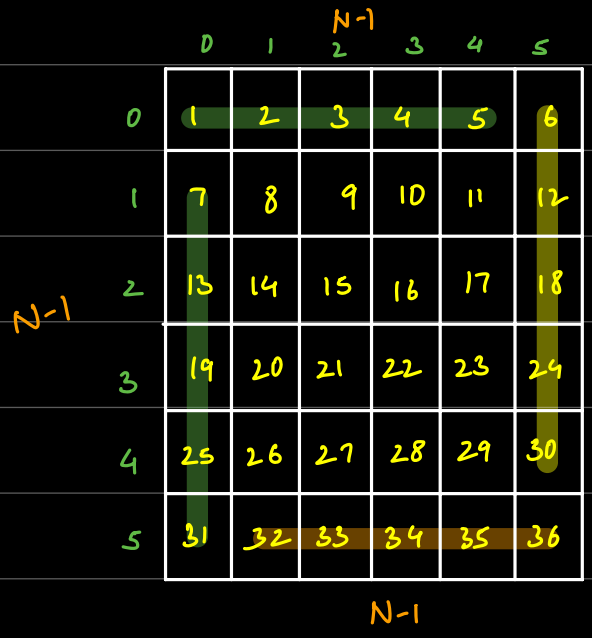
9 June Doubts

square

Q1) Given a N x N matrix, print the boundry elements in clockwise direction ?



00 → 01 → 02 → 03

N = 6

1 2 3 4 5 6 12 18 24 30 36 35 34 33 32 31

25 19 13 7

4 for loops

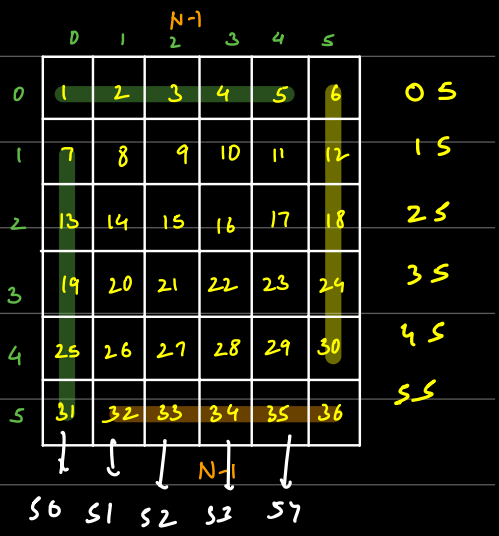
row = 0 col = 0

```
for(i=1; i<=N-1; i++) {
    print(A[row][col])
    col++
}
```

i :	1	2	3	4	5	6
r :	0	0	0	0	0	0
c :	0	1	2	3	4	5

A[0][0] A[0][1] A[0][2] A[0][3] A[0][4]

```
row = 0        col = 0
for (i=1 ; i <= N-1 ; i++) {
    print (A[row] [col])
    col++
}
for (i = 1 ; i <= N-1 ; i++) {
    print (A[row][col])
    row++
}
for (i=1 ; i <= N-1 ; i++) {
    print (A[row] [col])
    col--
}
for (i=1 ; i <= N-1 ; i++) {
    print (A[row][col])
    row--
}
```

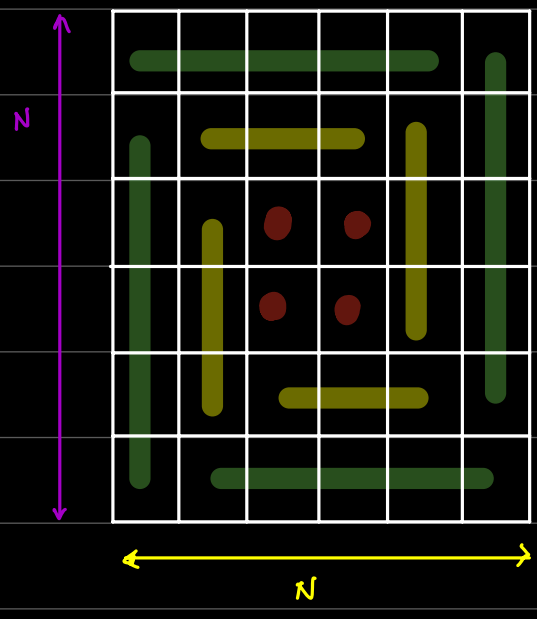


TC: O(N)

Meta, Google

Q2) Print a $(N \times N)$ matrix in spiral order?

r	c	N	iter
0	0	6	5
1	1	4	3
2	2	2	1
3	3	0	

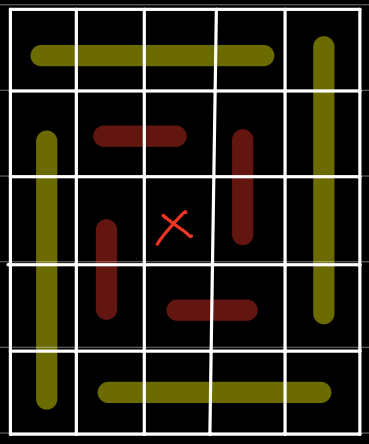


Size	iterations	row	Col
$N \times N$	$N-1$	0 $\curvearrowright +1$	0 $\curvearrowright +1$
$(N-2) \times (N-2)$	$N-3$	1 $\curvearrowright +1$	1 $\curvearrowright +1$
$(N-4) \times (N-4)$	$N-5$	2	2

