Today's agenda

- Max Chunks to make Sorted 1
- Max Chunks to make Sorted 2
- Print boundary
- Print Spiral Square matrix
- Print Spiral rectangle matrix

Q) max Chunks to make Sorted I

↳ Given an arr[N] containing permutation of the integers in the range [0, n-1]. you have to split array into maximum possible number of chunks such that after individually sorting each chunk, the array gets sorted.

Ex: arr[5]: {1 0 2 3 4}

↳ ans = 4

Ex: arr[9]: {2 0 1 4 3 6 7 5 8}

↳ ans: 4

Ex: arr[5]: {4 3 2 1 0}

↳ ans: 1

//idea

arr[i] = {0, n-1}

Ex: arr[9]: {2 0 1 | 4 3 | 6 7 5 | 8}

obs: if max(0...i) is equal to i → valid chunk.

//Pseudo code

```
Public int maxChunksSortedone (int arr[N]){

    int maxtillnow = -∞;
    int Count = 0;

    for(int i=0; i<N; i++){
        maxtillnow = mathmax(maxtillnow, arr[i]);

        if(maxtillnow == i){ Count++;}
    }

    return Count;
}
```

T.C: O(N)

S.C: O(1)

Tracing

arr[9]: { 2 0 1 | 4 3 | 6 7 5 | 8 }

(indices: 0 1 2 | 3 4 | 5 6 7 | 8)

```
int maxtillnow = -∞;
int Count = 0;

for(int i=0; i<N; i++){
    maxtillnow = mathmax(maxtillnow, arr[i]);

    if(maxtillnow == i){ Count++;}
}

return Count;
```

i	maxtillnow	Count
0	2	0
1	2	0
2	2	1
3	4	1
4	4	2
5	6	2
6	7	2
7	7	3
8	8	4

Q) max Chunks to make Sorted 2

Ls Given an arr[N] containing permutation of the integers. you have to split array into maximum possible number of chunks such that after individually sorting each Chunk, the array gets sorted.

arr[9] =	0	1	2	3	4	5	6	7	8
23	10	18	35	27	48	60	40	55

Ls ans = 3

arr[6] =	0	1	2	3	4	5
13	7	12	22	18	26

Ls ans = 3

//idea

```
if (max(0...i) <= min(i+1....n-1)){
    //valid chunk
}
```

	0	1	2	3	4	5	6	7	8 (i)
arr[9] =	23	10	18	35	27	48	60	40	55
Pref[inmax[9]]:	23	23	23	35	35	48	60	60	60
Suff[inmin[9]]:	10	10	18	27	27	40	40	40	55

+1

Count = 0 + 1 + 1 + 1

//Pseudo Code

```
int maxChunksSorted2 (int arr[N]){

    int [] PrefixMax = PrefixMax(arr);  → H.W
    int [] SuffixMin = SuffixMin(arr);  → H.W

T.C: O(N)       int count=0;
S.C: O(N)
    for(int i=0; i<N-1; i++){
        if(PrefixMax[i]<=SuffixMin[i+1]){
            count++;
        }
    }

    count++;
    return count;
}
```

Q) Print boundary

↳ Given `mat[N][N]`, Print boundary in Clockwise direction.

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

↳ 1 2 3 6 9 8 7 4

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

1 2 3 4 5 10 15 20 25 24 23 22 21 16 11 6

//idea

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

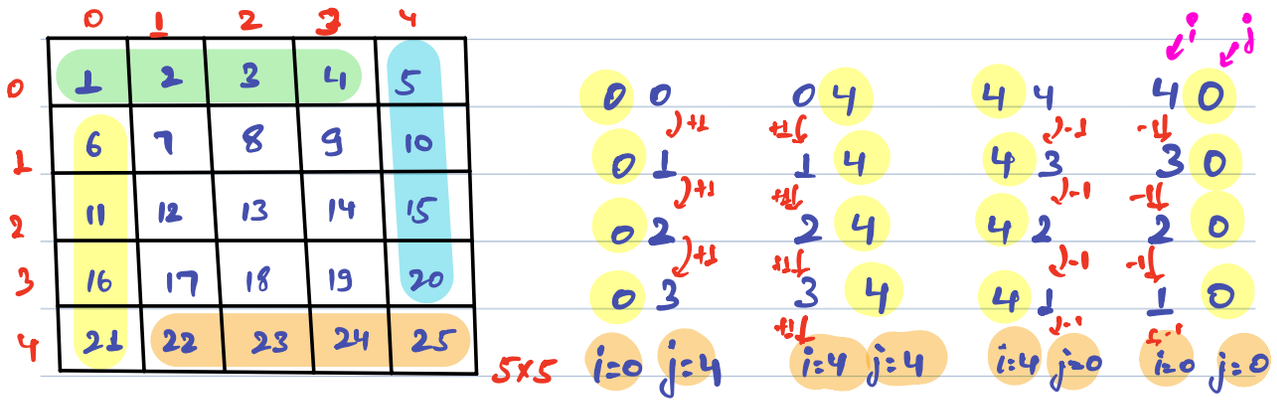
5x5

↳ 4 no. of 0th row
↳ 4 no. of last Col
↳ 4 no. of last row in reverse
↳ 4 no. of 0th Col in reverse

