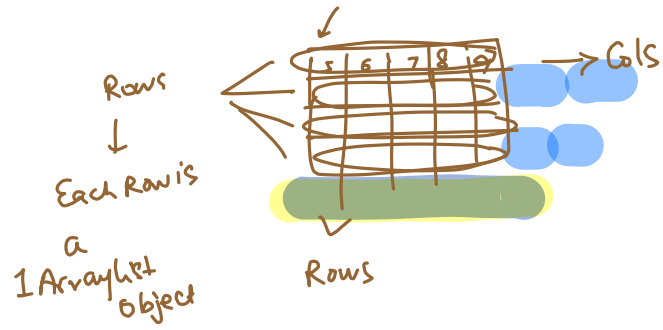
2D ARRAYS & ARRAYLISTS (2D)

`int[][] arr = new int[R][C];`

(Fixed Rows and cols)

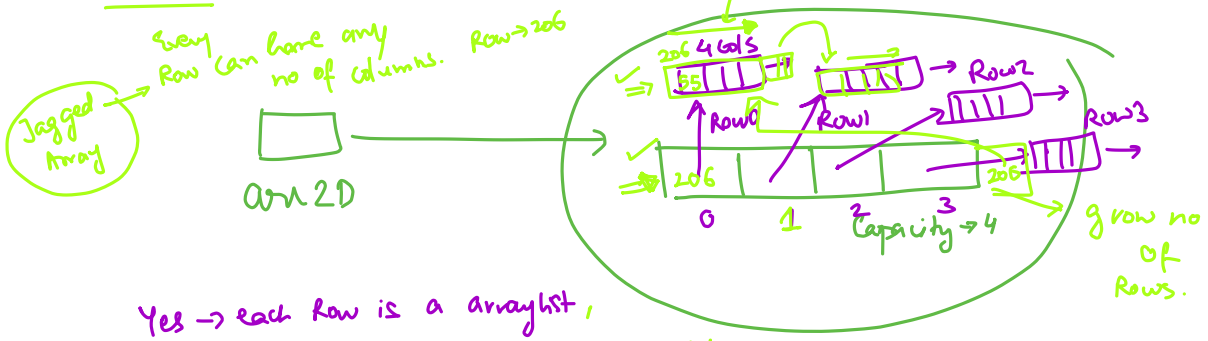
2D ArrayList

- Arraylist → Dynamic Array. [10|20|30|40] → grow in size
- 2D Arraylist → Dynamic 2D Array



`Arraylist < objects > arr2D = [ optional value → initial capacity`

- `Arraylist < Arralist <Integer> > arr2D = new Arraylist<>();`

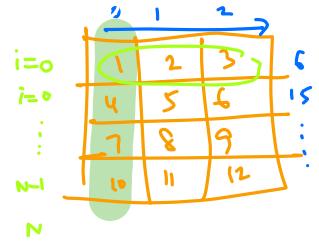


Yes → each Row is a arraylist, hence you can add more cols in any Row.

Arrays.asList(arr);

