Array Techniques

Content

- Reverse the array.
- Sum Queries.
- Equilibrium index
- # subarray of length K
- Max subarray with length k
- Sum of all subarray sums

Rules → Private chat to answer

→ Question tab to ask questions.

Q> Given an integer array. Reverse the array

SC : O(1)

	0	1	2	3	4
A =	1	2	3	4	5
	5	4	3	2	1

Two pointers

			l,r		
	0	1	2	3	4
A =	~~1~~	~~2~~	3	~~4~~	~~5~~
	5	4	3	2	1

l >= r

stop

Pseudocode

```
void reverse ( A[] ) {
    l = 0
    r = N - 1

    while ( l < r ) {
        temp = A[l]
        A[l] = A[r]
        A[r] = temp
        l++
        r--
    }
}
```

TC : O(N)

SC : O(1)

Q> Reverse the array from index L to R L < R

	0	1	2	3	4	5	6	7
A =	1	2	3	4	5	6	7	8
	1	2	7	6	5	4	3	8

L = 2 R = 6

subarray —— continuous part of an array

L — R

Pseudocode

```
void reverse ( A[], L, R) {
    l = L
    r = R
    while ( l < r) {
        temp = A[l]
        A[l] = A[r]
        A[r] = temp
        l++
        r--
    }
}
```

TC: O(N)

SC: O(1)

Q> Given an integer array with N elements & Q queries.

For each query L, R

— Calculate the sum of all elements from index L to index R

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

L	R	sum
4	8	9
3	7	10
1	3	12
0	4	14
7	7	-9

Bruteforce — Do as given in the question.

```
void querySum ( Queries[][], A[] ) {
    for i ⟶ 0 to Q-1 {
        L = Queries[i][0]
        R = Queries[i][1]

        total = 0
        for j ⟶ L to R {
            total += A[j]
        }
        print(total)
    }
}
```

TC: O(Q*N)

SC: O(1)

Over a period of 10 overs score of india is given

0	1	2	3	4	5	6	7	8	9	10
	2	8	14	29	31	49	65	79	88	97

How many runs were scored in 7th over

score[7] − score[6] =

65 − 49 = 16

How many runs were scored from 6th to 10th over

score[10] − score[5]

97 − 31 = 66

How many runs were scored in 10 over

score[10] − score[9] =

97 − 88 = 9

How many runs were scored from 3rd to 6th over

score[6] − score[2]

49 − 8 = 41

How many runs were scored from 4th to 9th over

score[9] − score[3] = 88 − 14 = 74

Prefix sum { cumulative sum }

ps[i] = sum of all elements from 0 to i

How to calculate ?

	0	1	2	3	4
A =	2	5	-1	7	1
ps	2	7	6	13	14

	0	1	2	3	4	5
A =	10	32	6	12	20	1
ps =	10	42	48	60	80	81

Bruteforce

```
int[] ps = new int[N]
for i ⟶ 0 to N-1 {
    total = 0
    for j ⟶ 0 to i {
        total += A[j]
    }
    ps[i] = total
}
```

ps[0] = A[0]

ps[1] = A[0] + A[1]

ps[2] = A[0] + A[1] + A[2]

ps[3] = A[0] + A[1] + A[2] + A[3]

$ps[3] = ps[2] + A[3]$

$ps[i] = ps[i-1] + A[i]$

	0	1	2	3	4
A =	2	5	-1	7	1
ps =	2	7	6	13	14

Pseudocode

```
int[] ps = new int[N]
ps[0] = A[0]
for i ⟶ 1 to N-1 {
    ps[i] = ps[i-1] + A[i]
}
```

TC : O(N)

SC : O(N)

	0	1	2	3	4	5	6	7	8	9
A =	-3	6	2	4	5	2	8	-9	3	1
ps =	-3	3	5	9	14	16	24	15	18	19

L	R		sum
4	8	ps[8] - ps[3] = 18 - 9 = 9	9
3	7	ps[7] - ps[2] = 15 - 5 = 10	10
1	3	ps[3] - ps[0] = 9 - (-3) = 12	12
0	4	ps[4] = 14	14
7	7	ps[7] - ps[6] = 15 - 24 = -9	-9

```
void querysum ( Queries[][], A[] ) {
    // create ps[N] } TC: O(N)   SC: O(N)

    for i ⟶ 0 to Q-1 { ⟶ TC: O(Q)  SC: O(1)
        L = Queries[i][0]
        R = Queries[i][1]

        total = 0
        if (L == 0) {
            total = ps[R]
        }
        else {
            total = ps[R] - ps[L-1]
        }
        print (total)
    }
}
```

