Lecture 5 ÷ Arrays - 2

Agenda.

- Prefix sum.
- App^n on prefix sum.

Class starts at 7:05 AM.

Qu1 Given arr[n], return pf[]

$$pf[i] = sum(arr[0], arr[1] \text{ ------ } arr[i]).$$

Example:

$$arr[\,] = \left[\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ 5 & 2 & 7 & -3 & 8 \end{array}\right]$$

$$pf[\,] = \left[\begin{array}{ccccc} 5 & 7 & 14 & \underline{11} & \underline{19} \end{array}\right]$$

$$pf[\underset{\substack{\uparrow \\ i}}{0}] = sum(arr[0], arr[0])$$

$$= arr[0] = 5.$$

$$pf[1] = sum(arr[0] \text{ --- } arr[1])$$

$$= arr[0] + arr[1] = 7$$

$$pf[2] = sum(arr[0] \text{ ----.. } arr[2])$$

$$= arr[0] + arr[1] + arr[2] = 14$$

obvious approach

$$arr[\,] = \left[\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ 5 & 2 & 7 & -3 & 8 \end{array}\right]$$

$$pf[\,] = [\;5. \quad 7 \quad 14 \quad \underline{11} \quad \underline{19.}\;]$$

i [0-n-1]	j [0 - i]	sum
0	j = 0	arr[0] = 5.
1	j = 0, 1.	arr[0] + arr[1] = 7.
2	j = [0, 2] = 0, 1, 2.	5 + 2 + 7 = 14.
3	j = [0, 3] = 0, 1, 2, 3	5 + 2 + 7 + (-3) = 11.
4	j = [0, 4] = 0, 1, 2, 3, 4	5 + 2 + 7 - 3 + 8 = 19.

```
int[]   prefixsum( int[] arr) {
        int  n =  arr.length;
       int[] pf  =  new int[n];

        for( i=0;  i<n;  i++)  {
             int  sum=0;
            for( j=0;  j<=i;  j++) {
               sum  = sum + arr[j];
            }

            pf[i] = sum;
       }
     return pf;
}
```

TC :- $O(n^2)$.

SC :- $O(1)$ / $O(n)$.

Constraint: TC: $O(n)$

$$\text{arr}[\,] = [\ \overset{0}{5}\quad \overset{1}{2}\quad \overset{2}{7}\quad \overset{3}{-3}\quad \overset{4}{8}\ ]$$

$$\text{pf}[\,] = [\ 5.\quad \underset{\uparrow}{7.}\quad \underline{14.}\quad 11.\quad\ ]$$

→ $\text{pf}[0] = \text{arr}[0].$

→ $\text{pf}[1] = \underbrace{\text{arr}[0]}_{\text{pf}[0]} + \text{arr}[1]$

$= \text{pf}[0] + \text{arr}[1].$

→ $\text{pf}[2] = \underbrace{\text{arr}[0] + \text{arr}[1]}_{\text{pf}[1]} + \text{arr}[2]$

$= \text{pf}[1] + \text{arr}[2].$

→ $\text{pf}[3] = \underbrace{\text{arr}[0] + \text{arr}[1] + \text{arr}[2]}_{\text{pf}[2]} + \text{arr}[3]$

$= \text{pf}[2] + \text{arr}[3]$

$\vdots \qquad\qquad \vdots$

$\text{pf}[i] = \text{pf}[i-1] + \text{arr}[i]$

Edge case:- $i = 0$

$\text{pf}[0] = \text{pf}[-1] + \text{arr}[0]$ → Invalid

$\text{pf}[0] = \text{arr}[0].$

```
int[]   prefixsumOptimal (int[] arr) {
        int n = arr.length;
        int[] pf = new int[n];
        pf[0] = arr[0];

        for( i=1; i<n; i++) {
            pf[i] = pf[i-1] + arr[i];
        }
        return pf;
}
```

TC : O(n).

Qu2   Given arr[n] and Q queries.

for every query, calculate sum of all el in given range

$$arr[10] = \left[\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ -3 & 6 & 2 & 4 & 5 & 2 & 8 & -9 & 3 & 1 \end{array}\right]$$

Q = 6

l	r	sum
4	8	9
3	7	10
1	3	12
0	4	14
6	9	3
7	7	-9.

```
void sumQuery(int[] arr, int[][] queries) {
    int q = queries.length;
    for( i=0; i<q; i++) {  → O(Q)
        int l = queries[i][0];
        int r = queries[i][1];
        int sum=0;
        for( j= l; j<=r; j++) {  → O(n)
            sum += arr[j];
        }
        print(sum);
    }
}
```

TC: O(n*q)

SC: O(1)

	l 0	r 1
0	4	8
1	3	7
→2	1	3
3	0	4
4	6	9
5	7	7

arr[2][1] = 3.

