1) Sliding Window

2) Few problems

3) TLE ??

Q Given an array of size N

Print the start & end index of all subarrays of size K
⇓
input.

N ↑

	0	1	2	3	4	5	6	7	8	9	10	11
A[12] =	3,	4,	2,	-1,	6,	7,	8,	9,	3,	2,	-1,	4

K = 3

s	e
→ 0	→ 2
1	3
2	4
3	5
⋮	⋮
9	11

⇓

12 - 3

s, e ⇒ l $(e - s + 1)$

s, length $(l + s - 1)$

for ($i = 0$; $\underset{}{i < \overset{N-1}{N}}$; i++)

$e = l + s - 1$

$l = e - s + 1$

$K = (N-1) - s + 1$

$s = N \cancel{-1} \cancel{+1} - K = \boxed{N-K}$

~~K > N~~

Code

```
                    i = N-K
        K => given
for( i = 0 ;  i <= N-K ;  i++) {
        j = i + K - 1 ;
        print (i, j);
}
```

T.C. = O(N-K)

≃ O(N)

K << N ×

Q Given an array of size N.

Find the max subarray sum of len = K

s e

	0	1	2	3	4	5	6	7	8	9
A =	-3,	4,	-2,	5,	3,	-2,	8,	2,	-1,	4

K = 5

S	e	Sum
0	4	7
1	5	8 ✓
2	6	12 ✓
3	7	16 ⇒ Ans
4	8	10
5	9	11

N - K

10 - 5 = 5

Soln 1

Brute force

$= \quad \neq$

⇒ for all subarrays of length K,
find the sum & take max.

Code

$\max S = N - K$
$\max E = N - 1$
$e < N$

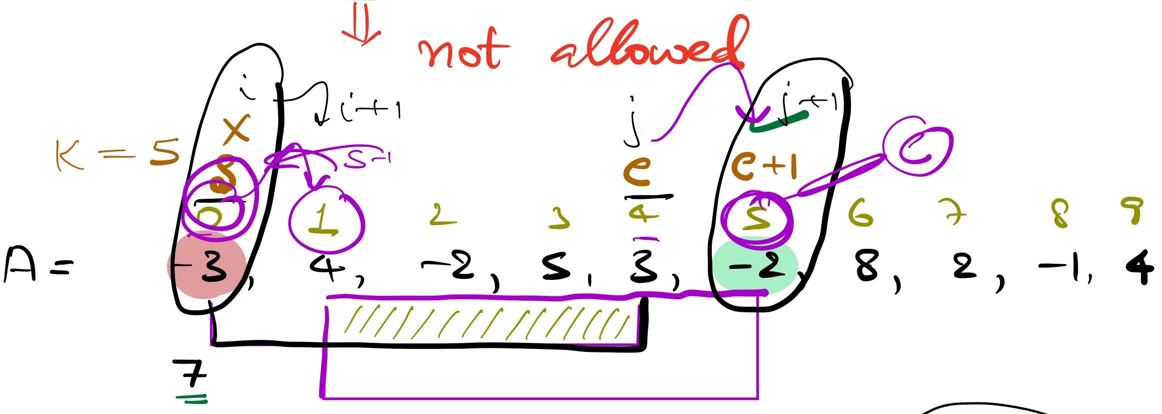
```
s = 0;
e = K-1;
maxSum = INT_MIN
while ( s <= N-K ) {

    // calculate sum of subarray from
       s to e

    - 1) loop from s to e & calc sum → O(K)
      OR
      2) Use PS to calc. sum from s to e

            PS[e] - PS[s-1];   ⇒ T.C. = O(1)
                                 S.C. = O(N)

    if (sum > maxSum) {
            maxSum = sum;
    }

    s++;
    e++;
}
```

$T.C. = O(N + N - K)$

$\simeq O(2N - \underbrace{K}_{N}) \Rightarrow \underline{O(N)}$

S.C. $= \underline{O(N)}$

$\Downarrow$ not allowed



$S = N - K$ ×

$\Rightarrow S[0 - 4] = 7$

$S[1 - 5] = 7 - A[0] + A[5]$

$= 7 - (-3) + (-2) = 8$

$\Downarrow$

$S[2 - 6] = 8 - A[1] + A[6] =$

$8 - 4 + 8 = \underline{12}$

$S[i \quad j] = S$

$S[i+1, j+1] = S - A[i] + A[j+1]$

Sliding Window.

