# Today's Agenda : (4 Problems)

- Pick from both sides
- Alternating Subarrays
- Row to Column Zero
- Spiral Matrix - 2

Q1) Given an integer array A of size N. You can pick B elements from either left or right end of the array A to get maximum sum. Find & return this maximum possible sum.

Eg :- [5, -2, 3, 1, 2] , B = 3 , ans = 8 .

Case 1 : Select first 3 elements : sum = 6 .

[5, -2, 3, 1, 2]

Case 2 : Select last 3 elements : sum = 6

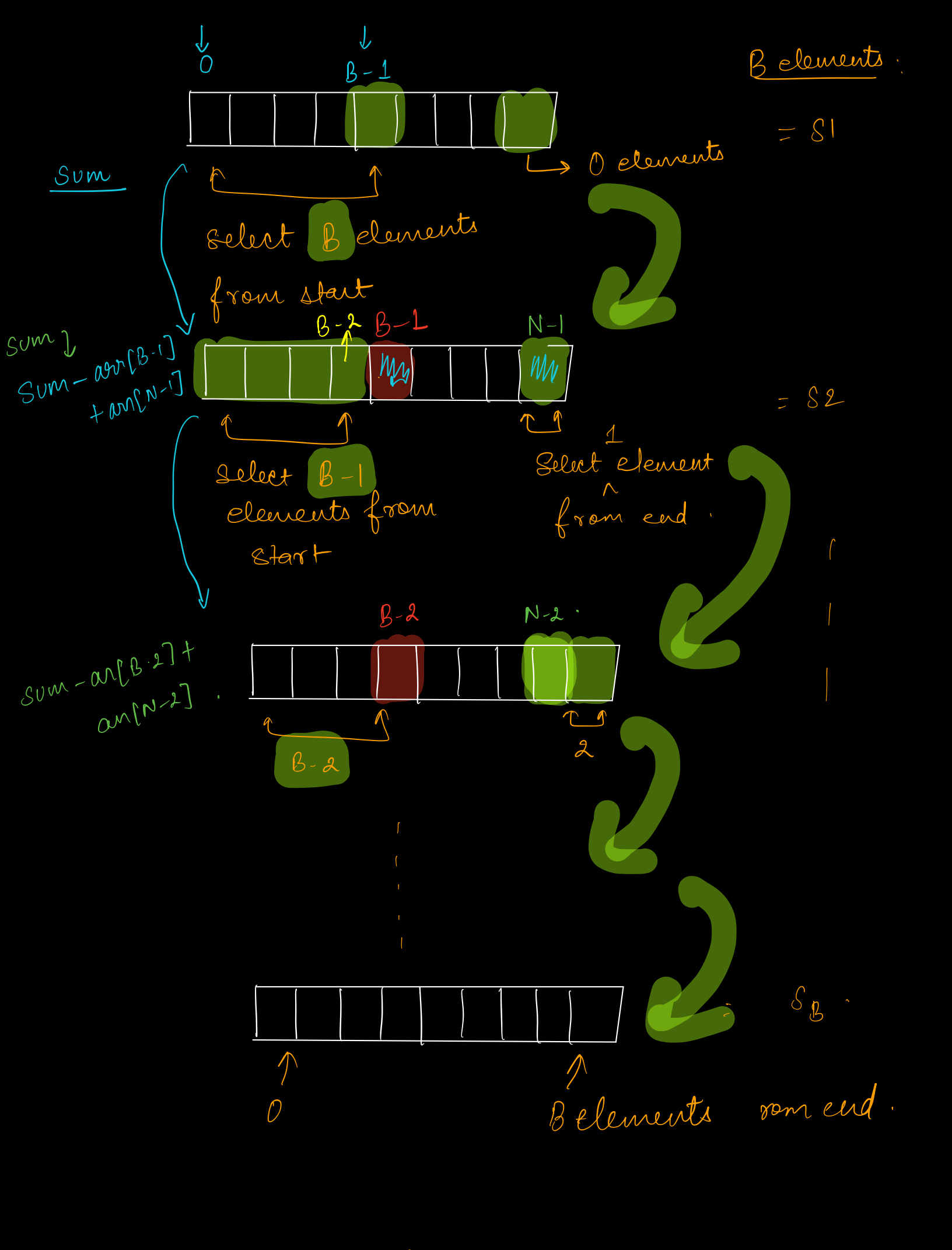
[5, -2, 3, 1, 2]