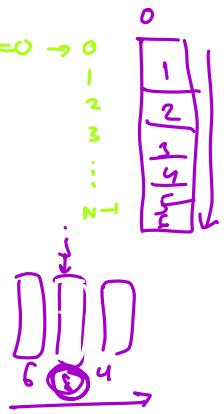
2D Array / Arraylist
- fixed
- dynamic

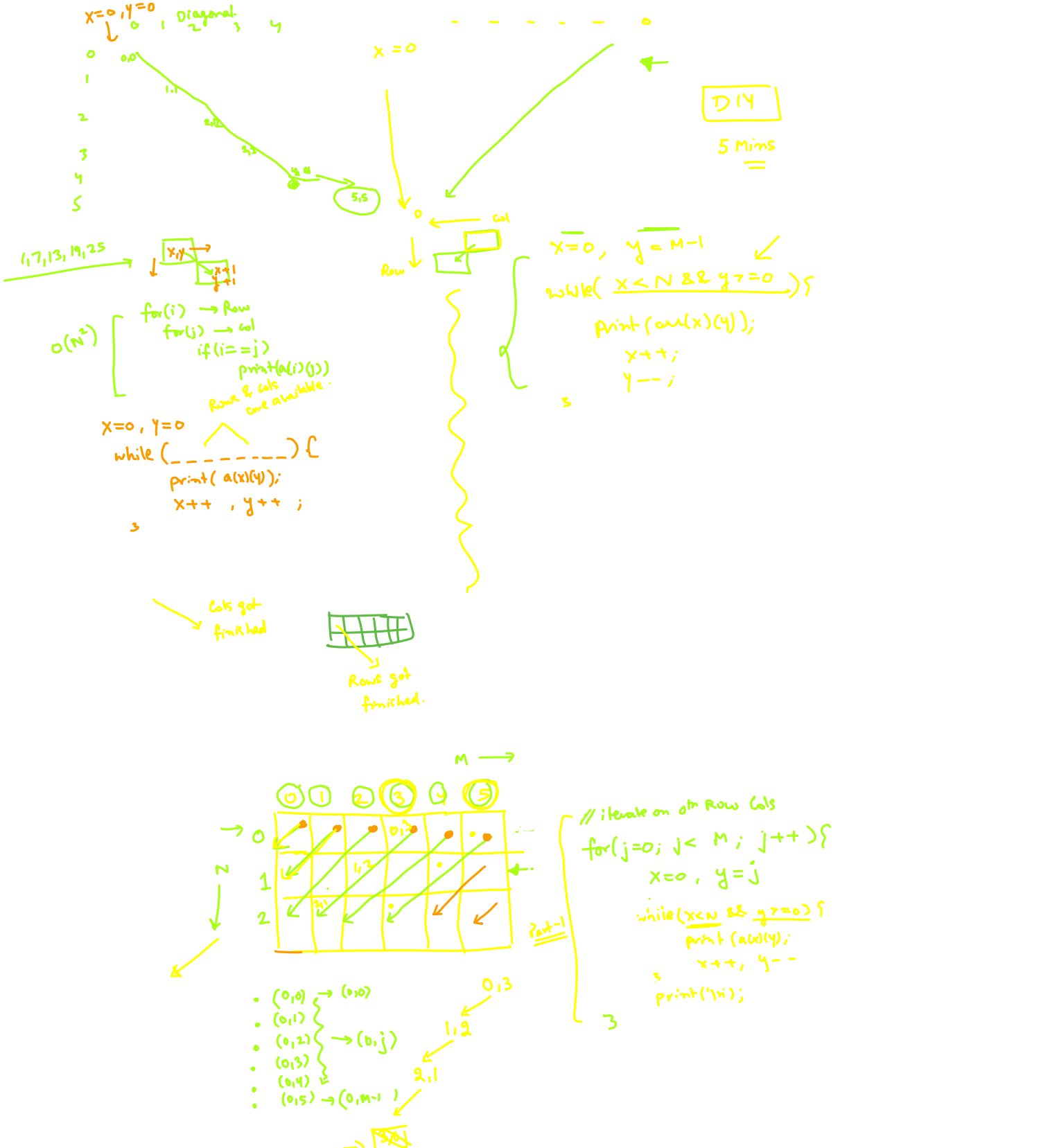
(8) Given mat[N][M] cols, print Row wise Sum.




```
for(i=0; i<N; i++){
    sum=0 // Reset sum for every Row.
    for(j=0; j<M; j++){
        sum = sum+a[i][j];
    }
    print(sum);
}
```

```
sum=0
for(row=0; row<N; row++){
    sum = sum + a[row][col];
}
if(sum > largest_sum){
    largest_sum = sum;
    best_col = col.
}
```




x=0, y=0
Diagonal
0 1 2 3 4
0 1 2 3 4 5
0,0
1,1
2,2
3,3
4,4
5,5
x = 0
0 1 2 3 4
0
DIY
5 Mins
Row
col
1,7,13,19,25
x,y
x+1
y+1
for(i) → Row
for(j) → col
if(i==j)
print(a[i][j])
O(N²)
Rows & cols are available.
x=0, y=0
while(_____){
print(a[x][y]);
x++ , y++ ;
}
x=0, y=M-1
while(x<N && y>=0){
print(a[x][y]);
x++;
y--;
}
Cols got finished
Rows got finished.
M
0 1 2 3 4 5
N
0
1
2
0,3
1,4
2,5
1,2
2,1
(0,0) → (0,0)
(0,1)
(0,2) → (0,j)
(0,3)
(0,4)
(0,5) → (0,M-1)
SOL
// iterate on all Row cols
for(j=0; j< M ; j++){
x=0 , y=j
while(x<N && y>=0){
print(a[x][y]);
x++, y--
}
print('\n');
}
Part-1

every cell once.

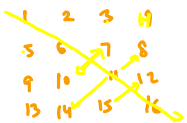
$O(N \cdot M)$

c=0 c=1 c=2 c=3

Strivers

i=1
i=2
i=3

Transpose.



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

a(i,j) a(j,i)

L ⟶ R R ⟶ L

go to every Row
↳ Reverse(it)

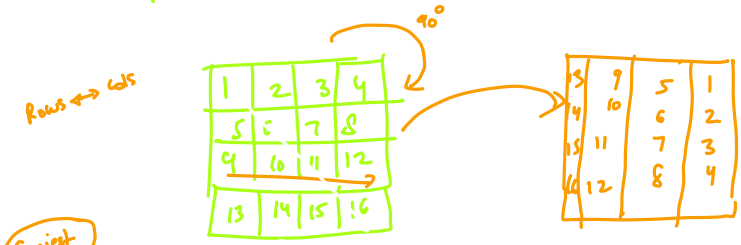
6, 5, 4, 3, 2, 1 ⟶

DM

i=0
j=N-1

while (i<j)
[swap(a[i], a[j]);
i++ j--]

Square matrix of N×N, rotate it 90°.



Easiest Hacky: