j-0+

1 october 2023

Arrays - (medium) :-

1.→ {Two Sum}

arr[] = { 2, 6, 5, 8, 11 }   target → 14

{8 + 6}

Brute:-
```
for (int i=0; i<n ; i++){
    for (int j=0; j<n ;j ++){
        if (i != j){
            if (target == arr[i]+arr[j] ){
                cout yes
                break;
            }
        }
    }
}
```

T.C → $O(n^2)$
S.C → $O(1)$

__/__/____

Better :- Hashing ---▷ T.C O(n)
S.C O(n)

```
map<int, int> mapp;
for (int i-▷0-▷size(n)){
    int find = target-arr[i]);
    if (mapp[find]){
        cout yes ...
    }
    else {
        mapp[arr[i]]++;
    }
}

return NO ;
}
```

Approach 2   target = 14   TC -▷ O(N) + O(NlogN)

Sort -▷ 2, 6, 5, 8, 11
i   j

i+j = 13 < 14 then increase small one
(i+1)+j = 6+11 = 17 > 14 then decrease bigger
6+8 = 14 --- found
j < i } cout "no".

2 -▷ { sort 0's, 1's and 2's }

Brute   sort (Merge sort)

T C -▷ N log N
S . C -▷ N

_/_/_

Better

count 0's, 1's and 2's   TC O(N) -- TC O(2N)

for (0 — 0's) 0 } SC O(1).

(0's — 1's) 1 } (N)

(1's — 2's) 2   for counting N &   {2N}

for distributing N

Optimal

Dutch National flag Algorithm :- T.C → O(N)

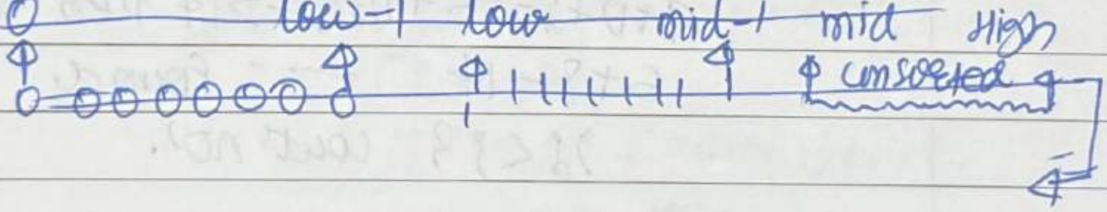
S.C → O(1).

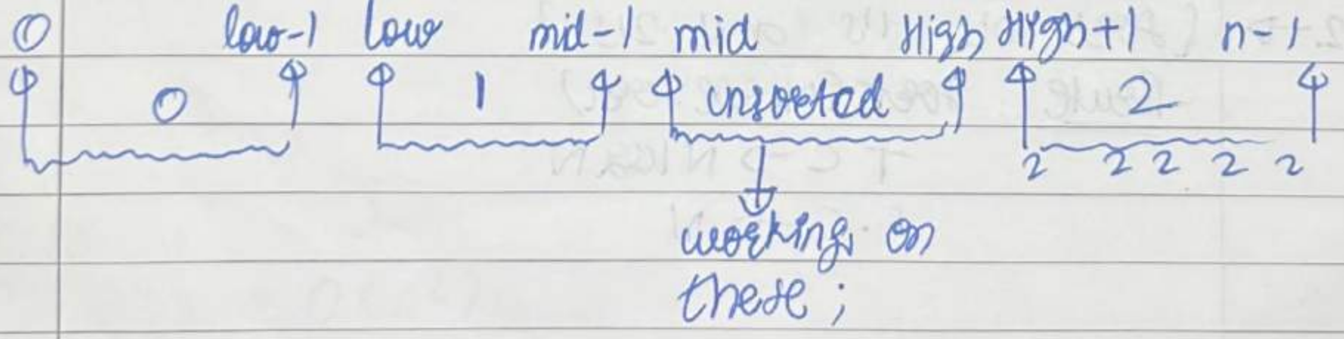
low } Pointer
mid
high

[0 ... low-1] → 0   extreme left

[low ... ~~H~~mid-1] → 1

[high+1, n-1] → 2   extreme right





```
      low
arr | 0 | 1 | 1 | 0 | 1 | 2 | 1 | 2 | 0 | 0 | 0 |
      mid                                     high
```

```
a[mid]=0    swap(arr[low], arr[mid]);
            low++ ; mid++;
a[mid]=1    mid++;

a[mid]=2    swap(arr[mid], arr[high]);
            ~~mid++~~ high--;
```

1> swap(arr(low), arr(mid)) low++, mid++;

```
0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0
 ↑  ↑
low mid
```

2> arr[mid]=1 mid++;

```
0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0
   ↑  ↑
  low mid
```

3> arr[mid]=1 mid++

```
0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0
   ↑     ↑
  low   mid
```

6> arr[mid]=0 swap(arr[low], arr[mid]),
low++, mid++.

