Prefix Sum

Vinay Neekhra

Senior Instructor & Mentor

Reachable in Scaler Lounge

"People worry that computers will get too smart and take over the world, but the real problem is that they're too stupid and they've already taken over the world - Pedro Domingos"

Agenda : ① Prefix Sum
② problems on prefix sum

Q. Given N array elements and Q (no. of queries) queries, for each query, calculate sum of all elements in range [L, R]. (L, R < N)

Note: L and R are indices such that L <= R

Constraint: $1 <= N, Q <= 10^5$

A = [-3 6 2 4 5 2 8 -9 3 1], N = 10

0 1 2 3 4 5 6 7 8 9

Queries ⇒ Q = 5

idea: for every query, calculate the sum (iterate from L to R)

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

```
1. void findSum(A, Q) {
2.     int N = A.size();
3.     while (Q > 0) {
           // get L, R
           sum = 0
4.
5.     for (int i = L; i <= R; i++) {
6.         sum += A[i]
       }
7.     // print(sum)
       Q--;
   }
}
```

Q = 1, L R

0 ⇔ 3

i <= 3

Q 1

L 0

R 3

i 0 1 2 3 4

sum = 0 - 3 5

4

sum = 4 #7

TC = O(Q * N)

SC = O(1)

$10^5 * 10^5$

$\approx 10^{10}$ = 16^8

TLE

Quiz: 7th 65 - 49 = 16

Quit: 6th to 10th one

$\Rightarrow$ score[10] − score[6−1]

$\Rightarrow$ score[10] − score[5]

Q. Given Indian cricket team scores for first 10 overs of batting. After every over, ~~total~~ score is given

Over	:	1	2	3	4	5	6	7	8	9	10
Score-board	:	2	8	14	29	31	49	65	79	88	97

→ ~~total~~ runs scored in the last over:

$97 - 88 = 9$

score[10] − score[9]

→ ~~total~~ runs scored in the 7th over

$65 - 49 = 16$

score[7] − score[6]

→ total runs scored from 6^{th} to 10^{th} over:

$$97 - 31 = 66$$

score[10] − score[6−1]

→ total runs scored from i^{th} to j^{th} over

score[j] − score[i−1]

idea: store the **prefix sum**

A = [−3 6 2 4 5 2 8 −9 3 1], N = 10

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

pSum[10]: [−3 3 5 9 14 16 24 15 18 19]

indices: 0 1

Queries ⇒ Q = 5

L R ans

4 8	9	→	psum[8] − psum[4−1] = 18 − 9 = 9
3 7	10	→	psum[7] − psum[3−1] = 15 − 5 = 10
1 3	12	→	psum[3] − psum[1−1] = 9 − (−3) = 12
0 4	14	→	psum[4] = 14
7 7	−9	→	psum[7] − psum[7−1] = 15 − 24 = −9

l r → psum[r] − psum[l−1]

O(1)

How to create prefix sum?

[3 −2 4 5 6]
0 1 2 3 4

psum[0] = A[0] = 3

psum[1] = A[0] + A[1] = 1

= psum[0] + A[1]

psum[2] = A[0] + A[1] + A[2] = 3 + (−2) + 4 = 5

= psum[1] + A[2]

$$psum[i] = psum[i-1] + A[i] \quad , \quad i > 0$$

⇒ Calculating prefix Sum:

```
int psum[10]; // defining the array
psum[0] = A[0];
for (i=1; i<N; i++) {
    psum[i] = psum[i-1] + A[i]
}
```

T.C. = O(N)
S.C. = O(N)

⇒ optimised code for Q1

```
void findsum ( A, Q ) {
    int psum[N];
```

fixed cost

```
psum[0] = A[0]

for (i=1; i<N; i++) {
    psum[i] = psum[i-1] + A[i]
}

while (Q>0) {
    // read L, R
    if (l==0) {
        print ( psum[R] );
    }
    else{
        print (psum[R] - psum[L-1] );
    }
    Q--;
}
```

T.C. $O(N)$

variable cost

TC $O(Q)$

SC

T.C. $= O(N+Q)$

S.C $= O(N)$

Obsn: N ⇒ Array size is 100

Q ⇒ 1 (5th index 1st index

→ Brute force is better

Q ⇒ 10,000 [5th 9th

[10th — 99th)

⋮

2nd soln is better

? Can we modify the existing array

A = [-3 6 2 4 5 2 8 -9 3 1], N = 10

0 1 2 3 4 5 6 7 8 9

Array is being read by other fn(s)? → side effects

A* : [-3 3 5 9 14 16 24, 15, 18, 19]

0 1

Advantage : SC. ⇒ O(1)

Dis-advantage: original array would be lost.

reset

10: 24

Q. Equilibrium index: Amazon Hike adobe

Given N array elements, count no. of eq^m index.
An index i is said to equilibrium index. if:

Sum of all the elements on left of i^{th} index = Sum of all the elements on right of i^{th} index

[Note: if i==0, left sum = 0
if i==N-1, right sum = 0]

