Must do **Array** questions

Topic:	Problem:
Array	Reverse the array
Array	Find the maximum and minimum element in an array
Array	Find the "Kth" max and min element of an array
Array	Given an array which consists of only 0, 1 and 2. Sort the array without usi
Array	Move all the negative elements to one side of the array
Array	Find the Union and Intersection of the two sorted arrays.
Array	Write a program to cyclically rotate an array by one.
Array	find Largest sum contiguous Subarray [V. IMP]
Array	Minimise the maximum difference between heights [V.IMP]
Array	Minimum no. of Jumps to reach end of an array
Array	find duplicate in an array of N+1 Integers
Array	Merge 2 sorted arrays without using Extra space.
Array	Kadane's Algo [V.V.V.V.V IMP]
Array	Merge Intervals
Array	Next Permutation
Array	Count Inversion
Array	Best time to buy and Sell stock
Array	find all pairs on integer array whose sum is equal to given number
Array	find common elements In 3 sorted arrays
Array	Rearrange the array in alternating positive and negative items with O(1) e
Array	Find if there is any subarray with sum equal to 0
Array	Find factorial of a large number
Array	find maximum product subarray
Array	Find longest coinsecutive subsequence
Array	Given an array of size n and a number k, fin all elements that appear more
Array	Maximum profit by buying and selling a share atmost twice
Array	Find whether an array is a subset of another array
Array	Find the triplet that sum to a given value

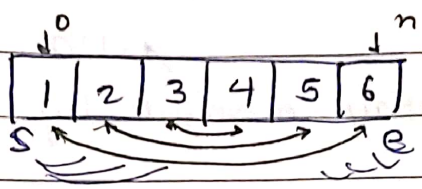
Array	Trapping Rain water problem
Array	Chocolate Distribution problem
Array	Smallest Subarray with sum greater than a given value
Array	Three way partitioning of an array around a given value
Array	Minimum swaps required bring elements less equal K together
Array	Minimum no. of operations required to make an array palindrome
Array	Median of 2 sorted arrays of equal size
Array	Median of 2 sorted arrays of different size

Page No.:
Date: 26/1/2022

Ques - Reverse the array (G4G)

| 1 | 2 | 3 | → | 3 | 2 | 1 |

Method :-



1) In a loop start with 0 & n-1
2) Swap arr[start] with arr[end].
start = start + 1 ; end = end = end - 1.

code :-

```
void reversearray(int arr[], int s, int e)
{
    while (s < end)
    {
        int temp = arr[s];
        arr[s] = arr[e];
        arr[e] = temp;
        start ++; end
        s++ ; e--;
    }
}
```

swapping

→ int main() { }

Print → for(int i=0; i<size; i++)

```
{
    cout<<" arr[i] <<" . ";
    cout << endl;
}
}
```

Ques → Find max. & min. of array using minimum no. of comparision (or 4or)

Soln →

```
struct pair {          // to return multiple
    int min;           // value, we use struct.
    int max; };

pair getMinMax (int arr[], int n) {
    struct pair minmax;   // object of struct
    int i;
if (n==1)   // if 1 element, return it as min & max.
    { minmax.max = arr[0];
      minmax.min = arr[0];
      return minmax; }
if (arr[0] > arr[1])   // if more than 1 elem,
    { minmax.max = arr[0];   // initialize min
      minmax.min = arr[1];   // & max.
    }
if else { minmax.max = arr[1];
          minmax.min = arr[0]; }

for (i=2; i<n; i++) {
    if (arr[i] > minmax)
        minmax.max = arr[i];
    else if (arr[i] < minmax.min)
        minmax.min = arr[i];
    }
    return minmax;
}

int main()
{ ---
  ---.
}
```

Page No.:
Date:

Ques: K^{th} smallest element (G4G)

arr[] given, K^{th} smallest element in arr[] find.

arr = | 7 | 10 | 4 | 3 | 20 | 15 | K = 3 (3^{rd} smallest)

O/P = 7

Code :-

2) Method → Broot F ; Min/Max heap (optimal)

Broot force.
→ sort array
print (k+1).
TC - (n log n) X

optimal :-
using min heap

code (optimal) :-

```
priority_queue<int> pq;
for(int i=0; i<n+1; i++)
{
    pq.push(arr[i]); //push ele. in prio. queue
    if (pq.size() > k)
    {
        pq.pop();
    }
}
return pq.top();
}
```

func^n

Page No.:
Date:
YOUVA

Ques- Sort an array 0s, 1s, 2s (Gr 4 Gr)

arr[] e contain 0s, 1s, 2s & arrange in ascending order.

| 0 | 2 | 1 | 2 | 0 | → | 0 | 0 | 1 | 2 | 2 |

Optimal Soln - (Dutch National flag Algo) ONF

| 0 | 2 | 1 | 2 | 0 |

l m h

→ while (m <= h)

→ [m == 0] → swap l & m
l++, m++

[m == 1] → m++; [m == 2] → swap h & m;
h--;

Code :-

```
int l=0; int m=0; int h= n-1;

while (m <= h) {
    int x = arr[m];
    if (x == 0) {
        swap(arr[l], arr[m];
        l++; m++; }
    else if ( x == 1)
        { m++; }

    else {
        swap(arr[h], arr[m]);
        h--;
    }
}
```

l - l के पिछे सारे 0s हैं।
h - h के बाद सारे 2 हैं।

***

Ques- Move all the element negative to one side of an array

arr = -12, -11, 10, 9, 8, -7

O/P => -12, -11, -7, 8, 9, 10

order doesn't matter

Code - (partion process of quick sort)

```
void rearrange (int arr[], int n)
{
    int j = 0;
    for (int i=0; i<n ; i++) {
        if (arr[i] < 0) {
            if (i != j)
                swap (arr[i], arr[j]);
            j++;
        } // if end
    } // for loop end
} // func^n end.
```

TC - O(n)

SC - O(1)

Method -2 -> 2 pointer approc.

