Arrays: Sliding Window

Friday - Contest
9:00 PM

10:30 PM - Discussion

Agenda

- Sliding Window Concept
- Minimum swaps
- Spiral Printing

Q1 Given array of N elements, find max subarray (window) sum of length=k (N >= k).

	0	1	2	3	4	5	6	7	8	9
arr[10] =	-3	4	-2	5	3	-2	8	2	-1	4

K = 5

s	e	sum
0	4	7
1 (2+1)	5 (2+1)	8
2	6	12
3	7	16
4	8	10
5	9	11

ans = 16

$e \to [k-1, N-1]$

$N-1-(k-1)+1$

$= N-\cancel{1}-k+\cancel{1}+1$

$= N-k+1$ iterations

Total Iterations

$= (N-k+1)\,k$

Idea

For every subarray of size k, iterate & calculate its sum. Overall take the max sum.

```
int maxSubarray(arr, k) {
    s=0, e=k-1
    ans = -∞
    while (e<N) {
        // Compute sum of [s e]
        sum=0
        for (i=s; i<=e; i++) {
            sum += arr[i]
        }
        ans = max(ans, sum)
        s=s+1
        e=e+1
    }
    return ans
}
```

k iterations

Worst Case –

$k=1 \Rightarrow (N-1+1)(1) = N$

$k=N \Rightarrow (N-N+1)(N) = N$

$k=\frac{N}{2} \Rightarrow (N-\frac{N}{2}+1)(\frac{N}{2}) = (\frac{N}{2}+1)(\frac{N}{2})$

$= O(N^2)$

Optimisation

Inner loop $\Rightarrow$ sum [s e]

↳ Prefix sum

1) Construct prefix array – pf [N]

2)

```
while ( e < N ) {
    // Compute sum of [s e]
    sum = 0
    if s == 0 :   sum = pf [e]
    else :   sum = pf [e] - pf [s-1]
    ans = max ( ans, sum )
    s = s+1
    e = e+1
}
return ans
```

$$\boxed{\begin{array}{l} TC : O(N) \\ SC : O(N) \end{array}}$$

Optimisation 2

$\boxed{K = 6}$

	0	1	2	3	4	5	6	7	8	9
ar[10] =	-3	4	-2	5	3	-2	8	2	-1	4

Sliding

s	e	sum
0	5	5
1	6	sum = sum − A[0] + A[6] = 5 − (-3) + 8 = 16
2	7	sum = sum − A[1] + A[7] = 16 − 4 + 2 = 14
3	8	sum = sum − A[2] + A[8] = 14 − (-2) + (-1) = 15
4	9	sum = sum − A[3] + A[9] = 15 − 5 + 4 = 14

Ans = 16

s e sum = sum − A[s-1] + A[e]

Carry forward	+	All subarrays of same size	⇒	Sliding Window

```
int maxWindowSum(int []A, int n, int k) {
        // Calculate sum of the first window → first k elements
        sum=0
        for (i=0; i<k; i++) {
                sum += A[i]
        }
        s=1, e=k
        ans = sum
        while ( e < N ) {
                // Calculate the sum [s e]
                sum = sum - A[s-1] + A[e]
                ans = max(ans, sum)
                s = s+1
                e = e+1
        }
        return ans
}
```

k iterations

N-k iterations

Quiz 1

$e \to [k, N-1]$

$\Rightarrow N \cancel{-1} - k \cancel{+1}$

$\Rightarrow N-k$

TC: O(N)

SC: O(1)

