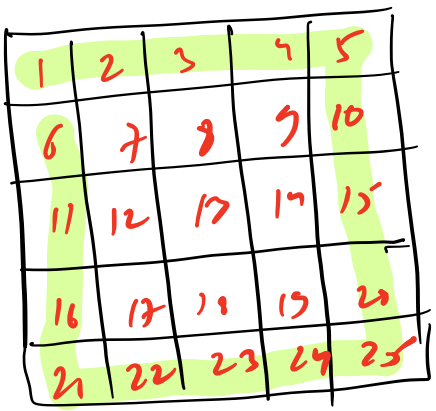
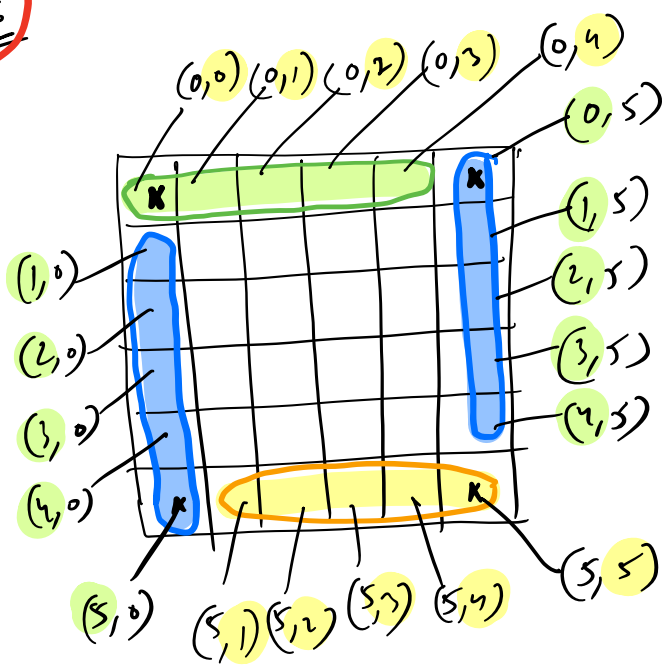
Q Given a N x N matrix.
Print the boundary in clockwise fashion!

I/P



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

N = 6



STEPS: N-1

1. 5 elements to right!
2. 5 ——— down!
3. 5 ——— left!
4. 5 ——— up!

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

N=6 → O(N)

k=1	2	3	4	5	6
i: 0	0	0	0	0	0
j: 0	1	2	3	4	5

O(N)

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

k=1	
i: 0	5
j: 5	5

O(N)

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

O(N)

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

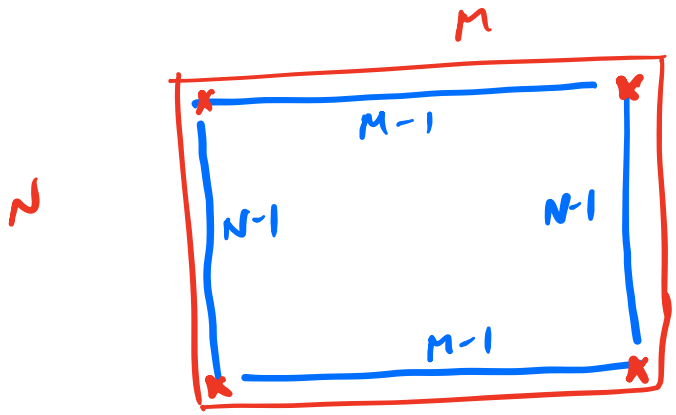
i → 0
j → 0

4N

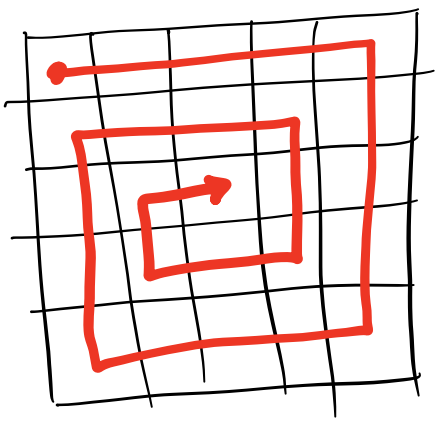
TC = O(N)

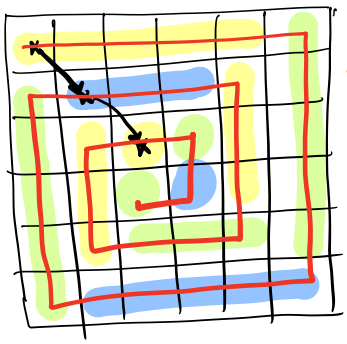
SC = O(1)

Q SAME Q for rect.. N x M



Q Given a N x N matrix.
Print it in SPIRAL fashion!