Page No.:
Date:
YOUVA

Ques → Union of two array

2 array a[] & b[],

a = 1 2 3 4 5 b = 1 2 3.

O/P = | 1 | 2 | 3 | 4 | 5 |.

we'll use **set library**.

★ set → returns unique element (sorted desc.)
★ unordered_set - returns randomly
★ initialization - set<data-type> set_name;

code :-

```
{   unordered_set<int> s;
    for(int i=0; i<n; i++)
    {
        s.insert(a[i]);
    }
    for(int i=0; i<n; i++)
    {
        s.insert(b[i]);
    }
    return s.size();
}
```

Page No.:
Date:
YOUVA

Ques- Cyclically rotate array by one (-or+or)

arr = {9, 8, 7, 6, 5, 4, 3}

=> 3, 9, 8, 7, 6, 5, 4

code :-

```
int temp= arr[n-1]; // Last element
for(int i= n; i>=1; i--)
{
    arr[i] = arr[i-1];
}
arr[0] = temp;
```

for between 0 to n-1

Last element changed with 1st

Page No.:
Date: 27/1/2022
YOUVA

Ques- Kadon's Algorithm

Maximum sum subarray return (G4G)

arr[] = {1, -2, 3, -2, 5}
drop drop

O/P = 9

2 solution optimal & brute force

Optimal :-

```
int MaxSumSubArr(int a[], int n){
    int maxsum = 0;
    int curr = 0;
for (int i=0; i<n; i++){
        curr = curr + a[i];
        if (curr > maxsum)
            maxsum = curr;
        else if (curr < 0)
            curr = 0; }

    return maxsum; }
```

// Now above soln works but not when all ele. will b -ve eg. {-1, -4, -6, -2}

Soln 2

```
int MaxSum(int a[], int n){
    int maxsum = a[0]; int curr = a[0];

for (int i=0; i<n; i++){
    curr = max(arr[i], curr + a[i]);
    maxsum = max(maxsum, curr);
}
return maxsum;
}
```

// This will work fine for all cases

Page No.:
Date:

Ques:- Minimize the height (Gr 4 Gr)

arr[] = given of N tower, array represent height
K is given, we can use or use hight by K
so that we get min. diff. b/w 2 element

Eg - N=5 | 3 | 9 | 16 | 12 | 20 | => K = 3

Let say I 1 | D | D | D => Diff = 17 - 6 => 11

Code :-

```
int GetMinDiff (int arr[], int n, int k) {

    sort(arr, arr+n); //so that we know the diff.
                      //between each element.
    int ans = arr[n-1] - arr[0]; //initial ans.
    int smallest = arr[0]+k;
    int largest = arr[n-1]-k;
    int mi, ma; // current minimum & max.

    for (int i=0; i<n-1; i++) {
if(arr[i]>=k) { mi = min(smallest, arr[i]-k);
        ma = max(largest, arr[i]+k);
        if (mi<0) continue; // if above apply this do.

        ans = ma-mi; min(ans, ma-mi);
    }}

    return ans;
}
```

Ques

Ex 1 -

Ex 2 -

DP question

Page No.: Date: YOUVA

Ques Min. no. of jumps to reach end (Gr 4 Gr)
Jump Game (Leetcode)

arr[] → given each element say no. of jumps possible. We've to reach to the end index. Return True if reach otherwise F.

Ex 1- num = [2, 3, 1, 1, 4] → If 2 → 3 → 1 → 4 ⇒ 2 → 3 → 4 → True
True
3 jump

Ex 2- num = [1, 2, 1, 0, 3]
False
1 → 2 → 1 → 0 ⇒ 1 → 2 → 1 → 0 → False
OR
1 → 2 → 0 → False

Code :-

We'll solve this using DP.

```
bool canJump(vector<int>& arr)
{
    int n = arr.size();
    int reachable = 0;
    for(int i=0; i<n; i++) {
        if(reachable < i)
            return False;
        reachable = max(reachable, i+arr[i]);
    }
    return true;
}
;
// understand this more when we'll study
// Dynamic Programming.
```

From 180 series

Page No.:
Date:

Ques - Find the duplicate number (leetcode)

nums → given of range [1, n]
→ there is 1 repeat no., return this no.
→ use only const. extra space.

Ex - [1, 3, 4, 2, 2] ⇒ O/P - [2] as 2 is duplicate

[3, 1, 3, 4, 2] ⇒ O/P - [3] → repeat 3 times

Mutiple soln available :-
optimal - Tortoise & Hare Algo ; Brute force

Steps (1) take 2 pointers fast & slow,
initially both will point to 1st element.
(2) Use slow by 1 & fast by 2. num[num[fast]]
(3) Step (2) will go as : slow = num[slow] ; fast = ;
(4) There will be 1 time when slow & fast will meet.
They will meet in cycle.
(5) At step 4 we've to stop.
(6) Now move fast to init pos. : fast = num[0]
(7) ~~Step~~ ~~increment will be done as~~
~~slow =~~ .
(7) Now move both slow & fast by 1 pos. till both meet
(8) Step 7 increment done as :
slow = num[slow] ; fast = num[fast] ;

(9) Node at which both slow & fast will meet be the duplicate no.

(10) Return slow or fast :) .