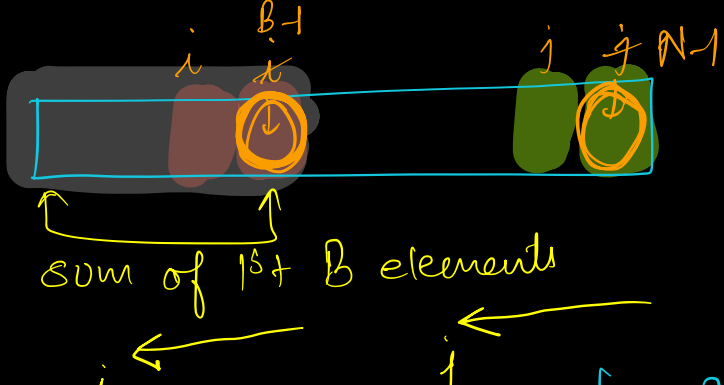
Case 3 : Select 1 element from start, 2 elements from end.
: sum = 8

[5, -2, 3, 1, 2]

Case 4 : sum = 5

[5, -2, 3, 1, 2]


0
B-1
B elements :
= S1
0 elements
Sum
Select B elements from start
B-2
B-1
N-1
Sum ↓
Sum - arr[B-1] + arr[N-1]
= S2
Select B-1 elements from start
Select 1 element from end.
B-2
N-2
Sum - arr[B-2] + arr[N-2]
B-2
2
$S_B$
0
B elements from end.




// found sum of first B elements [0, B-1] => sum.

B <= N

B > N

```
sum = 0
for(i=0; i < B; i++){
    sum = sum + A[i];
}
ans = sum;
i = B-1, j = N-1
```

B

```
while(i >= 0){
    sum = sum - Arr[i] + Arr[j]
    i--;
    j--;
    if(sum > ans){
        ans = sum;
    }
}
return ans;
```

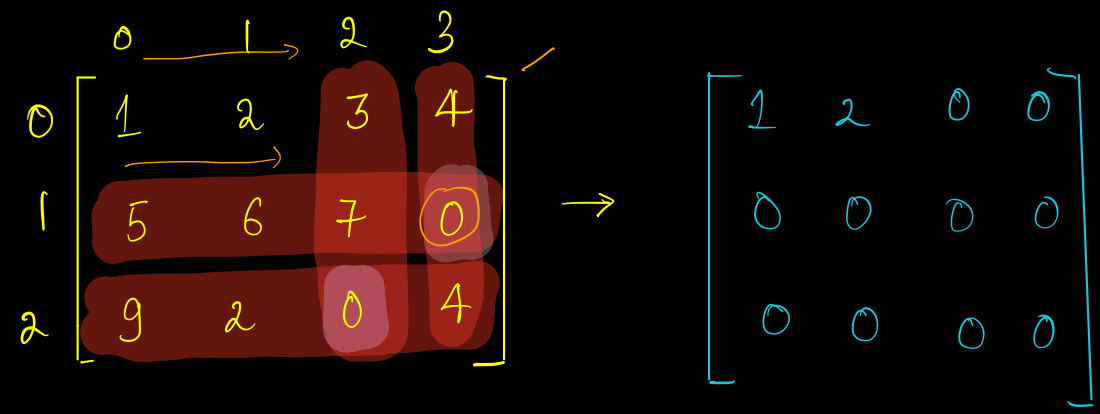
TC : O(B) ≈ O(N)

