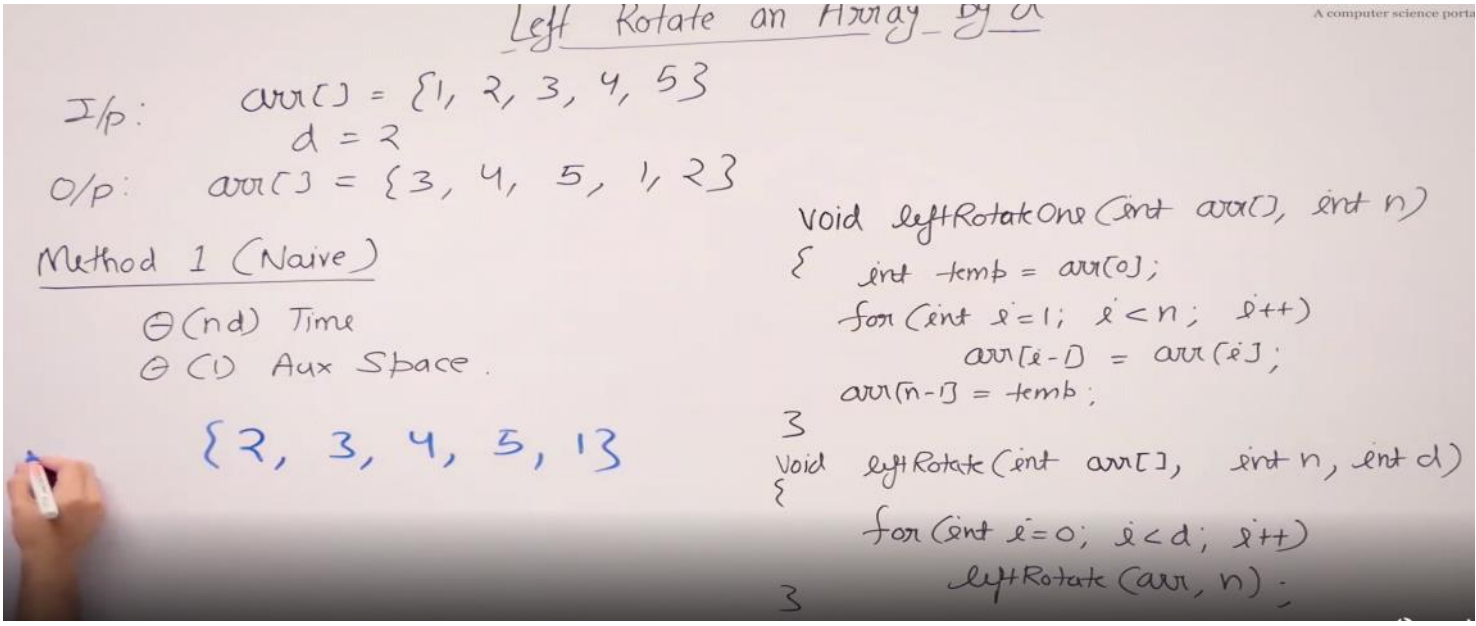
Thursday, 16 June 2022 10:56 AM

Input:
N = 5, D = 2
arr[] = {1,2,3,4,5}
Output: 3 4 5 1 2

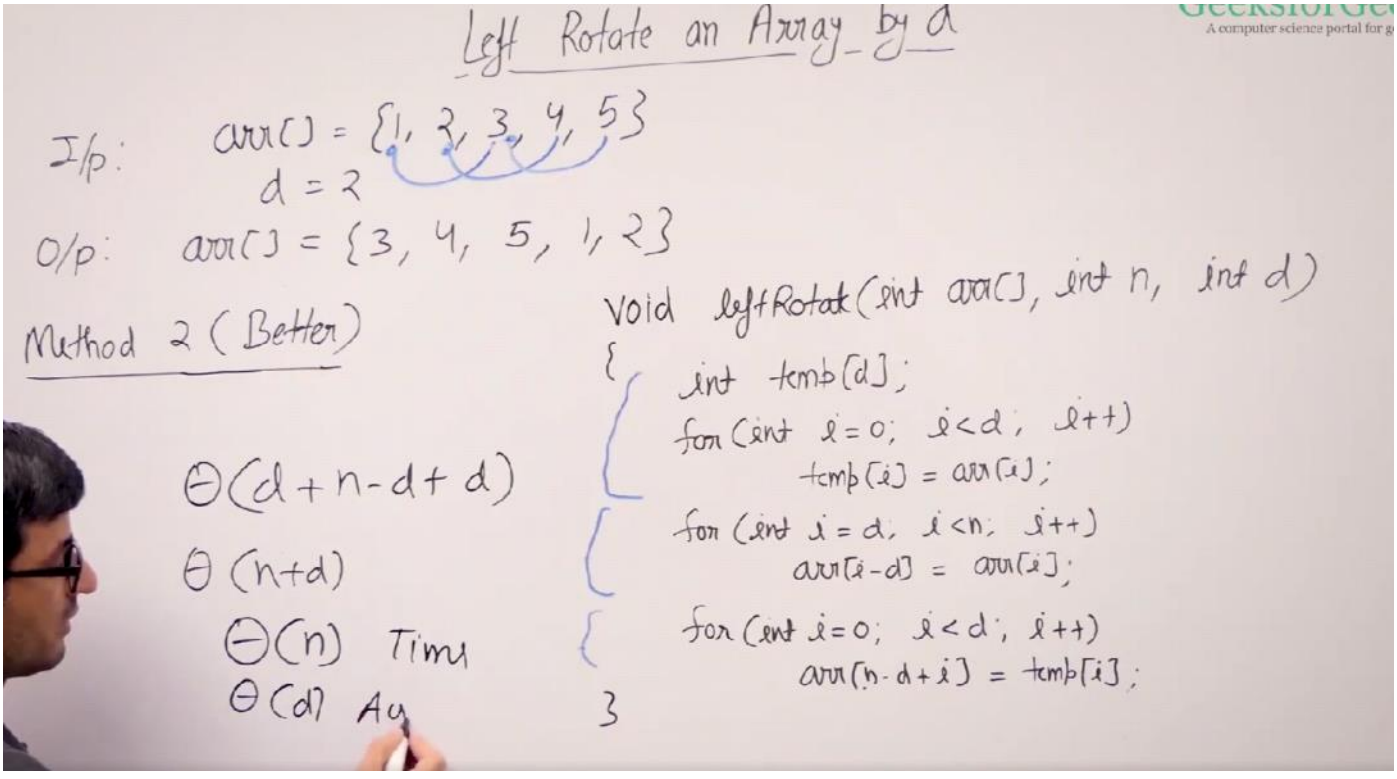
Explanation: 1 2 3 4 5 when rotated by 2 elements, it becomes 3 4 5 1 2.

From <https://practice.geeksforgeeks.org/problems/rotate-array-by-n-elements-1587115621/0/>

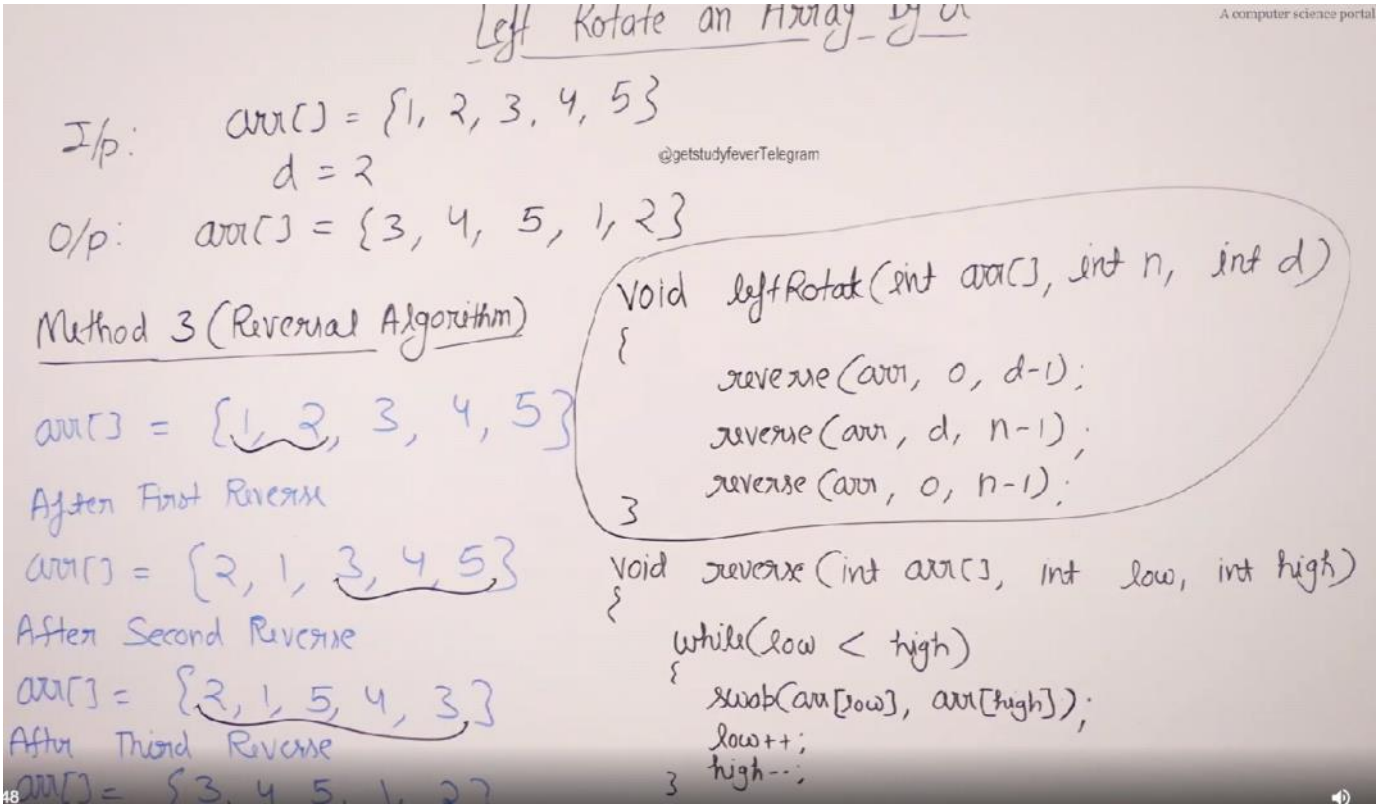
Method1



Method 2:



Mehod 3:


Left Rotate an Array by d
A computer science portal
I/p: arr[] = {1, 2, 3, 4, 5}
d = 2
@getstudyfeverTelegram
O/p: arr[] = {3, 4, 5, 1, 2}
Method 3 (Reversal Algorithm)
arr[] = {1, 2, 3, 4, 5}
After First Reverse
arr[] = {2, 1, 3, 4, 5}
After Second Reverse
arr[] = {2, 1, 5, 4, 3}
After Third Reverse
arr[] = {3, 4, 5, 1, 2}
void leftRotate(int arr[], int n, int d)
{
reverse(arr, 0, d-1);
reverse(arr, d, n-1);
reverse(arr, 0, n-1);
}
void reverse(int arr[], int low, int high)
{
while(low < high)
{
swap(arr[low], arr[high]);
low++;
high--;
}

Leaders in an array

Thursday, 16 June 2022 12:17 PM

An element from left is greater than all element from right

Example 1:

Input:
n = 6
A[] = {16,17,4,3,5,2}
Output: 17 5 2
Explanation: The first leader is 17 as it is greater than all the elements to its right. Similarly, the next leader is 5. The right most element is always a leader so it is also included.