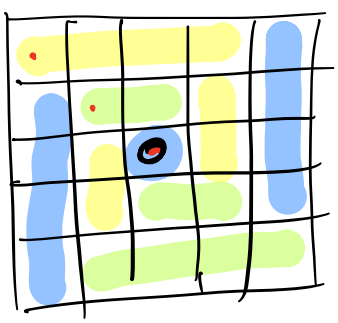


N = 6 → N = 4 → N = 2 → N = 0 (−2 each step)

i	j
0	0
1	1
2	2

(+1 each step)

```
i = 0, j = 0;
while (N > 1) {
    for (k = 1; k <= N-1; k++) {
        print(A[i][j]);
        j++;
    }
    for (k = 1; k <= N-1; k++) {
        print(A[i][j]);
        i++;
    }
    for (k = 1; k <= N-1; k++) {
        print(A[i][j]);
        j--;
    }
    for (k = 1; k <= N-1; k++) {
        print(A[i][j]);
        i--;
    }
    N = N-2, i++, j++;
}
```



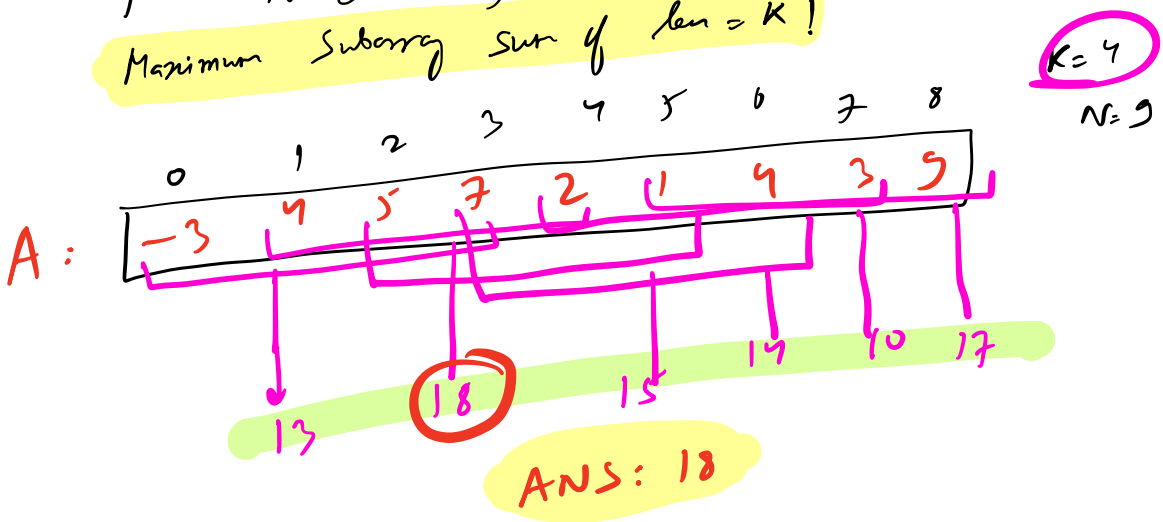
N	i	j
5	0	0
3	1	1
1	2	2

$TC = O(N^2)$

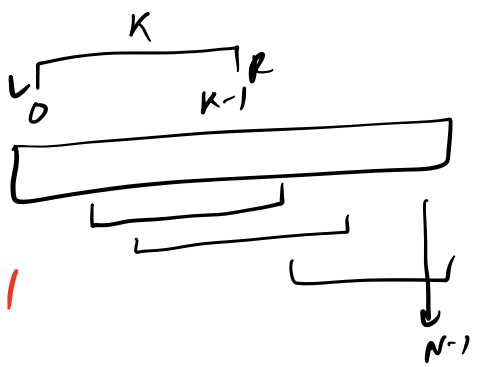
$SC = O(1)$

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

Q1 Given N elements, return the Maximum Subarray sum of len = K!