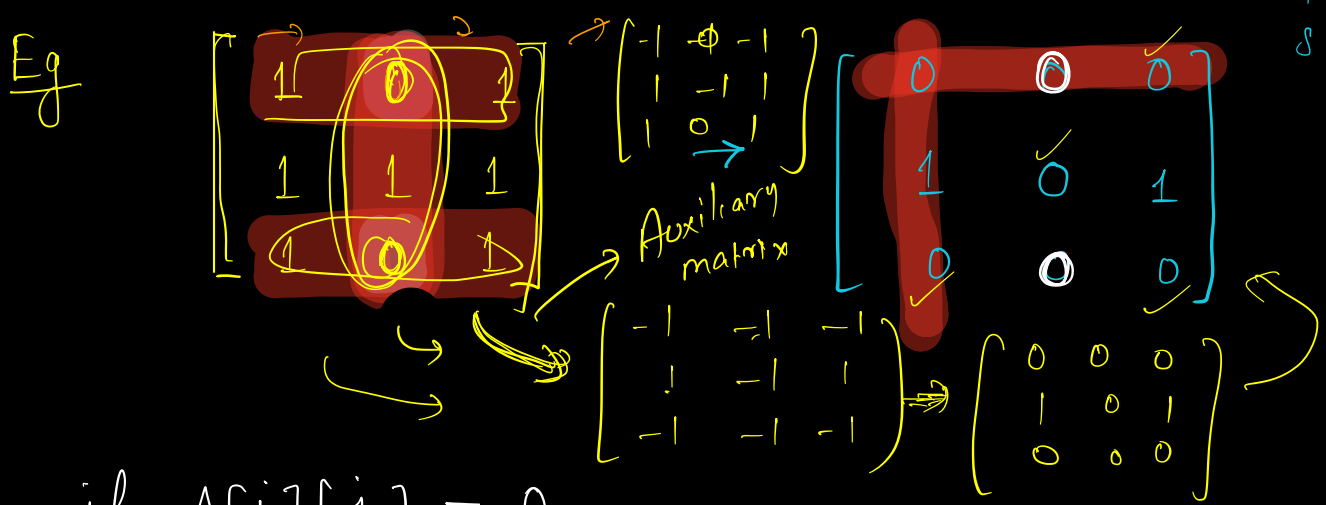
SC : O(1)

Q3) You are given a 2D integer matrix A, make all the elements in a row or column zero if A[i][j]=0.

Specifically, make entire ith row and jth column zero

Eg






//Best Approach
TC : O(N*M)
SC : O(1)

-50 - 50
51
-51

if A[i][j] = 0

↳ entire ith row should be 0 in the o/p.

↳ entire jth col. should be 0 in the o/p.

```
{ bool row[N], col[M]   // filled with false.
  (N) rows   (M) col.

  for(i=0; i<N; i++) {
      row[i] = false;
  }                                   } N
  for(i=0; i<M; i++) {
      col[i] = false;
  }                                   } M

  for(i=0; i<N; i++) {
      for(j=0; j<M; j++) {
          if(A[i][j] == 0) {
              // entire ith row & jth col should become 0.
              row[i] = true;
              col[j] = true;
          }
      }
  }                                   N*M

  for(i=0; i<N; i++) {
      for(j=0; j<M; j++) {            (i, j)
          // if row[i] == true or col[j] == true => 0
          if(row[i] == true or col[j] == true) {
              A[i][j] = 0;
          }
      }
  }                                   N*M
```

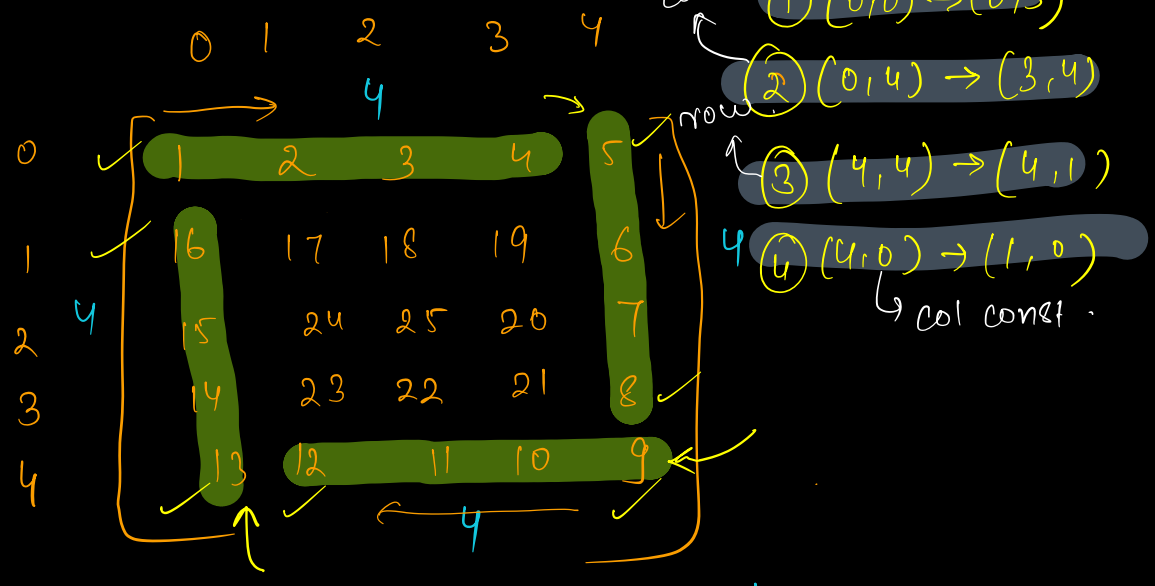
TC : O(N*M)

SC : O(N+M) → O(1)

Q4) Given an integer A, generate a square matrix filled with elements from 1 to $A^2$ in spiral order.

