Workshop #2
Prefix Sums

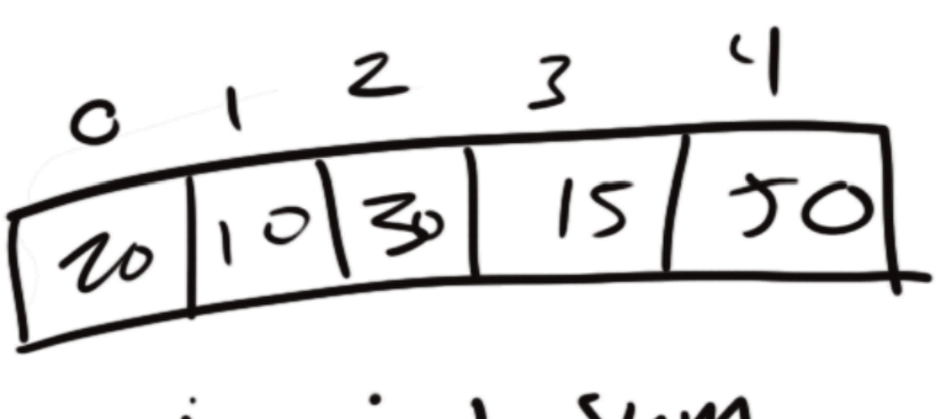
ACM ICPC Interview Track
Tuesday 4/23/19

Administrivia

- Mailing List
  - tinyurl.com/ICPC-Interview-Sign-Up
- Slides
  - tinyurl.com/ICPC-Interview-Slides
- Attendance Code
  - members.uclaacm.com/login
  - **uwu**

Motivation

- Given array **A** of size **N**, for all pairs of indices **i, j**, compute the sum of elements between the indices, inclusive.



i	j	Sum
0	1	30
1	3	55
1	4	105

Naive Solution

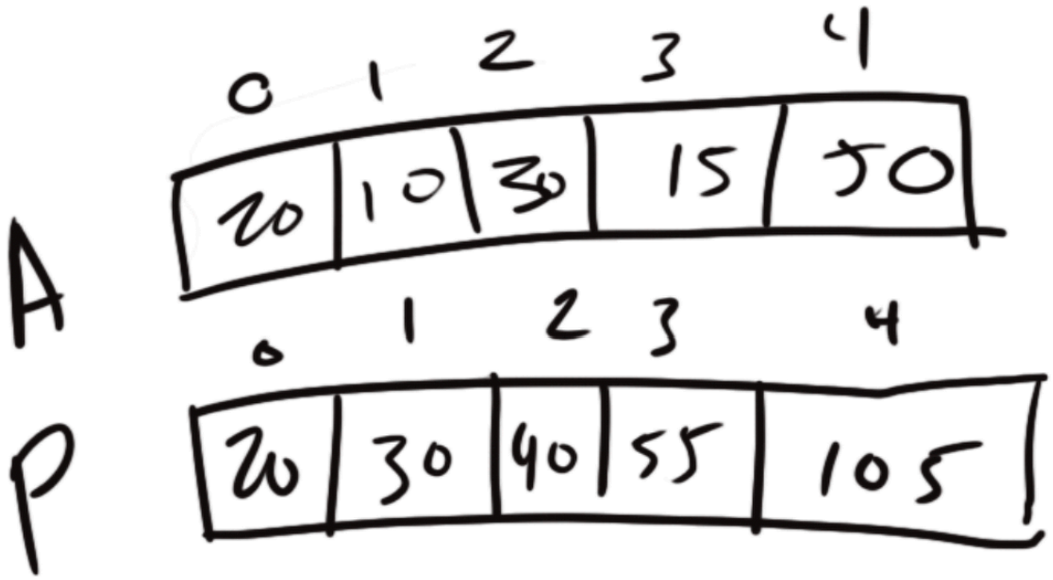
- Loop through all pairs **i, j**. Keep track of a running sum from **i** to **j**.