From <https://practice.geeksforgeeks.org/problems/leaders-in-an-array-1587115620/1/#>

{ 16 ,17 ,4 ,3 ,5 ,2 }
0 1 2 3 4 5

17 is greater from 1 to n
5 is greater from 4 to n
2 is greater from 5 to n

Method1:

```
vector<int> leaders(int a[], int n){
    vector<int> res;
    for(int i=0;i<n;i++)
    {
        int flag=0;
        for(int j=i+1;j<n;j++)
        {
            if(a[i]<=a[j])
            {
                flag=1;
                break;
            }
        }
        if(flag==0)
            res.push_back(a[i]);
    }
    return res;
```

Time compexity:O(n^2)

Method2:-

//hint traverse from right

```
 vector<int> leaders(int a[], int n){
    vector<int> res;
    res.push_back(a[n-1]);
    int max=a[n-1];
    for(int i=n-2;i>=0;i--)
    {
        if(max<=a[i])
            {
                max=a[i];
                res.push_back(a[i]);
            }
    }
    sort(res.begin(),res.end(),greater<int>());
    return res;
```

{ 16 ,17 ,4 ,3 ,5 ,2 }
0 1 2 3 4 5

Max=a[n-1]

17 is greater from 1 to n
5 is greater from 4 to n
2 is greater from 5 to n

Maximum Difference Between Increasing Elements

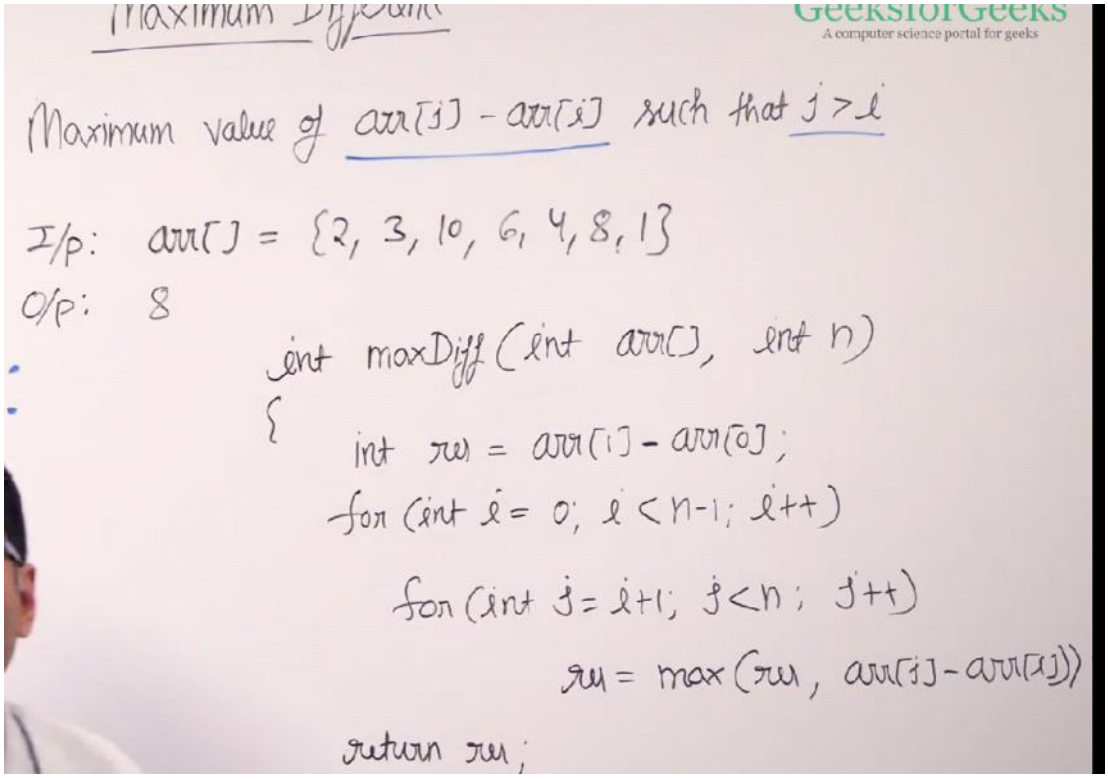
Thursday, 16 June 2022 2:52 PM

j - i >
1 - 7 = -6
5 - 1 = 4 - g
4 - 5 = -1

Input: nums = [7,**1**,**5**,4]
Output: 4
Explanation:
The maximum difference occurs with i = 1 and j = 2, nums[j] - nums[i] = 5 - 1 = 4.
Note that with i = 1 and j = 0, the difference nums[j] - nums[i] = 7 - 1 = 6, but i > j, so it is not valid.

From <https://leetcode.com/problems/maximum-difference-between-increasing-elements/>

Method 1



Method2:

```
int maximumDifference(vector<int>& nums) {
   int max=nums[1]-nums[0];
   int min=nums[0];
  for(int i=1;i<nums.size();i++)
  {

    if((nums[i]-min)>max)
          max=nums[i]-min;
    if(min>nums[i])
          min=nums[i];
  }
   return max;
```

Efficient:

arr[] = {2, 3, 10, 6, 4, 8, 1}

res = arr[1] - arr[0] = 1

~~minVal = arr[0] = 2~~

j = 1 : res = max(1, 3-2) = 1

minVal = min(2, 3) = 2

j = 2 :

res = max(1, 10-2) = 8

minVal = min(2, 10) = 2

j = 3

res = max(8, 6-2) = 8

minVal = min(2, 6) = 2

```
int maxDiff(int arr[], int n)
{
    int res = arr[1] - arr[0], minVal = arr[0];
    for (int j = 1; j<n; j++)
    {
        res = max(res, arr[j] - minVal);
        minVal = min(minVal, arr[j]);
    }
    return res;
}
```

Count number of occurrences (or frequency) in a sorted array

Thursday, 16 June 2022 4:56 PM

Input: arr[] = {1, 1, 1, 2, 3, 3, 5, 5, 8, 8, 8, 9, 9, 10}
Output: Frequency of 1 is: 3
Frequency of 2 is: 1
Frequency of 3 is: 2
Frequency of 5 is: 2
Frequency of 8 is: 3
Frequency of 9 is: 2
Frequency of 10 is: 1

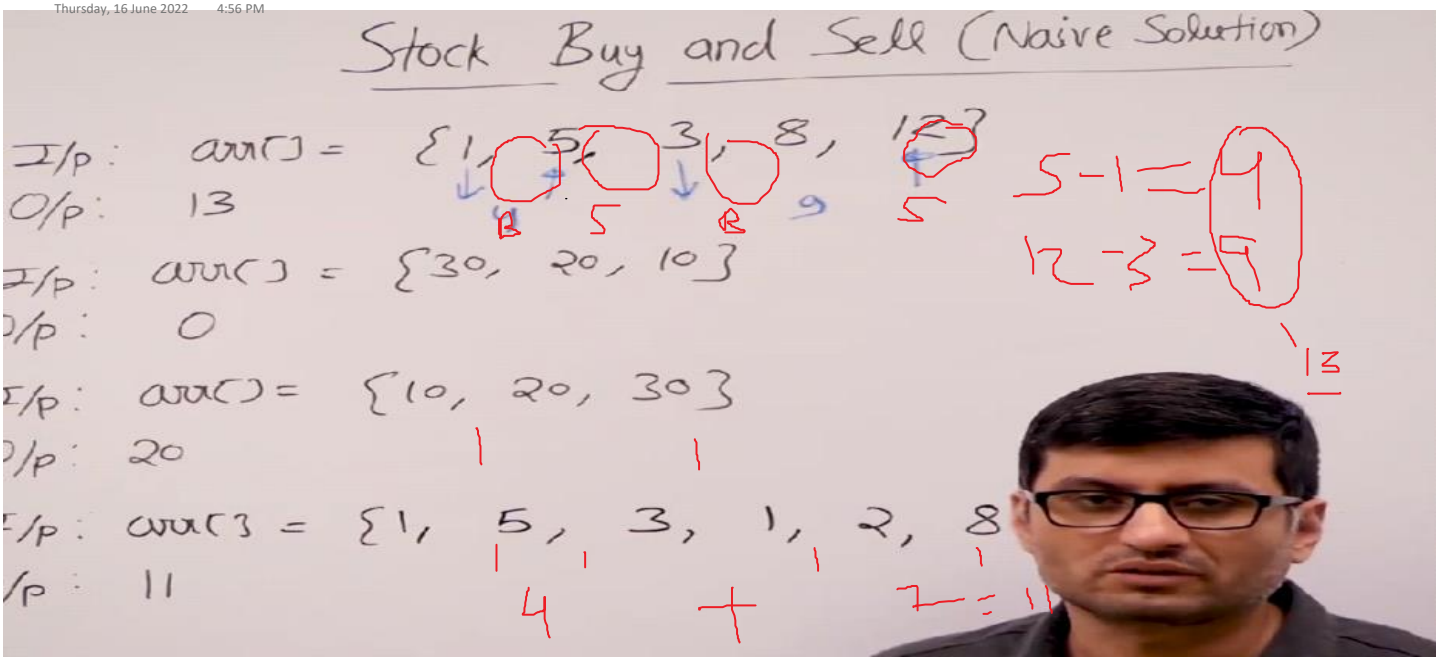
From <https://www.geeksforgeeks.org/find-the-frequency-of-each-element-in-a-sorted-array/>

```
void printFreq(vector<int> &arr, int N)
    {

        // Stores the frequency of an element
        int freq = 1;

       // Traverse the array arr[]
        for (int i = 1; i < N; i++)
        {

            // If the current element is equal
            // to the previous element
            if (arr[i] == arr[i - 1])
            {

                // Increment the freq by 1
                freq++;
            }

        // Otherwise,
            else {
                cout<<"Frequency of "<<arr[i - 1]<< " is: " << freq<<endl;
                // Update freq
                freq = 1;
            }
        }
```