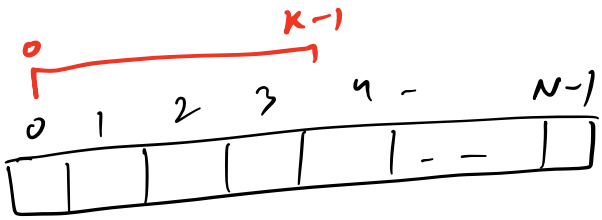
1) BF



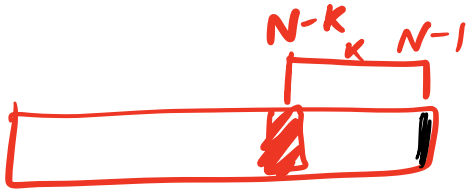

```
L=0, R=K-1, mC=-∞
while (R<N) {          → N-K+1
  // [L, R]
  s=0
  for(i=L; i<=R; i++) {
      s += A[i];
  }
  if( s > mC) { mC = s; }
  L++, R++;
}
ret mC;
```

O(K)

TC = O((N-K)K)



SA	L	R
1	0	K-1
2	1	K
3	2	K+1
⋮		
N-K+1	N-K	N-1

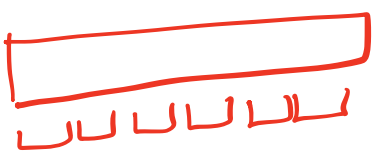


SA of len K in A of size N = N-K+1

O(N-K)

TC = (N-K+1)(K)

K = 1



TC = O(N)

K = N/2

N/2 x N/2

TC = N^2

K = N

TC = O(N)

