5. ARRAYS

- Advantages: ① Random access ② Cache friendly
- Vector advantages: ① Dynamic size ② Rich library function ③ Easy to know size ④ No need to pass size ⑤ Can be returned from a function ⑥ Initialized with default value ⑦ easy to copy a vector.

Left Rotate a given array by (d) places.

```
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 --;
    }
}
```

TC: O(n)
SC: O(1)

Maximum difference (= find max^m diff value in array).

```
int maxdiff ( int arr[], int n ) {
    int minval = arr[0];
    int res    = arr[1] - arr[0];
    for ( int i = 1; i < n; i++ ) {
        res = max ( res, arr[i] - minval );
        minval = min ( minval, a[i] );
    }
    return res;
}
```

TC: O(n)
SC: O(1)

Priyansh

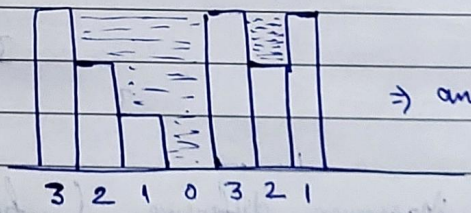
Max^m Consecutive 1's in a binary array

binary array → array of only '0' & '1' [0,1,0,1,0,1].

```
int maxcons ( int arr[], int n ) {
    int cur = 0;
    int res = 0;
    for( int i=0; i<n; i++ ) {
        if (arr[i] == 0)
            cur = 0;
        else {
            cur++;
            res = max(res, cur);
        }
    }
    return res;
}
```

TC: O(n)
SC: O(1)

Trapping rain water problem



⇒ ans = 7

consider all numbers in vector as bars, so find how many rain water can be stored.

```
int rainwater ( int a[], int n ) {
    int l[n], r[n];
    l[0] = a[0];
    for( int i=1; i<n; i++ ) {
        l[i] = max(l[i-1], a[i]);
    }
    r[n-1] = a[n-1];
    for ( int i=n-2; i>=0; i--) {
        r[i] = max( r[i+1], a[i]);
    }
```

TC: O(n)
SC: O(n)

```
    int ans=0;
    for(int i=0; i<n; i++){
        ans+= min(l[i], r[i]) - a[i];
    }
    return ans;
}
```

Stock buy and sell problem. [given a vector of price of a stock find maxm profit you can extract]

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

TC: O(n)
SC: O(1)

Maxm subarray sum [Subarray ⇒ contigous part of array
find a subarray with maxm sum value]

```
int maxmsubarray (int a[], int n){
    int res = a[0], maxm = a[0];
    for(int i=1; i<n; i++) {
        maxm = max( maxm +a[i], a[i]);
        res = max( maxm, res);
    }
    return res;
}
```

TC: O(n)
SC: O(1)

Priyansh

Largest even-odd length {Kadane Algorithm}

```
int evenodd( int a[], int n ) {
    int ans = 1, cur = 1;
    for( int i=1; i<n; i++ ) {
        if(( a[i]%2==0 and a[i-1]%2 !=0 ) or
           ( a[i]%2 !=0 and a[i-1]%2 ==0 ))
        {
            cur++;
            ans = max(ans, cur);
        }
        else
            cur = 1;
    }
    return ans;
}
```

TC: O(n)
SC: O(1)

Maxm circular Subarray sum

TC: O(n)

→ Here Method ①: take maxm subarray sum for normal part and for circular part,

maxm circular subarray sum = Total subarray sum − minm subarray sum

⇒ ans = maxm (maxm normal part , maxm circular part)

→ Here method ②: take maxm Subarray sum for normal part and for circular part,

we invert given array (* by (-1)) and then find maxm subarray sum and add it to total array sum.

Priyansh

⇒ Method ① : for normal part : kadane Algo
for circular part : modified kadane algo.

```
int maxcircularsubarray ( int a[], int n ) {
    int curn = a[0], maxn = a[0];
    for (int i=1; i<n; i++) {
        curn = max ( curn+a[i], a[i] );
        maxn = max ( maxn, curn );
    }
    int curc = a[0], resc = a[0];
    for (int i=1; i<n; i++) {
        curc = min ( curc+a[i], a[i] );
        resc = min ( resc, curc );
    }
    int tot = 0;
    for (int i=0; i<n; i++)
        tot += a[i];
    resc = tot - resc;
    return max ( maxn, resc );
}
```

⇒ Method ② : for normal part : kadane Algo.
for circular part : Invert + kadane algo + (sum + res)

```
int maxCircularSubarray ( int a[], int n ) {
    int curn = a[0], maxn = a[0];
    for (int i=1; i<n; i++) {
        curn = max ( curn+a[i], a[i] );
        maxn = max ( maxn, curn );
    }
    int tot = 0;
    for (int i=0; i<n; i++)
        tot += a[i];
```