TC: $O(N^2)$

SC: $O(1)$

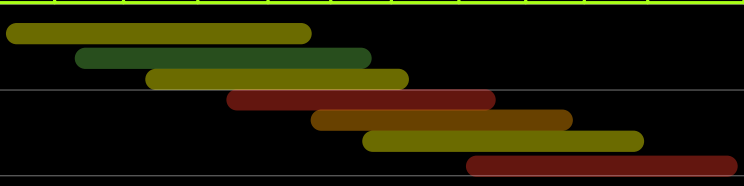
```
row = 0      col = 0
while ( N > 1 ) {
    for(i=1 ; i<=N-1 ; i++) {
        print (A[row][col])
        col++
    }
    for(i=1 ; i<=N-1 ; i++){
        print (A[row][col])
        row++
    }
    for(i=1 ; i<=N-1 ; i++) {
        print(A[row][col])
        col--
    }
    for(i=1; i<=N-1 ; i++){
        print (A[row][col])
        row--
    }
    row++
    col++
    N = N-2
}
if (N==1) { print (A[row][col]) }
```



r	c	N	itu
0	0	5	9
1	1	3	2
2	2	1	0
3	3	-1	

Q

0	1	2	3	4	5	6	7	8	9	10
3	2	5	-1	6	4	2	7	1	0	9



N = 11

N, K

K = 5

S	e
0	4
1	5
2	6
3	7
4	8
5	9
6	10

S	e
0	K-1
⋮	

N-K N-1

s e

[s N-1] len = k

N - x - s + x = k

N - s = k

s = N - k

[0 N-k] → N-k+1

First subarray of size k [0 , k-1]

Last subarray of size k [N-k , N-1]

Total no. of subarrays of size k → N-k+1

Break (10:49 - 11:00)

Q3) Given an array find number of subarrays of length k.

Q3) Given an array and subarray size, print the starting and ending of all subarrays of size = k

	0	1	2	3	4	5	6
A =	5	7	9	6	3	2	1

k = 4

s	e
0	3
1	4
2	5
3	6

```
s = 0          e = k-1
while (e <= N-1) {
    print (s, e)
    s++ , e++
}
```

(Q4) Find max sum subarray for all subarrays of size = k

N = 9 K = 4

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

s	e	sum
0	3	7
1	4	8
2	5	7
3	6	8
4	7	11
5	8	8

Ans : 11

s = 0 e = k-1 maxsum = $-\infty$

```
while (e <= N-1) {
    print (s, e)
    sum = 0
    for (i=s; i<=e; i++) {
        sum = sum + A[i]
    }
    maxsum = max (maxsum, sum)
}
```

sum (A[s:e])

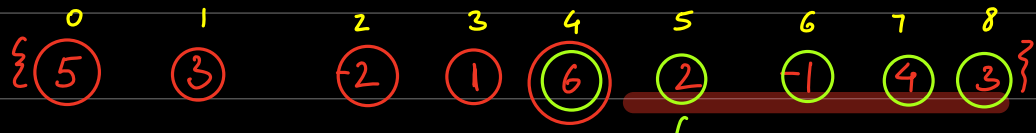
↓

pf array

TODO

New approach

N = 9 K = 4



s = 0 e = 3 (Iterate & Calculate the sum)

sum = 7

s = 1 e = 4

subtracting A[0] and adding A[4]

sum = 7 - 5 + 6 = 8

s = 2 e = 5

subtracting A[1] and adding A[5]

sum = 8 - 3 + 2 = 7

s = 3 e = 6

subtracting A[2] and adding A[6]

sum = 7 - (-2) + (-1) = 8

s = 4 e = 7

subtracting A[3] and adding A[7]

sum = 8 - 1 + 4 = 11

s = 5 e = 8

subtracting A[4] and adding A[8]

sum = 11 - 6 + 3 = 8

```
sum = 0 , max sum = -∞
for (i=0; i<=k-1; i++){
    sum = sum + A[i]
}
```

K — [k, N-1]

maxsum = max (maxsum, sum)

```
s = 1        e = k
while (e<=N-1){
    # index I am gaining → e
    # index I am losing → s-1
    sum = sum + A[e] - A[s-1]
    maxsum = max (maxsum, sum)
    s++, e++
}
return maxsum
```

N-k

TC: O(N)

SC: O(1)

[s i] len = k

Q5) Given an array of size N and a number B return min number of swaps required to bring all elements less than or equal to B together.

B = 6

A: { 1 10 12 14 3 7 5 }