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

6x6

↳ N-1 no. of 0th row
↳ N-1 no. of last Col
↳ N-1 no. of last row in reverse
↳ N-1 no. of 0th Col in reverse


0 1 2 3 4
0 1 2 3 4 5
1 6 7 8 9 10
2 11 12 13 14 15
3 16 17 18 19 20
4 21 22 23 24 25
5X5
0 0 0 4 4 4 4 0
0 1 1 4 4 3 3 0
0 2 2 4 4 2 2 0
0 3 3 4 4 1 1 0
i:0 j:4
i:4 j:4
i:4 j:0
i:0 j:0
i
j
+1
-1

// Pseudo Code

```
void PrintBoundary (int mat[N][N]){
    int i=0;
    int j=0;

    for( k=0; k<N-1; k++){
        Print(mat[i][j]);
        j++;
    }

    for( k=0; k<N-1; k++){
        Print(mat[i][j]);
        i++;
    }

    for( k=0; k<N-1; k++){
        Print(mat[i][j]);
        j--;
    }

    for( k=0; k<N-1; k++){
        Print(mat[i][j]);
        i--;
    }
}
```

T.C: O(N)

S.C: O(1)

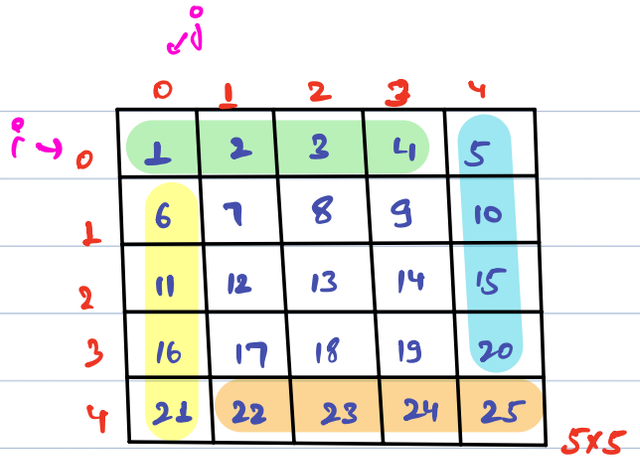
```
void PrintBoundary (int mat[N][N]){
    int i=0;
    int j=0;

    for( K=0; K<N-1; K++){
        Print(mat[i][j]);
        j++;
    }

    for( K=0; K<N-1; K++){
        Print(mat[i][j]);
        i++;
    }

    for( K=0; K<N-1; K++){
        Print(mat[i][j]);
        j--;
    }

    for( K=0; K<N-1; K++){
        Print(mat[i][j]);
        i--;
    }
}
```



1 2 3 4 5 10 15 20
25 24 23 22 22 16 11 6

Q) Square matrix Spiral

↳ Given mat[N][N], Print the Spiral form.

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

6×6

i	j	steps
0	0	N-1=5
+1 ↓	↓ +1	↓ -2
1	1	3
+1 ↓	↓ +1	↓ -2
2	2	1

↳ do PrintBoundary till your steps >=1.

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

5×5

i	j	steps
0	0	4
+1 ↓	↓ +1	↓ -2
1	1	2
2	2	0

//Pseudo code

```
void PrintSpiralSquare(int mat[N][N]){
    int i=0;
    int j=0;
    int steps = N-1;

    while(steps >= 1){
        for(k=0; k<steps; k++){
            Print(mat[i][j]);
            j++;
        }
        for(k=0; k<steps; k++){
            Print(mat[i][j]);
            i++;
        }
        for(k=0; k<steps; k++){
            Print(mat[i][j]);
            j--;
        }
        for(k=0; k<steps; k++){
            Print(mat[i][j]);
            i--;
        }
        i++;
        j++;
        steps = steps-2;
    }

    if(steps == 0){ Print mat[i][j]; }
}
```

T.C: $O(N^2)$

S.C: $O(1)$

```
void PrintSpiralSquare (int mat[N][N]){
    int i=0;
    int j=0;
    int steps = N-1;

    while (steps >= 1) {
        for (k=0; k<steps; k++){
            Print (mat[i][j]);
            j++;
        }
        for (k=0; k<steps; k++){
            Print (mat[i][j]);
            i++;
        }
        for (k=0; k<steps; k++){
            Print (mat[i][j]);
            j--;
        }
        for (k=0; k<steps; k++){
            Print (mat[i][j]);
            i--;
        }
        i++;
        j++;
        steps = steps-2;
    }
    if (steps == 0) { Print mat[i][j]; }
}
```

