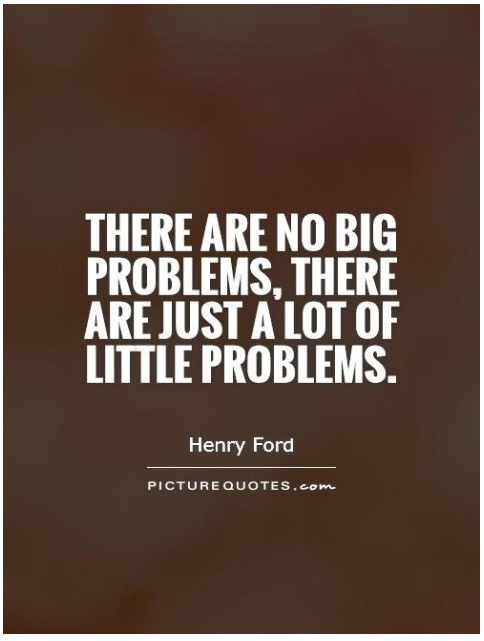
Sliding Window



Good Evening ☺

Today's content

01. Max subarray sum of len = k
02. Min swaps
03. Spiral matrix

01. Given N elements, print max subarray sum of length 'k'.

arr[10] =

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

k=5

Quiz → End index of first window ⇒ k-1

st index of first window = 0

Quiz End index of last window = n-1

st	end	sum
0	4	7
1	5	8
2	6	12
3	7	16 → Ans
4	8	10
5	9	11

02. $A = \{-3, 5, -1, 1\}$ $K = 2$

0 1 2 3

st	end	sum
0	1	2
1	2	4 → Ans
2	3	0

Brute force → For every subarray of size K, calculate the sum & update our maximum accordingly

```
int maxsub (int [] ar, int k)
{
    int st=0   e=k-1   ans = -∞
    while (e<n){
        int sum=0
        for(int i=st ; i<=e ; i++) {
            sum= sum+ ar[i];
        }                                  } K time
        st= st+1;   e=e+1;
    }
}
```

(n-K+1)

ar[] =

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

1

K = 1 ⟶ ans = 6 or n ⇒ n − 1 + 1

K = 2 ⟶ ans = 5 or n − 1 = n − 2 + 1

K = 3 ⟶ ans = 4 or n − 2 = n − 3 + 1

⋮

K ⟶ ans = n − K + 1 subarrays

Time complexity

⇒ (n − K + 1) * K

K = 1

TC = (n − 1 + 1) + 1

= n

= O(n)

K = n

TC = (n − n + 1) * n

= n

= O(n)

K = n/2

⇒ (n − n/2 + 1) * n/2

= (n/2 + 1) * n/2

= n²/4 + n/2

≃ O(n²)

TC = O(n²)

SC = O(1)

Idea 2 → Use prefix sum approach

```
int maxsub ( int [] ar , int k)
    Create psum array {TODO} ---> n
    int st=0   e=k-1   ans = -∞
    while (e<n){
        if (st==0) sum = pref [e] :
        else   sum = pf [e] - pf [st-1] :

        if ( sum > ans ) ans = sum
        st = st+1 ;   e=e+1 ;
    }
}
```

n-k+1

TC: n + n - k + 1

TC: 2n

TC: O(n)

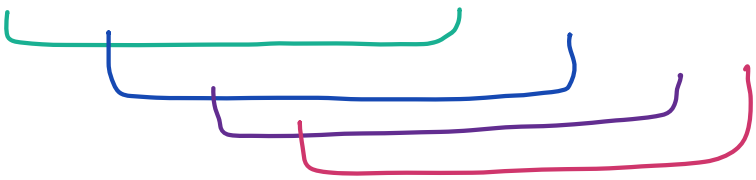
SC: O(n)

↓ sliding window

Idea 3 → Sliding window

arr[10] =

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



k=5

st	end	sum
0	4	7
1	5	8 = 7 − arr[0] + arr[5]
		= 7 − (−3) + (−2) = 8
2	6	12 = 8 − arr[1] + arr[6]
		= 8 − 4 + 8
3	7	16 = 12 − arr[2] + arr[7]
		= 12 − (−2) + 2
⋮		

Generalised = sum − arr[st−1] + arr[e]

Idea 3 → Carry forward approach + Window of size k

sliding window

```
int maxsum (int[] arr, int k)
{
    // sum for first window

      sum=0 , ans=-∞

    for( i=0 ; i<k ; i++){
    |   sum = sum + arr[i];
    }

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

      st=1 , e=k

      while (e<n)
      {
          // calculate sum for all windows

          sum = sum - arr[st-1] + arr[e];

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

          st = st+1 , e = e+1;
      }
    return ans;
}
```

TC: O(n)
SC: O(1)

02. Given arr[N] & a number B. Find & return the minimum no. of swaps to bring all the numbers $\leq B$ together (Google)