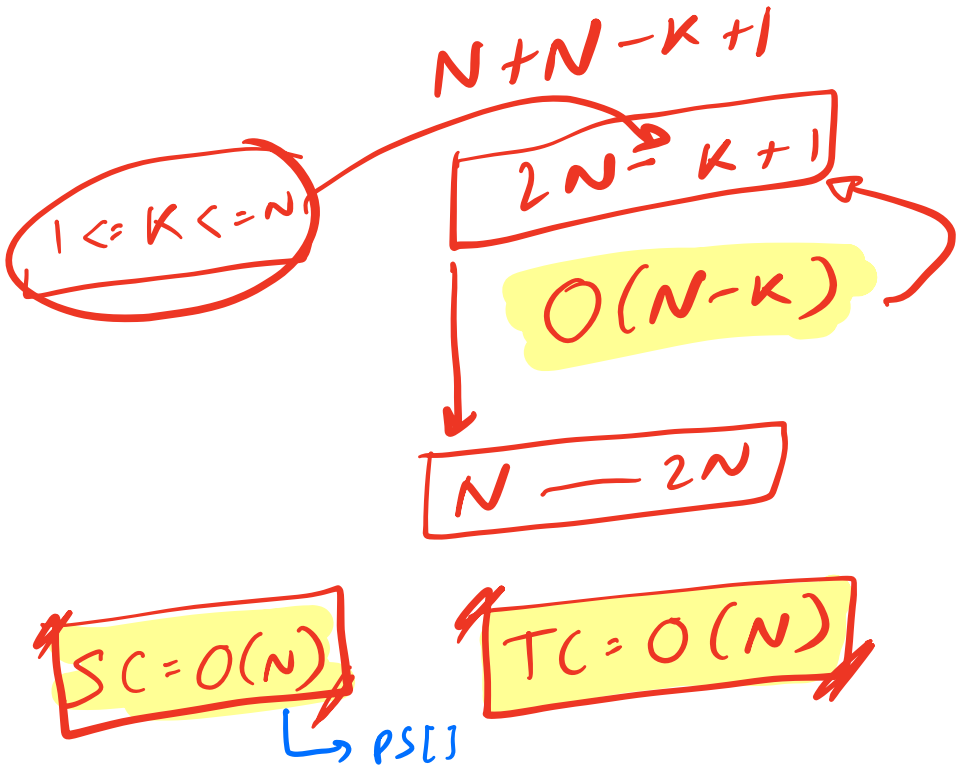
II Prefix Sums

1. Build the PS Array → O(N)

2. L = 0, R = K-1, mC = -∞ → O(N-K+1)

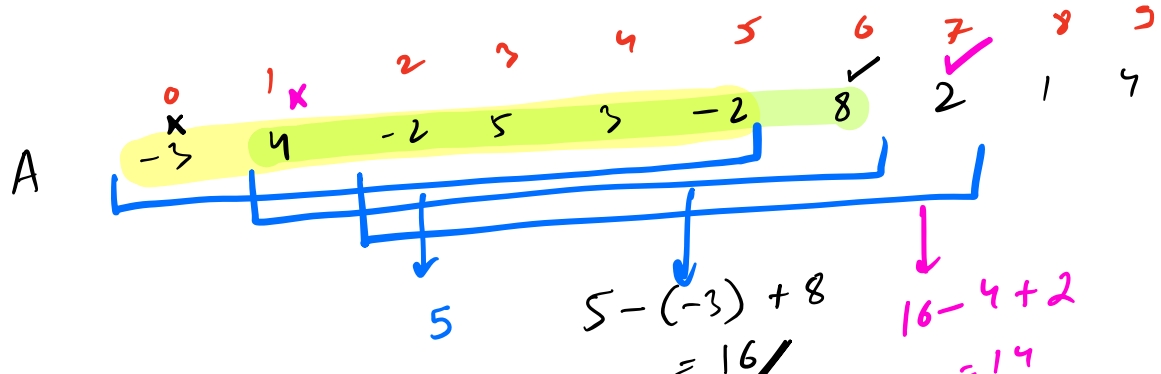
```
while (R < N) {                    // N-K+1
    // [L, R]
    s = ps[R] - ps[L-1];           // L==0 → O(1)
    if (s > mC) { mC = s; }

    L++, R++;
}
ret mC;
```



III SLIDING WINDOW →

K = 6



L = 0, R = K-1 → Iterate & find the Sum

L	R	
0	5	→ Sum = Sum − A[0] + A[6]
1	6 = K	→ Sum = Sum − A[1] + A[7]
2	7	→ Sum = Sum − A[2] + A[8]
3	8	→ ——— − A[3] + A[9]
4	9	→

Sum = Sum − A[L−1] + A[R]

```
s = 0
for(i=0; i<K; i++){
    s += A[i];
}
mC = s;
L=1, R=K
while (R<N){
    // [L, R]
    s = s - A[L-1] + A[R];
    if( s > mC) { mC = s;}
    L++, R++;
}
ret mC;
```

K → K

N-K+1 → while (R<N){

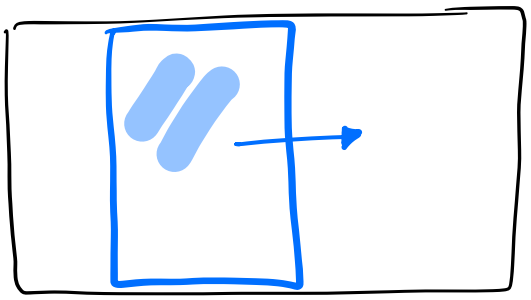
O(1) → s = s - A[L-1] + A[R]; if(s > mC) { mC = s;} L++, R++;

