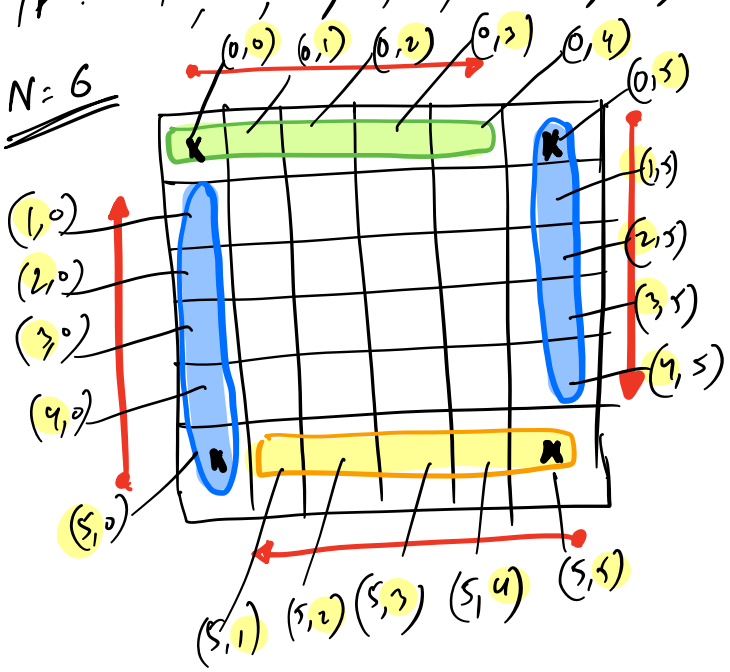
Q Given a N X N matrix.
Print the boundary in clockwise fashion!

I/P

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

O/P : 1, 2, 3, 4, 5, 10, 15, 20, 25, 24, 23, 22, 21, 16, 11, 6

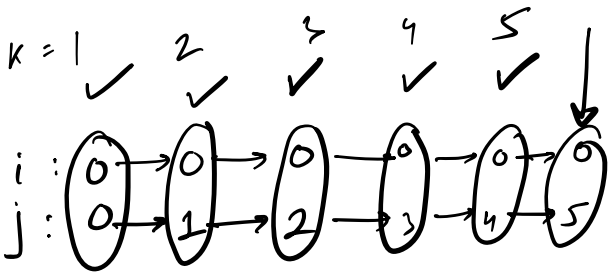
N = 6



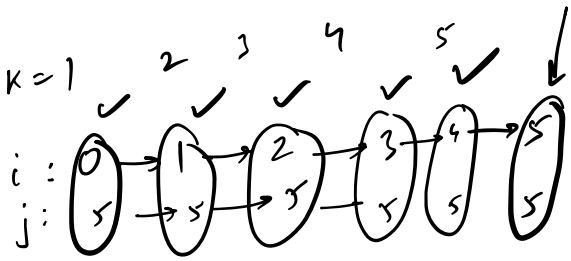
N-1

1. 5 elements to right
2. 5 — down
5 — left
5 — up

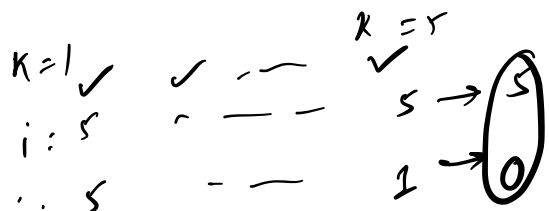
```
i = 0, j = 0;                                   O(N)
for (k = 1; k < N; k++) {
    print ( A[i][j]);
    j++;
}
```



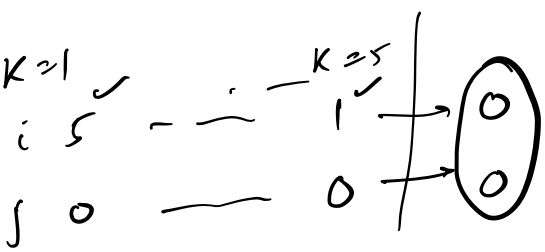

```
                                                O(N)
for ( k = 1; k < N; k++) {
    print( A[i][j]);
    i++;
}
```




```
                                                O(N)
for ( k = 1; k < N; k++) {
    print( A[i][j]);
    j--;
}
```




```
                                                O(N)
for ( k = 1; k < N; k++) {
    print( A[i][j]);
    i--;
}
```

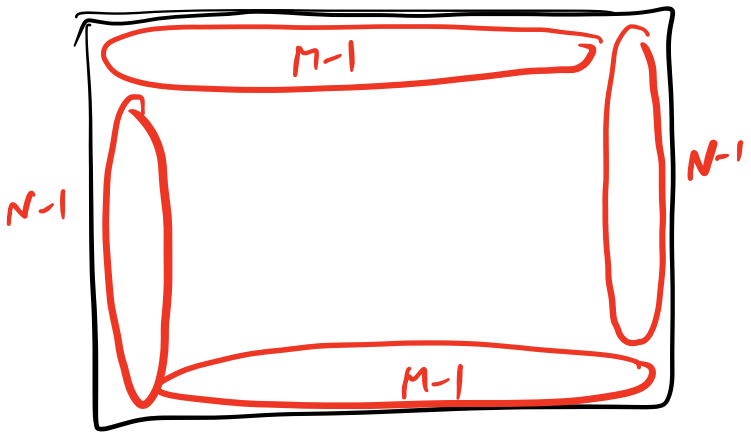


TC = $4 \times O(N)$

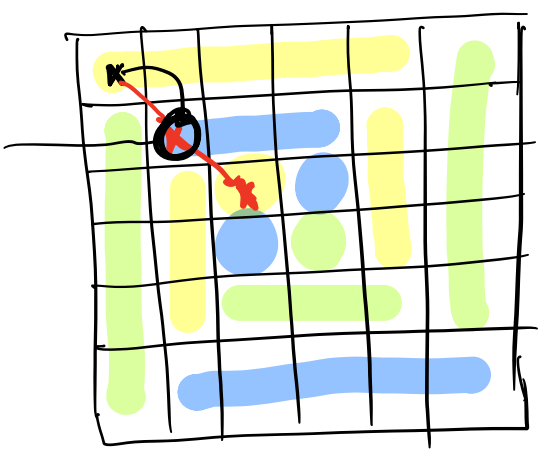
TC : O(N)

SC = O(1)

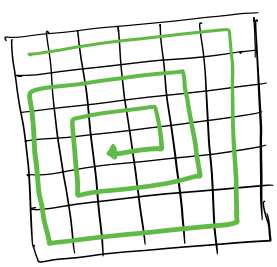
NXM matrix!



Given a NXN matrix.
Print it in SPIRAL fashion!



	i	j
N=6	0	0
N=4	1	1
N=2	2	2
N=0	STOP!	



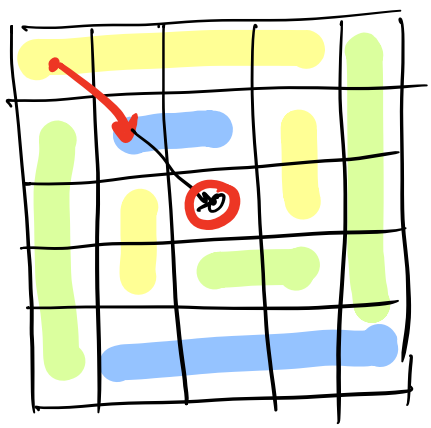
$TC = O(N^2)$

$SC = O(1)$

```
i = 0, j = 0;
While (N > 1) {

    for (k = 1; k < N; k++){
        print (A[i][j]);
        j++;
    }

    for (k = 1; k < N; k++) {
        print( A[i][j]);
        i++;
    }

    for (k = 1; k < N; k++) {
        print( A[i][j]);
        j--;
    }

    for (k = 1; k < N; k++){
        print( A[i][j]);
        i--;
    }

    N = N-2;
    i++;
    j++;

}
if (N == 1) {
    print(A[i][j]);
}
```

N → ODD

N = 5

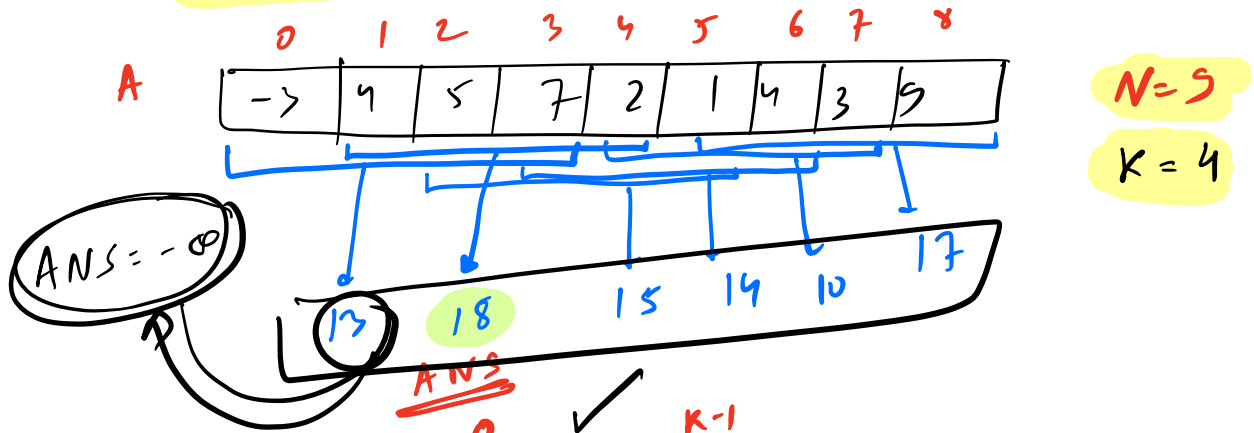


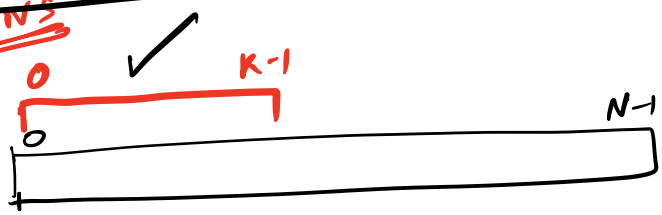
N	i	j
✓5	0	0
✓3	1	1
1	2	2

N = 1

N - 1 → 0

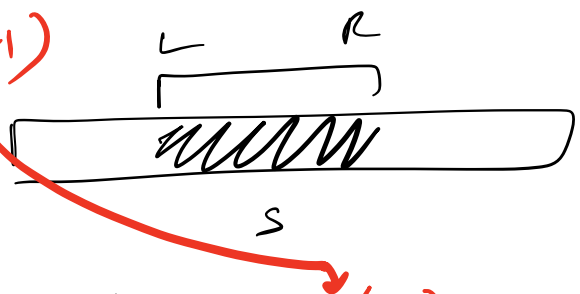
Q Given N elements, return the Maximum Subarray Sum of len = K !

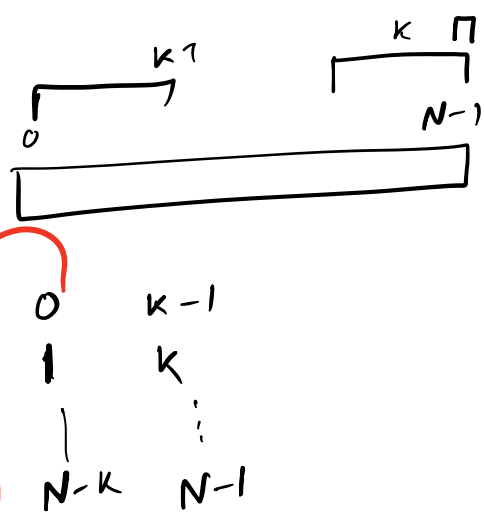




1) BF

```
L=0, R=K-1
ANS = -∞
while (R<N) {
  // [L,R]
    S=0;
    for(i=L; i<=R; i++){
        S = S + A[i];
    }
    ANS = max(ANS, S);

    L++, R++;
}
```





SA of size K in an arr of size N

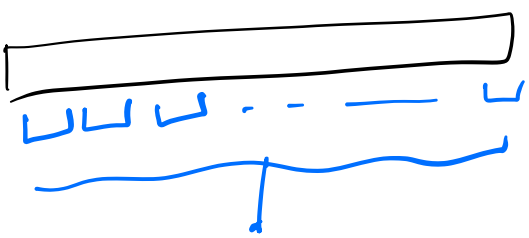
$= N - K + 1$

$TC = K \times (N-K+1)$

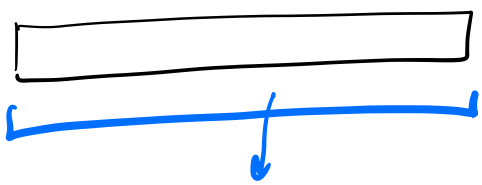
$TC = O(K(N-K))$

$K = N$

$K = 1$



$O(N)$



$O(N)$

$K = N/2$

$\#SA \rightarrow N/2 \times N/2 = N^2/4$

$\#SA \approx K$

$N - K = K$

$N = 2K$

$K = N/2$

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

II) Prefix Sum

1. Build the PS Array! → TC: O(N), SC: O(N)

2.
```
L = 0, R = K-1
ANS = -∞
while (R < N) {
  // [L, R]
  S = PS[R] - PS[L-1]
  ANS = max(ANS, S);

  L++, R++;
}
ret ANS //
```

N - K + 1

L = 0 → 0

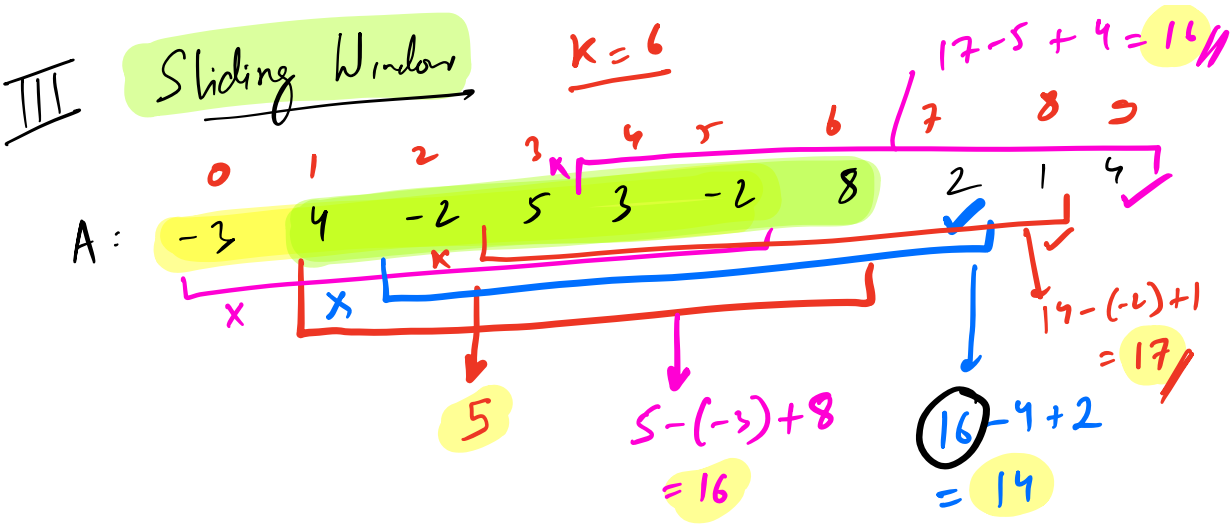
PS[L-1] — O(1)

TC = O(N + N - K)

TC = O(N - K) → O(N)

SC = O(N)

III Sliding Window



ANS = 17

K=6, N=10

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

ANS

Iterate over them & find the Sum

Sum = Sum − A[0] + A[6]

Sum = Sum − A[1] + A[7]

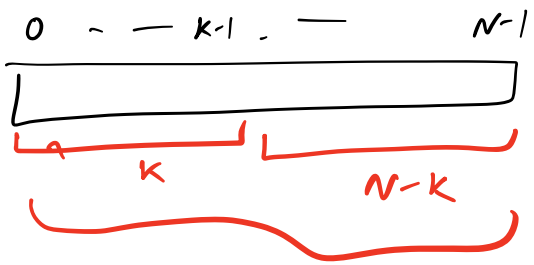
Sum = Sum − A[2] + A[8]

Sum = Sum − A[3] + A[9]

Sum = Sum − A[L-1] + A[R]

```
ANS = -∞
sum = 0
for (i=0; i<K; i++) {
    Sum = sum + A[i];
}
ANS = Sum
L = 1, R = K;
while (R<N) {
    sum = sum - A[L-1] + A[R];
    ANS = max(ANS, sum);
    L++, R++;
}
ret ANS
```

O(K)



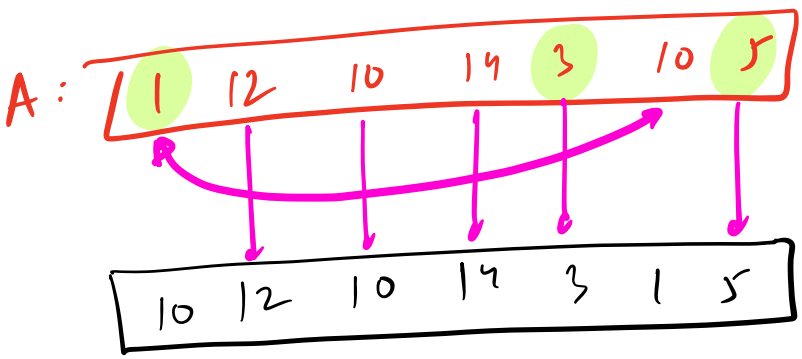
O(N-K)

TC = O(K+N-K)

TC = O(N)