Total iterations $= \cancel{k} + N - \cancel{k}$

$= N$

Q2

Given arr[N] and a number B, find minimum no of swaps to bring all numbers <= B

Flipkart /thoughtworks INTUIT

Example

arr = 1 12 10 3 14 10 5

B = 8

↳ ans = 2

Example **Quiz 2**

arr = 25 30 2 18 7 6 9 50 3

B = 10

↳ ans = 1

Example **Quiz 3**

arr = 19 11 3 9 7 25 6 20 4

B = 10

↳ ans = 1

→ Count of all elements <= B, say k

→ Size of subarray will be fixed - k

k = 5

	No of swaps
0 - 4	2
1 - 5	2
2 - 6	1
3 - 7	2
4 - 8	2

All elements <= B → good

All elements > B → bad

No of swaps in a window = No of bad elements in that windo

```
int minSwaps(int []A, int B) {
        // Calculate k
        k=0
        for (i=0; i<N; i++) {
                if (A[i] <= B)
                        k++
        }
        if (k==0 or k==1 or k==N)
                return 0

        // Start with sliding window
        // 1. Get result of first window
        bad =0
        for( i=0; i<k; i++) {
                if (A[i] > B)   bad ++
        }
        // 2. Carry forward the count

        ans = bad
        s = 1
```

```
e = k
while ( e < N ) {
        if ( A[s-1] > B )   bad--
        if ( A[e] > B )     bad++
        s++
        e++
        ans = min(ans, bad)
}

return ans
}
```

Quiz 4

TC : O(N)
SC : O(1)

Break till 10:35 PM

A[s-1]
⇓
Check if prev element was bad ⇒ bad--
↓

arr = 25 30 2 18 7 6 9 50 3

B = 10

count → 3

⇑
Check if next element is bad ⇒ bad++
⇑
A[e]

Q3

Given a matrix A[N][N], print it's boundary in clockwise direction

⇓
Square matrix

	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

1 2 3 4 5 10 15 20 25

24 23 22 21 16 11 6

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

1 2 3 6 9 8 7 4

Idea

1. First row N-1 →
2. Last col N-1 ↓
3. Last row N-1 ←
4. First col N-1 ↑

4 loops

```
printBoundaryClockwise(int [][]A, int N, int i=0, int j=0) {
        if(N=1)  {  print( A[i][j] ;  return; }

        1) Print (N-1) elements from first row  →
        for( k=1 ; k<N ; k++ ) {
                print ( A [i] [j])
                j++
        }
                                                  → i=0 , j=N-1

        2) Print (N-1) elements from last col  ↓
        for( k=1 ; k<N ; k++ ) {
                print ( A [i] [j])
                i ++
        }
                                                  → i=N-1 , j=N-1

        3) Print (N-1) elements from last row  ←
        for( k=1 , k<N ; k++ ) {
                print ( A [i] [j])
                j - -
        }
                                                  → i= N-1 , j=0

        4) Print (N-1) elements from first col  ↑
        for( k=1 , k<N ; k++ ) {
                print ( A [i] [j])
                i - -
        }
                                                  → i=0 , j=0
}
```

Total iterations - $(N-1) \times 4$
$= 4N-4$ iterations

Quiz 5

TC : $O(N)$
SC : $O(1)$

Q4 Given a matrix A[N][N], print it in spiral form.

square

Adobe

	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

First cell index

i	j
0	0
1 (2+1)	1 (2+1)
2 (2+1)	2 (2+1)

Size of box

N
5
3 (2-2)
1 (2-2)
⇓
-1

	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

First cell index

i	j
0	0
1 (2+1)	1 (2+1)
2 (2+1)	2 (2+1)
i++	j++

Size of box

N
6
4 (2-2)
2 (2-2)
⇓
N=0

N -= 2

1) Keep on printing boundary of box till you can.

2) Reduce the size of box after each cycle.

```
printSpiralClockwise(int [][]A, int N) {
        i=0,    j=0
        while ( N >0 ) {
                printBoundaryClockwise (A, N, i, j)  ← Q1
                i ++
                j ++
                N - =2
        }
}
```

Quiz 6

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

Traversing over matrix in spiral order

N^2 elements in matrix

↳ N^2 iterations in total

$\Downarrow$

TC: $O(N^2)$

Doubts

Thank You

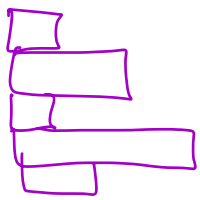
Assignments before HW

2D Array containing all subarrays

No of subarrays = $\frac{N(N+1)}{2}$

Jagged Arrays

ans = $\frac{N(N+1)}{2}$ x ___



int [][] ans = new int [$\frac{N(N+1)}{2}$][]

for (i = 0; i < $\frac{N(N+1)}{2}$; i++) {

subarray = ___

ans [i] = subarray

}

ans = [0] * $\frac{N(N+1)}{2}$

Good Night

Thank You

9 PM on Friday - Contest

10:30 PM - Contest Discussion