- Runtime: **O(N^3)**

```
for i=0...N
  for j=0...N
    sum=0
    for k=i..j
      sum += A[k]
```

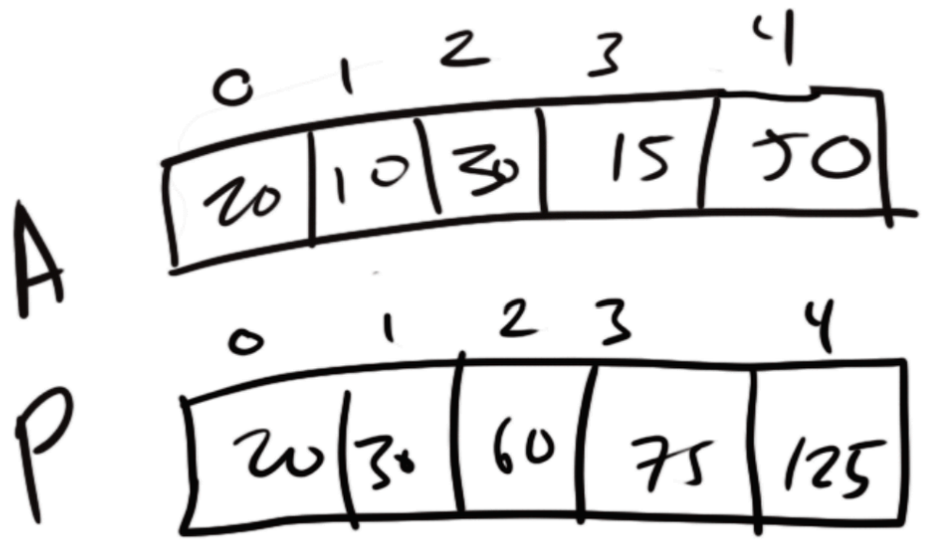
Prefix Sums

$P[i] = A[0] + A[1] + \cdots A[i]$



i	sum breakdown	P[i]
0	20	20
1	20 +10	30
2	20 +10 +30	60
3	20 +10 +30 +15	75
4	20+10+30+15+15	125

Range Sums



Assume P[-1] = 0

RANGE_SUM(i,j) = P[j] - P[i-1]

Examples

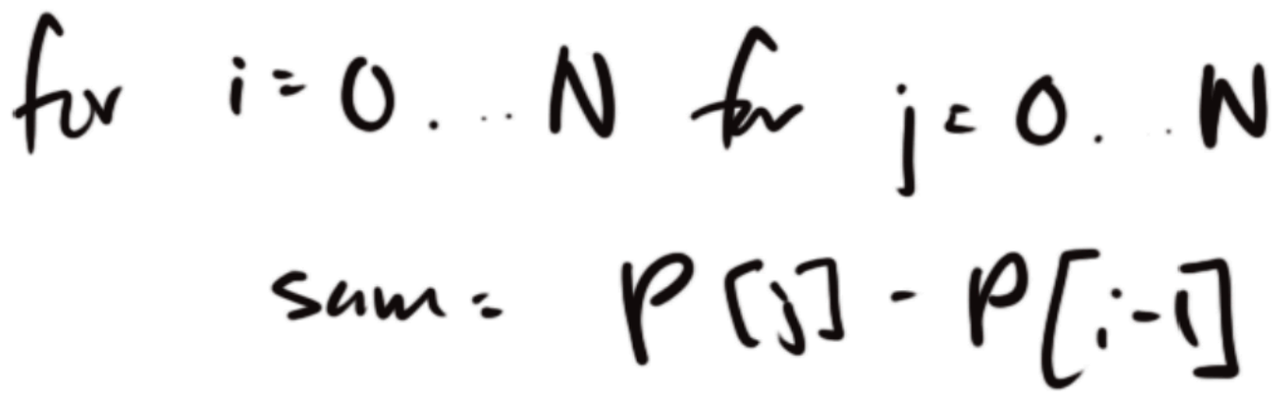
i	j	RS(i,j)	P[j] - P[i-1]
0	3	55	55 - 0 = 55
2	4	95	125 - 30 = 95
1	2	40	60 - 20 = 40

Efficient Solution

- Loop through all pairs **i, j**.Compute range sum with prefix sums.