Code

```
ans = INT_MIN;
sum = 0;                                  [0, K-1]

for (i=0; i<K; i++) {
        sum = sum + A[i];                  O(K)
}
ans = sum;
```

s = 1;   e = K $(\overset{1}{\cancel{s}}+K-\cancel{1})$

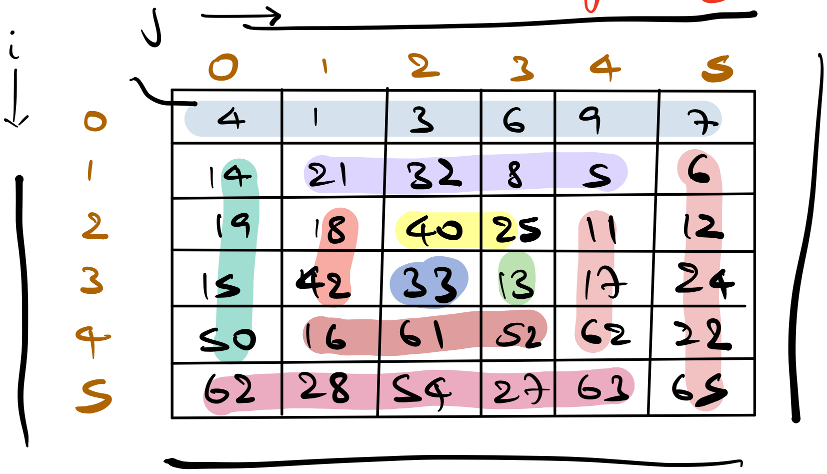
```
while (s<=N-K) {
    sum = sum - A[s-1] + A[e];
    ans = max(ans, sum);                  O(N-K)

    s++;
    e++;
}
```

T.C. = O(N)

S.C. = O(1)

Google, Adobe, Apple, FB, Oracle, Microsoft.

Q Given a matrix of size N×N



i \ j	0	1	2	3	4	5
0	4	1	3	6	9	7
1	14	21	32	8	5	6
2	19	18	40	25	11	12
3	15	42	33	13	17	24
4	50	16	61	52	62	22
5	62	28	54	27	63	65

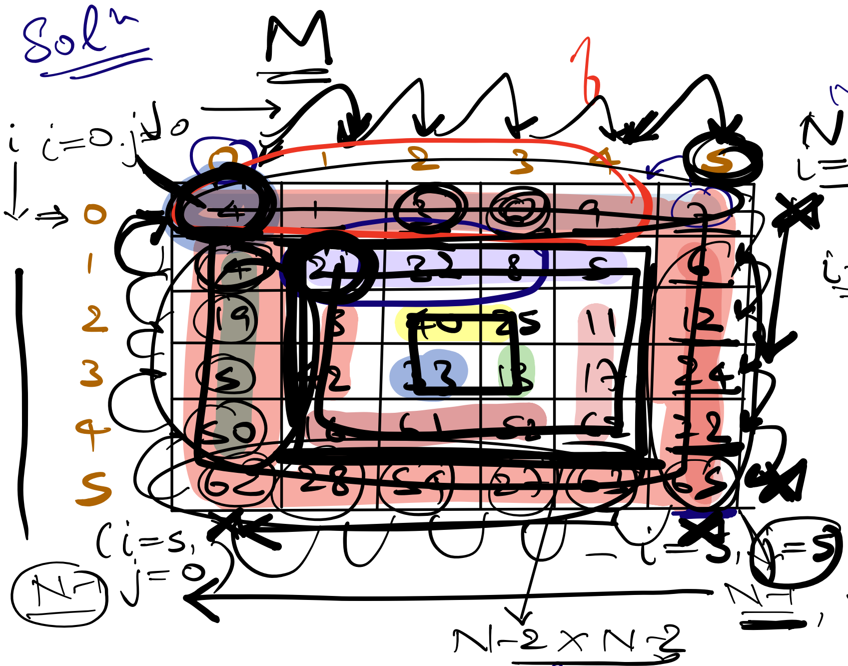
Print the matrix in spiral manner.

Print

4, 1, 3, 6, 9, 7, 6, 12, 24, 22, 65, 27. ---

36

Soln



6 ⇒ 4 ⇒ 2

(0, 5)

[0, 0] → [0, 4] = N-1

i++

[0, 5] - [4, 5] = N-1

[5, 5] - [5, 1] = N-1

[5, 0] - [1, 0] = N-1

N-2 × N-2

[1, 1] - [1, 3] ⇒ N-3

Code

```
i = 0; j = 0;
while (   N > 1          ) {
    count = 0;
    while (count < N-1) {
        print M[i][j];
        count++;
        j++;
    }

    count = 0;
    while (count < N-1) {
2       print(M[i][j];
        count++;
        i++;
⇒   }

    count = 0;
    while (count < N-1) {

3       print(M[i][j]);

        count++;
        j--;
    }

    count = 0;
    while (count < N-1) {

4       print (M[i][j]);
```

$(N-1) \cancel{-1} \cancel{+1}$

$= N-1$

s   e +1

N-1

[1, N-1]

