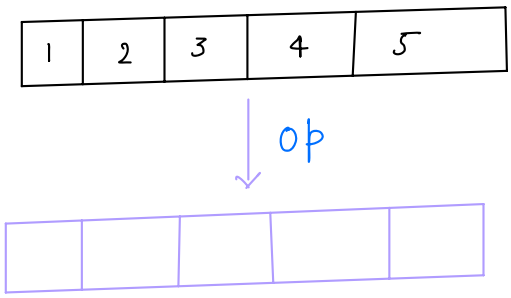
Lecture :- Array Techniques

Agenda

- Array manipulations
- Prefix sum concept
- Equilibrium indices
- Sliding window
- Sum of all subarray sums

Qu-1 Reverse the entire array.

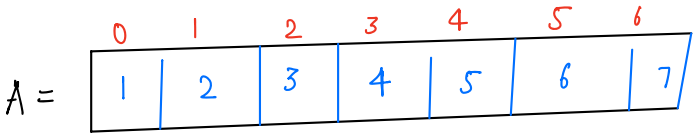


Approach

Code

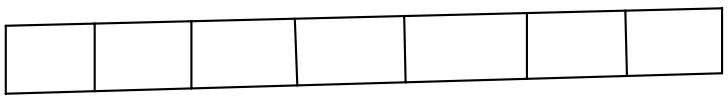
```
void reverseEntireArray(int[] arr) {
```

Qu: Reverse subarray from start to end



start = 2 end = 5

Output:



Algorithm

```
void reverse(int[] arr, int s, int e) {
```

Reverse entire array.

Qu: Given arr[n] and Q queries. for each query, calculate sum of all elements from L to R [0 based index]

A =

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

Queries

l	r	sum [l, r]
4	8	9
3	7	10
0	4	14
1	3	12
7	7	-9

4	8
3	7
0	4

queries[q]
[2]

Brute force

```
void printQuery(int[] A, int[][] queries) {
    q = queries.length;
    n = A.length;
Q ——— for(int i=0; i<q; i++) {
        l = queries[i][0]
        r = "  [i][1];
        sum=0;
n ——— for(j=l; j<=r; j++) {
        sum+ = A[j];
      }
    } print(sum);
}
```

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

$n <= 10^5$
$Q <= 10^5$
$n * Q = 10^{10} > 10^8$ ——— TLE.

Quiz 1 Given scores of 10 overs of a cricket match.

score[] = {2 (1), 8 (2), 14 (3), 29 (4), 31 (5), 49 (6), 65 (7), 79 (8), 88 (9), 97 (10)}

How much score was scored in 7th over?

a. 16

b. 20

c. 18

d. 17

score[7] − score[6]
65 − 49 = 16 Ans

Quiz 2 Given scores of 10 overs of a cricket match.

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

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

a. 66

b. 72

c. 68

d. 90

score[10] − score[5]
97 − 31

66 Ans

Quiz 3

Given scores of 10 overs of a cricket match.

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

How much score was scored in 10th over?

a. 7

b. 8

c. 9

d. 10

score[10] − score[9]

97 − 88 = 9 Ans

Prefix array

A[] =

0	1	2	3	4
2	5	-1	7	1

pf[] =

2	7	6	13	14

pf[i] ⇒ sum of all el from 0th idx to ith idx

10	32	6	12	20	1

pf =

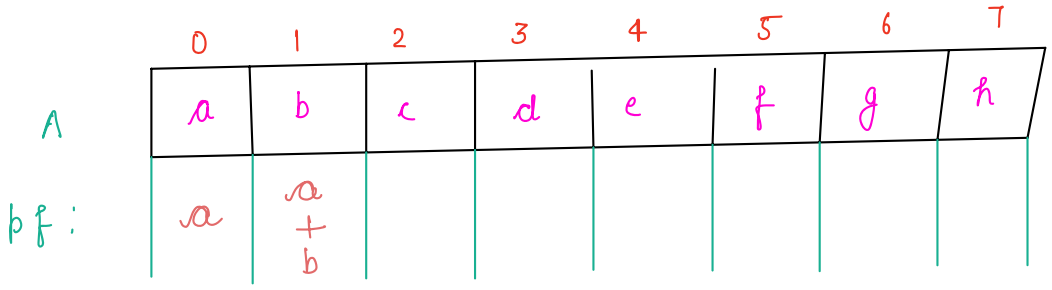
10	42	48	60	80	81

Brute force approach for creating prefix array.

```
int[] getPrefixSumArray(int[] A) {
    pf[n];
    for(i=0; i<n; i++) {
        sum=0
        for(j=0; j<=i; j++) {
            sum += A[j];
        }
        pf[i] = sum;
    }
    return pf;
}
```