- Runtime: **O(N^2)**

- Time/Space Tradeoff



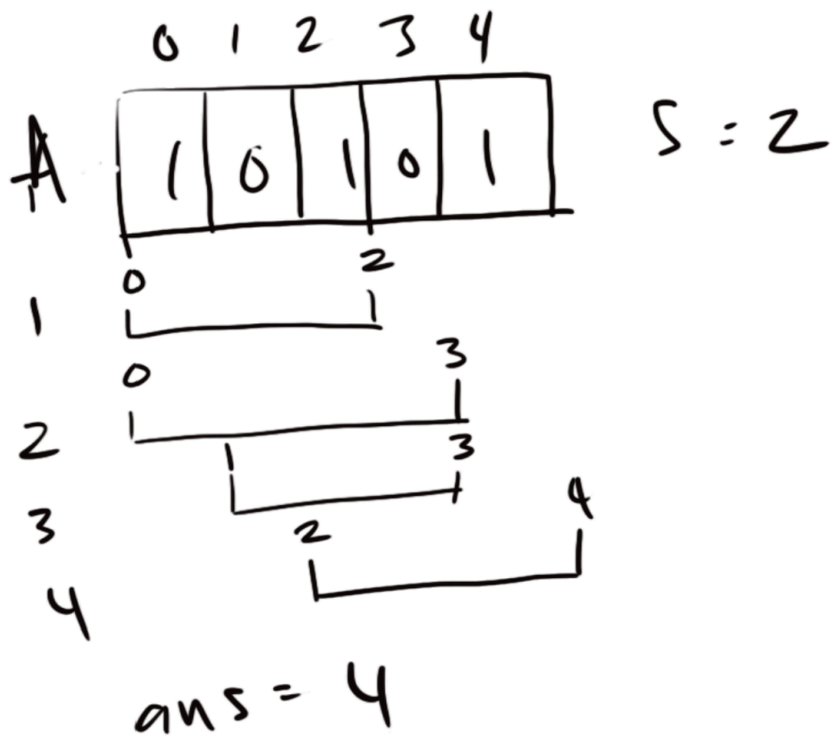
Leetcode 930
Binary Subarrays with Sum

930. Binary Subarrays With Sum

Oct. 27, 2018 | 6.3K views

Average Rating: 4.69 (13 votes)

