7:05AM

sliding window → 

Topic 8 8

- Sliding Window
  - max subarray sum len k
  - min swaps to group

*please be patient we optimize in multiple steps ;)

- 2 problems on 2D arrays
  - print boundry
  - print spiral

P1 Given an array of N elements, return max subarray sum of len = k k>=1 k<=N

ex a[]: { -3 | 4 | -2 | 5 | 3 | -2 | 8 | 2 | -1 | 4 } k=5
0 1 2 3 4 5 6 7 8 9

All sub arrays with len = 5

idea1: Brute Force

Choose private chat for ideas

s	e	sum
0	4	7
1	5	8
2	6	12
3	7	16 ← ans
4	8	10
5	9	11

Quiz
[0, k-1]

Quiz

TC: O(n^2)
SC: O(1)

O((n-k+1) x k) (x: n-k+1, y: k)

k=n → O((n-n+1)xn) → O(n)

k=1 → O((n-1+1)x1) → O(n)

k=n/2 → O((n-n/2+1)x n/2) → O(n^2/2 - n^2/4 + n/2) → O(n^2)

```
int maxSubArr1(int a[], int k){
    n = a.len
    s=0   e=k-1
    ans = int.MIN_VALUE
    while(e<n){
        sum=0
        for(i=s; i<=e; i++){        // n-k+1 * k
            sum+=a[i]
        }
        ans = math.max(ans, sum)
        s++; e++;
    }
    ret ans
}
```

a[]: { -3 | 4 | -2 | 5 | 3 | -2 | 8 | 2 | -1 | 4 }
0 1 2 3 4 5 6 7 8 9

k	#window
1	n
2	n-1
3	n-2
⋮	⋮
k	n-k+1

n-(k-1)

6 ↔ 10-5+1

n-k+1

n=10
k=5

Can we optimize TC?

idea 2 : Prefix sum

Quiz*
TC: O(n)
SC: O(n)

```
int maxSubArr2(int a[], int k){                    TC   SC
    // PS[] ← PrefixSum(a)                         O(n) ; O(n)
    n = a.Len
    s = 0    e = k-1
    ans = int.MIN_VALUE
    while(e<n){                                              n-k+1
        sum = 0
        if(s==0) sum = PS[e]
        else sum = PS[e] - PS[s-1]
        ans = math.max(ans, sum)
        s++; e++;
    }
    ret ans
```

$O(n + \{n-k+1\})$

$k=1$	$k=n/2$	$k=n$
$O(n)$	$O(n)$	$O(n)$

Can we optimize SC?

sliding window

a[]: { -3 4 -2 5 3 -2 8 2 -1 4 }
0 1 2 3 4 5 6 7 8 9

sum = 7

out = -3, in = -2

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

out = 4, in = 8

sum = 8 - (4) + (8) = 12

out = -2, in = 2

sum = 12 - (-2) + 2 = 16

All sub arrays with len = 5

S	e	sum
0	4	7
1	5	8
2	6	12
3	7	16
4	8	10
5	9	11

n
S e

sum = sum - a[s-1] + a[e]

O(k)

= 7 - a[0] + a[5] = 8

= 8 - a[1] + a[6] = 12

= 12 - a[2] + a[7] = 16

= 16 - a[3] + a[8] = 10

= 10 - a[4] + a[9] = 11

O(1)

Carry Forward + all sub arry same size → sliding window

Final Code:

$k <= a.Len$

Quiz*

TC: O(n)

SC: O(1)

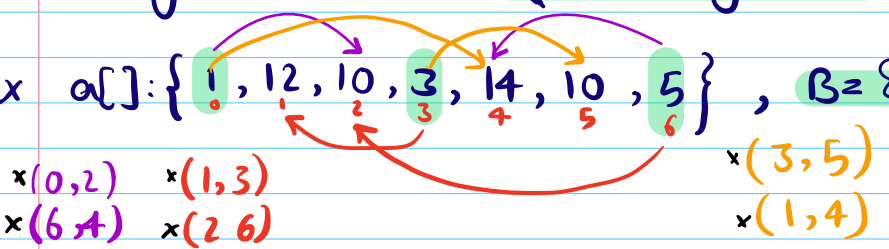
```
int maxSubArr3(int a[], int k){
    n = a.Len , sum = 0
    for(i=0; i<k; i++){          // O(k)
        sum += a[i]
    }
    s = 0   e = k-1
    s = 1   e = k
    ans = sum
    while(e<n){
        sum = sum - a[s-1] + a[e]
        ans = math.max(ans, sum)
        s++; e++;
    }
    ret ans
}
```

$O(k+n-k+1)$

a[]: {10, 12, 1, 3, 5, 10, 14}
(indices: 10=2, 12=1, 1=0, 3=3, 5=4, 10=5, 14=4)

P2 Given an array of n elements and a number B, return minimum number of swaps to bring all numbers <= B together

ex a[]: {1, 12, 10, 3, 14, 10, 5}, B = 8
(indices: 1=0, 12=1, 10=2, 3=3, 14=4, 10=5, 5=6)



ans = 2

Quiz a[]: {19, 11, 3, 9, 7, 25, 6, 20, 4}, B = 10
(indices: 19=0, 11=1, 3=2, 9=3, 7=4, 25=5, 6=6, 20=7, 4=8)

ans = 1

n-k+1

Quiz a[]: {25, 30, 2, 18, 7, 6, 9, 50, 3}, B = 10
(indices: 25=0, 30=1, 2=2, 18=3, 7=4, 6=5, 9=6, 50=7, 3=8)

ans = 1

idea

Please pay attention

① Count the elements <= B
② length of final answer group
③ swap == # Bad numbers
④ Move the window calculate # of swaps ← how to do efficiently →

<= B good number
> B bad number

hey, Can you explain again?