~~K + N-K+1~~

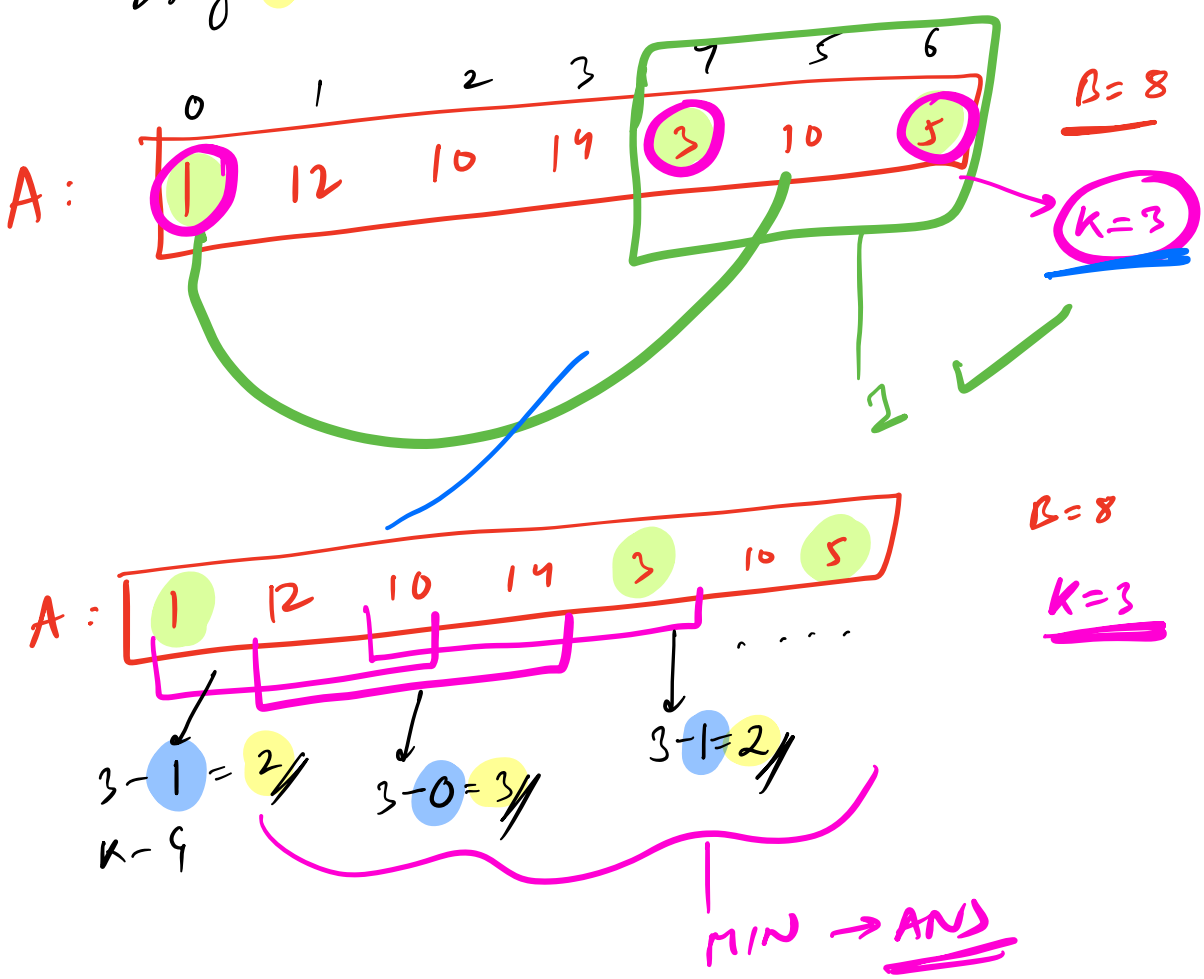
TC = O(N)

SC = O(1)

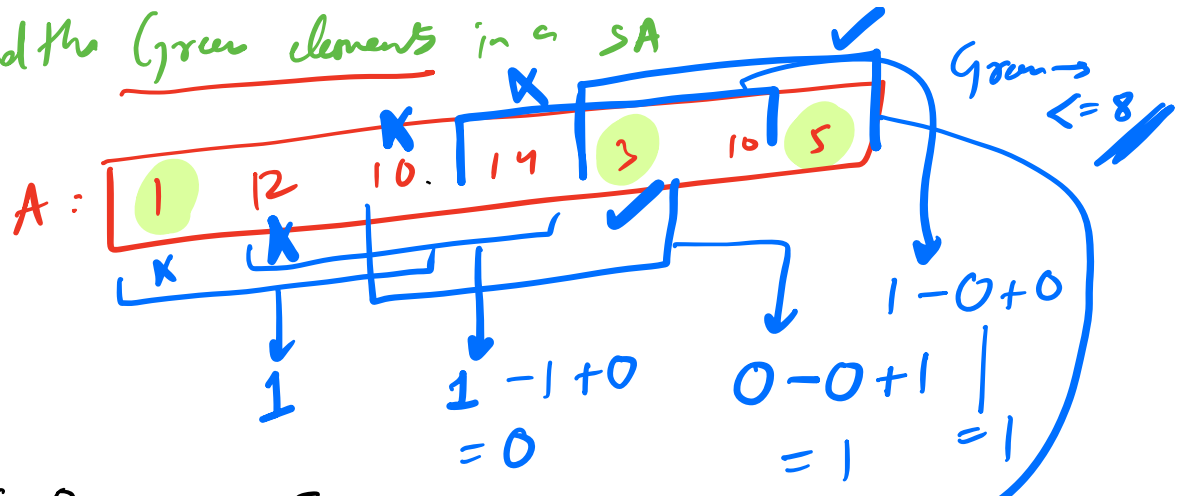
WINDOW Size is FIXED in Sliding Window!



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



(b) find the Green elements in a SA




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