Page No.:
Date:
YOUVA

Code :-

```
int findDuplicate(vector<int>& nums)
{
    int slow = num[0];
    int fast = num[0];
do {
    slow = nums[slow];  // by 1 step
    fast = num[num[fast]];  // by 2 step
}
while (slow != fast);

    fast = num[0];
while( slow != fast)
{
    slow = num[slow];
    fast = num[fast];
}

    return slow;
}
```

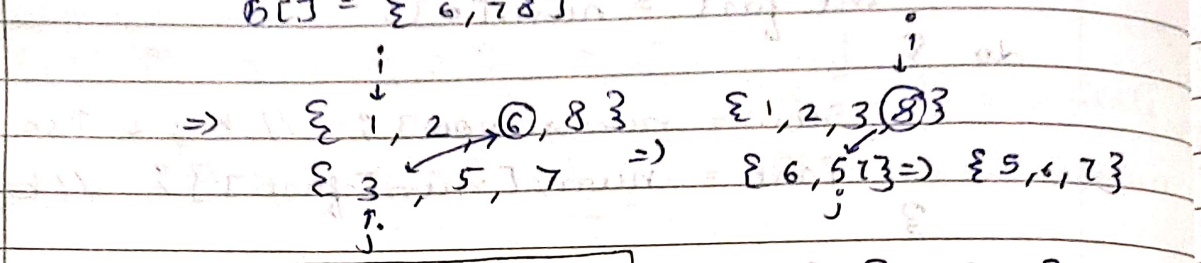
From 180 Striver

Page No.:
Date:
YOUVA

Ques- Merge 2 sorted array without using extra space

Input ⇒ a[] = {1, 2, 6, 8}
b[] = {3, 5, 7}

O/P = a[] = {1, 2, 3, 5}
b[] = {6, 7, 8}



⇒ a[] {1, 2, 3, 5} b[] = {6, 8, 7}

B[] ⇒ {6, 7, 8}

⭐ Code :- (Ques slight diff in GfG & leetcode)

```
void merge (long long arr1[], long long arr2[], int n, int m)
{
        int i = n-1;
        int j = 0;
while( i>=0 && j<m) {
    if(arr1[i] > arr2[j]) {
        swap (arr1[i], arr2[j])
        i--;
    } j++;
    else
        Break;
}
Sort (arr1, arr1 +n);
Sort (arr2, arr2 +m);
}
```

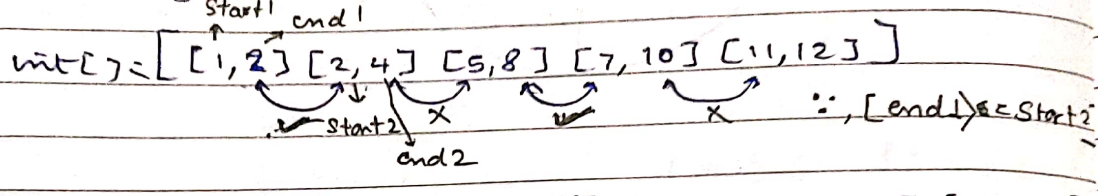
180 Striver

Page No.:
Date: 28/1/22
YOUVA

Ques - **Merge Interval (leetcode)**

array Intervals given ⇒ intervals[i] = [start, end]
merge all overlapping intervals, return array
[end1 >= start2]

explain - covering all interval non & on overlap.



→ ans = [[1,4], [5,8] [2,4], [5,10], [11,12]]

here ans ⇒ [start1, max(end1, end2)]

Algo :- (TC(n log n))

1) The base case : if no interval return [],
2) sort the interval
3) while traverse 2 condition,
   - max element in case of overlapping will be
     [1,4] [2,8] ⇒ [1,8] ∵ since 8 > 4.
   - if no overlapping, push this into result vector

CODE :-

```
vector<vector<int>> merge(vector<vector<int>>& intervals) {
    if (intervals.size() <= 1) return intervals; // base case

    vector<vector<int>> result; // resultant vector
    sort(intervals.begin(), intervals.end()) // TC - n log n
    result.push_back(intervals[0]); // Insert 1st interval
    for (int i=1; i<intervals.size(); i++) { // Traverse whole vector
        // if they are overlapping.
        if (result.back()[1] >= interval[i][0])
            result.back()[1] = max(result.back()[1], interval[i][1])
        else // they are not overlapping
            result.push_back(interval[i]); }
    return result;
};
```

From 160 striver

Page No.: Date: YOUVA

* Ques 1 - Next Permutation

an array of num given we've to find next permutation.

example → [1,2,3] => [1,2,3], [3,1,2], [2,3,1]

if they ask next Per. of [1,2,3] then [3,1,2] is ans.

if the last one they aking for then 1st will be the answeree.

* Algo.:-

(1) → find a break point from last, where a single element is not in increasing order i.e.- ar[i] < ar[i+1] & keep index "ind1".

(2) → Find again from last that, which element is just greater than ind1, as to follow the dictionary order to place bigger element than ind1 & name "ind2".

(3) → Swap ar[ind1] & ar[ind2].

(4) → reverse from the next element of ind1. i.e. ind1 + 1 to last of the array.