~~0, 0, 1, 1, 0~~

```
0, 0, 1, 1, 1, 2, 1, 2, 0, 0, 0
      ↑     ↑  ↑
     low   mid mid
```

7> arr[mid]=1 mid++

8) arr[mid] = 2

swap (arr[mid], arr (high));

high--;

0, 0, 1, 1, 1, 0, 1, 2, 0, 0, 2

low mid high

9) arr (mid) = 0 ~~mid++~~ swap(low, mid)

0, 0, 0, 1, 1, 1, 1, 2, 0, 0, 2

low mid high

10) mid++ {because 1}

11) mid++

0, 0, 0, 1, 1, 1, 1, 2, 0, 0, 2

low mid high

12) swap(mid, high) arr(mid) = 2

0, 0, 0, 1, 1, 1, 1, 0, 0, 2, 2

low mid high

13) arr(mid) = 0

0, 0, 0, 0, 1, 1, 1, 1, 0, 2, 2

low mid, high

arr[mid] = 0

swap with low

14)

{0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2}

sorted

3.→ Majority element ($> N/2$ times)

arr[] = {2, 2, 3, 3, 1, 2, 2}

ans = 4

Brute

```
for(i → n)
    for(j → n)
        - —
O(n^2)
```

Better

{use map}

unordered N

- map N log N
- → T.C O(N) + N log N

Optimal

Moore's voting Algorithm :-

If the element is greater than $> \frac{N}{2}$ times its count in different sub-array can't be zero.

{7, 7, 5, 7, 5, 1, 5, 7, 5, 5, 7, 7, 5, 5, 5, 5}

```
ele = 7                     ele = 5  ele = 5    ele = 5
cnt = 1 2 1 2 1 0   1 0     1 2 1 0  1 2 3 4
  {7 7 5 7 5 1}{5 7}        5 5 7 7  5 5 5 5
```

{cnt = 4 is not cancelled}

check that element occurs $\{> \frac{N}{2}\}$ time in array again.

{ mooeel voting Algo }

```
int majoeityElement (vectoe <int>& v)}
        int cnt=0;
        int el;
        for (int i =0  i<n  i++){
            if (cnt==0){
                el= v[i];  cnt=1;
            }
            if (v[i] ==el){
                cnt++)
            }
            else{
                cnt--;
            }
        }
        //again iterate
        for (int i=0 to n){
            if (el ==v[i]) count++;
        }
        if (count > (v.size()/2)){
            eetuen el;
        }
        eetuen -1;
    }
```

T.C → O(N) + O(N)

S.C → O(1)

21 sept 2023

LOVE BABBAR DSA SHEET / A2Z

Largest subarray sum Problem

Kadane algorithm
{-2, -3, 4, -1, -2, 1, 5, 3}

~~max_so_far = INT-MIN~~ ;
~~max ending here = 0~~ ;
~~i=0 , a[0] = -2~~
~~max ending here~~ += (-2) ;
~~max ending here = 0 -- because~~
~~max-end-here < 0~~

~~set max-so-far = -2~~
~~i=1 a[1] = -3~~
~~max-endin += (-3)~~
~~-4 = 0 --- < 0~~
~~next~~

```
int maxsubarraySum( int arr[], int n){
    int mini = INT_MIN;
    for(int i=0 ; i<n ; i++){
        for(int j=i ; j<n ; j++){
            int sum = 0;        N^2 -> sum take(s)
            for (int k = i ; k <= j ; k++){
                sum += arr[k];
            }
            mini = max(mini, sum);
        }
    }
    ~~retu~~ mini;
}
```

}} -> O(n^3)

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

maxi = ~~INT_MIN~~ ~~-2~~ ~~4~~ 7

sum < 0

sum = ~~0~~ ~~-3~~ ~~4~~
~~-2~~ 0 ~~3~~
0 ~~1~~
~~2~~
~~7~~ 4

```
long long maxSub( int arr[], int n ){
        long long sum = 0, maxi = LONG_MIN;
          for(int i=0 ; i<n ; i++){
              sum += arr[i]; --> if(sum == 0){
                                      start = i;
                                  }
              if (sum > maxi){
                  maxi = sum; --> { ansStart
                                    = start;
                                    ansEnd = i }
              }
              if(sum < 0){
                  sum = 0;
              }
          }
        return maxi;
}
```