Size of window?
5

a[]: { 25 30 2 18 7 6 9 50 3 } B=10

0 1 2 3 4 5 6 7 8

ans=3

bad=3 → #swaps

a[]: { 25 30 2 18 7 6 9 50 3 }

0 1 2 3 4 5 6 7 8

ans=2

bad=3-1+0=2 → #swaps

a[]: { 25 30 2 18 7 6 9 50 3 }

0 1 2 3 4 5 6 7 8

ans=1

bad=2-1+0=1 → #swaps

a[]: { 25 30 2 18 7 6 9 50 3 }

0 1 2 3 4 5 6 7 8

ans=1

bad=1-0+1=2 → #swaps

⋮

Code 8

Quiz*

TC: O(n)

SC: O(1)

```
int minSwaps(int a[], int B){
    int n = a.Len
    // count # elem. <= B   TC = O(n) to find k
    if(k==0 || k==1) ret 0;

    int bad=0;
    for(i=0; i<k; i++){ // count # of bads in the first sub arr
        if(a[i]>B) bad++
    }
    // apply sliding window
    ans = bad; s=1; e=k      // 0, k-1
    while(e<n){
        if(a[s-1]>B) bad--
        if(a[e]>B) bad++
        ans = math.Min(ans, bad)
        s++; e++;
    }
    ret ans
}
```

n

n-k+1

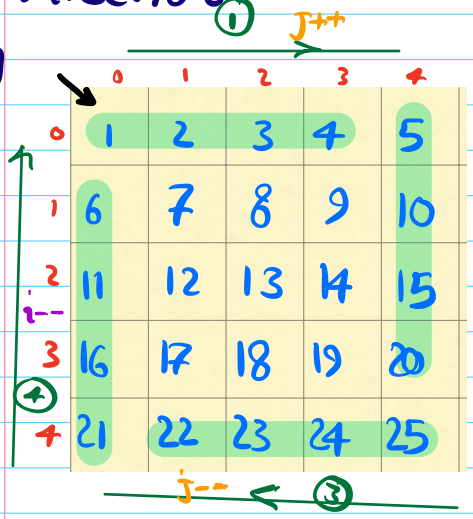
$O(n + k + n-k+1)$

$O(2n+1)$

square

P3 Given matrix a[N][N], print boundry in clockwise direction

a[5][5]

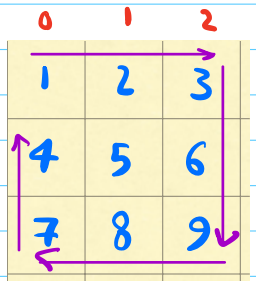


i → cur. row

j → cur. col

a[i][j]

a[3][3]



output: 1,2,3,6,9,8,7,4

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

idea?

① n-1 → j++

② n-1 ↓ i++

③ n-1 ← j--

④ n-1 ↑ i--

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

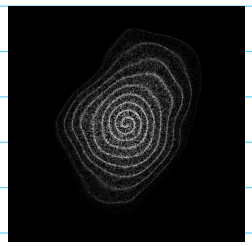
```
void printBoundryClockwise(int a[][]){
    n = a.Len
    // → ①
    i=0   j=0
    for(k=0; k<N-1; k++){
        print(a[i][j])
        j++
    }

    // ↓ ②
    for(k=0; k<N-1; k++){
        print(a[i][j]
        i++
    }

    // ← ③
    for(k=0; k<N-1; k++){
        print(a[i][j]
        j--
    }

    // ↑ 4
    for(k=0; k<N-1; k++){
        print(a[i][j]
        i--
    }
}
```

① n-1 → j++
② n-1 ↓ i++
③ n-1 ← j--
④ n-1 ↑ i--

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

optional HW?
Can you merge 4 directions into one for loop
for(d=0; d<4; d++)...

P4 Spiral printing of a[n][n]



	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30
5	31	32	33	34	35	36

Start Corner

i	J	N
0 (+1)	0 (+1)	6 (-2)
1 (+1)	1 (+1)	4 (-2)
2 (+1)	2 (+1)	2 (-2)
3	3	0

	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

i	J	N
0 (+1)	0 (+1)	5 (-2)
1 (+1)	1 (+1)	3 (-2)
2 (+1)	2 (+1)	1 (-2)
3	3	-1

Quiz

TC: $O(n^2)$
SC: $O(1)$

```
void printSpiralClockwise(int a[][]){
    n = a.Len
    i = 0   j = 0
    while (n > 1){
        // → ①
        for(k = 0; k < n-1; k++){
            print(a[i][j]
            j++
        }

        // ↓ ②
        for(k = 0; k < n-1; k++){
            print(a[i][j]
            i++
        }

        // ← ③
        for(k = 0; k < n-1; k++){
            print(a[i][j]
            j--
        }

        // ↑ ④
        for(k = 0; k < n-1; k++){
            print(a[i][j]
            i--
        }
        i++; j++; // ↘ move corner
        n -= 2;
    }
    if(n == 1) print(a[i][j])
}
```

n = 5 → 3 → 1

i = 1
j = 1

int a = $10^9 \times 2$

int b = $10^9 \times 2$

long c = (a+b)

int

int

int

overflow

long c = (long) a + (long) b