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

Optimised Approach



$pf[0] = a$

$pf[1] = a + b \longrightarrow pf[0] + A[1]$

$pf[0] + b$ (b = A[1])

$pf[2] = \underbrace{a + b}_{pf[1]} + \underbrace{c}_{A[2]} \longrightarrow pf[1] + A[2]$

$pf[i] \longrightarrow pf[i-1] + A[i]$

Edge case: $i = 0$

$pf[0] = A[0]$

Code:

```
int[] getPrefixSumArray(int[] A) {
    pf[n];
    pf[0] = A[0];
    for(i=1; i<n; i++) {
        pf[i] = pf[i-1] + A[i];
    }
    return pf;
}
```

TC: O(n)
&C: O(n)

Answer the queries

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

sum[4,8] →
A[4] + A[5] + A[6]
+ A[7] + A[8]

{A[0] + A[1] + A[2] +
A[3] + A[4] + A[5]
+ A[6] + A[7] + A[8]}

−

{A[0] + A[1] + A[2]
+ A[3]}

pf[8] − pf[3]

Queries

l	r	sum [l, r]
4	8	pf[8] − pf[3] = 18 − 9 = 9

Generalised equation

sum[l, r] → pf[r] − pf[l−1]

Edge case: l = 0

sum[0, 3] → A[0] + A[1] + A[2] + A[3] ⇒ pf[3]

sum[0, r] = pf[r]

Code:

```
void printQuery(int[] A, int[][] queries) {
    q = queries.length;
    n = A.length;
    pf[] = getPrefixSumArray(A);
    for(int i=0; i<q; i++) {
        l = queries[i][0]
        r = queries[i][1];
        sum = 0;
        if(l==0) {
            sum = pf[r];
        } else {
            sum = pf[r] - pf[l-1];
        }
    }
}
```

n ——— pf[] = getPrefixSumArray(A);

Q ——— for(int i=0; i<q; i++) {

TC: O(n+Q)

SC: O(n)

Qu Given A[n] and Q queries with s and e index. for every query, return the sum of all even indexed elements from s to e.

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

Query:

s	e	Ans
1	3	1
2	5	A[2] + A[4] = 1+4 = 5
0	4	2 + 1 + 4 = 7.
3	3	0

Approach

	0	1	2	3	4	5
A =	2	3	1	6	4	5
pf array for even indexed sum →	2	2	3	.	. . .	—

pf[i] ⇒ sum[0..i] for only even indexed elements

pf[0] = 2

pf[1] = pf[0] + i % 2 == 0 ? A[i] : 0

+0 = pf[0]

pf[2] = pf[1] + A[2] = 3

2 + 1

pf[i] = pf[i-1] + i % 2 == 0 ? A[i] : 0

Edge case:

i = 0, pf[0] = A[0].

Pseudocode

```
int[] getPrefixEvenIndexSumArray( int[] A ) {
    pf[n];
    pf[0] = A[0];
    for(i=1; i<n; i++) {
        pf[i] = pf[i-1] + i % 2 == 0 ? A[i] : 0 ;
    }
    return pf;
}
```

```
void printQuery ( int[] A , int[][] queries) {
    q = queries.length
    n = A.length;
n ——— pf[] = getPrefixEvenIndexSumArray (A);
Q ——— for(int i=0; i<q; i++) {
        l = queries[i][0]
        r =    "    [i][1];
        sum = 0;
        if ( l == 0) {
            sum = pf[r];
        } else {
            sum = pf[r] - pf[l-1];
        }
}
```

Extension :

Given A[n] and Q queries with s and e index. for every query, return the sum of all ~~even~~ odd indexed elements from s to e.

Break: 9:50 — 9:57

Special Index { Amazon, Cred, Walmart }

Given A[], count the no of special indices in array.

Special index: Those index after removal of which, sum of all even indexed elements is equal to odd indexed elements.

	0	1	2	3	4	5
A:	4	3	2	7	6	-2

Ans = 2

Ans ⇒

idx.	updated array	Se	So	Special
0	3 2 7 6 -2	3+7-2 = 8	2+6 = 8	yes
1	4 2 7 6 -2	4+7-2 = 9	2+6 = 8	No
2	4 3 7 6 -2	4+7+(-2) = 9	3+6 = 9	yes
⋮	⋮	⋮	⋮	⋮