(*) Edge Case: If array is already in Desc order from the last than just reverse it

Code :-

```
int n= nums.size;
int k;  // to find break point
int l;  // to traverse from last & find
greater element has break point.

for (k=2; k>=0; k--) // step 1
{
    if (nums[k]<nums[k+1])
        break;
}
if (k<0)  // edge case
    reverse (nums.begin(), nums.end());
else
{   for (l=n-1; l>k; l--) // step 2
    {
        if (num[l] > num[k])
            break;
    }
    swap (num[k], num[l]); // step # 3
    reverse (nums.begin()+k+1, num.end());
                                // step 4
}
};
```

TC- O(n log n)

180 Revision

Ques ★★ Count Inversion (arr, r)

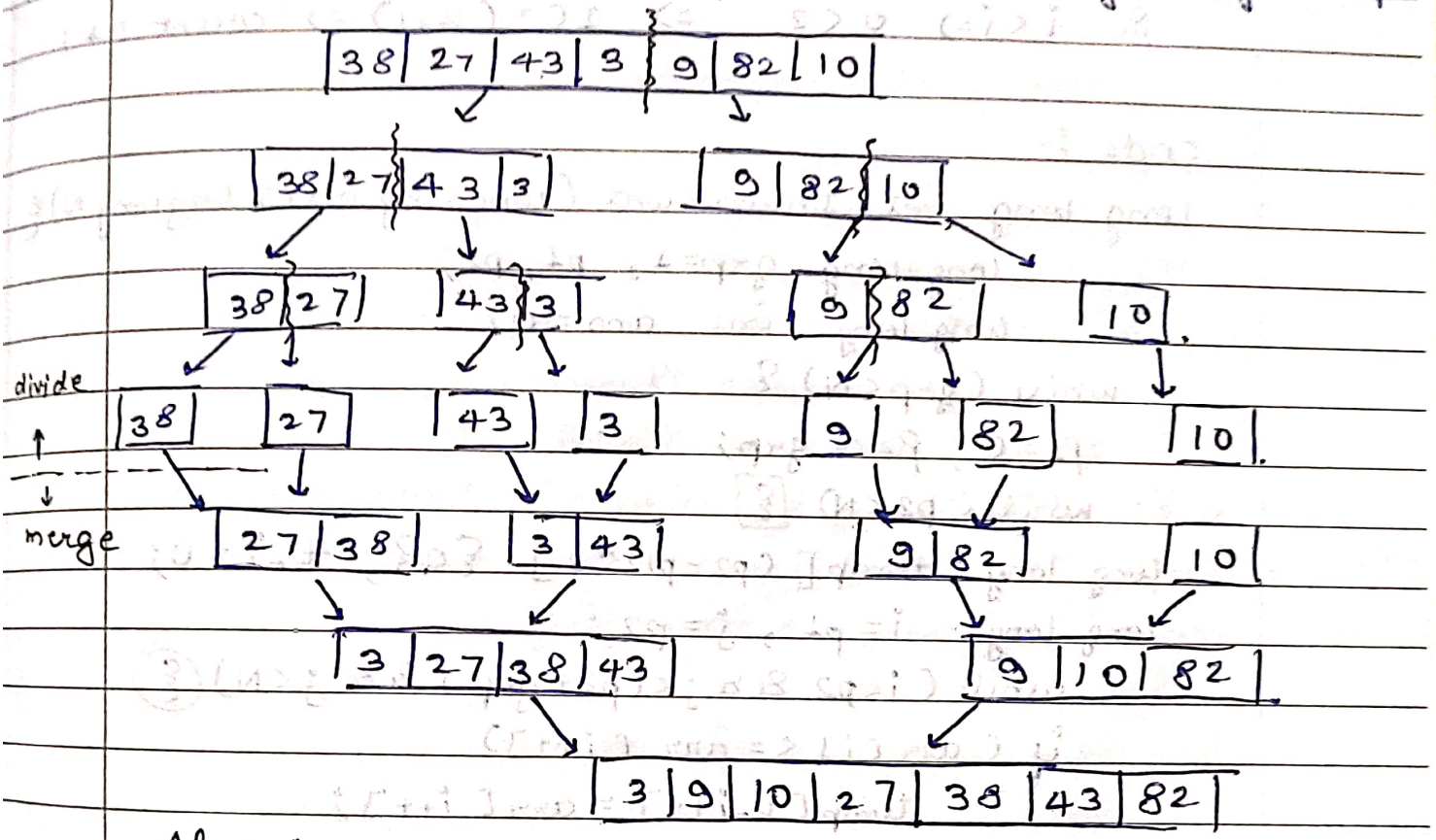
Hard

Before solving this question you must be aware of merge sort.

Merge Sort :-

→ Divide into 2 halves till size becomes 1

→ Once size become 1, merge come place, start merging array back till array merge complete



Algo :-

MergeSort (arr[], int l, int r),

If r > l.

1) Find middle point to divide array in 2 halves
middle m = l + (r-1)/2.

2) call mergesort for 1st half : mergesort (arr, l, m).

3) " " for 2nd half : mergesort (arr, m+1, r)

4) Merge 2 half sorted in step 2 & 3 :
call merge (arr, l, m, r)

→ Inversion Count: How far or close the array to being sorted. If it is already sorted IC = 0. Array is in reverse order sorted then IC = MAX.

Like- a[i] > a[j] and i < j then it IC

eg:- arr[] = {2, 4, 1, 3, 5} (i: 0 1 2 3 4)

i = 2, j = 2 · j++ => i = 2 j = 4 =>

here condition true a[i] = 2 a[j] = 1 => a[i] > a[j] = 2 > 1.