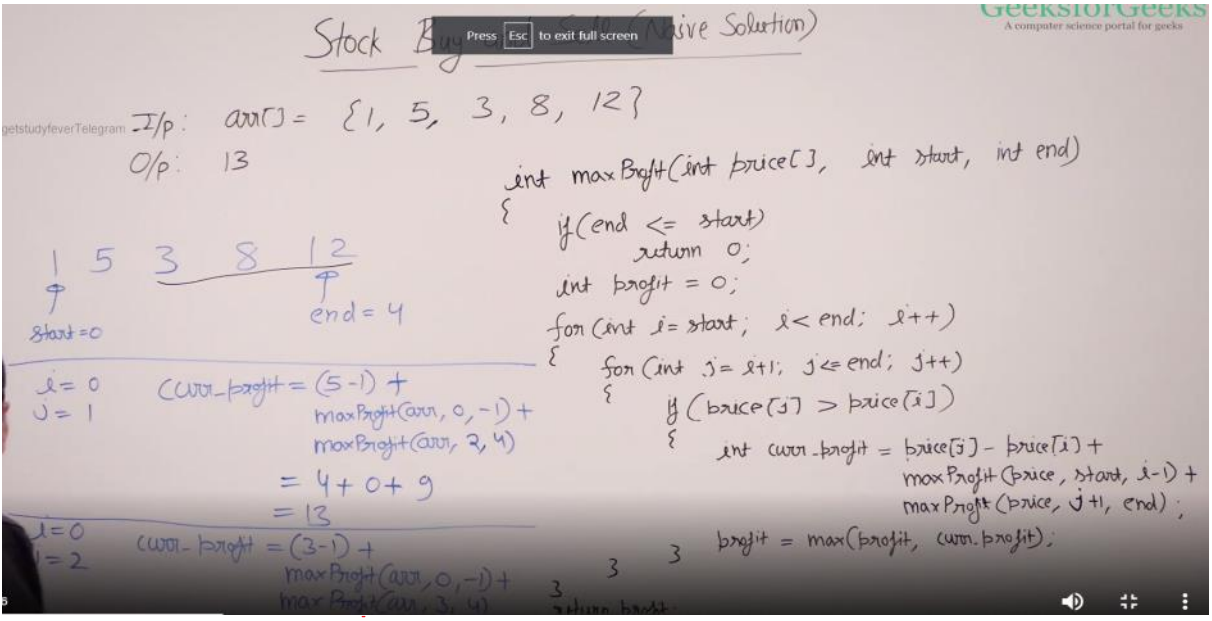
From <https://www.geeksforgeeks.org/find-the-frequency-of-each-element-in-a-sorted-array/>

Stock Buy Sell to Maximize Profit

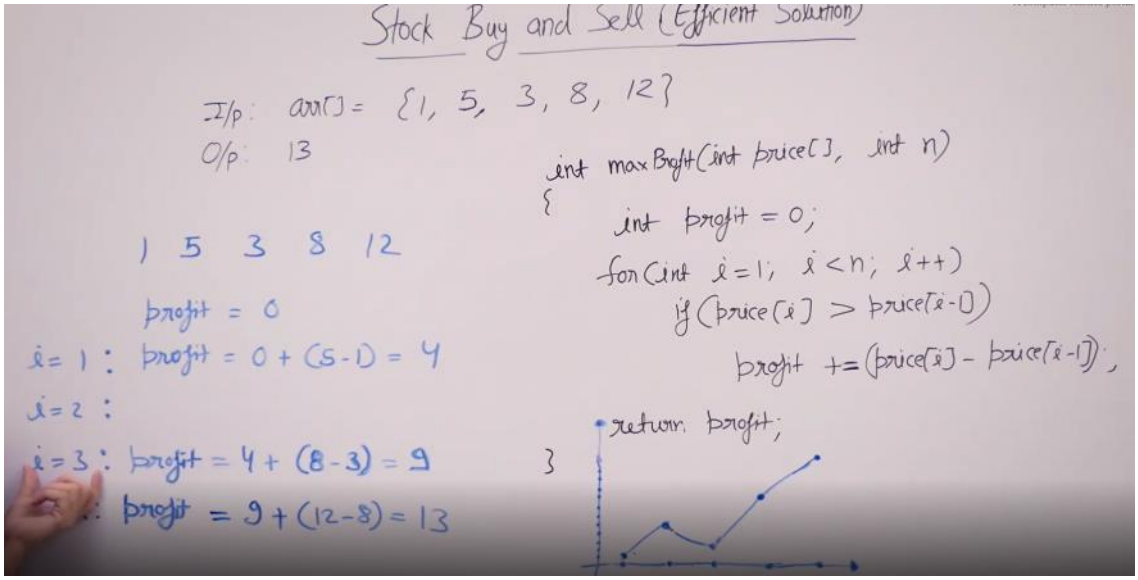
Thursday, 16 June 2022 4:56 PM



M1:-



M2:


Stock Buy and Sell (Efficient Solution)
I/p: arr[] = {1, 5, 3, 8, 12}
O/p: 13
1 5 3 8 12
profit = 0
i = 1 : profit = 0 + (5 - 1) = 4
i = 2 :
i = 3 : profit = 4 + (8 - 3) = 9
profit = 9 + (12 - 8) = 13
int maxProfit(int price[], int n)
{
int profit = 0;
for (int i = 1; i < n; i++)
if (price[i] > price[i-1])
profit += (price[i] - price[i-1]);
return profit;
}

Buy and Sell a Share at most twice

Thursday, 30 June 2022 10:33 AM

Buy and Sell a Share at most twice

From <https://practice.geeksforgeeks.org/problems/buy-and-sell-a-share-at-most-twice/1/#>

Example 1:

Input:
6
10 22 5 75 65 80
Output:
87
Explanation:
Trader earns 87 as sum of 12, 75
Buy at 10, sell at 22,
Buy at 5 and sell at 80

M1:

```
int maxProfit(vector<int>&price){
   //Write your code here..
   int n=price.size();
   vector<int>temp(n,0);
   int maxele=price[n-1];
   int maxdiff1=0;
   for(int i=n-2;i>0;i--)
   {
      maxele=max(maxele,price[i]);
      maxdiff1=max(maxdiff1,maxele-price[i]);
      temp[i]=maxdiff1;
   }
   int minele=price[0];
   int maxdiff2=0;
   int res=0;
   for(int i=1;i<n;i++)
   {
      minele=min(minele,price[i]);
      maxdiff2=max(maxdiff2,price[i]-minele);
      res=max(res,temp[i]+maxdiff2);
      temp[i]=res;
   }
   return temp[n-1];
}
```

Steps:

1. Find from last by get max diff
2. Find from beg by get max diff
3. Add max(1+2 step)

12 MAX

Prices[i]

10	22	5	75	65	80

Temp[i]

0	0	0	0	0	0

Temp1[i]

75	75	75	15	15	0

Start from last max=prices[n-1]

Temp2[i]

0	12	12	70	7 0	75

Temp[i]

00	87	87	87	87	87

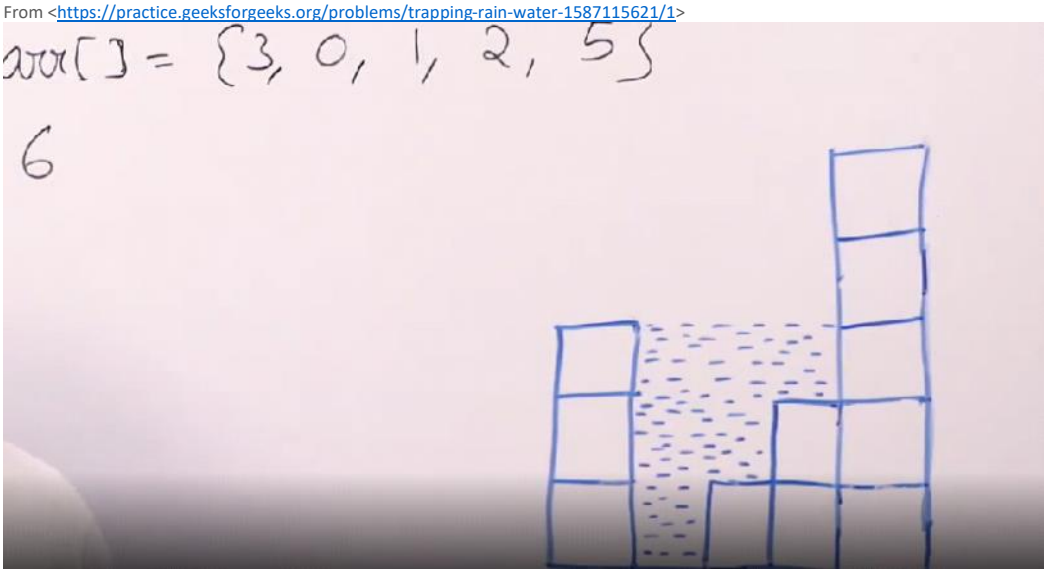
Start from beg min=prices[0]

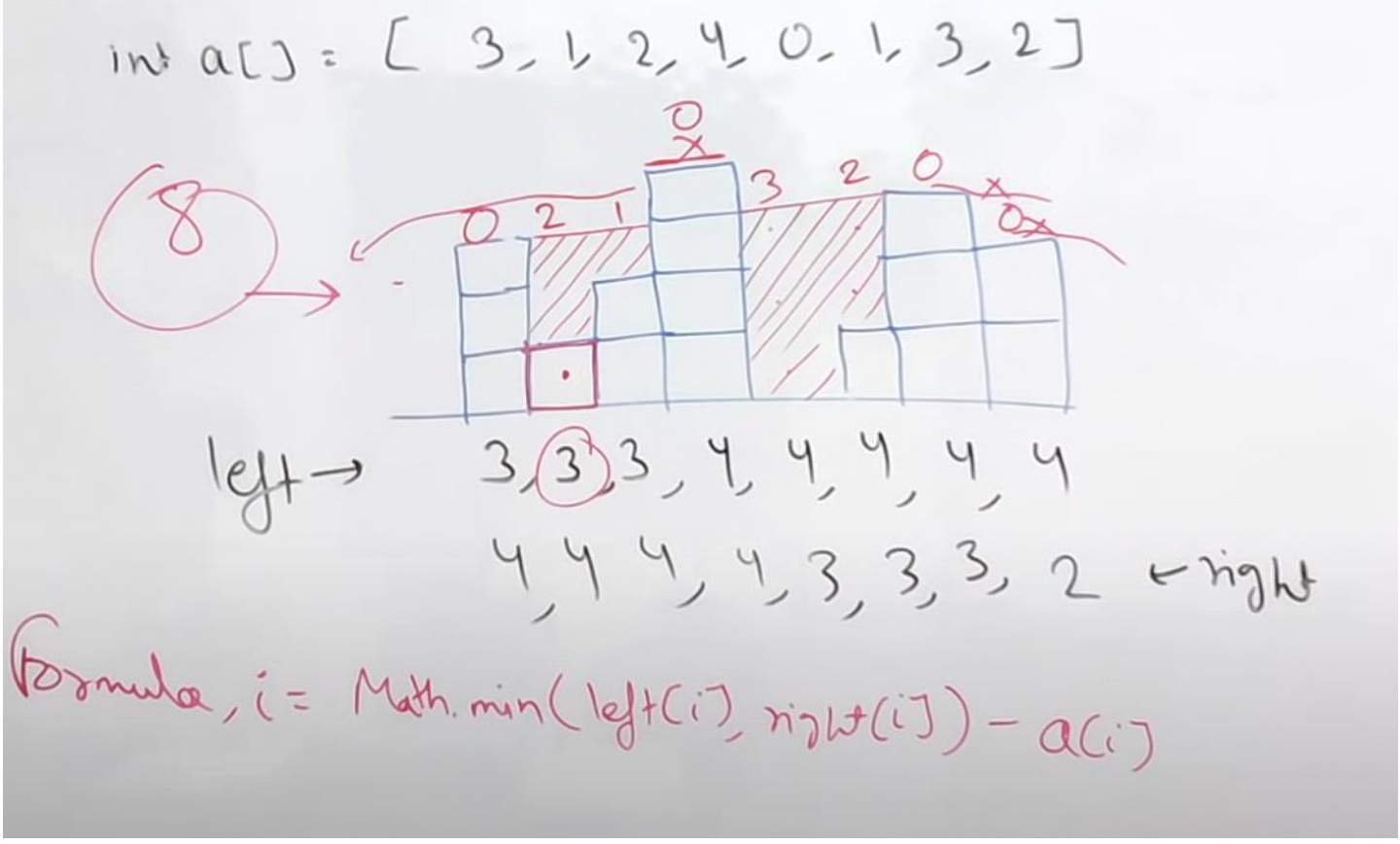
Trapping Rain Water

Friday, 17 June 2022 5:42 PM

Input:
N = 6
arr[] = {3,0,0,2,0,4}
Output:
10

From <https://practice.geeksforgeeks.org/problems/trapping-rain-water-1587115621/1>