0	1	2	3	4
4	1	3	7	10

Sum of elements at odd index after removal of index = 2

4 1 7 10 ⇒ 1 + 10 = 11 Ans

0	1	2	3	4	5	6	7	8	9
2	3	1	4	0	-1	2	-2	10	8

sum of odd indices
after removal of idx = 3

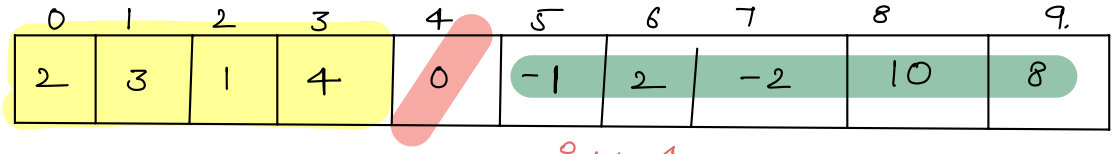
0	1	2	3	4	5	6	7	8	9
2	3	1	4	0	-1	2	-2	10	8

sum of even indices
after removal of idx = 3

0	1	2	3	4	5	6	7	8
2	3	1	0	-1	2	-2	10	8

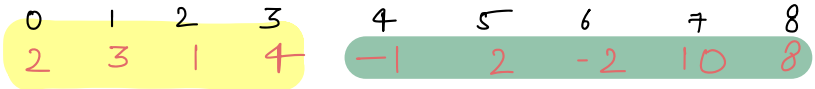
= 8 Ans

Approach



remove idx = 4.

sum of all odd idx el. ⟶
" " " even " ". ⟶ prefix sum concept



In original array, the el at odd indices are now present at even indices in updated array.
&
same behaviour is for even indices.

After deletion of idx = i ——

s_o = pfodd[i-1] + sumeven[i+1, n-1]
pfeven[n-1] - pf[i]

s_e = pfeven[i-1] + sumodd[i+1, n-1]
pfodd[n-1] - pf[i]

i = 0 ——

s_o = pfeven[n-1] - pf[i]
s_e = pfodd[n-1] - pf[i]

Pseudocode

```
int countSpecialIndices(int[] A) {
    pfodd[] = getPrefixOddSum(A);          ——— O(n), O(n)
    pfeven[] = getPrefixEvenSum(A);        ——— O(n), O(n)
    count = 0;
    for(i=0; i<n; i++) {                   ——— O(n)
        if(i==0) {
            so = pfeven[n-1] - pf[i]
            se = pfodd[n-1] - pf[i]
        } else {
            so = podd[i-1] + sumeven[i+1, n-1]
                 pfeven[n-1] - pf[i]

            se = pfeven[i-1] + sumodd[i+1, n-1]
                 pfodd[n-1] - pf[i]
        }
        if(so == se) {
            count++;
        }
    }
    return count;
}
```

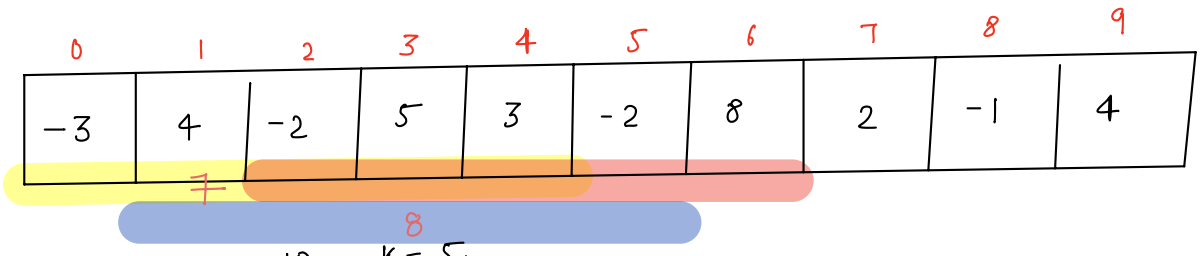
A:

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

TC: O(n)

SC:

Qu: Given arr[n], print max subarray sum for all subarrays of length = K.



n = 10, K = 5.

start	end	sum
0	4	7 {sum[0,4]}
1	5	8 {sum[1,5]}
2	6	12 {sum[2,6]}
3	7	16
4	8	10
5	9	11