l → [i][0]

r [i][1]

constraint: TC : O(n)

$$arr[10] = \left[\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ -3 & 6 & 2 & 4 & 5 & 2 & 8 & -9 & 3 & 1 \end{array}\right]$$

$$pf[10] = \left[\begin{array}{cccccccccc} -3 & 3 & 5 & 9 & 14 & 16 & 24 & 15 & 18 & 19 \end{array}\right]$$

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

1.) $sum(4, 8) = arr[4] + arr[5] + arr[6] + arr[7] + arr[8]$

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

red - yellow = green.

red = $sum(0, 8) = pf[8]$.

yellow = $sum(0, 3) = pf[3]$.

green = $sum(\underset{\substack{\uparrow\\ l}}{4}, \underset{\substack{\uparrow\\ r}}{8})$ = red − yellow

$$= \underset{\substack{\uparrow\\ r}}{pf[8]} - \underset{\substack{\uparrow\\ l-1}}{pf[3]}.$$

Generalisation

$$sum(l, r) = pf[r] - pf[l-1]$$

Edge case :-

$l = 0 \rightarrow sum(0, r) = pf[r] - pf[0-1]$

$= pf[r] - pf[-1]$ Invalid

$$sum(0, r) = pf[r].$$

$sum(0, 3) = pf[3]$

$sum(0, 2) = pf[2]$

```
        void sumQuery( int[] arr, int[][] queries) {
SC: O(n) <--  int[] pf = prefixsumoptimal (arr); --> TC: O(n)
              int q = queries.length;
              for( i=0; i<q; i++) { ----> O(Q).
                  int l = queries[i][0];
                  int r = queries[i][1];
                  if( l==0) {
                      print ( pf[r]);
                  } else {
                      print( pf[r] - pf[l-1]);
                  }
              }
        }
```

TC: O(n+q).

SC: O(n)

Break :- 8 : 13 AM

Qu3 Equilibrium idx. [Amazon, microsoft].

Given arr[n], count no. of equilibrium idx.

An idx is said to be equilibrium if.

sum of all el before idx = sum of all el after idx.

Example:

arr[4] = [-3 2 4 -1] → ans = 1.

-3	2	4	-1
ls = 0	ls = -3	ls = -1	ls = 3
rs = 5	rs = 3	rs = -1	rs = 0

arr[2] = [1 0] → ans = 1

1	0
ls = 0	ls = 1
rs = 0	rs = 0

arr[7] = [-7 1 5 2 -4 3 0] ans = 2.

-7	1	5	2	-4	3	0
			ls = -1			ls = 0
			rs = -1			rs = 0

$arr[8] = [\ \overset{0}{-6}\quad \overset{1}{1}\quad \overset{2}{5}\quad \overset{3}{2}\quad \overset{4}{-4}\quad \overset{5}{3}\quad \overset{6}{1}\quad \overset{7}{3}\ ]$

Dry runs:

$i = 4$. :- $ls = arr[0] + arr[1] + arr[2] + arr[3]$

$ls \Rightarrow pf[\underset{\underset{4-1}{\uparrow}}{3}]$

$rs \Rightarrow arr[5] + arr[6] + arr[7]$

$rs \Rightarrow sum(5, 7)$

$\Rightarrow pf[7] - pf[5-1]$

$\Rightarrow pf[\underset{\underset{n-1}{\uparrow}}{7}] - pf[\underset{\underset{i}{\uparrow}}{4}]$

$i = 3$. :- $ls \Rightarrow arr[0] + arr[1] + arr[2]$

$\Rightarrow pf[\underset{\underset{3-1}{\uparrow}}{2}]$

$rs \Rightarrow sum(4, 7) = pf[\underset{\underset{n-1}{\uparrow}}{7}] - pf[\underset{\underset{i}{\uparrow}}{3}]$.

$\vdots$

$i \qquad ls = pf[i-1]$

$\qquad rs = pf[n-1] - pf[i]$

$\underset{\underset{i}{\uparrow}}{i = 7}$:- $ls = arr[0] + arr[1] --- arr[6]$

$= sum(0, 6) = pf[\underset{\underset{i-1}{\uparrow}}{6}]$

$i = \underline{4}$:- $rs = sum(5, 7)$

$pf[\underset{\underset{n-1}{\uparrow}}{7}] - pf[\underset{\underset{i}{\uparrow}}{4}]$

Edge case:

$i = 0$

$\quad \hookrightarrow ls = 0$

$\quad \hookrightarrow rs = pf[n-1] - pf[0]$.