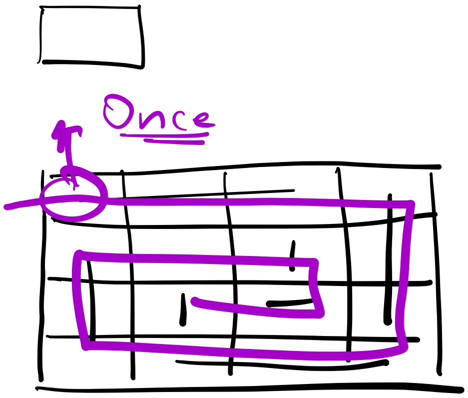
⇒ N-1

```
            count++;
            i--;
    }

    // i=0, j=0
    N = N-2;
    i++;  j++;

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

8 → 6 → 4 → 2 → 0

10 → 8

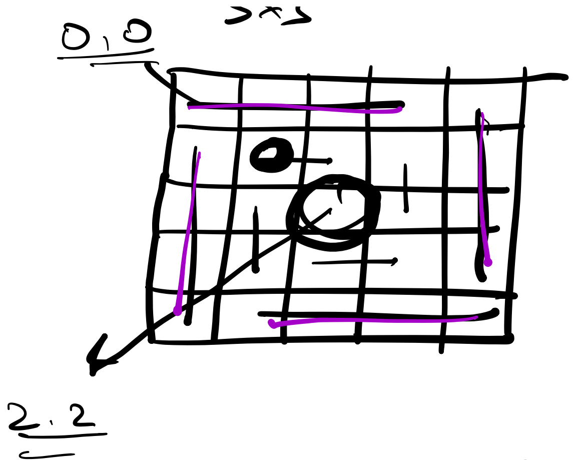
7 → 5 → 3 → ①



N×N ⇒ N^2

N=4 ⇒ N=2

N=2 ⇒ N=0



$N=5 \Rightarrow N=3 \quad (0,0)\ (1,1)$

$N=3 \Rightarrow N=1 \quad (1,1)\ (2,2)$

$N=1 \Rightarrow M[i][j]$

$$T.C. = O(1 \times n^2) = \boxed{O(\underline{n^2})}$$

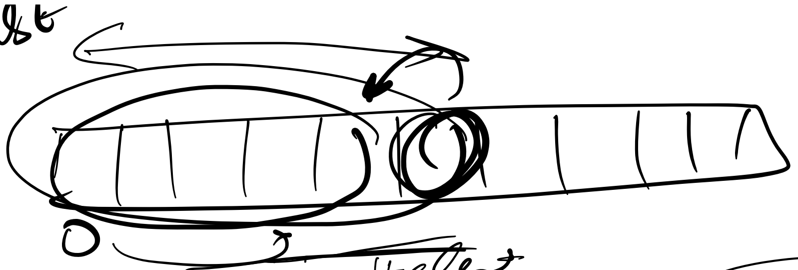
$$4(n-1) + 4(n-3) + 4(n-5) \ldots\ldots\ldots\ 0$$

H.W. Use AP

$$4\Big(\overbrace{(n-1) \underset{\times}{+} (n-3) \underset{\times}{+} (n-5) \ldots\ldots\ 0}\Big)$$

$4\Big($

Doubt



#elem

0, 0 ⇒ A[0] ⇒ Sum / # elements

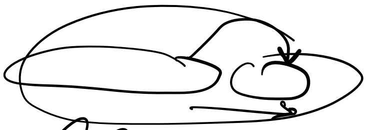
0, 1 ⇒ $\frac{(avg) \times \#(prev) + A[1]}{prev + 1}$

```
for (i=0; i<N; i++){
    for (j=i; j<N; j++){
```

[1, 1, 1, 1, 1, ... 10^5] ⇒ 1

S, S ⇒ Σ

$\frac{Sum}{\#elements}$



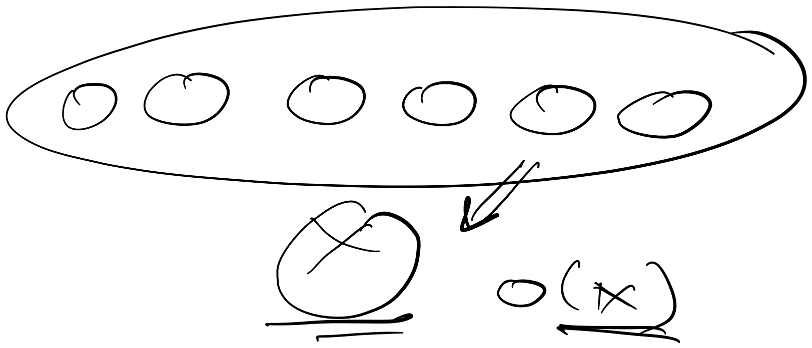
CF ⇒

Ass ⇒ Class

PF ⇒ range

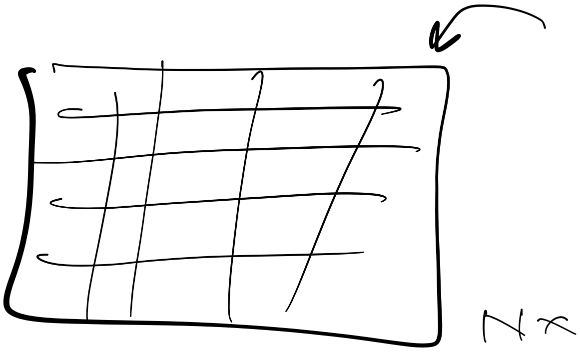
H.W New prob

SW ⇒ Size of subarray is fixed



$O(\underline{x})$

All subarray $\Rightarrow$ $\boxed{O(\underline{n^2})}$ $\longrightarrow$ T.C. $= O(\underline{n^2})$



$N \times M = O(\underline{N \times m})$