& i < j => 0 < 2 => IC = (2, 1) => count = 1;

Code :-

```
long long int InversCount (long long arr[], long long N) {
    long long grp = 1, p1, p2;
    long long int ans = 0;
  while (grp < N) {
    p1 = 0; p2 = grp;
    while (p2 < N) {
  long long temp[ (p2 - p1) * 2 ] = {0}, t_i = 0;
  long long i = p1, j = p2;
    while (i < p2 && j < (p2 + grp) && j < N) {
    if ( arr [i] <= arr [i++])
            temp [t_i++] = arr[ i++];
    else if ( arr [i] > arr [j])
            temp [t_i++] = arr [j++], ans += (p2 - i);
  }
while (i < p2)   temp[ t_i++] = arr [ i++];
while (j < (p2 + grp) && j < N)  temp [t_i++] = arr [j++];
long long cnt = t_i;
while (i < cnt) arr [t_i++] = temp [i++];
    p1 = p2 + grp, p2 = p1 + grp;
  grp *= 2;
} return ans; } };
```

Page No.:
Date:
YOUVA

Ques → Best time to buy and sell stocks (leetcode)

an array price[i], is given, we've to maximize profit by choosing day to buy stock & choose another day to sell those stocks

Input → [7, 1, 5, 3, 6, 4], O/P => 5

Bcoz Buy 2nd day = 1
sell 5th day = 6; 6-1 = 5

[7, 6, 5, 4, 3, 2, 1]

O/P = 0 ; Sell date > Buy date

Code :-

```
int maxprofit (vector<int> &prices) {
    int maxPo = 0;
    int minPrice = INT_MAX;
For (int i = 0; i < prices.size(); i++) {
    minPrice = min (minPrice, prices[0]);
    maxPo = max (maxPo, Prices[i] - minPr
}
return maxPo;
```

Ques- Count Pair with given sum (easy)

Pair of element whose sum is equal to K, return pair count.

example: arr = {1, 5, 7, 1} if K=6
=) 2 ∵ arr[0] + arr[1] = 1+5 = 6
[1] + [3] = 5+1 = 6

Code :-

```
unordered_map <int, int > mp;
    int count = 0;
for( int i=0;  i<n;  i++)
{
    count = mp[ K - a[i] ];
    mp[a[i]]++;
}
    return count;
}
};
```

Ques. Common elements :-

3 array given in sorted order, find element that is common in all three.

a = {1, 5, 10, 20, 40, 80} b = {6, 7, 20, 80, 100}
c = {3, 4, 15, 20, 30, 70, 80, 120}

output = 20, 80

code :-

```
int i=0, j=0, k=0;
vector<int> v;
while( i<n1 and j<n2 and k<n3) {
  if (a[i]==b[j] and b[j]== c[k])
{ v.push_back(a[i]); i++; j++; k++; }
  else if ( a[i] < b[j])
  { i++ }
  else if (b[j] <c[k]) j++;
      else
          k++;
  int xx = a[i-1];
while (a[i] ==xx) i++;
  int yy= b [i-1];
while ( b[j -1]) j++;
    int zz = c[k-1];
while ( c[k-1]) k++;
```

If array is like
3 3 3
3 3 3
3 3 3
=> then it'll give

(3) if (v.size() == 0) return {-1};
    return v;
}

Page No.:
Date:
YOUVA

Ques- Rearrange array in alternate +ve & -ve items.

array given with mix (+ve & -ve) no.
Arrange as every +ve no. is followed by -ve or vice-versa.
no. of +ve & -ve should be equal & alternate & if +ve no. are more put them at the end.

example- {1,2,3,-4,-1,4} ⇒ {-4,1,-1,2,3,4}
{2,3,-4,6,-1,-9} ⇒ {-4,2,-1,3,-9,6}
N P N P N P

Code :-

```
void rearrange(int arr[], int n)
{
    int i = -1, j = n;
    while (i<j)
    {
        while (i<=n-1 and arr[i]>0)
            i+=1;
        while(j>=0 and arr[j]<0)
            j-=1;
        if (i<j)
            swap(arr[i], arr[j]);
    }
    if (i==0 || i==n)
        return;
    int k=0;
    while (k<n && i<n)
    {
        swap(arr[k], arr[i]);
        i=i+1;
        k=k+2;
    }
}
```

Ques- Factorial of large number (GFG)

N given, find its factorial
suppose N = 10000! = (?).

There are 2 approches for using array & linked list.

5! = 5 x 4 x 3 x 2 x 1

We'll print this list as ans.

5! = [1] → [2] → [0]

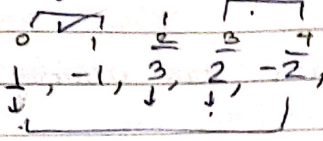
In link list approche we'll print array using recursion.

```
vector<int> factorial (int n) {

    vector<int> res;
    res.push_back(1);
for(int j=2; j<=n; j++) {
    int carr = 0;
    for(int i=0; i<res.size(); i++) {
        int val = res[i] * j + carr;
        res[i] = val%10;
        carr = val/10;
    }
    while(carr!=0) {
        res.push_back(carr%10);
        carr/=10;
    }
    reverse(res.begin(), res.end());
    return res;
}
};
```

180 Striver

Ques- ★★ Largest Subarray with 0 sum (GFG)

Array of +ve and -ve numbers. find max. length subarray whose sum = 0.