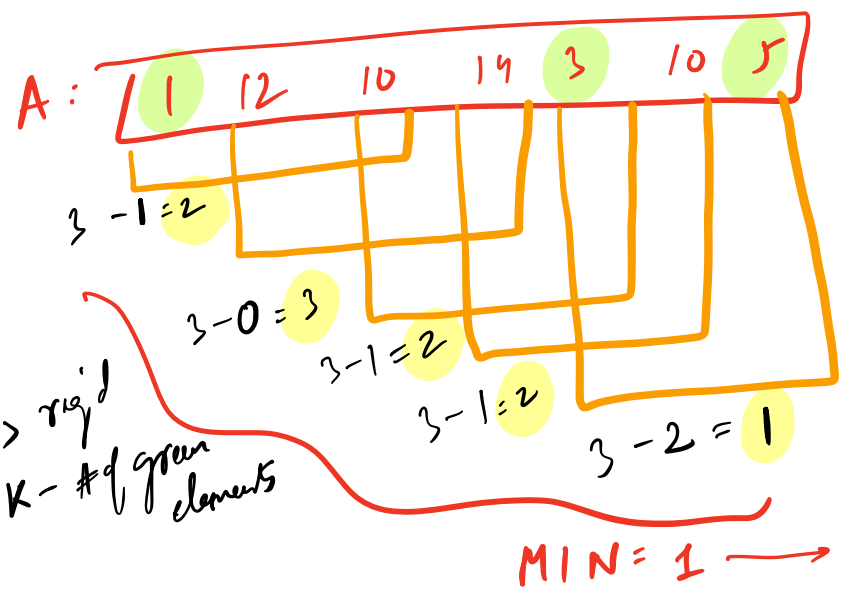
SC = O(1)

Q Given an array A & an int B
find the MIN no. of SWAPS required to bring
all the elements <= B together!



B = 8

ANS
1



B = 8

Cnt of elements
<= B = 3

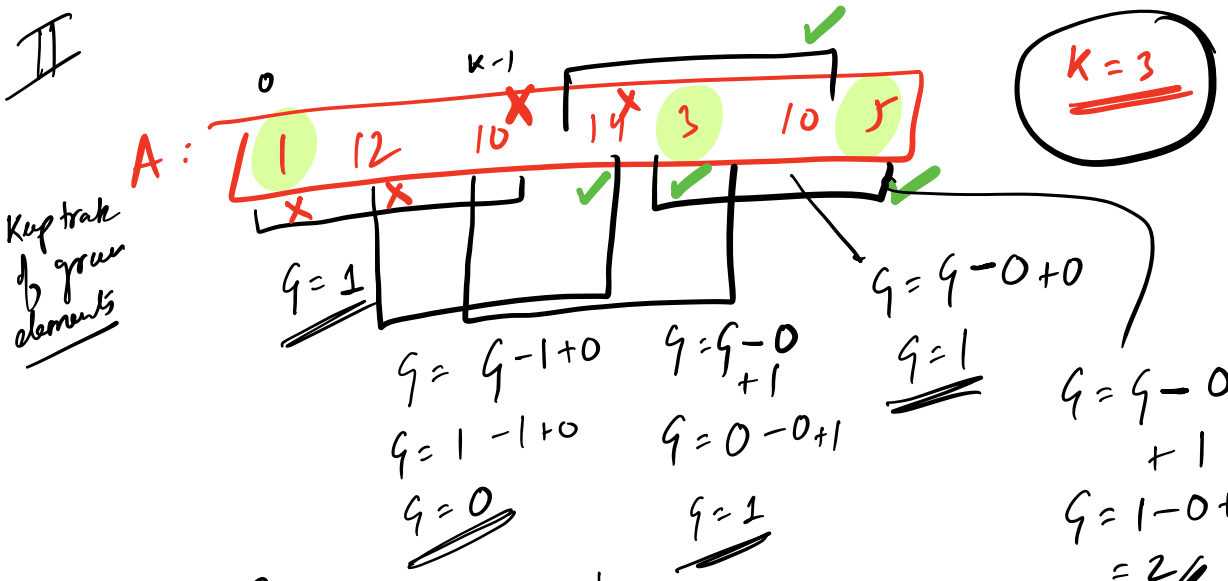
K = 3

Swaps req'd
= K - # of good elements

MIN = 1 → ANS!

1) BF → TC: $O(N^2)$

II




```
ANS = ∞
q = 0
for(i=0; i<K; i++){
    if(A[i] <= B) q++;
}
ANS = K - q

L = 1, R = K;
while(R < N){
    if(A[L-1] <= B) q--;
    if(A[R] <= B) q++;
    ANS = min(ANS, K - q);
    L++, R++;
}
ret ANS
```

```
K = 0
for(i=0; i<N; i++){
    if(A[i] <= B) K++;
}
```

TC = O(N)

SC = O(1)