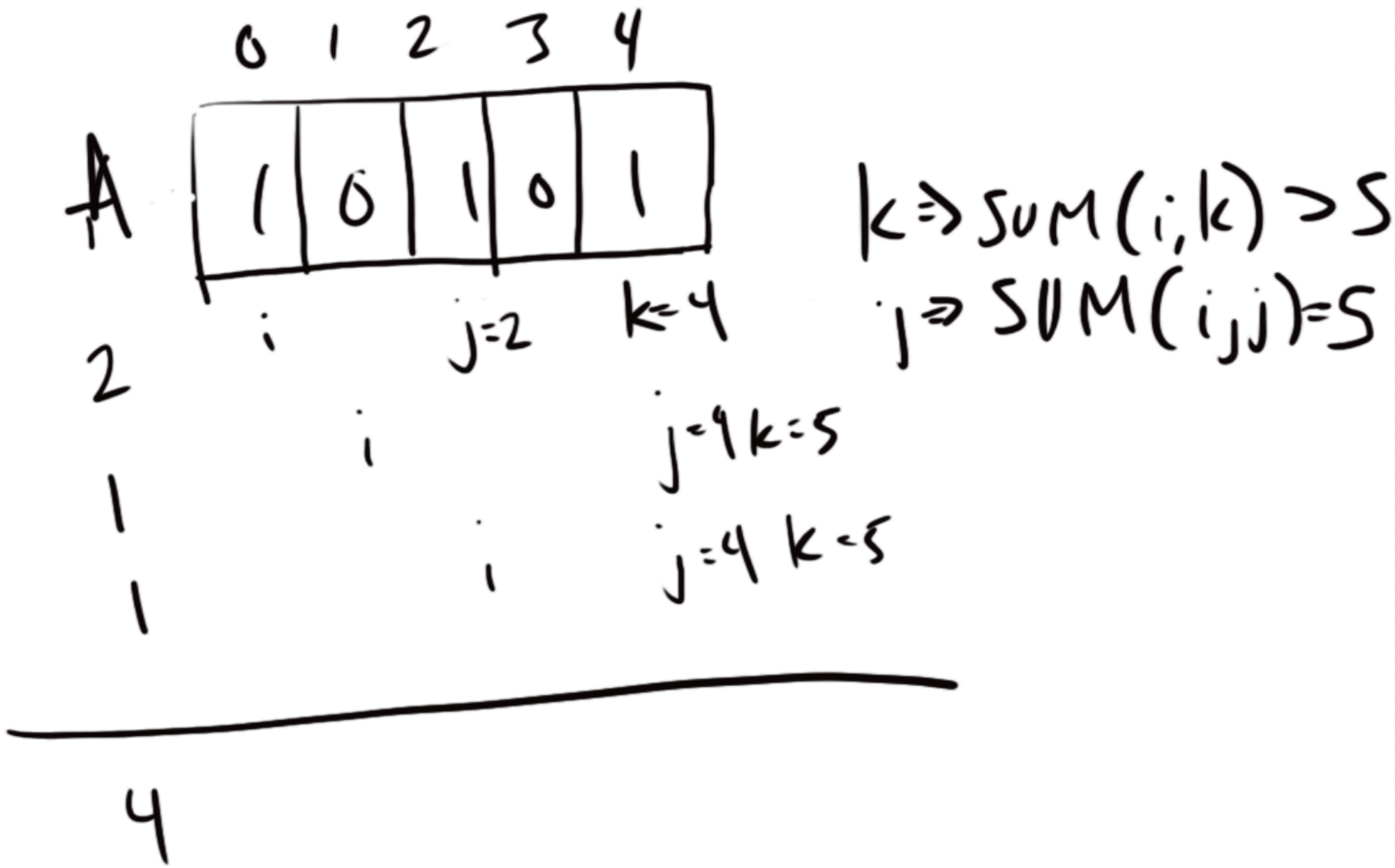
In an array `A` of `0`s and `1`s, how many **non-empty** subarrays have sum `S`?