Brute force Approach

```
int maxSubarraySumOfLengthK(int[] arr, int k) {
```

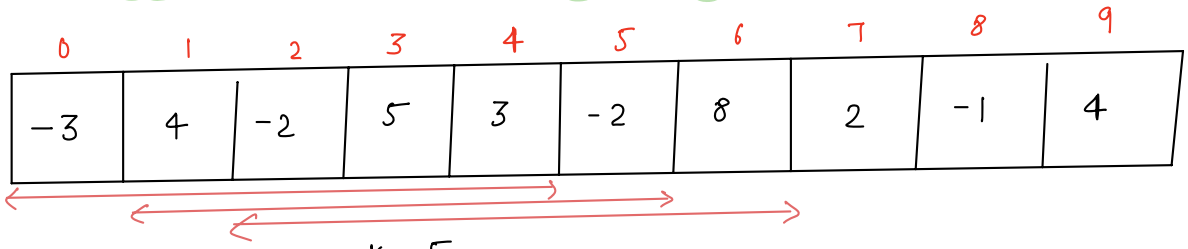
Optimised Approach

```
int maxSubarraySumOfLengthK(int[] arr, int k) {
    pf[] = getPrefixSum(A);
    max = 0;
    s = 0
    e = k-1;
    while(e<n) {
        if(s==0) {
            sum = pf[e]
        } else {
            sum = pf[e] - pf[s-1];
        }
        max = Maths.max(max, sum);
        s++;
        e++;
    }
    return max;
}
```

TC: $O(n) + O(n) \simeq O(n)$
SC: $O(1)$

Expected TC: $O(n)$
$O(1)$

Optimised Approach using Sliding window



n = 10, K = 5.

window	start	end	sum
1st	0	4	$-3+4-2+5+3 = 7$
2nd	1	5	sum = sum − A[s−1] + A[e] = 7 − (−3) + (−2) = 10 − 2 = 8
3rd	2	6	sum = sum − A[s−1] + A[e] = 8 − 4 + 8 = 12 Ans
⋮			

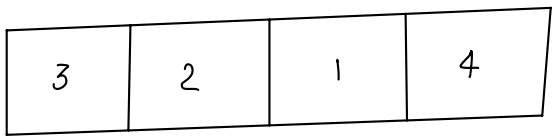
Code:

```
int maxSubarraySumOfLengthK(int[] arr, int k) {
    Handle 1st window alone
    sum = 0;
    for(i = 0; i <= k-1; i++) {
        sum += A[i];
    }
    max = sum;
    s = 1;
    e = k;
    while(e < n) {
        sum = sum - A[s-1] + A[e];
        max = Math.max(sum, max);
        s++;
        e++;
    }
    return max;
}
```

TC: O(n)

SC: O(1)

Qu: find sum of all subarray sums. {fb}



3
3 2
3 2 1
3 2 1 4
2
2 1
2 1 4
1
1 4
4

All subarrays:

Subarrays	Sum.
3	3
3 2	5
3 2 1	6
3 2 1 4	10
2	2
2 1	3
2 1 4	7
1	1
1 4	5
4	4

46 Ans

Contribution technique

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

3	2	1	4

All subarrays:

Subarrays	Sum.
3	3
3 2	3 + 2
3 2 1	3 + 2 + 1
3 2 1 4	3 + 2 + 1 + 4
2	2
2 1	2 + 1
2 1 4	2 + 1 + 4
1	1
1 4	1 + 4
4	4

3 * 4 + 2 * 6 + 1 * 6 + 4 * 4

A[0] occ_0 A[1] occ_1 A[2] occ_2 A[3] occ_3

Quiz

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

In how many subarrays, the element at index 1 will be present?

3 -2
3 -2 4
3 -2 4 -1
3 -2 4 -1 2
3 -2 4 -1 2 6

-2

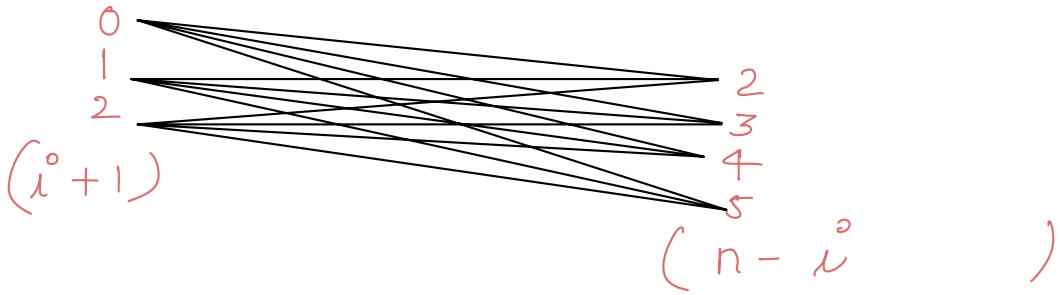
-2 4
-2 4 -1
-2 4 -1 2
-2 4 -1 2 6

Quiz

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

In how many subarrays, the element at index 2 will be present?

start indices : $i = 2$ end indices



Total occ of idx $i \Longrightarrow (i+1) * (n-i)$

```
int sumofAllsubArrays(int[] arr) {
    sum=0
    for(i=0; i<n; i++) {
        sum+ = A[i] * (i+1) * (n-i)
    }
    returnsum;
}
```

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

Thankyou ☺