__/__/____

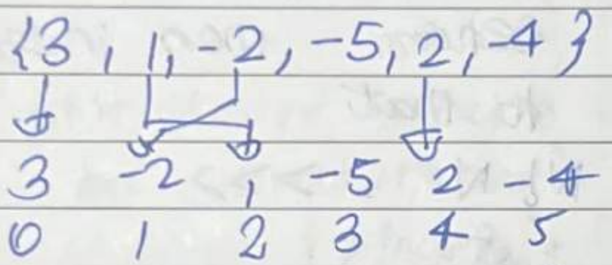
4. → Rearrange array element by sign
arr[ ] = {3, 1, -2, -5, 2, -4}
arr[ ] = {3, -2, 1, -5, 2, -4}

Brute:-

maptuh two array positive, negative
T.C O(N) + O(N)
S.C O(N)

Opthimal:

TC → O(N)
S.C → (O(N)



```
function (vector<int> &v){
    vector <int> answer;
    int iter=0, iter1 = 1;
      if (v[i] <0){
           answer[iter1] = v[i];
              iter1 += 2;
           }
      else {
           answer[iter] = v[i];
              iter += 2;
           }
      }
    return answer;
}
```

__/__/__

5. → Next Permutation

arr[] = [3 1 2]

1, 2, 3   ans → [3, 2, 1]
1, 3, 2
2, 1, 3
3, 1, 2
3, 2, 1

Brute : generate all permutations
then linear search
return then index for index next
to that

TC N! N >>>>

Better : next_permutation (arr.begin, arr.end);
inside the

Optimal :

1. Longer prefix match (a[i] < a[i+1])
2. first greater than arr[i] but smallest one
3. sort —

arr = [2, 1, 5, 4, 3, 0, 0]
2 1 → < 3
2, 3 {5, 4, 1, 0, 0}
sort
{2, 3, 0, 0, 1, 4, 5}

dip

{1, 2, 3, 4, 5}

{5, 4, 3, 2, 1} last permutation

```
              i  n-1
            o o  o
            n-2     {i<n-1}

index = -1;
 for (i=n-2 ; i>=0 ; i--){
    if (arr[i] < arr[i+1]){
              index = i;
              break;
           }
    }
  if (index == -1){
      reverse (arr);
   }                         ____4____
                          { 2, 1, 5, 4, 3, 0, 0 }
  else{
   // first element greater element at index
   //   but smallest one
                    i > index
   for (i= n-1 ; ~~i>=0~~ ; i--){
        if (arr[i] > arr[index]){
          swap (arr[i], arr[index]);
          break;
     }
  }          // 2, 3, 5, 4, 1, 0, 0
                                arr.end()
   reverse (arr.begin() + index+1, n-1);


TC -> O(3N)
SC -> O(1)
```

6. → Leader in the array
{10, 22, 12, 3, 0, 6} everything in the right
{22, 12, 6} must be smaller

brute

```
for (i 0 → n)
    for (j i+1 → n)
        - -

        - -.
```

T.C → O(N2).
S.C → O(1). — storing O(N)

Optimal

Start from backward and maintain maxi.
stored in answer.
sort - -
T.C → ~~O(N2)~~ O(NlogN)
S.C → O(N)

__/__/____

03 october 2023

Longest consecutive sequence in an array :-

Brute :-

```
for (int i=0 , i<n) {
    int start = arr[i];
    int cnt =1 ;
    while ( bool (start+1, arr)) {
            start++;
            cnt++;
    }
    ans = max (ans, cnt) ;
  }
  return ans ;
}
```

```
bool bool ( ) {
 for(i->n){
   if(start==arr[i])
     return true;
   }
 }
return false;
```

$O(N^2)$ TC
$O(1)$ SC

{1,1,1,2,2,2,3,3,4,100,100,101,101,102}

Better :- sort()

TC -> NlogN
SC -> O(1)

```
for(int i=0, i<n ; i++){
   if (arr[i]-1 == lastSmaller){
         cnt++;
         lastSmaller = arr[i];
   }
   else if (arr[i] != lastSmaller){
         cnt =1
         lastSmaller = arr[i];
   }
   longest = max (cnt, longest);
} return longest;
```