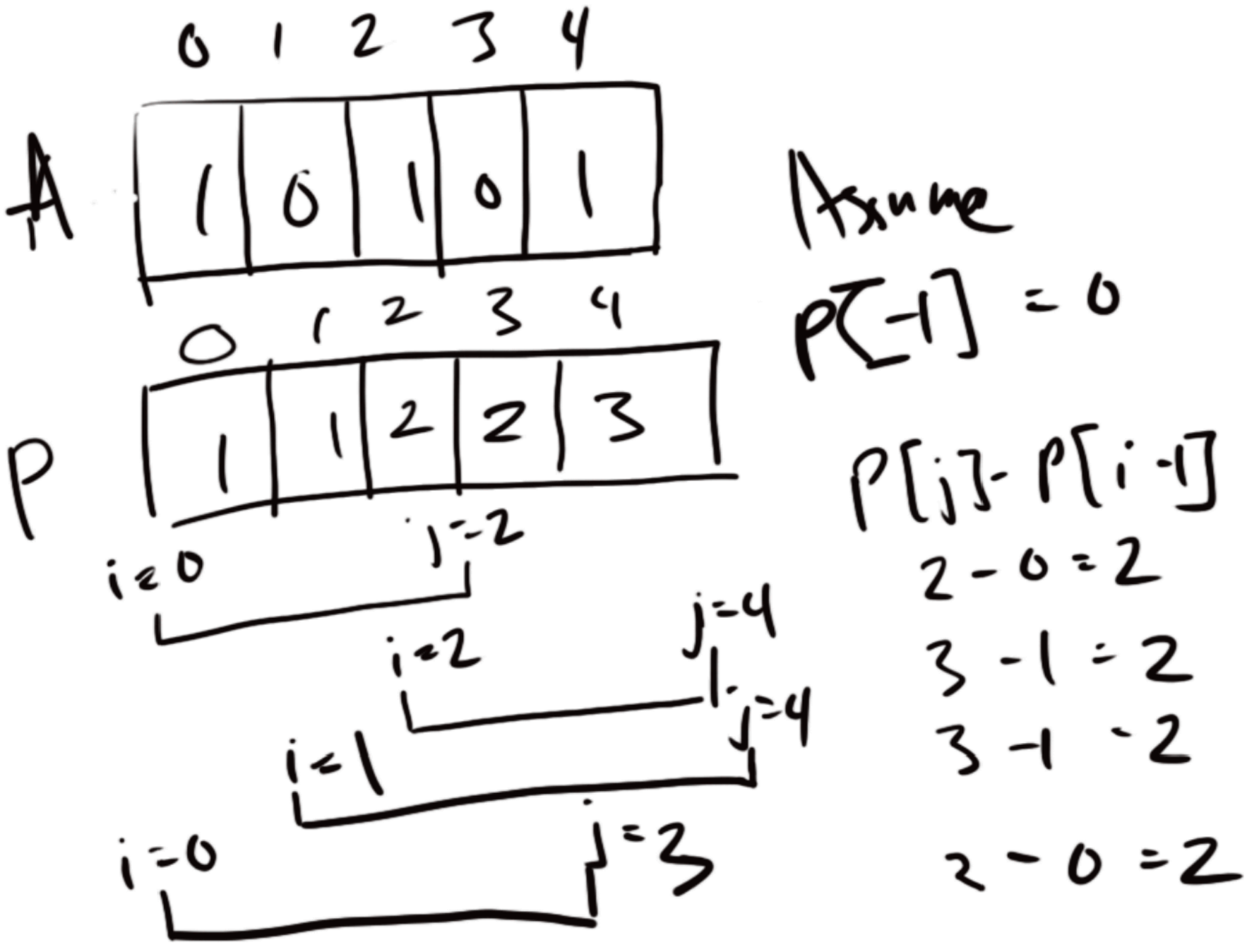
Naive Solution

- For each index **i**, find
  - **j>i** where **SUM(i,j)=S**
  - **k>i** where **SUM(i,k)>S**
- Add **k-j** to the answer.
- Runtime: **O(N^2)**