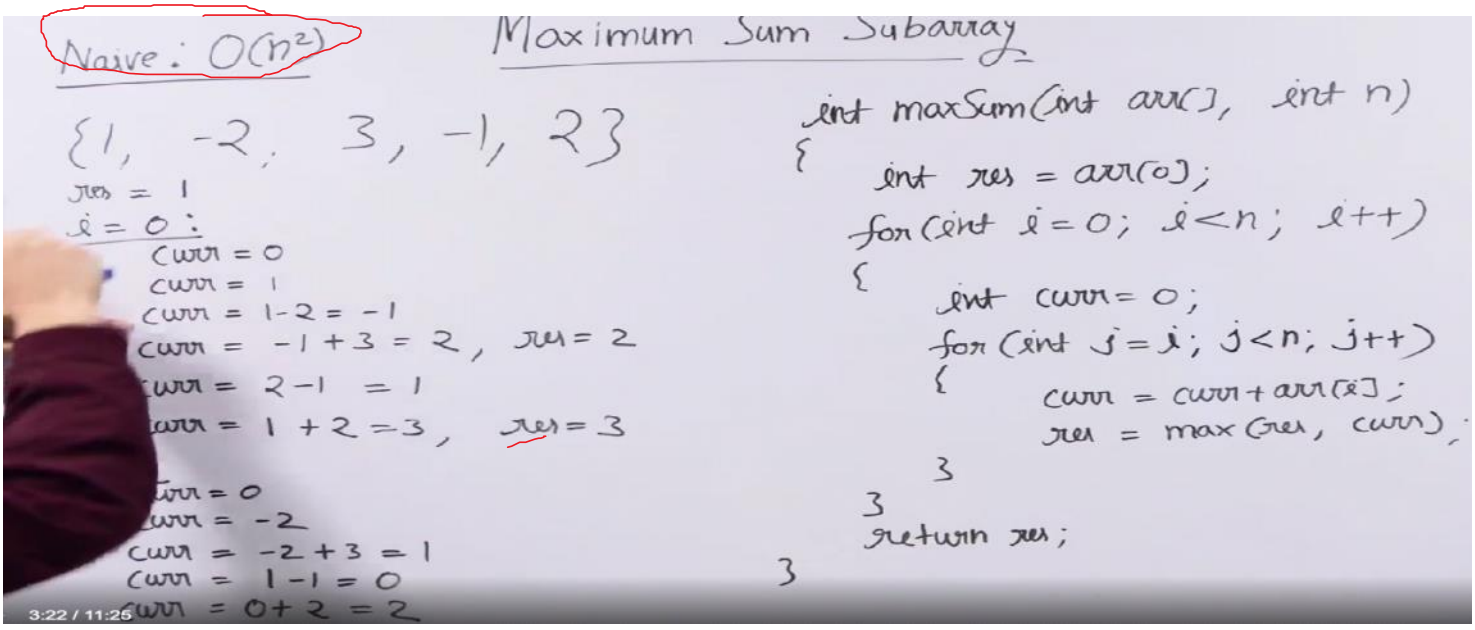

```
long long trappingWater(int arr[], int n){
    // code here
    long long res=0;
    int left[n];
```

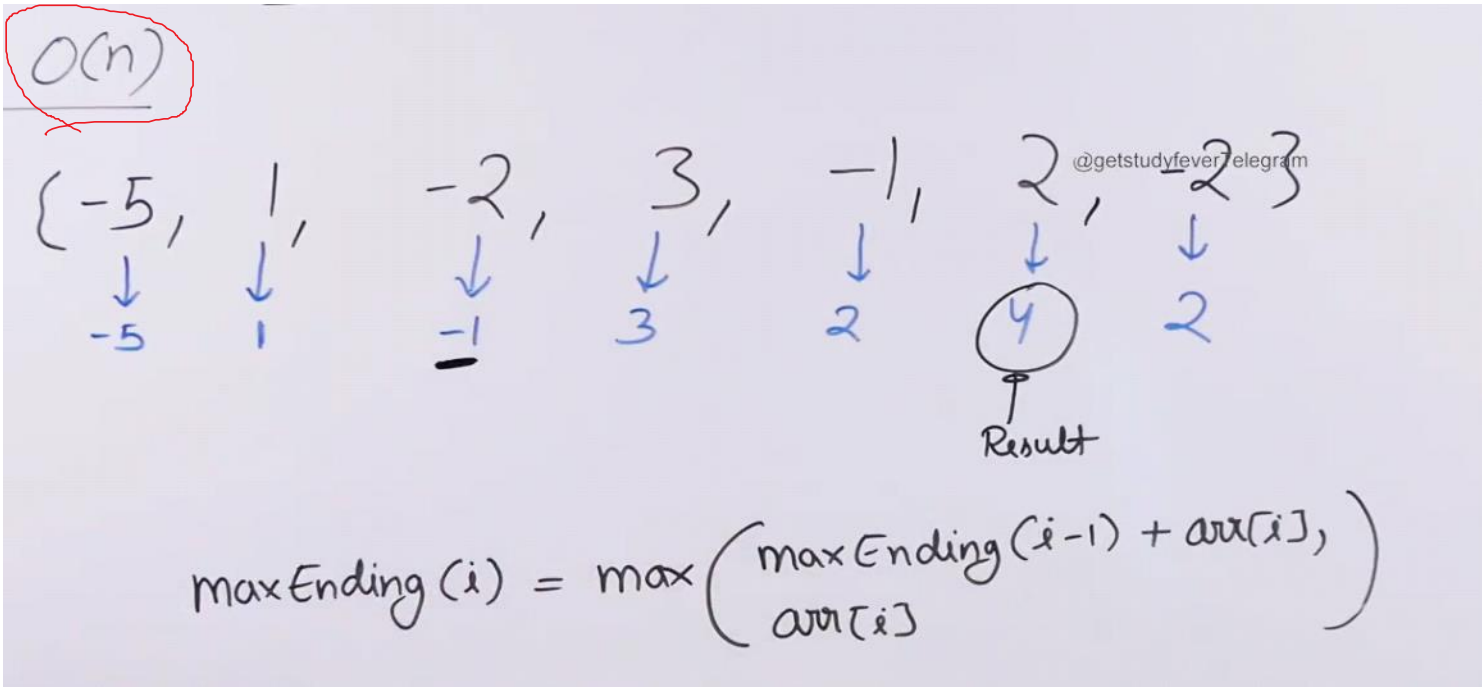
```
    int right[n];
    left[0]=arr[0];
    for(int i=1;i<n;i++){
        left[i]=max(left[i-1],arr[i]);
    }
    right[n-1]=arr[n-1];
    for(int i=n-2;i>=0;i--)
    {
        right[i]=max(right[i+1],arr[i]);
    }
    for(int i=0;i<n;i++)
    {
        res+=(min(left[i],right[i])-arr[i]);
    }
    return res;
}
```