Overall
TC : O(N+Q)
SC : O(N)

↓ O(1)

use the original array as prefix sum.

```
for i ⟶ 1 to N-1 {
    A[i] += A[i-1]
```

Find the no. of equilibrium index

sum of all to left of index == sum of all to right of index

	0	1	2	3
A[] =	-3	2	4	-1

count = 1

	0	1	2	3	4	5	6
A[] =	-7	1	5	2	-4	3	0

Bruteforce

```
count = 0                                          TC : O(N^2)
                                                   SC : O(1)
for i ⟶ 0 to N-1 {
    left = 0
    for j ⟶ 0 to i-1 {        // sum [0, i-1]
        left += A[j]              ps[i-1]
    }
    right = 0
    for j ⟶ i+1 to N-1 {      // sum [i+1, N-1]
        right += A[j]             ps[N-1] - ps[i]
    }
    if (left == right)  count++
}
print(count)
```

Optimised approach using Prefix sum

// sum [0, i-1] == // sum [i+1, N-1]

ps[i-1] == ps[N-1] - ps[i]

ps[N-1] → total of entire array

carry forward technique

```
psminus = 0      // ps[i-1]
psi = 0          // ps[i]
total = 0        // total sum of array

for i ⟶ 0 to N-1 {
    total += A[i]
}
```

```
for i ⟶ 0 to N-1 {
    psi += A[i]  ⟶ carry forward.

    // leftsum == rightsum
    if (psminus == total - psi) {
        count++
    }
    psminus = psi
}
```

TC : O(N)

SC : O(1)

A[] =

0	1	2	3
-3	2	4	-1

i	psminus		total	psi	count
	0		2	0	0
0	0		2	-3	0
1	-3		2	-1	0
2	-1	==	2	3	1
3	3		2	2	1

Break : 10 : 37

Q> Total # subarrays of length K (<= N) ?

	0	1	2	3	4	5
A =	3	-2	4	-1	2	6

K = 3

ans = 4

K	start	last start
1	0	N-1
2	0	N-2
3	0	N-3
⋮		
K	0	N-K

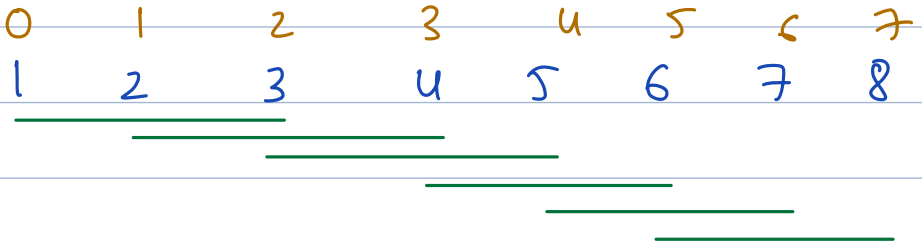
$[0, N-K] = (N-K) - 0 + 1 = N-K+1$

N = 7 K = 4 $= 7-4+1 = 4$

1 2 3 4 5 6 7

Given A[], print start and end indices of subarrays of length k.

N = 8 k = 3.



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

Pseudocode

```
s = 0
e = k - 1
```
} for the first subarray

```
while (e < N) {
    print (s + " " + e)
    s++
    e++
}
```

TC : O(N-k)

SC : O(1)

Q) Given an integer array. Find max subarray sum of subarray of length = K

	0	1	2	3	4	5	
A = [	3	-2	4	-1	2	6	]

K = 4

s	e	sum
0	3	4
1	4	3
2	5	11

max

ans = 11

Bruteforce

For each subarray, calculate sum and keep track of max sum

How to denote a subarray

s e

```
s = 0
e = k - 1
maxtotal = -∞   // INT_MIN
while (e < N) {
    total = 0
    for j ⟶ s to e {
        total += A[j]
    }
    maxtotal = max(maxtotal, total)
    s++
    e++
}
print(maxtotal)
```

Optimization

```
for j ⟶ s to e {
    total += A[j]
}
```

⟶ ps[e] − ps[s−1]

HW try prefix sum for above code

TC : O(N)

SC : O(N)

```
// calculate ps[N]

s = 0
e = k − 1
maxtotal = −∞   // INT_MIN
while (e < N) {
        total = 0
        if (s==0) { total = ps[e] }
        else { total = ps[e] − ps[s−1]

        maxtotal = max (maxtotal, total)
        s++
        e++
}
print (maxtotal)
```

sliding window technique.