Naive Solution Visualized



Prefix Sums Visualization



Prefix Sums Solution

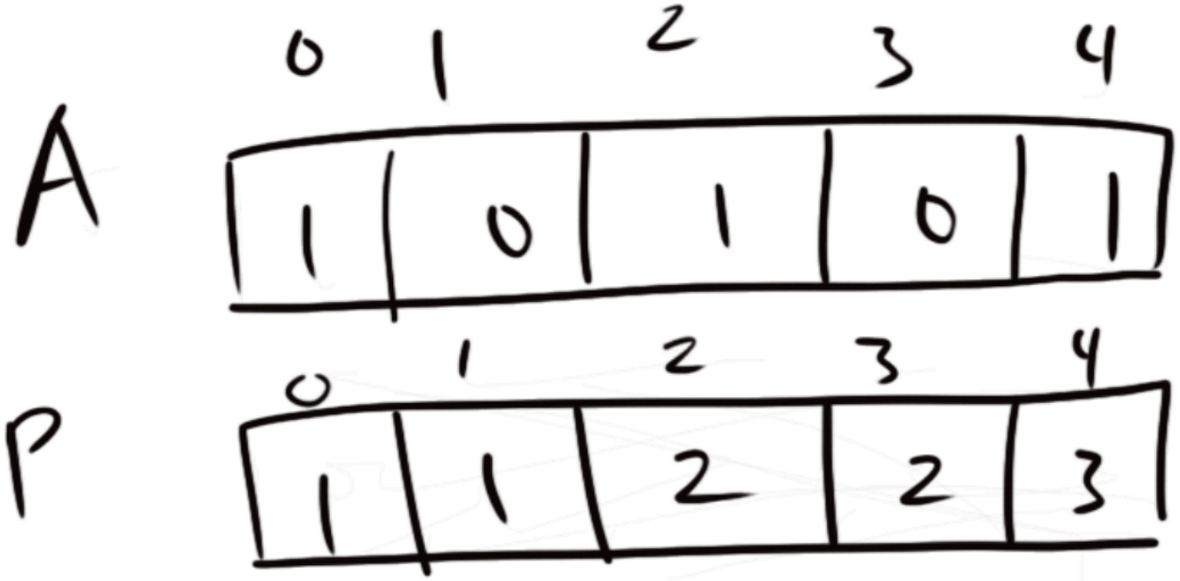
- We create a prefix array **P**.
- Observation. The number of subarrays summing to **S** ending at index **j** is the number of indices **i<j** where **P[j]-P[i-1]=S**.
- Create a map **m**. For each index **i**, **m[p[i-1]]++**. Then, we are looking for the number of previous indices with count **P[i-1]=P[j]-S**. So we add **m[p[i]-S]** to the answer.
- Runtime: **O(N)**

Psuedocode

```
# create prefix sums
for i = 0 ... N
    P[i+1] = P[i] + A[i]

# calculate ans
map m;
ans = 0
for i = 1 ... N (include N)
    m[P[i-1]]++
    ans += m[P[i]-S]
```

Prefix Sums Walkthrough



Assume
P[-1] = 0

① m[P[j-1]] += 1

m

i	m[i]
0	1
1	
2	
3	

^
j=0

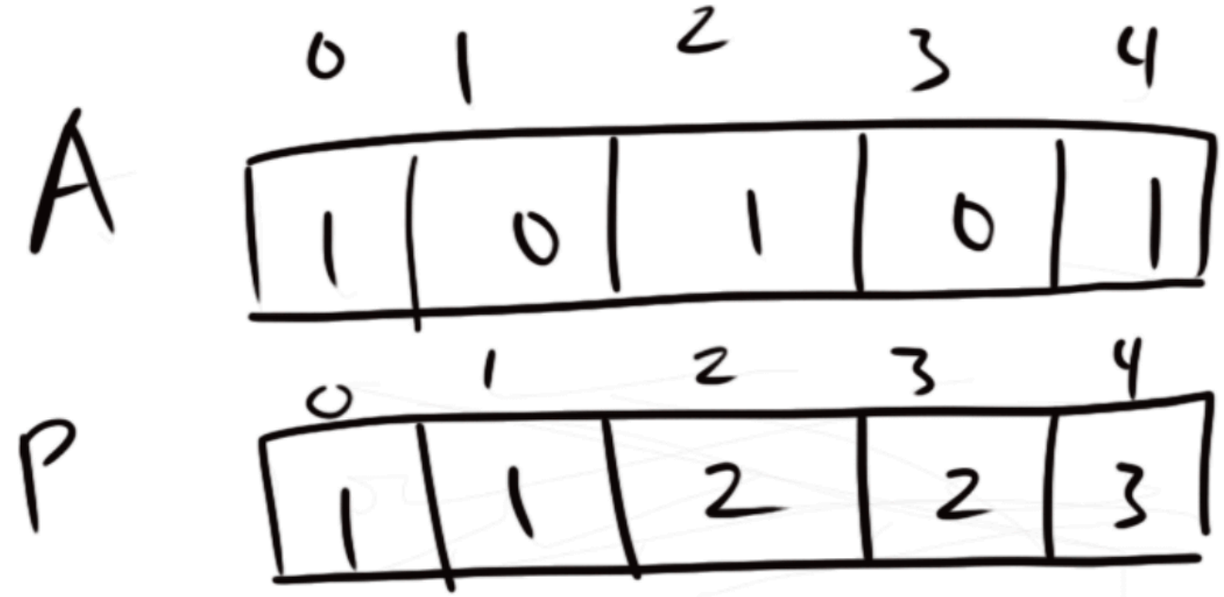
Recall
P[j] - P[i-1] = s

P[j] - s = P[i-1]