__/__/____

Optimal :-

```
longest (vector<int> a) {
 if (!a.size()) { return 0; }
  int longest = 1;
  unordered_set <int> st;
  for (int i → 0 → n) {
        st.insert (a[i]);
  }
 for (auto it : st) {
      if (st.find(it-1) == st.end()) {
           int cnt = 1
           int x = it;
           while (st.find(x-1) != st.end()) {
             ~~int cnt = 1;~~
             cnt++;
                x++;
           }
          longest = max (longest, cnt);
       }
     return longest;

  T.C → O(3N)
  SC → O(N)
```

__/__/__

Set matrix zeros

Brute :-

```
for(int i=0 to n){              } n×m
    for(int j=0 to m){
        if (arr[i][j]==0){
            markRow(i);         } n+m
            markCol(j);
        }
    }
}
markRow(i){
    ~~if(arr[i~~
    for(int j=0 to m){
        if (arr[i][j] != 0){
            arr[i][j] = -1;
        }
    }
}
markCol(j){
    for(i=0 to n){
        if (arr[i][j] != 0){
            arr[i][j] = -1;
        }
    }
}
```

T.C → $(n\times m)\times(n+m) + (n\times m)$

$\approx \{n^3\}$

S.C $\approx O(1)$

Put all zero where -1 exists } $n\times m$

__/__/____

Better

maintain two set
{st1, st2}

```
for (->){
    for (->){
        if (arr[i][j] ==0){
            ~~Row.push~~
            st1. insert(i);
            st2. insert(j);
        }
    }
}
for (->){
    for (->){
        if (st1. find(i) != st1.end()){
            matrix [i][j] =0;
        }
        if (st2. find(j) != st2.end()){
            matrix [i][j] =0;
        }
    }
}
```

TC -> O(2n*m)
SC -> O(n)+ O(m).

__/__/____

Optimal

```
int col0 = 1;
for (int i = 0; i < n; i++){
    for (int j = 0; j < m; j++){
        if ( matrix [i][j] == 0){
{mark ith row}   matrix [i][0] = 0;
            if (j != 0){
                matrix [0][j] = 0;
            }
            else {
                col0 = 0;
            }                              TC -> O(n*m)
        }                                  SC -> O(1).
    }
}
for (int i = 1; i < n; i++){
    for (int j = 1; j < m; j++){
        if (matrix [i][j] != 0){
    {check col & row}
            if (matrix[0][j] == 0 || matrix [i][0] ==0){
                matrix [i][j] = 0;
            }
        }
    }
}       if (matrix [0][0] == 0){
~~set~~             j -0 to m  matrix [0][j] = 0; }
        if ( col0 == 0){
            i -> 0 to n  matrix [i][0] = 0; }
```

__/__/____

Rotate Matrix :-

Brute{

TC → $O(N^2)$

SC → $O(N^2)$

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 & 8 \\ 2 & 9 & 10 & 11 & 12 \\ 3 & 13 & 14 & 15 & 16 \end{array} \rightarrow \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 13 & 9 & 5 & 1 \\ 1 & 14 & 10 & 6 & 2 \\ 2 & 15 & 11 & 7 & 3 \\ 3 & 16 & 12 & 8 & 4 \end{array}$$

i j	i j	i j	i j
[0,0] →	[0,3]	[1,0] →	[0,2]
[0,1] →	[1,3]	[1,1] →	[1,2]
[0,2] →	[2,3]	[1,2] →	[2,2]
[0,3] →	[3,3]	[1,3] →	[3,2]
(i, j)	(i₁, j₀)		

$(j) = (n-1)-i$

ans[j - - -]

~~arr~~[j][n-i-1] = arr[i][j]

Better / Optimal :-

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 & 8 \\ 2 & 9 & 10 & 11 & 12 \\ 3 & 13 & 14 & 15 & 16 \end{array} \xrightarrow{\text{Transpose}} \begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix}$$

↓ reverse each row

$$\begin{bmatrix} 13 & 9 & 5 & 1 \\ 14 & 10 & 6 & 2 \\ 15 & 11 & 7 & 3 \\ 16 & 12 & 8 & 4 \end{bmatrix}$$

__/__/____

```
Funct(){
  for(int i->0 to n){
      for(j=i+1 to m){
          swap(mat[i][j], mat[j][i]);
      }
  }
  for(int i->0 to n){
      int start=0, end = mat[0].size()-1;
      while(start <= end){
          swap(mat[i][start], mat[i][end]);
          start++;
          end--;
      }
  }
}
```

TC -> $O(N/2 * N/2)$ + $O(N * N/2)$

SC -> O(1).