$i = n-1$ [Pass]

```
int   equilibriumIdx( int[] arr) {

      int[] pf = prefixsumoptimal(arr); → o(n)
       int  count = 0;
      //  i = 0, handle it  alone
            ls             rs
      if ( 0  = =  pf[n-1] - pf[0]) {

            count++;

      }

      for( i=1;  i<n;  i++){    → o(n)

            ls =  pf[i-1];

            rs =  pf[n-1] — pf[i];

            if( ls == rs){

               count++;
            }
      }

    return count;
}
```

TC: O(n)

SC: O(n)

Small Assignment:- Try solving in O(1) space

Qu4: Given arr[n], and Q queries. for each query [l, r], count even no. in that range

arr[10] =

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

	0 L	1 r	even no
0	2	6	2
1	4	8	3
2	0	6	4
3	⋮	⋮	⋮
	⋮	⋮	⋮

```
void countEvenQuery(int[] arr, int[][] queries) {
    int q = queries.length;
    for( i=0; i<q; i++) {  ⟶ O(Q)
        int l = queries[i][0];
        int r = queries[i][1];
        int cnt = 0;
        for( j=l; j<=r; j++) {  ⟶ O(n)
            if( arr[j] % 2 == 0) {
                cnt++;
            }
        }
    print (cnt);
}
```

TC: $O(Q * n)$

SC: $O(1)$

constraint:- TC: O(n) → linear

$$\begin{array}{lcccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ arr[10] = [& 2 & 4 & \underline{3} & 7 & 9 & \underline{8} & 6 & 5 & 4 & 9 \;] \\ pf[\,] = [& 1 & 2 & 2 & 2 & 2 & 3 & 4 & 4 & 5 & 5 \;] \end{array}$$

pf[i] = countEven(0, i)

→ pf[0] = if (arr[0] % 2 == 0) → 1
else → 0.

→ pf[1] = pf[0] + if (arr[1] % 2 == 0) → 1
else → 0

→ pf[2] = pf[1] + if(arr[2] % 2 == 0) → 1
else → 0

⋮ ⋮

pf[i] = pf[i-1] + if (arr[i] % 2 == 0) → 1
else → 0

Edge case: i = 0 → pf[0] = $\underbrace{pf[-1]}$ + Invalid.

```
int[]   prefixEvenCount( int[] arr){
        int n = arr.length;
        int[] pf = new int[n];
        // i=0, handle alone
        if (arr[0] %2 == 0) {
            pf[0] = 1;
        } else{
            pf[0] = 0;
        }

        for( i=1; i<n; i++) {
            if ( arr[i] % 2==0) {
                pf[i] = pf[i-1] +1;
            } else{
                pf[i] = pf[i-1];
            }
        }
      return pf;
}
```

TC : O(n)

SC: O(n).

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

	0 L	1 r	even no
0	2	6	2
1	4	8	3
2	0	6	4
3	⋮	⋮	⋮
	⋮	⋮	⋮

countEven(2,6) = pf[6] − pf[1].

idx: 0 1 2 3 4 5 6 7 8 9.

count[L,R] = pf[R] − pf[L−1].

Edge case:- L == 0

count(0, R) = pf[R].

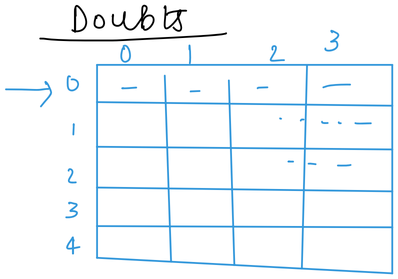
```
void countQuery( int[] arr, int[][] queries){
    int[] pf = prefixEvenCount(arr); ⟶ o(n).
    for( i=0; i< queries.length; i++) { ⟶ O(Q)
        int l = queries[i][0];
        int r = queries[i][1];
        if (l==0) {
            print( pf[r]);
        } else{
            print( pf[r] - pf[l-1]);
        }
    }
}
```

TC: O(n+Q)

SC: O(n)

Thankyou ☺

Doubts



$\underset{\text{length}}{\underset{\uparrow}{mat[5]}}[4]$

rows = mat.length

col = mat[0].length.

$$arr[7] = \left[\begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ \boxed{-7} & 1 & 5 & 2 & -4 & 3 & 0 \end{array}\right]$$

$\uparrow$ ls = 0

i — ls = sum(0, i−1).

rs = sum(i+1, n−1).

ls == rs. [i is an eq idx]

$\boxed{i=3} \rightarrow$ ls = sum(0, 2) = $pf[\underset{i-1}{\underset{\uparrow}{2}}]$.

rs = sum(4, 6) = $pf[\underset{n-1}{\underset{\uparrow}{6}}] - pf[\underset{i}{\underset{\uparrow}{3}}]$

sum(l, r) = pf[r] − pf[l-1]

i — ls = pf[i−1]

rs = pf[n−1] − pf[i].

i = 0. ✓ edge case

ls = pf[-1] → idx out of bound exception.

ls = 0

rs = pf[6] − pf[0]

$\boxed{i = n-1}$ ✓.