Priyansh

```
for (int i=0; i<n; i++)
    a[i] = -a[i];
int curc = a[0], maxc = a[0];
for (int i=1; i<n; i++){
    curc = max(curc+a[i], a[i]);
    maxc = max(curc, maxc);
}
maxc = tot + maxc;
return max(maxn, maxc);
}
```

Majority Element {Boyer-Moorey Majority voting}

Majority is when count > n/2

```
int majority(int a[], int n){
    int count=1, res=0;
    for(int i=1; i<n; i++){
        if(a[res]==a[i])
            count++;
        else
            count--;
        if(count==0){
            res = i;
            count = 1;
        }
    }
    int ans = 0;
    for(int i=0; i<n; i++){
        if(a[res]==a[i])
            ans++;
    }
    if(ans < n/2)
        return -1;
    else return a[res];
}
```

→ check for candidate

candidate = a[res]

→ check if a[res] is an actual majority element or not.

TC: O(n)
SC: O(1)

Priyansh

classmate
Date
Page

Min^m consecutive flips for binary array.

if(a[0] == a[n-1]) ⇒ no. of set's of 1's − no. of sets of 0's = 1
else ⇒ no of set's of 1's = no of sets of 0's

```
void minmconsec ( int a[], int n ) {
    for( int i=1 ; i<n ; i++ )
    {   if( a[i] != a[i-1] ) {
            if ( a[i] != a[0] )
                cout << "from" << i ;
            else
                cout << i-1 ;
        }
    }
    if ( a[n-1] != a[0] )
        cout << n-1 ;
}
```

Sliding Window

① given array a and a int k, find max^m sum of k consecutive element.

```
int maxmK ( int a[], int n, int K) {
    int sum = 0 ;
    for(int i=0; i<k ; i++ ).
        Sum += a[i] ;
    int ans = sum ;
    for ( int i=k ; i<n ; i++ )
    {
        sum += a[i] - a[i-k] ;
        ans = max( ans , sum ) ;
    }
    return ans ;
}
```

TC: O(n)
SC: O(1)

Priyansh

② find subarray with a given Sum S.

```
bool sol (int a[], int n, int sum) {
    int cur = a[0]; s = 0;
    for (int e = 1; e < n; e++) {
        while (cur < sum and s < (e-1)) {
            cur -= a[s];
            s++;
        }
        if (cur == sum)
            return true;
        if (e < n)
            cur += a[e];
    }
    return (cur == sum);
}
```

TC: O(n)
SC: O(1)

③ N^{th} - fibonacci number:

```
void sol() {
    int n, m;
    cin >> n >> m;
    int a[m];
    for (int i=0; i<n; i++)
        a[i] = 0;
    a[n-1] = 1;
    int sum = 1;
    for (int i = n; i<m; i++) {
        a[i] = sum;
        sum += a[i] - a[i-n];
    }
}
```

Priyansh

★ Prefix Sum.

① given an array arr and 2 int l and r find sum for range l → r

```
int range ( int arr[], int n, int l, int r ) {
    int prefix[n];
    int sum = 0;
    for ( int i = 0; i < n; i++ ) {
        sum += arr[i];
        prefix[i] = sum;
    }
    if ( l == 0 )
        return prefix[r];
    else
        return prefix[r] - prefix[l-1];
}
```

TC: O(n)
SC: O(n)

② find if equilibrium point exist { ie, sum of left = sum of right }

```
bool eqpos ( int a[], int n ) {
    int lsum = 0; sum = 0;
    for ( int i = 0; i < n; i++ )
        sum += a[i];
    for ( int i = 0; i < n; i++ ) {
        if ( lsum == sum - a[i] )
            return true;
        lsum += a[i];
        sum -= a[i];
    }
    return false;
}
```

TC: O(n)
SC: O(1)

Priyansh

★★★ find frequency in limited range

① Subtract 1 from all element {indexing from 0}.
② add n to every point the element point
③ freq = a[i]/n ;

```
void sol( int a[], int n ){
    for(int i=0; i<n; i++)
        a[i]--;
    for(int i=0; i<n; i++)
        a[a[i]] += n;
    for(int i=0; i<n; i++)
        cout<< a[i]/n ;
}
```

leader in an array [every max^m is a leader. when we go right to left].

```
void sol(int a[], int n) {
    int maxm = Int_min;
    for(int i=n-1; i>=0; i--) {
        if( a[i] > maxm ) {
            cout<<a[i];
            maxm = a[i];
        }
    }
}
```

TC: O(n)
SC: O(1)