Eg A = 2 ⇒ $\begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}$

Eg A = 5



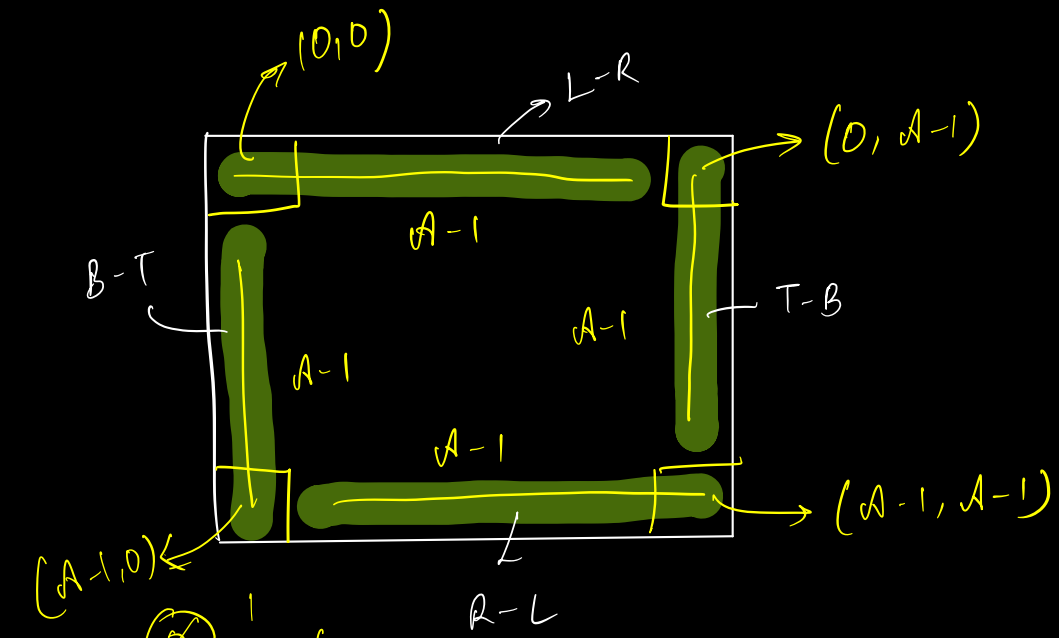

Direction 1 (Left to Right) → Row const

Direction 2 (Top to Bottom) → col const

Direction 3 (Right to left) → Row const

Direction 4 (Bottom to top) → col const.

A-1 times




A = 3

i = 0 1 2 3

j = 0 1 2 3

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 8 | 9 | 4 |
| 7 | 6 | 5 |

(0,0) → (1,1)

A = 1

A = 5

A = 3 (A = 1)

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 16 | 17 | 18 | 19 | 6 |
| 15 | 24 | 25 | 20 | 7 |
| 14 | 23 | 22 | 21 | 8 |
| 13 | 12 | 11 | 10 | 9 |

4

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | |
| 12 | 13 | 14 | 5 | |
| 11 | 16 | 15 | 6 | |
| 10 | 9 | 8 | 7 | |