Sum=0, maxSum = ~~0~~ 5

=> 1-1 = 0,

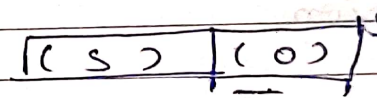
~~3, 5,~~ (3) ~~-5, -4,~~ (3)

diff = 5

(K, V) map

(preSum, Index)
(3, 9)
(13, 8)
(-4, 6)
(-5, 5)
(5, 3)
(3, 2)
(1, 0)

→ Sum=0 and



Code :-

```
HashMap<Integer, Integer> mpp = new HashMap<int,int>();
int maxi = 0;
int sum = 0;
for (int i = 0; i < n; i++) {
  sum += A[i];
  if (sum == 0) {
      maxi = i+1; }
  else {
      if (mpp.get(sum) != null) {
          maxi = math.max (maxi, i - mpp.get(sum)
      }
      else {
          mpp.put (sum, i); }
      }
  }
return maxi;
}
```

Page No.:
Date:
YOUVA

① good

Ques- Maximum product subarray (g4g) / Leetcode

array given, find subarray with maximum product

eg - [2, 3, -2, 4]

BF - find all sub-array, find its prod, return max.
TC $O(n^2)$

Algo → ma = max(a[i], ma*a[i]) ; mi = min(a[i], mi*a[i])
if a[i] = -ve, swap(ma, mi) ; ans = 0.

dry Run

[2, 3] -2, 4 => 2 ma = 6 mi = 3 ans = 6

ma = (3, 6) -2 mi = (3, 6) ma = -2, 3x-2 = -2, 6 ; mi = -2, -6 => -2

swap

CODE

```
int MaxPro(vector<int>& nums) {
    int ans = nums[0];
    int ma = ans; int mi = ans;

    int n = nums.size();
    for(int i=1; i<n; i++){
        if(nums[i]<0){
            swap(ma, mi);
        }
        ma = max(nums[i], ma*nums[i]);
        mi = min(nums[i], mi*nums[i]);
        ans = max(ans, ma);
    }

    return ans;
}
```

180 Striver using set

Page No.: Date: YOUVA

Ques ** Longest consecutive subsequence :- (g4g)

Ques-

unsorted array nums, longest consecutive subsequence return.

nums [100, 4, 200, 1, 3, 2] => ② subseq. → ① [1, 2, 3, 4] = 4, [100, 200] = ②

we'll take a set, [4 | 1 | 7 | 2 | 3 | 8]

start, check for 2 then 3,

4's already present in the set we'll not consider.

same goes with [7, 8] = 2 & [1 | 2 | 3 | 4] = 4 ans

Code

```
int LongConsecutive(int a[], int n){
  set<int> se;
  for(int i=0; i<n; i++){
int maxLen = 1; currLen = 1; auto it = se.begin();
  int prev = *it;
      it++;
while(it != se.end()){
    int now = *it;
if(now - prev == 1){
      currLen++; prev = *it; }
else{
    if(maxLen < currLen)
      maxLen = currLen;
currLen = 1; prev = *it;
}
  prev = *it;
    it++;
}
if(currLen > maxLen)
  { maxLen = currLen; }
return maxLen;
}
```

Page No.:
Date:
YOUVA

Ques- Given an array of size n and a number K, find all element that appear more than n/K times.

arr = {3, 1, 2, 2, 1, 2, 3, 3}, K = 4, n = 8.

n/K = 8/4 = (2) => 3 = 3 times, 2 = 3 times

[2, 3] = Ans.

code :-

```
void morethanNbyK(int a[], int n, int k) {
    int x = n/k;
    unordered_map<int, int> freq;

    for(int i=0; i<n; i++) {
        freq[a[i]]++;
    }

    for(auto i: freq) {

        if(i.second > x)
        {
            cout<<i.first<<endl;
        }
    }
}
```

Page No.:
Date: 30/1/2022 YOUVA

Ques - Maximum profit by buying & selling atmost 2x ; Ques -
(Hard)

Trader allowed to make 2 trans., 2nd T is done only after complete 1 tans:- (Buy → sell → Buy → sell).
find out max. profit-

example :- { 10, 22, 5, 75, 65, 80 }
Buy sell Buy sell

⇒ 22-10 = 12 ; 80-5 = 75

profit = 87.

* If optimal approch is using DP (TC - O(N))

Simple Sol^n :- (TC - O(n²))

```
int mx=0; mn=INT_MAX, n= price.size();
vector<int> f(n), b(n);                    upto ith ind
f[0]=0; // store max profit till ith index, when perfo. 1T b
for (int i=0; i<n; i++) {
    f[i] = max(mx, prices[i]-mn);
    mn = min(mn, price[i]);
    mx = max(mx, f[i]);
}
    mx=0; // b store max. profit after if we perform
// 1 T after ith index (included)
int mxx=0, and=0;
// b[i]+f[i-1] will give max profit if we do 1 T. before ith ind ex
for(int i=n-1; i>0; i--) {          // & 1 T after ith index.
b[i] = max(mx, mxx-prices[i]);
mxx = max(mxx, prices[i]);
mx = max(mx, b[i]);
    if(i>0) ans = max(b[i]+f[i-1], ans);
}

ans = max(b[0], ans); // ans will store overall
return ans;           // max. profit.
}
};
```

Page No.:
Date:
YOUVA

Ques- Array subset of another array

2 array given a1[] & a2[], we've to find out whete a2[] is subset of a1[] 'Yes or No'

Soln- 4 approches.

M1: TC - (O(N²) / O(1) M2: (Sorting + Binary search)

M3: (Sorting + 2 ptr) TC/SC: O(nlogn + mlogn) / O(1)

TC/SC: O(nlogn + mlogn) / O(1),

M4:- (hashing)

TC/SC = O(N) / O(N).