04 october 23

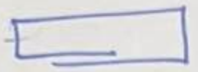
Spiral Matrix

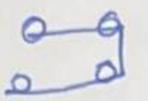
	0	1	2	3	4	5	
0	1	2	3	4	5	6	(0,5)
1	7	8	9	10	11	12	(1,5)
2	13	14	15	16	17	18	
3	19	20	21	22	23	24	
4	25	26	27	28 (5,4)	29 (4,4)	30	
5	31	32	33	34	35	36	(5,5)

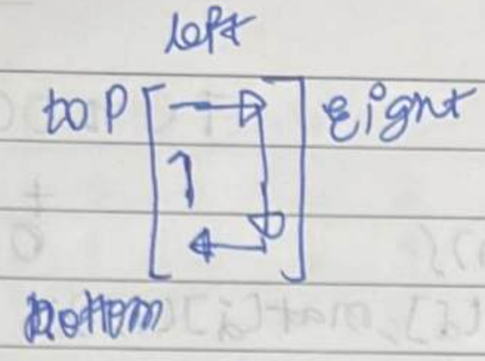
(5 4)

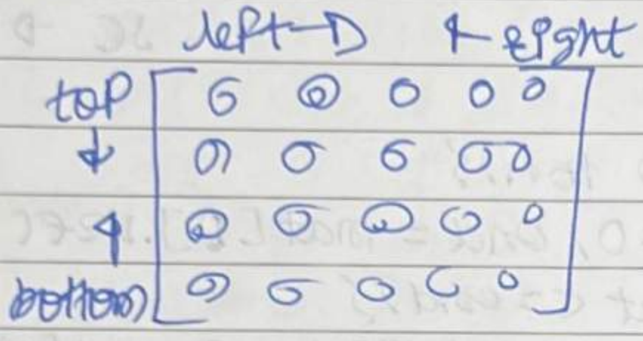
6/2 = 3

$$\left\{\begin{matrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 & 1 \\ x & x & x & \propto & \propto \\ x & x & \propto & x & x \end{matrix}\right\}$$

edge 







```
if(left <= right){
for(int i = bottom to top)
  ans.push_back
  (mat[..
   -- =
   =
}
```

bottom = mat.size()-1
left = top = 0 right = mat[0].size()-1

```
Funct(){
vector<int> ans ;
  while ( left <= right and top <= bottom ){
     for (int i = left ;  i <= right ; i++){
         ans.push_back( mat [top][i] ) ;
        }
       top++;
    for (int i = top ; i <= bottom ; i++){
         ans.push_back( mat[i][right]);
        }
       right --;
{edge case} if (top <= bottom){
     for (int i = right ; i >= ~~bott~~ left ; i--){
         ans.push_back (mat[i] [left]);
        }
       left++;
   }
```

Numbers of sub-arrays with sum K

Brute:-

```
for (int i=0 to n){
    for(int j=i to n){
        for (int k = i  k<=j ; k++){
            sum+=nums[k];
            }
        if(sum==k){
            ans++;
        }
    }   }
```

TC = O (N3)

```
for (int i=0 to n){
    ~~for (in~~ sum=0
    for(int j=i to n){
        sum += nums[j];
        if(sum==k){
            ans++;

    }   }
```

TC = O (N2)

TC : O(N × log N)
SC O(N)

<u>Optimal approach with Prefix sum :-</u>

arr[] = { 1,2,3,-3,1,1,1,4,2,-3 } {K = 3}

prefSum = ~~0~~ ~~1~~ ~~3~~ ~~6~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~10~~ ~~12~~
present 9

cnt = ~~0~~ ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~6~~ 8

(9,1)
(12,1)
(10,1)
(5,1)
(4,1)
(6,2)
(3,2)
(1,1)
(0,1)

arr[] = { 3,-3,1,1,1 }

without putting zero in prefix sum

PS = ~~3~~ ~~0~~ ~~1~~ ~~2~~ 3 ~~0~~
cnt = 0 0 1

1

(2,1)
(1,1)
(0,1)
(3,1)

prefSum = ~~0~~ ~~3~~ ~~0~~ ~~1~~ ~~2~~ 3
cnt = ~~0~~ ~~1~~ 2 2 2 $\boxed{4}$ → ↓
{3}, {3,-3,1,1,1}
{1,1,1} {-3,1,1,1,1,3}

(2,1)
(1,1)
(3,2)
(0,2)

{1,-1,0} K=0

(0,1)
~~(0,1)~~

0,1,0 0
1 2

prefSum = ~~0~~ 1 0 0
~~0~~ 1 2 3