A = 4

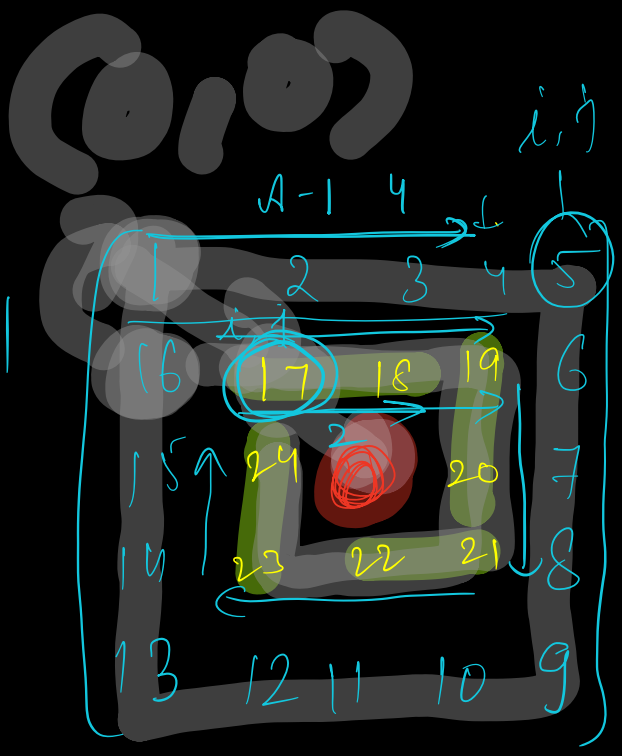
A = 2

A = 0

A = 4

```
int mat[A][A];

counter = 1
i = 0
j = 0
while(A > 1) {
  // Left to right A-1 times
  for(k=1; k<A; k++) {
    mat[i][j] = counter
    counter++;
    j++;
  }

  // Top to Bottom.
  for(k=1; k<A; k++) {
    mat[i][j] = counter;
    counter++;
    i++;
  }

  // Right to left.
  for(k=1; k<A; k++)
    mat[i][j] = counter.
    counter++;
    j--;

  // Bottom to Up.
  for(k=1; k<A; k++)
    mat[i][j] = counter.
    counter++;
    i--;

  A = A-2; i++; j++;
}
```

D1




D2

✓ A is changing
✓ i, j is changi-

i, j = (2, 2)
A = 1

D3

TC : O(A*A)
SC : O(1)

D4

```
if(A==1)
  mat[i][j] = counter;
return mat;
```

Q2) You are given an integer array A of length N comprising of 0's and 1's and an integer B.

You have to tell all indices of the array A that can act as centre of 2*B+1 length 0-1 alternating subarrays.

o/p array

2B+1

// 0-1 Alternating Subarray.

Eg: [ 1 0 0 1 0 1 0 1 0 0 0 1 0 ]

[ 0 1 0 1 0 1 . . .

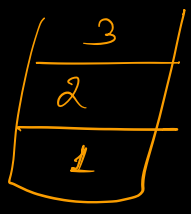
[ 1 0 1 0 1 0 . . .

Eg arr: [ 1 0 1 0 1 ]  (0 1 2 3 4)

B = 1

K = 2*B+1 = 3

ans = [1, 2, 3]




Eg:- [ 0 0 0 1 1 0 1 ]  (0 1 2 3 4 5 6)

B = 0

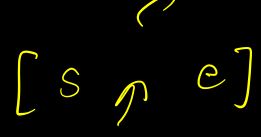
2*B+1 = 1

[0]

[1]

ans = [0, 1, 2, 3, 4, 5, 6]

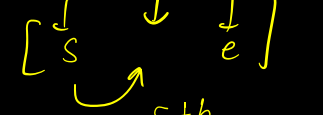

Alt. $\frac{s+e}{2}$




list ans;

$K = 2*B+1$

$s = 0$, $e = K-1$ // $s <= N-K$




N

N-K ← while (end < N) {

// [s, e]

TC : O(N*N)

SC : O(1)

s-e+1 → N

→ if (isAlternating(arr, s, e)) {

ans.add(s+b);

}

{ s++; e++; }

}

return ans;

// Given a start and end index, you need to return true if the subarray [s.. e] is alternating, else return false..

→ bool isAlternating (int arr[], int s, int e) {

for (i = s+1; i <= e; i++) {

if (arr[i] == arr[i-1]) {

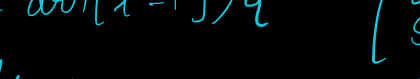
return false;