Method 4:-

Code:-

(1)

unordered_map<int, int> m1;

for(int i=0; i<m; i++) // array 1 or a2[]

m1[a[i]]++;

(2)

int c=0;

for(int i=0; i<n; i++) // array b[]

(2) if(m1[b[i]])

c++;

(3)

if(c!=n) cout<<"No"<<endl;

else cout<<"Yes"<<endl;

(3)

Page No.:
Date:

Ques - Triplet sum in array :-

an array of size N & int x given- find triplet which sum up to x,

ex- : x = 13 arr[] = {1, 4, 45, 6, 10, 8}

{1, 4, 8} = (1).

Code - TC- $O(N^2)$ - SC = (O(1))

```
bool find3Triplet (int a[], int n, int sum)
{       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] == sum)
{
    cout << a[i], << a[l] << a[r];
    return true;
}
else if ( a[i] + a[l] + a[r] < sum)
        l++;
  else
      r--;
}
}
return false;
}
```

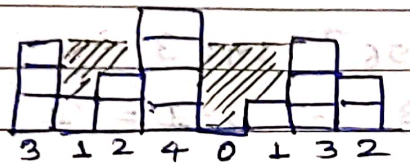
180 Striver

Page No.: Date: YOUVA

Ques: Trapping Rain Water :-

array arr[], each element represent height of 1 block, find out how much water will trap.

arr[] = {3, 1, 2, 4, 0, 1, 3, 2} o/p = 8



left → 3, 3, 3, 4, 4, 4, 4, 4
4, 4, 4, 4, 3, 3, 3, 2 ← right

Formula = min. (left[i], right[i]) - a[i]

Code :- int rainWater(int a[]) {
int n = a.length;
int left[] = new int[n];
int right[] = new int[n];
left[0] = a[0];
for (int i=1; i<n; i++) {
left[i] = max.(left[i-1], a[i]); }

right[n-1] = a[n-1];
for (int i = n-2; i>=0; i--) {
right[i] = max (right[i-1], a[i]);
}
int ans = 0;
for (int i=0; i<n; i++) {

ans = ans + (min (left[i], right[i])
- a[i]);
↓ right block

}
return ans;
}

Page No.:
Date:
YOUVA

Ques- Choclate distribution problem

Ques- arr[], contain N element represent no. of choclate packet. M is no. of student distribute such that: (1) each student (M) get one packet (2) diff. b/w no. of choc. with max. & min is min.

arr = {7, 3, 2, 4, 9, 12, 56} m = 3

=> {2, 3, 4} => 4 - 2 = 2

Approch-s sort + traverse

diff = a[i + (m-1)] - a[i]

while => (i + m - 1 < n)

Code :-

```
sort(a, a+n);
int min = INT_MAX;
for(int i=0; i+m-1<n; i++)
{
    int d = a[i+m-1] - a[i];
    if(d < min)
        min = d;
}
return min;
}
```

Page No.:
Date:
YOUVA

Ques:- Smallest subarray with sum greater than given value :-

array and no. x given, find smallest subarray with sum ~~equal to~~ greater than x

arr = {1, 4, 45, 6, 0, 19} x = 51.

sub array = {4, 45, 6} = (3)

Code :-

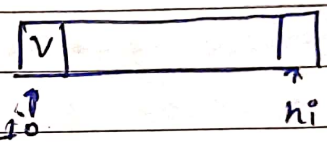
```
int smallestSubWithSum(int arr[], int n, int x)
{
int curr_sum = 0, end = 0, star = 0, min_len = n+1;
while( end < n) {
    while (curr_sum <= x && end < n)
    curr_sum += arr[end++];
    while (curr_sum > x && start < n)
    {
        if (end - start < min_len)
            min_len = end - start;

    curr_sum -= arr[star++];
    }
}
return min_leng;
}
```

Page No.:
Date:

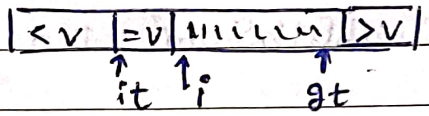
Ques- Three way partition (gfg)

Array A[] & rang [a,b] given :-
(i) element smaller than a come first
(ii) " in range of [a,b] come between
(iii) " bigger than b comes at last

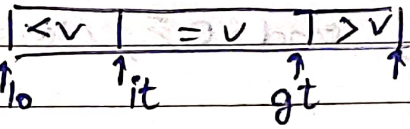
Approach :- Befor



Duration



After



→ 3 way partition

Code :- (DNF based quick Sort)

```
void ThreeWay(int arr[], int low, int high)
{
    int start=0, end=n-1;
for (int i=0; i<end; ) {

    if (arr[i] < low)
    {   start++; i++; }
    else
        swap(arr[i++], arr[start++]);

else if (arr[i] > high)
    swap(arr[i], arr[end--]);

else
    i++;
}
```

TC - SC - O(N) & O(1)

Page No.:
Date: 31/1/2022 YOUVA

Ques- Minimum swap and k together

array arr[], & k int given. You can swap any no. of times. Find min. no. of swap req. to bring all no. <= k together. make a contigious subarray.

example :-

{ 2, 1, 5, 6, 3 }
  0 1 2 3 4

swap index 2 & 4 => 5 & 3.