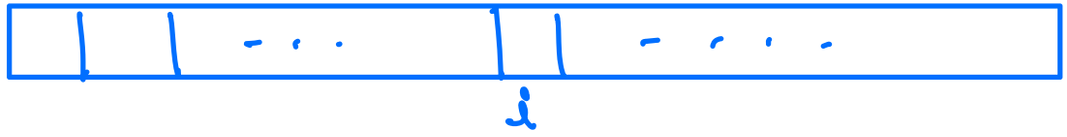
ex.: A = [-3 2 4 -1]
0 1 2 3

index → 0 ✗ is eq^m index? (leftsum 0 ≠ 5 rightsum)
→ 1 is eq^m index (-3 ≠ 3)
→ 2nd ✓ → (-1 = -1)

→ 3rd → 3 ≠ 0

ans = 1

Obsn



Sum [0, i-1] == Sum [i+1, N-1]

⇓

Obsn:

pSum [i-1] == pSum [N-1] − pSum [i]

Check this for index.

```
int countEquilibriumIndices(A){
    // create the prefix sum array
    int pSum[N]
    pSum[0] = A[0]
```

```
for (i=1; i<N; i++) {
        psum[i] = psum[i-1] + A[i]
}
```

```
count = 0
for (int i=0; i<N; i++) {
    TODO when i==0
    if (psum[i-1] == psum[N-1] - psum[i]) {
        count += 1
    }
}
return count
}
```

Bug for i = 0

T.C. = O(N)
S.C. = O(N)

TODO: find out the T.C & S.C. of Brute force soln.

Q. Given N array elements and Q queries.

for each query (l to r), find count of even numbers in the given range

ex: $A = [2\ 4\ 3\ 7\ 9\ 8\ 6\ 5\ 4\ 9]$
$\quad\quad\ \ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

Queries, Q = 3

l	r	ans
4	8	3
3	9	3
0	4	2

Brute force:
for each query, iterate from l to r, and count the even numbers

T.C. = $O(Q * N)$
S.C = $O(1)$

```
void countEvens(A, Q){
    while(q>0){
        // Read l, r
        int count = 0
        for(i=L; l<=r; i++){
```

```
        if (A[i] % 2 == 0) {
            count += 1
        }
    }
    print(count);
q--;
}
return;
}
```

optimisation:

(!) remember count of even numbers.

$A = [2\ 4\ 3\ 7\ 9\ 8\ 6\ 5\ 4\ 9]$
$\quad\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

pSum ⇒ count of even numbers till index i

$pSum = [1\ 2\ 2\ 2\ 2\ 3\ 4\ 4\ 5\ 5]$
$\quad\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

$$pSum[i] = pSum[i-1] + (A[i] \% 2 == 0)$$

count of even numbers from l to r $= pSum[r] - pSum[l-1]$

```
void countEvenNumbers(A, Q)
```

creating prefix array

```
int pSum[N];
pSum[0] = A[0] % 2 == 0 ? 1 : 0
for(i=1; i<N; i++) {
    pSum[i] = pSum[i-1] + (A[i] % 2 == 0 ? 1 : 0)
}
```

O(N)

```
while(q>0) {
    // read l, r
```

O(q)

```
    if (l==0) { print(psum[r]) }
    else { print(psum[r] - psum[l-1]) }
    q--;
  }
}
```

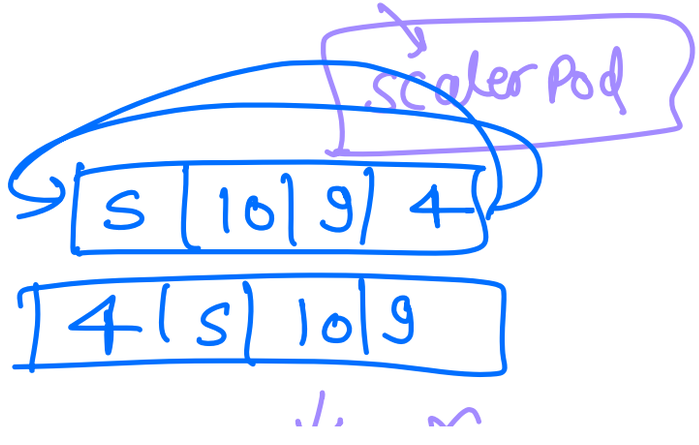
T.C = O(N + Q)

S.C = O(N)

↑ we are not modifying the input array

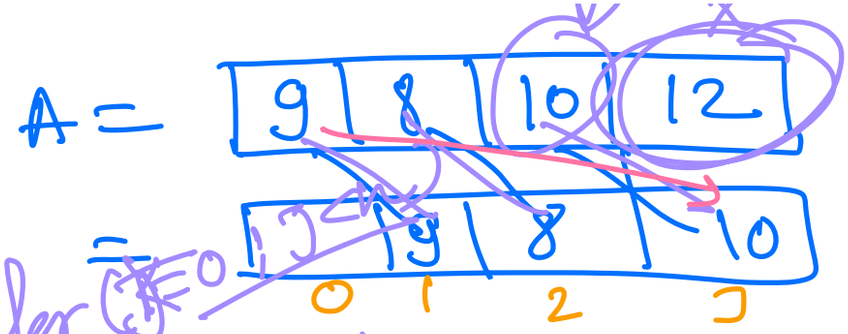
Doubt session

Scaler Youtube → search ⇒ Backend



rotate probⁿ.

K ≥ 10^12



for (i=0

for (i=1; i<N-1; i++){

temp = A(i)

A[i] = A(i-1)

}

A[i%N] = temp

* (K*N)