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

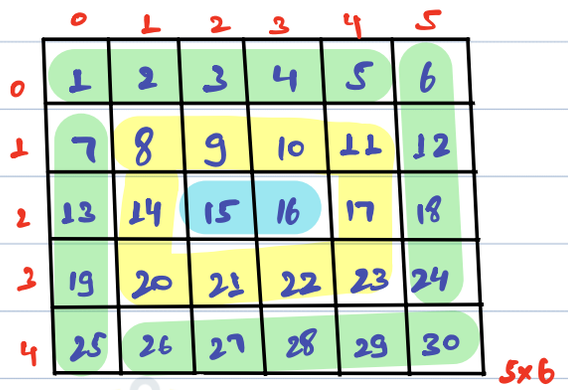
6x6

i	j	steps
0	0	5
1	1	3
2	2	1
3	3	-1

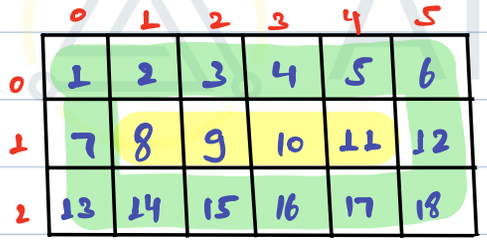
(, exit

Q) Spiral matrix

↳ Given `mat[n][m]`, return all elements of the matrix in spiral order.



i	j	oSteps	cSteps
0	0	4	0



i	j	oSteps	cSteps
0	0	4	2
1	1	2	0

```
void PrintSpiralSquare(int mat[n][m]){
    int i = 0;
    int j = 0;
    int rSteps = n-1;
    int cSteps = m-1;

    while(rSteps >= 1 && cSteps >= 1){
        for(k=0; k<cSteps; k++){
            Print(mat[i][j]);
            j++;
        }

        for(k=0; k<rSteps; k++){
            Print(mat[i][j]);
            i++;
        }

        for(k=0; k<cSteps; k++){
            Print(mat[i][j]);
            j--;
        }

        for(k=0; k<rSteps; k++){
            Print(mat[i][j]);
            i--;
        }
        i++;
        rSteps = rSteps-2;
        j++;
        cSteps = cSteps-2;
    }

    if(rSteps == 0){
        for(int k=0; k<cSteps+1; k++){
            Print(mat[i][j]);
            j++;
        }
    }
}
```

T.C: O(n*m)

S.C: O(1)

```
        else if (csteps==0){
            for(int k=0; k<csteps+1; k++){
                Print(mat[i][j]);
                i++;
            }
        }
    }
```

Q) Flip bits

↳ Given an arr[N] that contains 1 and 0 only. Find the subarray when flipped can give max 1's in the entire array. (flipping once)

ex: arr[5]:

	0	1	2	3	4	
	1	0	1	0	0	
1-4	1	1	0	1	1	: 4
0-2	0	1	0	3	4	: 1
2-4	1	0	0	1	1	: 3

arr[7]:

	0	1	2	3	4	5	6	
	0	0	1	0	0	1	0	
0-4	1	1	0	1	1	1	0	: 5

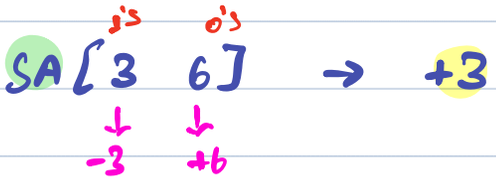
//idea 1

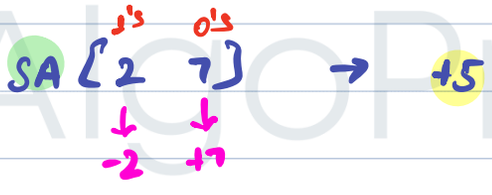
↳ Try flipping every subarray and choose the one which gives us the maximum.

T.C: $O(N^2) * O(N) \approx O(N^3)$

//idea2

Obs1:





→ flip subarray with max no. of net gain

net gain:

↳ 1 → 0 : -1

↳ 0 → 1 : +1

	0	1	2	3	4	5	6
arr[i]:	0	0	1	0	0	1	0

⇓

arr[i]:	+1	+1	-1	+1	+1	-1	+1

↓

find the subarray with max gain

↓

max subarray sum

↳ Kadane's algo

arr[i]:	+1	+1	-1	+1	+1	-1	+1

cmax : 0 1 2 1 2 3 2 3

omax : 0 1 2 3

ans = 5

//Pseudo code

```
int maxones (int arr[N]){
    int count=0;
    for (int i=0; i<N; i++) {
        if (arr[i]==0) {
            arr[i]=1;
        }
        else {
            arr[i]=-1;
            count++;
        }
    }
    int csum=0;
    int osum=-∞;

    for (int i=0; i<N; i++) {
        if (csum>=0) { csum+=arr[i]; }
            else { csum=arr[i]; }

        osum = Math.max(osum, csum);
    }
    if (osum>0){ return osum+count; }
        else { return count; }
}
```

T.C: O(N)

S.C: O(1)

↳ you didn't come this far only to come this far.