	0	1	2	3	4	5	
A = [	3	-2	4	-1	2	6	]

K = 4

s	e						sum
		0	1	2	3		
0	3	3	-2	4	-1		4
		~~0~~	1	2	3	4	
1	4	~~3~~	-2	4	-1	2	4 − 3 + 2 = 3
			~~1~~	2	3	4	5
2	5		~~-2~~	4	-1	2	6	3 − (−2) + 6
							3 + 2 + 6 = 11

Pseudocode

```
s = 0
e = k - 1
total = 0
for i ⟶ s to e {
    total += A[i]
}                          // total of the first window.
maxtotal = total
e++
while (e < N) {
    total += A[e]          // new value to window
    total -= A[e-k]        // outside of window.
    maxtotal = max(maxtotal, total)
    e++
}
print(maxtotal)
```

TC: O(N)

SC: O(1)

**

Q> Given an integer array.
Calculat sum of [sum of all possible subarray]

A = [1 2 3]

	total
1	1
1 2	3
1 2 3	6
2	2
2 3	5
3	3

1 + 3 + 6 + 2 + 5 + 3
20

Bruteforce

```
overall = 0
for s ⟶ 0 to N-1 {
    for e ⟶ s to N-1 {
        // calculate sum of subarray s to e
        total = 0
        for j ⟶ s to e {
            total += A[j]
        }
        overall += total
    }
}
print(overall)
```

TC : O(N^3)

A = [1 2 3]

Contribution Technique

1

1 2

1 2 3

2

2 3

3

$1*3 + 2*4 + 3*3$

$= 3 + 8 + 9$

$= 20$

$$\sum_{i=0}^{N-1} A[i] * \{ \text{No. of times } A[i] \text{ appears in all subarrays of } A \}$$

$$A = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 \\ [3 & -2 & 4 & -1 & 2 & 6] \end{matrix}$$

# of subarrays index 1 is present in

s	e
0	1
0	2
0	3
0	4
0	5
1	1
1	2
1	3
1	4
1	5

$$A = [\underset{0}{3} \quad \underset{1}{-2} \quad \underset{2}{4} \quad \underset{3}{-1} \quad \underset{4}{2} \quad \underset{5}{6}]$$

s

# of subarrays index 2 is present in

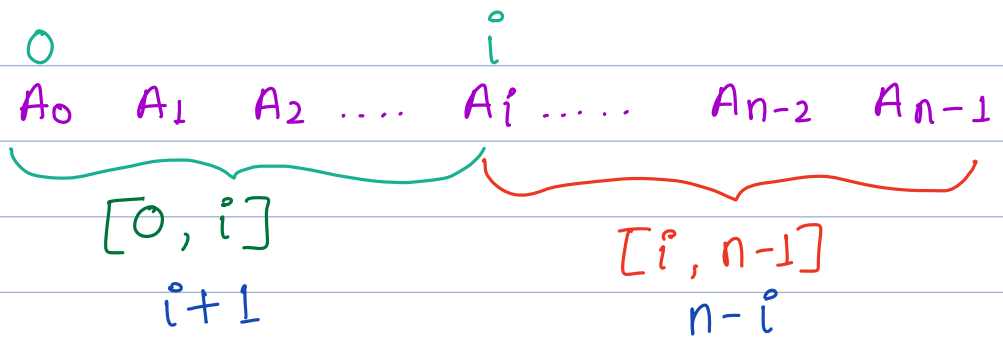
s	e	# subs
0	[2 5]	4
1	[2 5]	4
2	[2 5]	4
		12

No. of posible options for s = [0, 2] = 3 options

No. of posible options for e = [2, 5] = 4 options

3 * 4

# of subarrays index i is present in



$$(i+1)*(n-i)$$

$$\sum_{i=0}^{N-1} A[i] * (i+1)*(n-i)$$

TC : O(N)

SC : O(1)

Doubt session { attendance in DS is completely optional }

```
for i = 0 ; i * i < N ; i++
```

min i = 0

max i = $\sqrt{N}$

i = N

```
while i > 0 {
    i /= 2
```

$N \longrightarrow \frac{N}{2} \longrightarrow \frac{N}{4} \ldots .$

$\log N$

f(n) ⟶ $O(N)$

g(n) ⟶ $O(N^2)$

f(n) + g(n) $O(N+N^2)$

f(a)
g(n)

f(a) + g(n)

Problem Solving Framework

1) Understand question
   - input
   - output
2) Brute force
3) Optimise

exceeded 25 mins {Revise question}.

→ Hint 1

→ Hint 2

→ watch video solution

→ Reach out to TA {video call help}.

→ Text me over whatsapp / slack.