Eg :- arr = { 1, 12, 10, 3, 14, 10, 5 } B=8

Ans = 2 swaps

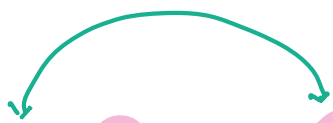
arr = { 3 7 6 13 2 15 } B=7

Ans = 1 swap

arr = { 25 30 2 18 7 6 9 50 3 } B=10

Ans = 1

arr = { 19 11 3 9 7 25 6 20 4 } B=10



Ans = 1

→ Count of ele $\leq B$ (count)

→ we have to make a subarray of fixed size → count

arr = { 19 11 3 9 7 25 6 20 4 } B = 10

0 1 2 3 4 5 6 7 8

count = 5

Ele > B → Bad ele

Ele ≤ B → Good ele

st	end	Bad ele	
0	4	2	→ 2 swaps
1	5	2-1+1	→ 2 swaps
2	6	2-1+0	→ 1 swap } → Ans
3	7	1-0+1	→ 2 swap
4	8	2-0+0	→ 2 swap

---

arr[ ] = { 10, 12, 11, 14, 7 } B = 3 → count = 0 ⇒ ans = 0

arr[ ] = { 10, 13 12, 11 } B = 10 → count = 1 ⇒ ans = 0

arr[ ] = { 10, 12, 11, 17 } B = 20 → count = 3 , ans = 0

arr = { 1, 12, 10, 3, 14, 10, 5 } B = 8

0 1 2 3 4 5 6

count = 3

ele > B ⇒ Bad ele

ele ≤ B ⇒ Good ele

st	end	count
0	2	2 bad ele → 2 swaps
1	3	2 − 0 + 0 → 2 swaps
2	4	2 − 1 + 1 → 2 swaps
3	5	2 − 1 + 1 → 2 swaps
4	6	2 − 0 + 0 → 2 swap → Ans

Brute force

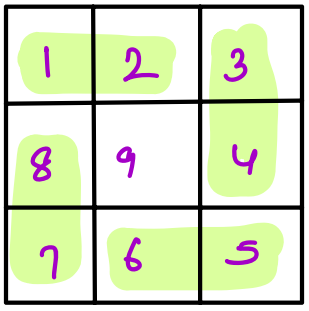
```
int minswaps (int [] ar, int B)
{
    count = 0

    for( int i=0 ; i<n; i++) {
        if (ar[i] ≤ B) count++;
    }

// count → No. of ele we have to group together.

    if (count == 0 || count == 1 || count == n)
        return 0;

// Calculate the bad ele in every window of
                                size ⇒ count :

    st = 0 , e = count - 1 , ans = ∞

    while (e < n)
    {
        int bad = 0

        for( i=st ; i≤e ; i++) {
            if (ar[i] > B) bad++ ;
        }
        if (bad < ans) ans = bad:
        st = st+1 , e = e+1 ;
    }
}
```

```
int minswaps (int [] ar, int B)
    count = 0
    for ( int i=0 ; i<n; i++) {
        if (ar[i] ≤ B) count++;
    }
// count → No. of ele we have to group together.
    if (count == 0 || count == 1 || count == n) return 0;

// Calculate ans for first window
      bad = 0 , ans = ∞
    for ( i = 0 ; i< count ; i++ ){
        if (ar[i] > B ) bad ++;
    }

    if (bad < ans) ans = bad ;
// Apply slidind window technique
      s = 1   , e = count
    while (e<n)
        if ( ar[s-1] > B)  bad--;  //dropping bad
        if (ar[e] > B) bad++;      // adding bad
        if (bad < ans) ans = bad ;
        s++ , e++;
    }
    return ans;
}
```

TC: O(n)

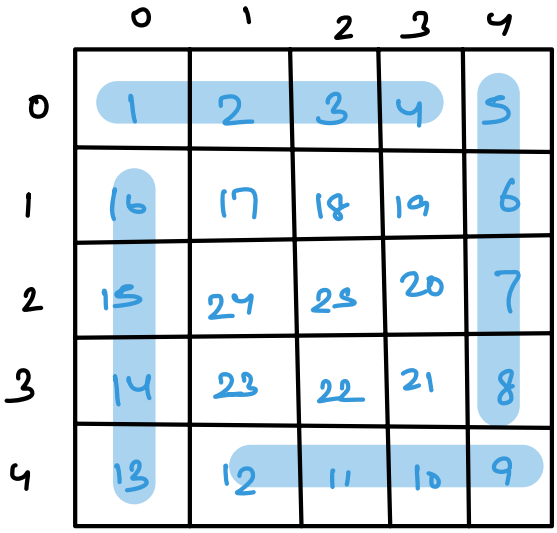
SC: O(1)

Q → Given a mat[N][N], print boundary in clockwise direction.