=> { 2, 1, 3, 5, 6 } => 1 swap

Approch1: count all element <= k. traverse for each sub array & swap who's value greater than k

Tc - O(n^2)

Approch -2 - Using 2 pointer and sliding window

Code :-

```
int minSwap(int arr*arr, int n, int k) {
    int count = 0;
    for(int i = 0; i < n; ++i)
    {   if (arr[i] <= k)
            ++count;
    }
    int bad = 0;
    for (int i=0; i<count; ++i) {
        if (arr[i] > k)
            ++bad;
    }

    int ans = bad;
    for (int i=0, j=count; j<n; j=count; ++i, ++j) {
        if (arr[i] > k)  --bad;
        if (arr[j] > k)  ++bad;
        ans = min(ans, bad);
    }

    return ans;
}
```

Ques → Palindrome Array (gfg)

an array[] given. return 1 if all element are palindrome otherwise return 0.

example :-

{ 111, 222, 333, 4443 } => Yes it is palindrom

111 = 111 ; 222 = 222 ; 121 = 121

Approch :- (121) ans = 0;

formula : ans = (ans x 10) + remainder (r);

while()

r = n%10;

= (0 x 10) + 1; = 1

for (21) => (1 x 10) + 2 = 12

= (12 x 10) + 1; = (121) = (121)

Code :-

```
int Pallindrome(int a[], int n)
{
    for(int i = 0; i<n; i++){
    int ans = 0, int temp = a[i];
while( temp >0){
    int r = temp%10;
    tem /= 10;
    ans = (ans*10) + r;
}
    if( ans != a[i])
      return 0;
}
}
```

Page No.:
Date:
YOUVA

* * *
Ques- Find the median of 2 sorted of equal size

2 array of same size given, we've to find its median.

a[] = {1, 12, (5), 26, 38}
{2, 13, (17), 30, 45}

{1, 2, 12, 13, (15, 17) 26, 30, 38, 45}; M = 15+17/2 = 16 (Ans)

Method:- We use the concept of "Merge 2 sorted array without using extra space" (Ques-12)
Then calculate is median! TC-O(n²) SC-O(N)

Method-2 - TC- O(N).

```
int i=0, j=0, count;
m1 = m2 = -1;
for (count = 0; count <= n; count++)
{   if (a[i] < b[j]) {
        m1 = m2;
        m2 = a[i];
        i++      }
    else {
        m1 = m2;
        m2 = b[j];
        j++;     }
    if (i == n)
    {   m1 = m2;
        m2 = b[0];
        break;   }
    else if (j == n) {
        m1 = m2;
        m2 = a[0];
        break; }
}   (m1 + m2)/2 = ans;
```

189 Striver

Page No.:
Date:

Ques- Median of two sorted array of diff size.

(Hard)

arr[] = {1, 2, 3, 4, 5} N=5

brr[] = {3, 4, 5, 6, 7, 8} M=6.

Merge = [1, 2, 3, 4, 5, 6, 7, 8] Median = [1, 2, 3, 3, 4, 4, 5, 5, 6, 7, 8] = 11

Dry Run

arr[] = [10, 20, 30, 40, 50] brr = [5, 15, 25, 35, 45]

Merg = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50] Median = 27.5

Naive Approch :-

1) creat a temp[] of size (m+n)
2) copy elem. of arr[] & brr[] in temp[].
3) if (extra) sort temp[]
4) if (n+m) is odd return middle of temp
5) else return avg. of middle 2 elemun.

→ Better approch :-

assume :- N <= M

arr[] = [10, 20, 30, 40, 50] N=5
i1

brr[] = [5, 15, 25, 35, 45, 55, 65, 75, 85] M=9
i2

left half	Right half
arr[0 --- (i1-1)]	arr[i1 --- (n-1)]
brr[0 --- (i2-1)]	brr[i2 --- (m-1)]

Now → If arr[] & brr[] as

N <= M

arr[] = [10, 20, 30, 40, 50] N=5
max1 i1 (min1)

brr[] = [5, 15, 25, 27, 28, 55, 65, 75, 85] M=9
max2 i2 min2

min1 : minimum element of right side of arr[] (30)

max1 : Max. ele. of left side of arr[] (20)

min2 : min. ele. of right side of brr[] (5s)
max2 : max. ele of left side of brr[] (28)

Dry Run

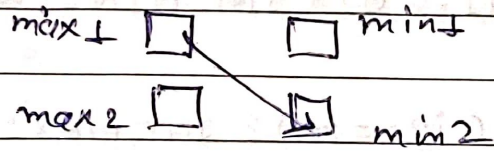
⇒ arr[] = [10, 20, 30] n=3 brr = [5, 15, 25, 35, 45] m=5

$i_1 = 1$ (begin=0, end=3) ⇒ (0+3)/2 = 1.

$i_2 = (3+5+1)/2 - 1 = 3$

(min1 = 20 min2 = 35
max1 = 30 10 max2 = 25)

max1 > min2 ← left
else right shift



[10, 20, 30], [5, 15, 25, 35, 45]

(begin = 2 end = 3)

$i_1 = 2$

$i_2 = (3+5+1)/2 - 2 = 2$

[min1 = 30 min2 : 25
max1 = 20 max2 : 15]

20 25

median = $\frac{20+25}{2}$ = 22.5

⇒ (max1 <= min2) and (max2 <= min1)

Code :-

```
int FindMedian(int arr[], int n, int brr[], int m)
{
    int begin1 = 0; int end1 = n;
    while( begin1 <= end1)
    {
        int i1 = (begin1 + end1)/2;
        int i2 = (n+m+1)/2 - i1;

        int min1 = (i1==n)? INT_MAX : arr[i1];
        int max1 = (i1==0)? INT_MIN : arr[i1-1];

        int min2 = (i2==m)? INT_MAX : brr[i2];
        int max2 = (i2==0)? INT_MIN : brr[i2-1];
        if(max1<=min2 && max2<=min1)
        {
            if((n+m)%2==0)
            return ((double)(max(max1,max2)+min(min1,min2))/2);
            else
            return (double)(max(max1,max2));
        }
        else if(max1>min2)
            end = i1-1;
        else
            begin = i1+1;
    }
}
```