Priyansh

Smallest positive missing number

```
int missingnumber(int a[], int n){
    int p=0;
    for(int i=0; i<n; i++){
        if(a[i]==1){
            p++;
            break;
        }
    }
    if(p==0)
        return 1;
    for(int i=0; i<n; i++){
        if(a[i]<=0 or a[i]>n)
            a[i]=1;
    }
    for(int i=0; i<n; i++)
        a[(a[i]-1)%n] += n;
    for(int i=0; i<n; i++)
        if(a[i]<=n)
            return i+1;
    return n+1;
}
```

TC: O(n)
SC: O(1)

Rearranging sorted array alternatively (maxm minm format)

```
void sol(int a[], int n){
    int maxm = a[n-1]+1;
    int mini = 0, maxi = n-1;
    for(int i=0; i<n; i++){
        if(i%2==0){
            a[i] += (a[maxi]%maxm)*maxm;
            maxi--;
        }
```

TC: O(n)
SC: O(1)

Priyansh

```
else {
    a[i] += ( a[mini] % maxm ) * maxm;
    mini ++;
}
3.
for (int i=0; i<n; i++)
    a[i] = a[i] / maxm;
3.
```

To store 2 value at 1 index
increment by (arr[arr[i]%n]*n)
and to get old value a[i]%n
to get new value a[i]/n

*** Rearrange an array with O(1) extra space.
↳ a[i] = a[a[i]]

```
void sol (int a[], int n) {
    for (int i=0; i<n; i++)
        a[i] += ( a[a[i]%n] * n );
    for (int i=0; i<n; i++)
        a[i] = a[i]/n;
}
```

Largest subarray sum [kadane algo] {2nd implementation}

```
int max_so_far = Int_min, max_end_here = 0;
int start = 0, end = 0, s = 0;
for (int i=0; i<n; i++) {
    max_end_here += a[i];
    if ( max_so_far < max_end_here ) {
        max_so_far = max_end_here;
```

Priyansh

```
        start = s;
        end = i;
    }
    if (max_end_here < 0) {
        max_end_here = 0;
        s = i+1;
    }
}
cout << max_so_far << start << end;
```

find max^m diff. (j-i) such that A[i] ≤ A[j] and i ≤ j

```
int maxdiff(int a[], int n) {
    int l[n], r[n];
    l[0] = a[0];
    for(int i=1; i<n; i++)
        l[i] = min(l[i-1], a[i]);
    r[n-1] = a[n-1];
    for(int i=n-2; i>=0; i--)
        r[i] = max(r[i+1], a[i]);
    int i=0, j=0, diff = -1;
    while(i<n and j<n) {
        if(l[i] <= r[j]) {
            diff = max(diff, j-i);
            j++;
        } else {
            i++;
        }
    }
    return diff;
}
```

TC: O(n)
SC: O(n)

for eg:
a = 34, 8, 10, 3, 2, 80, 30, 33, 1
l = 34, 8, 8, 3, 2, 2, 2, 2, 1
r = 80, 80, 80, 80, 80, 80, 33, 33, 1

diff = 6

Priyansh

Sieve of Sundaram (print all primes <= n)

of the form
i+j+2*i*j

```
void fun(int n){
int newn = (n-1)/2;
bool arr[newn+1];
memset(arr, false, sizeof(arr));
for(int i=1; i<=newn; i++){
    for(int j=i; (i*j+2*i*j)<=newn; j++)
        arr[i+j+2*i*j] = true;
if(n>2)
    cout<<2<<endl;
for(int i=1; i<=newn; i++)
    if(arr[i]==false)
        cout<<2*i+1<<" ";
```

TC: O(n)
SC: O(n)

3.

Longest span with same sum of 2 binary array

```
int sol(int a[], int b[], int n){
    int c[n], map<int,int> m;
    for(int i=0; i<n; i++)
        c[i] = a[i]-b[i];
    int cur=0, res=0;
    for(int i=0; i<n; i++){
        cur += c[i];
        if(cur==0)
            res = i+1;
        if(m.find(cur)!=m.end())
            res = max(res, i-m[cur]);
        else
            m[cur]=i;
    }
    return res;
}
```

a= 0, 1, 0, 0, 0, 0
b= 1, 0, 1, 0, 0, 1
c= -1, 1, -1, 0, 0, -1

TC: O(n)
SC: O(n)

Priyansh

Maxm occurred integer in n ranges.

```
int sol( int l[], int r[], int n ){
    int M = 0;
    vector<int> a(100000);
    for( int i=0; i<n; i++) {
        a[ l[i] ]++;
        a[ R[i]+1 ]--;
        if ( R[i] > M )
            M = R[i];
    }
    int s = a[0], ind;
    for( int i=1; i<=M+1; i++ ) {
        a[i] += a[i-1];
        if( a[i] > s) {
            s = a[i];
            ind = i;
        }
    }
    return i;
}
```