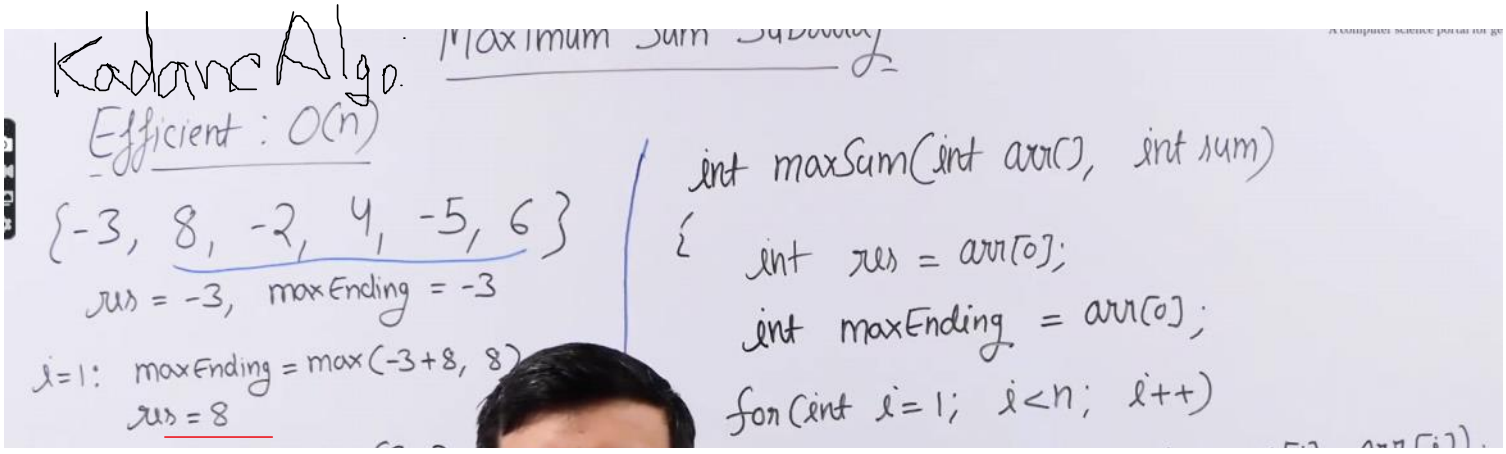
Maximum Sum Subarray=kadane Algo

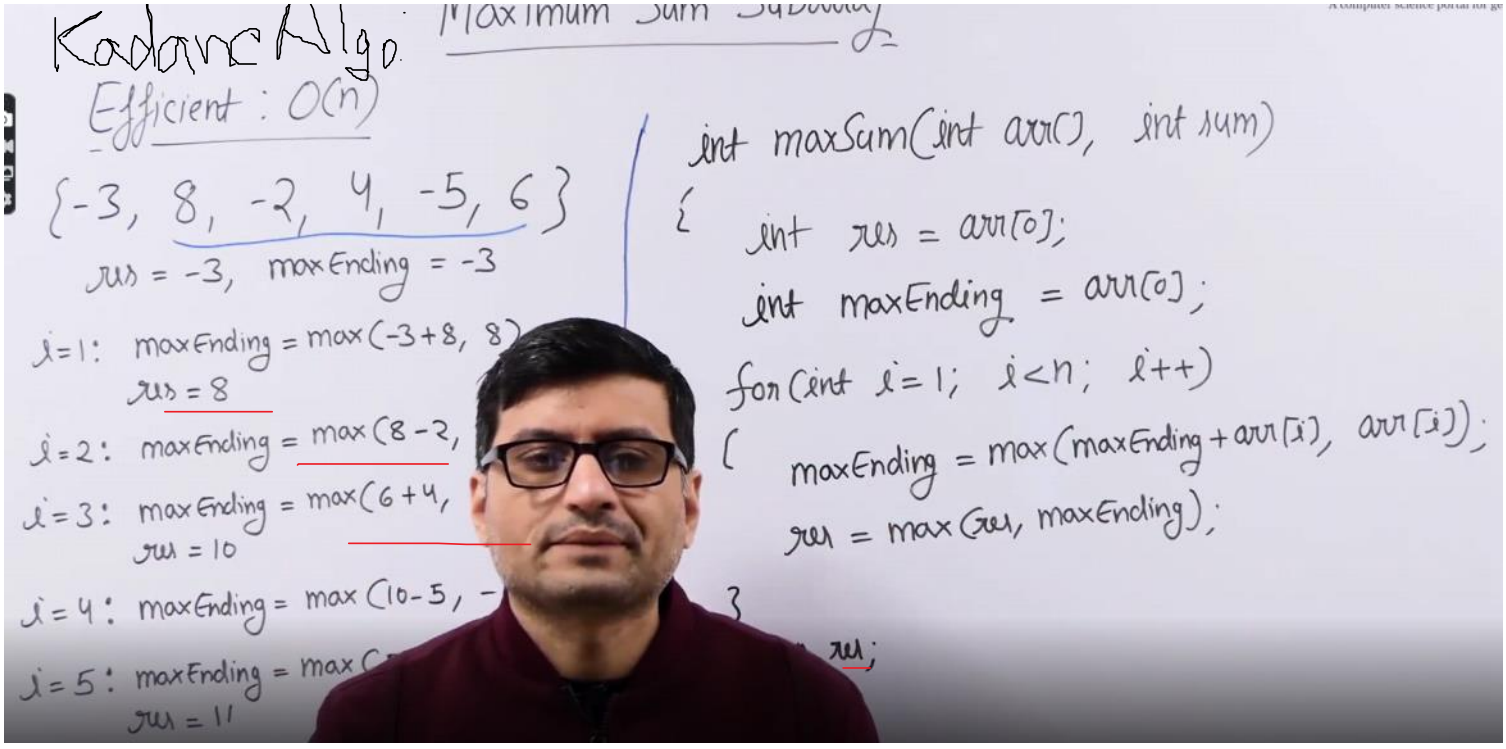
Saturday, 18 June 2022 11:54 AM

Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.








Kadane Algo
Maximum Sum Subarray
Efficient : O(n)
{-3, 8, -2, 4, -5, 6}
res = -3, maxEnding = -3
i=1: maxEnding = max(-3+8, 8)
res = 8
i=2: maxEnding = max(8-2,
i=3: maxEnding = max(6+4,
res = 10
i=4: maxEnding = max(10-5, -
i=5: maxEnding = max(
res = 11
int maxSum(int arr[], int sum)
{
int res = arr[0];
int maxEnding = arr[0];
for(int i=1; i<n; i++)
{
maxEnding = max(maxEnding + arr[i], arr[i]);
res = max(res, maxEnding);
}
res;

Length of the longest alternating even odd subarray

Sunday, 19 June 2022 7:58 AM

Input: a[] = {1, 2, 3, 4, 5, 7, 9}
Output: 5
Explanation:
The subarray {1, 2, 3, 4, 5} has alternating even and odd elements.
Input: a[] = {1, 3, 5}
Output: 0
Explanation:
There is no such alternating sequence possible.

From <https://www.geeksforgeeks.org/length-of-the-longest-alternating-even-odd-subarray/>

M1:

```
int longestEvenOddSubarray(int a[], int n)
{
    // Length of longest
    // alternating subarray
    int ans = 0;

    // Iterate in the array
    for (int i = 0; i < n ; i++) {
        int cnt = 1;
        // Iterate for every  subarray
        for (int j = i + 1; j < n; j++) {
            if ((a[j - 1] % 2 == 0 && a[j] % 2 != 0)|| (a[j - 1] % 2 != 0 && a[j] % 2 == 0))

                cnt++;
            else
                break;
        }
        // store max count
        ans = max(ans, cnt);
    }

    return ans;
}
```

M2:

From <https://www.geeksforgeeks.org/length-of-the-longest-alternating-even-odd-subarray/>

Hints :-

1. *Check for the previous element .*

```
int maxEvenOdd(int arr[], int n)
{
    if (n == 0)
        return 0;

    int maxLength = 0;
    bool prevOdd = arr[0] % 2; // stroring the nature of first element, if remainder = 1, it
is odd
    int curLength = 1;

    for (int i = 1; i < n; i++)
    {
        if (arr[i] % 2 != prevOdd) // everytime we check if previous element has opposite
even/odd nature or not
```

odd != even

```
            curLength++;
        else
            curLength = 1; // reset value when pattern is broken

        if (curLength > maxLength) // changing value  when new maximum subaaray is found
            maxLength = curLength;

        prevOdd = arr[i] % 2; // updating even/odd nature of prev number encountered
everytime
    }

    return maxLength;
}
```

From <https://www.geeksforgeeks.org/length-of-the-longest-alternating-even-odd-subarray/>

Majority Element

Tuesday, 21 June 2022 7:50 AM

Given an array **A** of **N** elements. Find the majority element in the array. A majority element in an array A of size N is an **element that appears more than N/2 times in the array**.

Example 1:

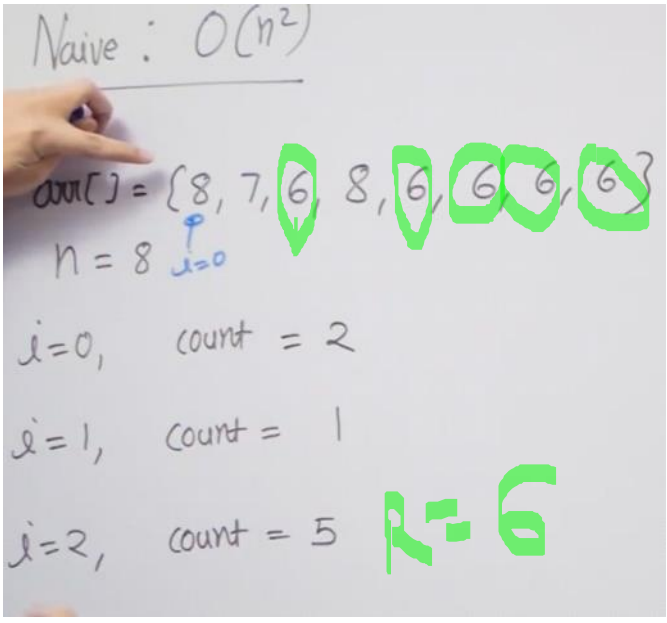
Input:
N = 3
A[] = {1,2,3}
Output:
-1
Explanation:
Since, each element in {1,2,3} appears only once so there is no majority element.

From <https://practice.geeksforgeeks.org/problems/majority-element-1587115620/1>

Input:

N = 5
A[] = {3,1,3,3,2}
Output:
3
Explanation:
Since, 3 is present more than N/2 times, so it is the majority element.

From <https://practice.geeksforgeeks.org/problems/majority-element-1587115620/1>

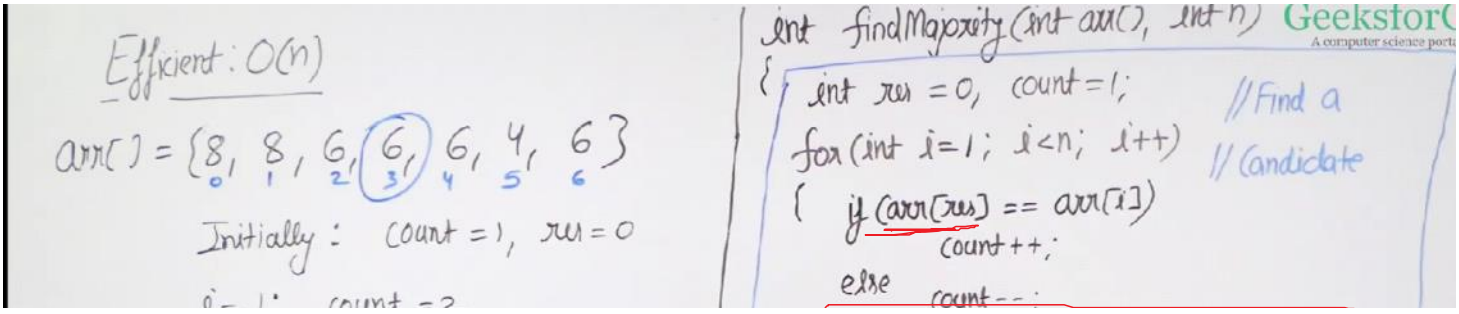


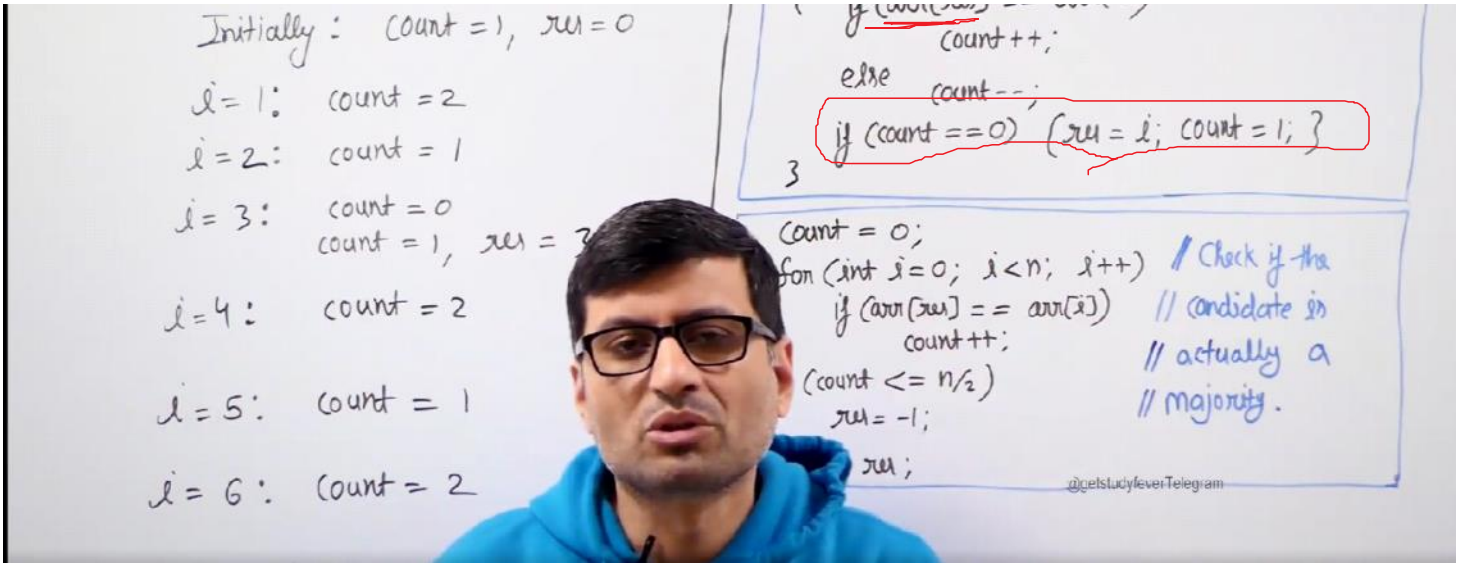
6 appear 5 times>8/2

M1:-O(n^2)

```
int majorityElement(int a[], int size)
  {
     for(int i=0;i<size;i++)
     {
        int count=1;
        int res=-1;
        for(int j=i+1;j<size;j++)
        {
           if(a[i]==a[j])
              count++;
        }
        if(count>(size/2))
        {

           res=a[i];
        }count=1;
     }
     return res;
  }
```

M2:Moore Voting Algorithim




Initially : count = 1, res = 0
i = 1: count = 2
i = 2: count = 1
i = 3: count = 0
count = 1, res = 3
i = 4: count = 2
i = 5: count = 1
i = 6: count = 2
count++;
else count--;
if (count == 0) {res = i; count = 1;}
}
count = 0;
for (int i = 0; i < n; i++)
if (arr[res] == arr[i])
count++;
(count <= n/2)
res = -1;
res;
// Check if the
// candidate is
// actually a
// majority.
@getstudyfever Telegram

Window Sliding Technique-Max Sum Subarray of size K

Tuesday, 21 June 2022 11:03 AM

Max Sum Subarray of size K

Example 1:

Input:N = 4, K = 2
Arr = [100, 200, 300, 400]
Output:700
Explanation: Arr3 + Arr4 =700,which is maximum.