}

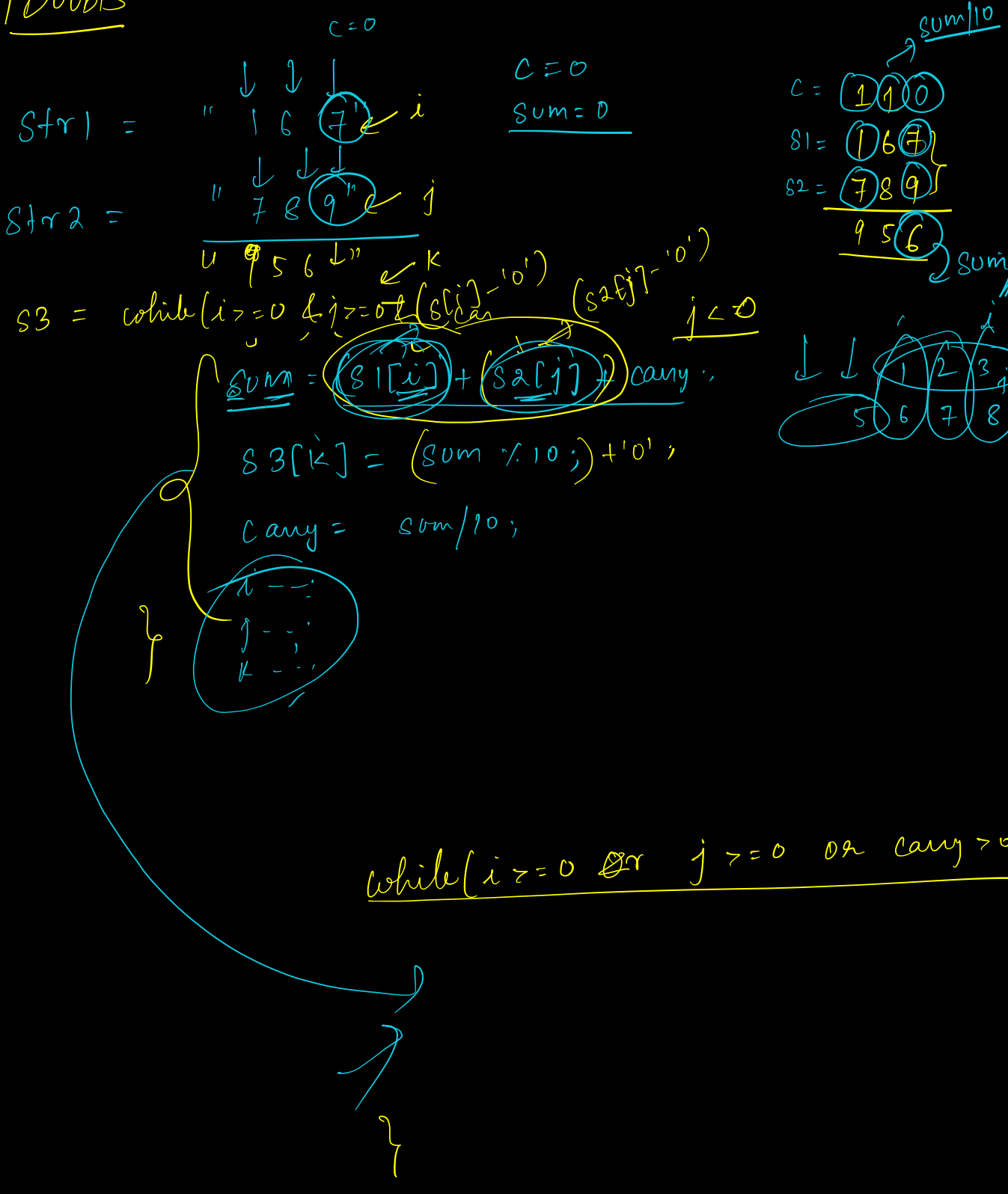
}

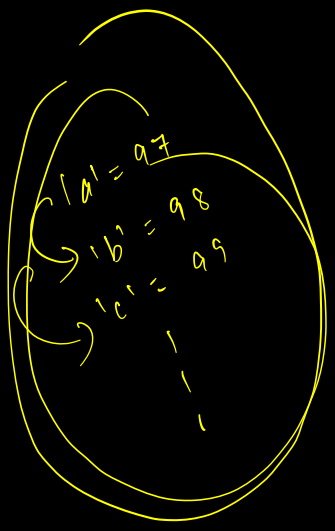
return true;

}



# //Doubts


c=0
Str1 = "167"
Str2 = "789"
i
j
c=0
sum=0
s3 = cohile(i>=0 && j>=0 && (s1[i]-'0') (s2[j]-'0')
k
j<0
sum = (s1[i]) + (s2[j]) + carry;
s3[k] = (sum %10;) +'0';
carry = sum/10;
i--;
j--;
k--;
sum/10
c = 110
s1 = 167
s2 = 789
956
sum/10
1 2 3 i
5 6 7 8
while(i>=0 or j>=0 or carry>0){




x x+1

'0' '1' '2' '3' '4'

'0' - '0' = 0

'1' - '0' = 1

'4' - '0' = 4

```
if(B > N){
    return -1;
}
```