3 * 3

Output → 1 2 3 4 5 6 7 8



5 * 5

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

N = 3

(n-1) 2ele →

2ele ↓

2ele ←

2ele ↑

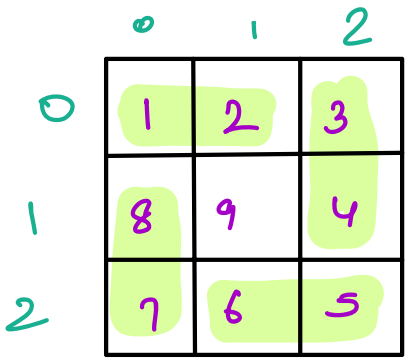
N = 5

(n-1) 4ele →

4ele ↓

4ele ←

4ele ↑



3 * 3

```
void printbound (int [][] arr)
```

i=0 , j=0

// (n-1) ele ⟶

```
for(int k=1 ; k<n ; k++){
    print(ar[i][j])
    j++;
}
```

k=1 1<3
2 2<3
3 3<3

i=0 j=0+2

ar[0][0] = 1

ar[0][1] = 2

// i=0 , j=2 → print (n-1) ↓

```
for(int k=1 ; k<n ; k++)
    ⇒ print(ar[i][j])
    i++;
```

k=1 1<3
2 2<3
3 3<3

i=0+1 j=2

ar[0][2] = 3

// i=2 , j=2 , print (n-1) ⟵

ar[1][2] = 4

```
for(int k=1 ; k<n ; k++)
    → print(ar[i][j])
    j=j-1;
```

k=1 1<3
2 2<3
3 3<3

i=2 j=2+0

ar[2][2] = 5

ar[2][1] = 6

// i = 2 , j = 0 → print (n-1) ↑

```
for(int k=1 ; k<n ; k++)
    → print(ar[i][j])
      i--;
3
```

k=1	k<3
2	2<3
3	3<3 x

i = 2 → 1 → 0 , j = 0

print ar[2][0] = 7

ar[1][0] = 8

⇒ i = 0 , j = 0

TC = O(n)

SC = O(1)

Q → Spiral matrix

	0	1	2	3	4	5
0	1	2	3	4	5	6
1	6	7	8	9	10	11
2	12	13	14	15	16	17
3	18	19	20	21	22	23
4	24	25	26	27	28	29
5	30	31	32	33	34	35

6 + 6

i	j	N	in loop
0	0	6	(sele)
1	1	4	(3elc)
2	2	2	(1elc)
3	3	0	

n == 0

stop

void spiral matrix (int [][] ar)

i=0 , j=0 , N

while (n > 1)

→

↓

←

↑

i = i+1 , j = i+1

N = N-2

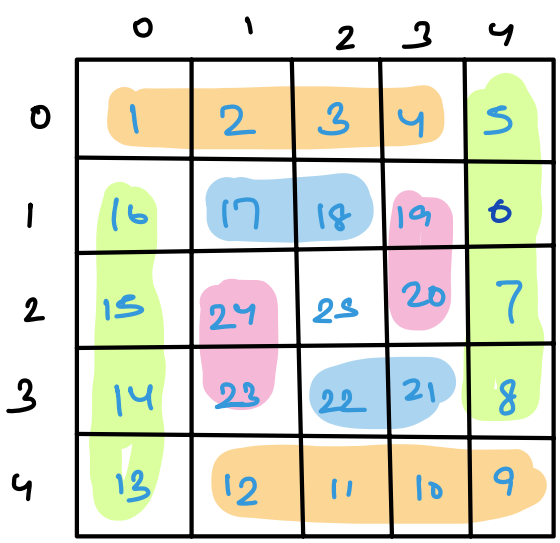
3

if (n == 1) print (arr[i][j]);

3

TC: O(n²)

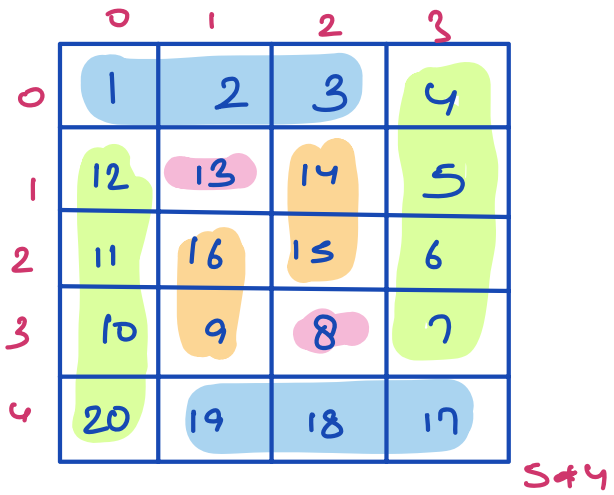
SC: O(1)



5*5

i	j	N	in one loop
0	0	5	4 ele
1	1	3	2 ele
2	2	(1)	0 ele

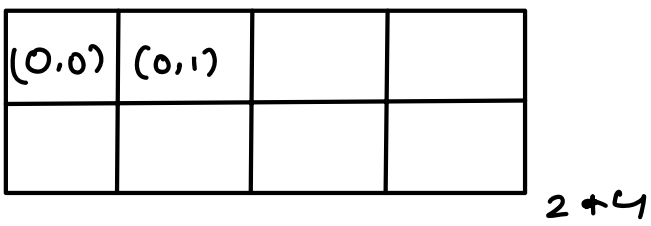
Doubt session



n-1 el

i	j	n	m
0	0	5	4
1	1	3	2

m-1 el



⇒ create a new matrix n*m

$\frac{4}{0}$