From <https://practice.geeksforgeeks.org/problems/max-sum-subarray-of-size-k5313/1/#>

M1:

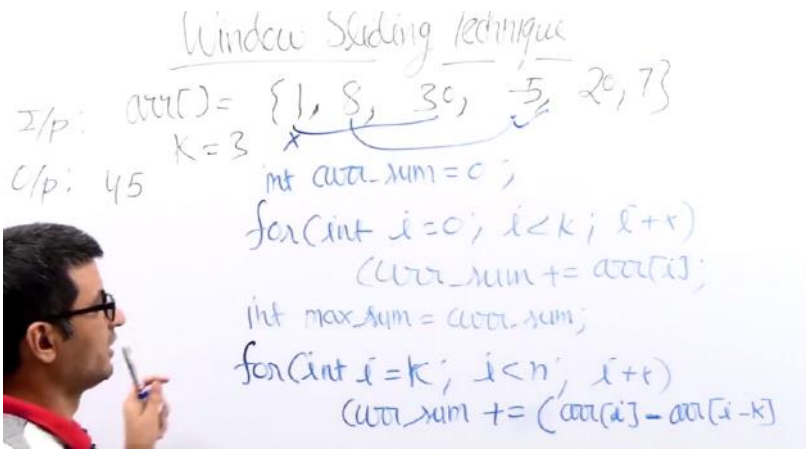
```
long maximumSumSubarray(int k, vector<int> &arr , int n){
    // code here
  int maxsum=INT_MIN;
  for(int i=0;i<n-k+1;i++)
  {
    int currsum=0;
    for(int j=0;j<k;j++)
    {
      currsum+=arr[i+j];
    }
    maxsum=max(maxsum,currsum);
  }
  return maxsum;
  }
```

Time complexity:O(n^2)

M2:

```
  long maximumSumSubarray(int k, vector<int> &arr , int n){
    // code here
  int maxsum=0;
  int currsum=0;
  if(n<k)
    return -1;
  for(int i=0;i<k;i++)
  {
    currsum+=arr[i];
  }
  maxsum=currsum;
  for(int j=k;j<n;j++)
  {
    currsum+=arr[j]-arr[j-k];
    maxsum=max(maxsum,currsum);
  }
  return maxsum;
  }
```

Time complexity:O(n)

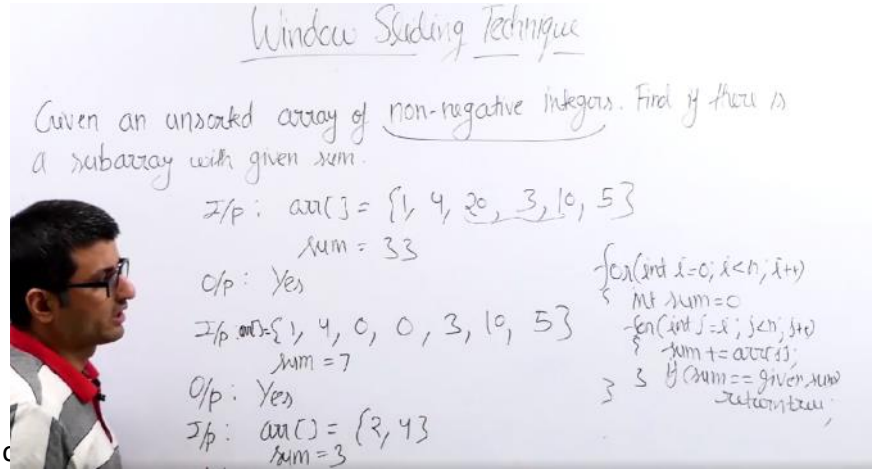


Subarray with given sum

Tuesday, 21 June 2022 11:06 AM

Given an unsorted array **A** of size **N** that contains only non-negative integers, find a continuous sub-array which adds to a given number **S**.

In case of multiple subarrays, return the subarray which comes first on moving from left to right.



Example 1:

Input:
N = 5, S = 12
A[] = {1,2,3,7,5}
Output: 2 4
Explanation: The sum of elements from 2nd position to 4th position is 12.

From <https://practice.geeksforgeeks.org/problems/subarray-with-given-sum-1587115621/1/#>

```
vector<int> subarraySum(int arr[], int n, long lo
{
   // Your code here
vector<int>res;

int start=0,currsum=0;
    for(int i=0;i<n;i++)
    {
       currsum+=arr[i];
       while(currsum>s && start<i)
       {
          currsum-=arr[start];
          start++;
       }
       if(currsum==s)
       {
          res.add(start+1);
          res.add(i+1);
          break;
       }
    }

}
if(res.empty())
   res.push_back(-1);
return res;
}
```

S=12

p= 0 1 2 4 5
A[] = {1, **2 ,3 ,7** ,5} currsum= 0,count=-1
En=0 currsum=1,count=0
En=1 currsum=3,count=1
En=2 currsum=6,count=2
En=3 currsum=13,count=3

Currsum>S (13>12)

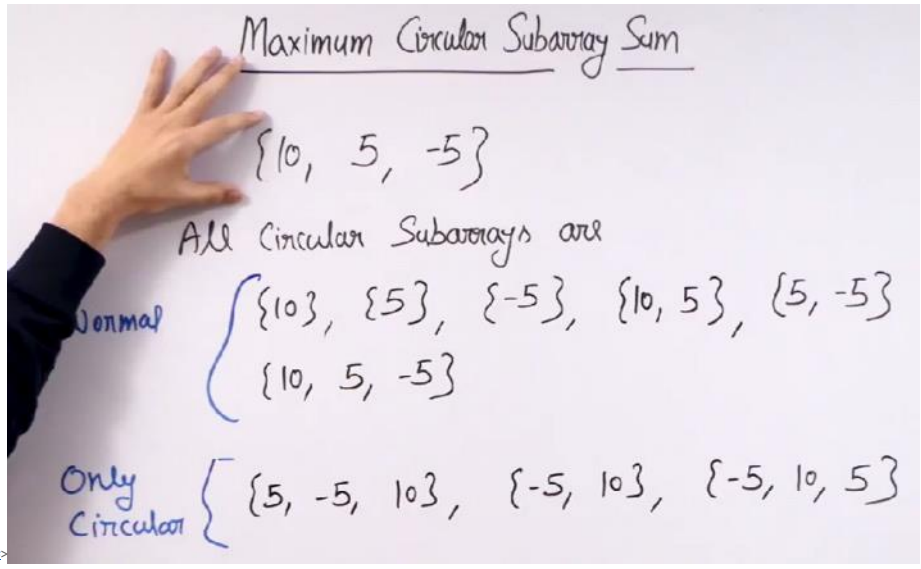
En=3 currsum=13-arr[3-3],count=2
En=3 currsum=12,count=2

Starting=en-count+1=3-2+1=2
End=en+1=4

Max Circular Subarray Sum

Monday, 20 June 2022 8:39 AM

Input:
N = 7
arr[] = {8,-8,9,-9,10,-11,12}
Output:
22
Explanation:
Starting from the last element of the array, i.e, 12, and moving in a circular fashion, we have max subarray as 12, 8, -8, 9, -9, 10, which gives maximum sum as 22.



From <https://practice.geeksforgeeks.org/problems/max-circular-subarray-sum-1587115620/1/#>

M1:

```
int circularSubarraySum(int arr[], int num){
   int res=arr[0];
   for(int i=0;i<num;i++)
   {
      int curr_sum=arr[i];
      int curr_max=arr[i];
      for(int j=1;j<num;j++)
      {
         int index=(i+j)%num;                    //index for previous
Element
         curr_sum+=arr[index];
         curr_max=max(curr_max,curr_sum);
      }
      res=max(curr_max,res);
   }
      return res;
   }
```

{8, -8, 9, -9,10 ,-11 ,12 }

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

Index=(i+j)%n

Suppose i=3
j=1to n

index
(3+1)%7=4
(3+2)%7=5
(3+3)%7=6
(3+4)%7=0
(3+5)%7=1
(3+6)%7=2

Using Kadane Algo

1.find the mininum substring array sum using kadane algo
-4,3,-5=-6
2.find sum of array= -6+12=6
3.substract sum-mininum substring=6-(-6)=12 {8,4}

Efficient Solution : O(n)

arr[] = {8, -4, 3, -5, 4}

max_normal = 8

arr_sum = 6

After inversion

arr[] = {-8, 4, -3, 5, -4}

max_circular = 6 + 6

= 12

res = max (max_normal, max_circular)

= max (8, 12)

= 12

```
int normalMaxSum(int arr[], int n)
{
    int res = arr[0], maxEnding = arr[0];
    for(int i=1; i<n; i++)
    {   maxEnding = max(arr[i], maxEnding + arr[i]);
        res = max(res, maxEnding);
    }
    return res;
}

int overallMaxSum( int arr[], int n)
{
    int max_normal = normalMaxSum(arr, n);
    if (max_normal < 0)
        return max_normal;

    int arr_sum = 0;
    for (int i=0; i<n; i++)
    {   arr_sum += arr[i];
        arr[i] = -arr[i];
    }
    int max_circular = arr_sum + normalMaxSum(arr, n);
    return max (max_normal, max_circular);
}
```

Normal Sum

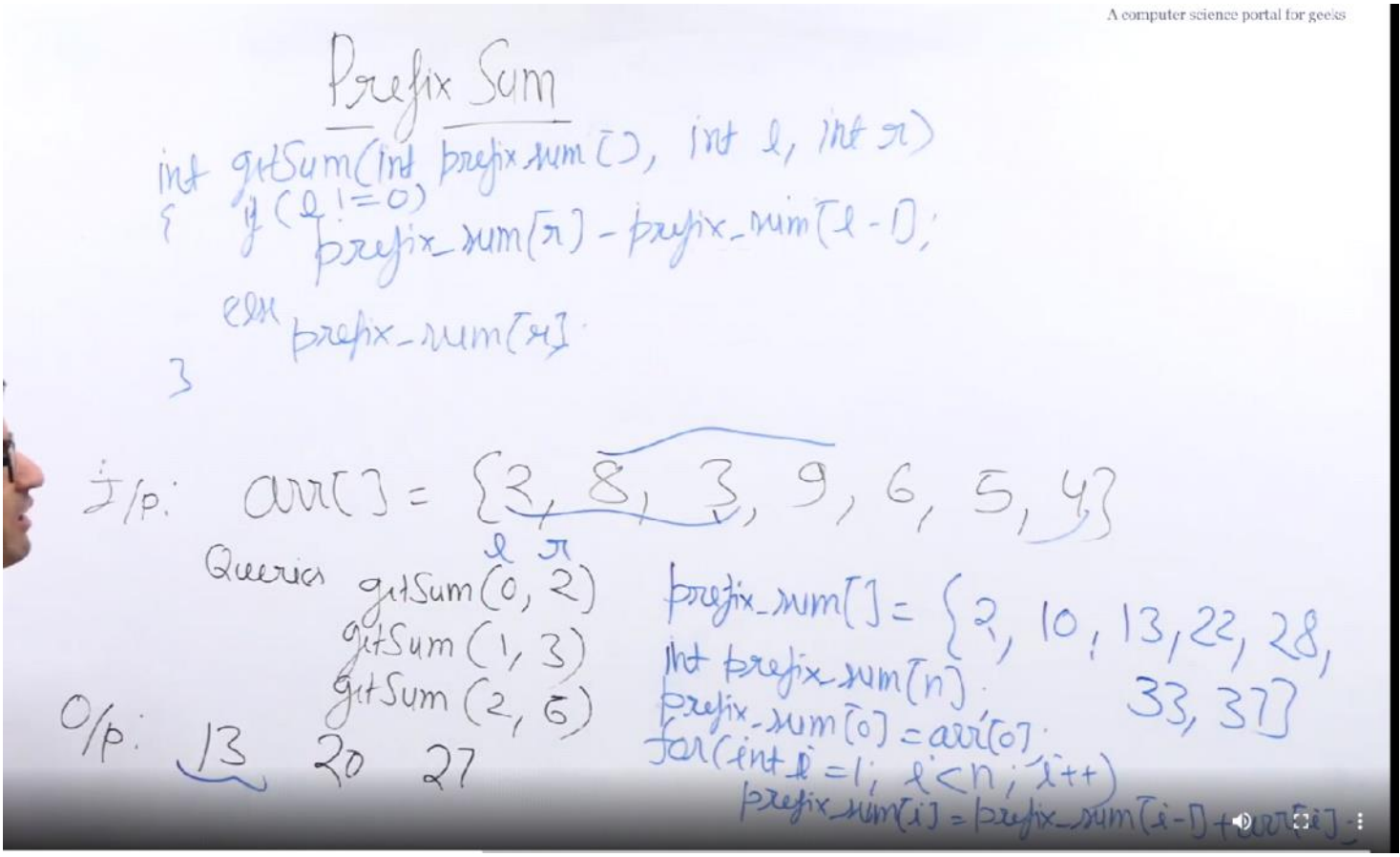
Circular Sum

Sum

Min Sub array

Prefix sum

Thursday, 23 June 2022 12:32 PM



Equilibrium Point

Wednesday, 22 June 2022 1:32 PM

Given an array A of n positive numbers. The task is to find the first Equilibium Point in the array.

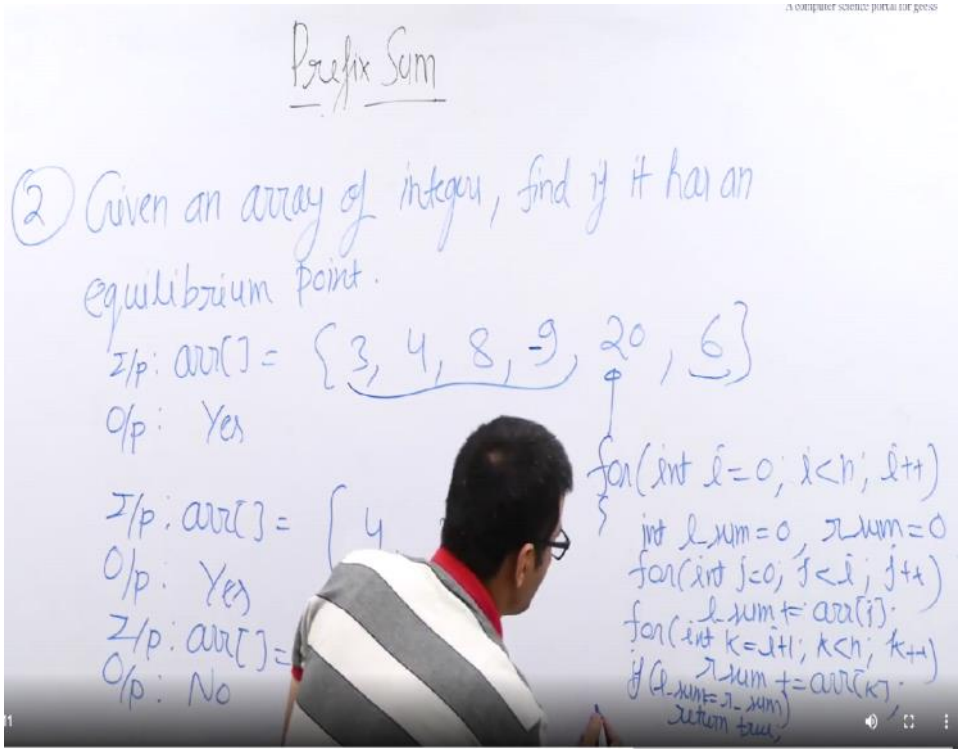
Equilibrium Point in an array is a position such that the sum of elements before it is equal to the sum of elements after it.

Note: Retun the index of Equilibrium point. (1-based index)

Example 1:

Input:
n = 5
A[] = {1,3,5,2,2}
Output: 3
Explanation: For second test case equilibrium point is at position 3 as elements before it (1+3) = elements after it (2+2).

From <https://practice.geeksforgeeks.org/problems/equilibrium-point-1587115620/1>



M2:

```
int equilibriumPoint(long long a[], int n) {

int sum=0,rsum=0,lsum=0;
for(int i=0;i<n;i++)
{
   sum+=a[i];
}
rsum=sum;
for(int i=0;i<n;i++)
{
   rsum-=(a[i]);
   if(lsum==rsum)
      return i+1;
   lsum+=a[i];
}
return -1;
}
```

Pos: 0 1 2 3 4
A[] = {1, 3, 5 ,2 , 2}
1.total sum
2.rsum=total sum;
3.lsum=0
4.for find ith pos rsum=rsum-a[i]
5then compare if(lsum==rsum)
6.lsum+=a[i]

Tsum=19
i=0 lsum=0 rsum=13-1=12
i=1 lsum=1 rsum=12-3=9
i=2 **lsum=4 rsum=9-5=4**
Return i+1=3

Sort 0,1,2-Love babbar array start

Thursday, 23 June 2022 1:12 PM

Input:
N = 5
arr[]= {0 2 1 2 0}
Output:
0 0 1 2 2
Explanation:
0s 1s and 2s are segregated
into ascending order.

[Sort an array of 0's 1's & 2's | Leetcode | C++ and Java | Brute-Better-Optimal]

Method1:

From <https://practice.geeksforgeeks.org/problems/sort-an-array-of-0s-1s-and-2s4231/1#>

```
void sort012(int a[], int n)
{
    int count0=0,count1=0,count2=0;
    for(int i=0;i<n;i++)
    {
        if(a[i]==0)
            count0++;
        else if(a[i]==1)
            count1++;
        else
            count2++;
    }
    for(int i=0;i<n;i++)
    {
        if(i<count0)
            a[i]=0;
        else if(i>=count0 && i<count0+count1)
            a[i]=1;
        else if(i>=count0+count1 && i<count0+count1+count2)
            a[i]=2;
    }
}
```

Method 2:

Dutch national flag problem.


```
void sort012(int a[], int n)
{
    int low=0,med=0,high=n-1;
    while(med<=high)
    {
     switch(a[med]){
        case 0:
            swap(a[med++],a[low--]);
             break;
        case 1:
            med++;
        case 2:
            swap(a[med],a[high--]);
            break;
       }
    }
}
```

arr[]= {0 2 1 2 0}
low=0,med=0,high=4;
Med<=high

Count pairs with given sum

Thursday, 23 June 2022 3:26 PM

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.

From <https://practice.geeksforgeeks.org/problems/count-pairs-with-given-sum5022/1#>

[Count pairs with given sum | Array | Love Babbar DSA Sheet | Amazon](#)

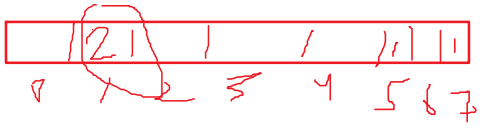
M1:

```
int getPairsCount(int arr[], int n, int k) {
    // code here
    int count=0;
    for(int i=0;i<n-1;i++)
    {
      for(int j=i+1;j<n;j++)
      {
        if(arr[i]+arr[j]==k)
          count++;
      }
    }
    return count;
  }
```

M2:-

```
int getPairsCount(int arr[], int n, int k) {
    // code here
    int ans=0,b=0;
    unordered_map<int,int> mp;
    for(int i=0;i<n;i++)
    {
      b=k-arr[i];
      if(mp[b])
        ans+=mp[b];
      mp[arr[i]]++;
    }
    return ans;
  }
```

Map[]=



arr[] = {1, 5, 7, 1} K=6

6-1=5

Subarray with 0 sum

Sunday, 26 June 2022 8:37 AM

Given an array of positive and negative numbers. Find if there is a subarray

(of size at-least one) with 0 sum.

Example 1:

Input:
5
4 2 -3 1 6
Output:
Yes

Explanation:
2, -3, 1 is the subarray
with sum 0.

[Find if there is any subarray with sum equal to 0 | Q21 | Love Babbar DSA Sheet | Leetcode | Amazon]

From <https://practice.geeksforgeeks.org/problems/subarray-with-0-sum-1587115621/1#>

```
bool subArrayExists(int arr[], int n)
  {
     unordered_map<int,int>m;
     int s=0,f=0;
     for(int i=0;i<n;i++)
     {
        s+=arr[i];
        if(s==0 or arr[i]==0 or m[s])
           {
              f=1;
              break;
           }
        else
           m[s]=1;
     }
     return f;
  }
```

Case 1: when 0 in array
Case2:Sum is zero
Case3 Sum is already present in map

Arr[]= 4 2 -3 1 6

Prefix Sum []=**4** 6 3 **4** 10

. Merge Intervals

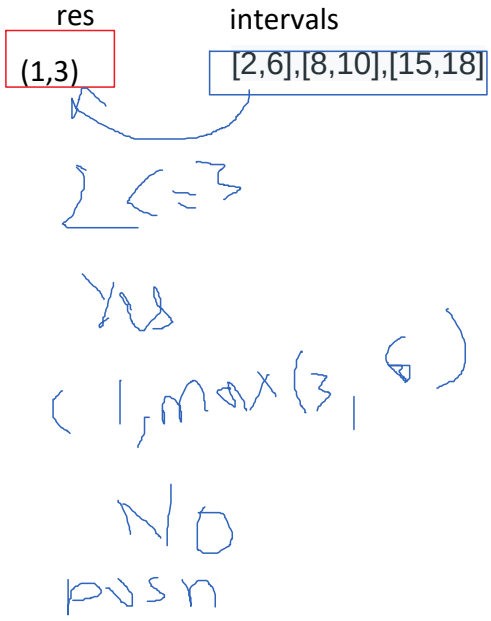
Sunday, 26 June 2022 11:10 AM

[Merge Overlapping Intervals | Q13 | Love Babbar DSA sheet | Amazon | Google]

Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input*.

Example 1:
Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].

From <https://leetcode.com/problems/merge-intervals/>




```
vector<vector<int>> merge(vector<vector<int>>& intervals) {
   sort(intervals.begin(),intervals.end());
   vector<vector<int>>res;
   res.push_back(intervals[0]);
   for(int i=1;i<intervals.size();i++)
   {
      if(intervals[i][0]<=res.back()[1])
         res.back()[1]=max(intervals[i][1],res.back()[1]);
      else
         res.push_back(intervals[i]);
   }
   return res;
   }
```

Common elements

Sunday, 26 June 2022 11:48 AM

Given three arrays sorted in increasing order. Find the elements that are common in all three arrays.

Note: can you take care of the duplicates without using any additional Data Structure?

Example 1:

Input:
n1 = 6; A = {1, 5, 10, 20, 40, 80}
n2 = 5; B = {6, 7, 20, 80, 100}
n3 = 8; C = {3, 4, 15, 20, 30, 70, 80, 120}
Output: 20 80
Explanation: 20 and 80 are the only common elements in A, B and C.

From <https://practice.geeksforgeeks.org/problems/common-elements1132/1#>

```
vector <int> commonElements (int A[], int B[], int C[], int n1, int n2, int n3)
{
vector<int>res;
unordered_map<int,int>m1;
unordered_map<int,int>m2;
unordered_map<int,int>m3;
for(int i=0 ;i<n1;i++)
{
   m1[A[i]]++;
}
for(int i=0 ;i<n2;i++)
{
   m2[B[i]]++;
}
for(int i=0 ;i<n3;i++)
{
```

```
    m3[C[i]]++;
}
for(int i=0 ;i<n1;i++)
{
    if(m1[A[i]] and m2[A[i]] and m3[A[i]])
    {
        res.push_back(A[i]);
        m1[A[i]]=0;
    }
}
return res;
}
```

Maximum Product Subarray

Sunday, 26 June 2022 5:04 PM

Given an array **Arr[]** that contains **N** integers (may be **positive**, **negative** or **zero**). Find the product of the maximum product subarray.

Example 1:

Input:
N = 5
Arr[] = {6, -3, -10, 0, 2}
Output: 180
Explanation: Subarray with maximum product is [6, -3, -10] which gives product as 180

From <https://practice.geeksforgeeks.org/problems/maximum-product-subarray3604/1#>

```
long long maxProduct(vector<int> arr, int n) {
    long long max_end=1,min_end=1;
    long long max_so_far=INT_MIN;

    for(int i=0;i<n;i++)
    {
      if(arr[i]>0)
      {
        max_end=max((long long)
    arr[i],max_end*arr[i]);
        min_end=min((long long)arr[i],min_end*arr[i]);

      }
      else if(arr[i]==0)
      {
        max_end=1;
        min_end=1;
      }
```

```
    else if(arr[i]<0)
    {
        long long temp=max_end;
        max_end=max((long long)
  arr[i],min_end*arr[i]);
        min_end=min((long long)arr[i],arr[i]*temp);

    }
    max_so_far=max(max_so_far,max_end);
  }

return max_so_far;


  }
```

Longest consecutive subsequence

Tuesday, 28 June 2022 9:40 PM

Given an array of positive integers. Find the length of the longest sub-sequence such that elements in the subsequence are consecutive integers, the **consecutive numbers can be in any order.**

Example 1:

Input:
N = 7
a[] = {2,6,1,9,4,5,3}
Output:
6
Explanation:
The consecutive numbers here are 1, 2, 3, 4, 5, 6. These 6 numbers form the longest consecutive subsquence.

a[] = {2,6,1,9,4,5,3}
1.Sort=[1,2,3,4,5,6,9
2.Remove duplicate element
3.then check for previous element
For consecutive if(res[i+1]==res[i]+1)

From <https://practice.geeksforgeeks.org/problems/longest-consecutive-subsequence2449/1#>

M1:

```
int findLongestConseqSubseq(int arr[], int n)
 {
   sort(arr,arr+n);
   vector<int>res;
   res.push_back(arr[0]);
   for(int i=1;i<n;i++)
   {
      if(arr[i-1]!=arr[i])
      {
        res.push_back(arr[i]);

      }
   }
   int count=1,maxcount=1,m=res.size();
   for(int i=0;i<n;i++)
   {
     if(res[i+1]==res[i]+1)
        count++;
     else
        count=1;
     maxcount=max(maxcount,count);
   }
   return maxcount;

 }
```

Remove duplicate element

check for previous element
For consecutive

M2:

Triplet Sum in Array

Wednesday, 29 June 2022 7:57 AM

Given an array arr of size n and an integer X. Find if there's a triplet in the array which sums up to the given integer X.

Example 1:

Input:
n = 6, X = 13
arr[] = [1 4 45 6 10 8]
Output:
1
Explanation:
The triplet {1, 4, 8} in
the array sums up to 13.

From <https://practice.geeksforgeeks.org/problems/triplet-sum-in-array-1587115621/1>

```
bool find3Numbers(int A[], int n, int x)
  {
    //Your Code Here
  int l,r;
  sort(A,A+n);
  for(int i=0;i<n-2;i++)
  {
    l=i+1;
    r=n-1;
    while(l<r)
    {
      if(A[i]+A[l]+A[r]==x)
          return true;
      else if(A[i]+A[l]+A[r]>x)
        r--;
      else if(A[i]+A[l]+A[r]<x)
          l++;

    }
  }
  return false;
```

example

```
arr[] = [1 4 45 6 10 8]
Sort=[1  4  6  8  10  45]
        i   i+1          r
if(A[i]+A[l]+A[r]==x)
            return true;
else if(A[i]+A[l]+A[r]>x)
         r--
else if(A[i]+A[l]+A[r]<x)
         l++
Sort=[1  4  6  8  10  45]
       i   i+1     r
```

Smallest subarray with sum greater than x

Wednesday, 29 June 2022 10:59 AM

Given an array of integers (A[]) and a number x, find the smallest subarray with sum greater than the given value.

From <https://practice.geeksforgeeks.org/problems/smallest-subarray-with-sum-greater-than-x5651/1#>

Example 1:

Input:
A[] = {1, 4, 45, 6, 0, 19}
x = 51
Output: 3
Explanation:
Minimum length subarray is
{4, 45, 6}

Two pointer approach

```
int smallestSubWithSum(int arr[], int n, int x)
  {
     int sum=0;
     int count=INT_MAX,j=0;
    for(int i=0;i<n;i++)
    {
       sum+=arr[i];
       while(sum>x)
       {
          count=min(count,i-j+1);

          sum-=arr[j++];

       }
    }
    return count;
  }
```

Arr[]= {1, 4, 45, 6, 0, 19}
1.calculate sum until sum>X
2.Remove the element fromj=0
And increase j++
3.calculate length of sub array
Length=i-j+1

```
Arr[]= {1,  4,   45, 6,  0,  19}
        0  1    2   3   4   5
        ij
Arr[]= {1,  4,   45, 6,  0,  19}
        0  1    2   3   4   5
        I            j
```

Sum=56>51
Remove from j=0

Sum=55>51 True
Len=i-j+1=3-1+1=3

Minimum swaps and K together

Wednesday, 29 June 2022 11:25 AM

Given an array **arr** of **n** positive integers and a number **k**. One can apply a swap operation on the array any number of times, i.e choose any two index i and j (i < j) and swap arr[i] , arr[j] . Find the **minimum** number of swaps required to bring all the numbers less than or equal to **k** together, i.e. make them a contiguous subarray.

Input :
arr[ ] = {2, 1, 5, 6, 3}
K = 3
Output :
1
Explanation:
To bring elements 2, 1, 3 together,swap index 2 with 4 (0-based indexing),i.e. element arr[2] = 5 with arr[4] = 3 such that final array will be- arr[] = {2, 1, 3, 6, 5}

From <https://practice.geeksforgeeks.org/problems/minimum-swaps-required-to-bring-all-elements-less-than-or-equal-to-k-together4847/1>

```
int minSwap(int *arr, int n, int k) {

    // Find count of elements which are
    // less than equals to k
    int count = 0;
    for (int i = 0; i < n; ++i)
        if (arr[i] <= k)
            ++count;

    // Find unwanted elements in current
    // window of size 'count'
    int bad = 0;
    for (int i = 0; i < count; ++i)
        if (arr[i] > k)
            ++bad;

    // Initialize answer with 'bad' value of
    // current window
    int ans = bad;
    for (int i = 0, j = count; j < n; ++i, ++j)
{

        // Decrement count of previous window
        if (arr[i] > k)
            --bad;

        // Increment count of current window
        if (arr[j] > k)
            ++bad;

        // Update ans if count of 'bad'
        // is less in current window
        ans = min(ans, bad);
    }
    return ans;
}
```

>

Hints:-

1.find total number of element<k
2.find no. of greater until count
3.using slinding window technique
Increment bad when arr[j]>k
decrement bad when arr[i]>k

{2, 1, 5, 6, 3} k=3
Total element less than k is 1 2 3
{2, 1, 5,}
Bad element=1 because 5>3
Bad=1
Then check a[0]>3
Decrement bad--;
a[count]>3
Increment bad++;
Min(bad,ans)

Next Permutation

Thursday, 30 June 2022 3:58 PM

A **permutation** of an array of integers is an arrangement of its members into a sequence or linear order.

- For example, for arr = [1,2,3], the following are considered permutations of arr: [1,2,3], [1,3,2], [3,1,2], [2,3,1].

The replacement must be **in place** and use only constant extra memory.

Example 1:
Input: nums = [1,2,3]
Output: [1,3,2]
Example 2:
Input: nums = [3,2,1]
Output: [1,2,3]

Next Permutation 🔥🔥 | Array | Love Babbar DSA Sheet | Leetcode 31 | Amazon | Google

From <https://leetcode.com/problems/next-permutation/>

```
void nextPermutation(vector<int>& nums) {
    if(nums.size()==1)
        return;
    int n=nums.size();
    int idx1;
    for(int i=n-2;i>=0;i--)
    {
        if(nums[i]<nums[i+1])
        {
            idx1=i;
            break;
        }
    }
    int idx2;
    if(idx1<0)
        reverse(nums.begin(),nums.end());
    else
    {
        for(int i=n-1;i>=0;i--)
        {
            if(nums[i]>nums[idx1])
            {
                idx2=i;
                break;
            }
        }
    }
    swap(nums[idx1],nums[idx2]);
    reverse(nums.begin()+idx1+1,nums.end());
}
```