② ans += m[P[j] - s]
m[1-2 = -1]
= 0

0 → 0

Prefix Sums Walkthrough



Assume
P[-1] = 0

① m[P[j-1]+1

m	
i	m[i]
0	1
1	1
2	
3	

② ans += m[P[j]-S]

m[1-2=-1]

= 0

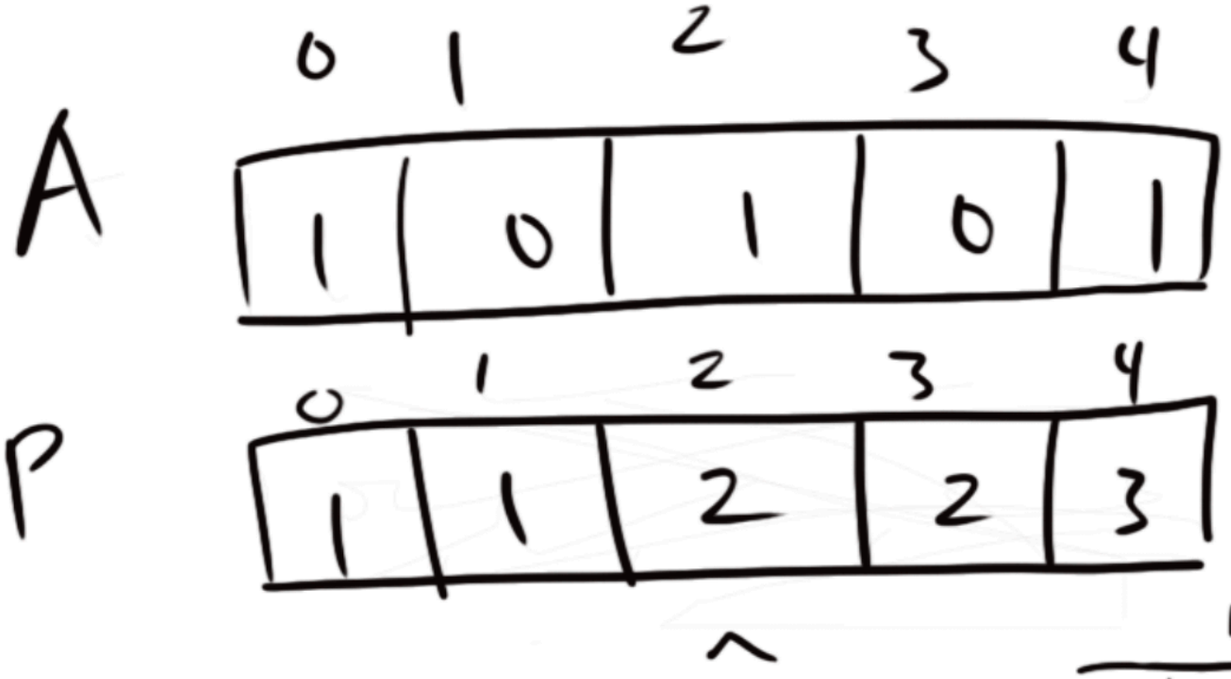
0 → 0

Recall

P[j]-P[i-1]=S

P[j]-S=P[i-1]

Prefix Sums Walkthrough



Assume
P[-1] = 0

① m[P[j-1]]++

m

i	m[i]
0	1
1	2
2	
3	

j = 2

② ans += m[P[j]-s]

m[2-2] = 1

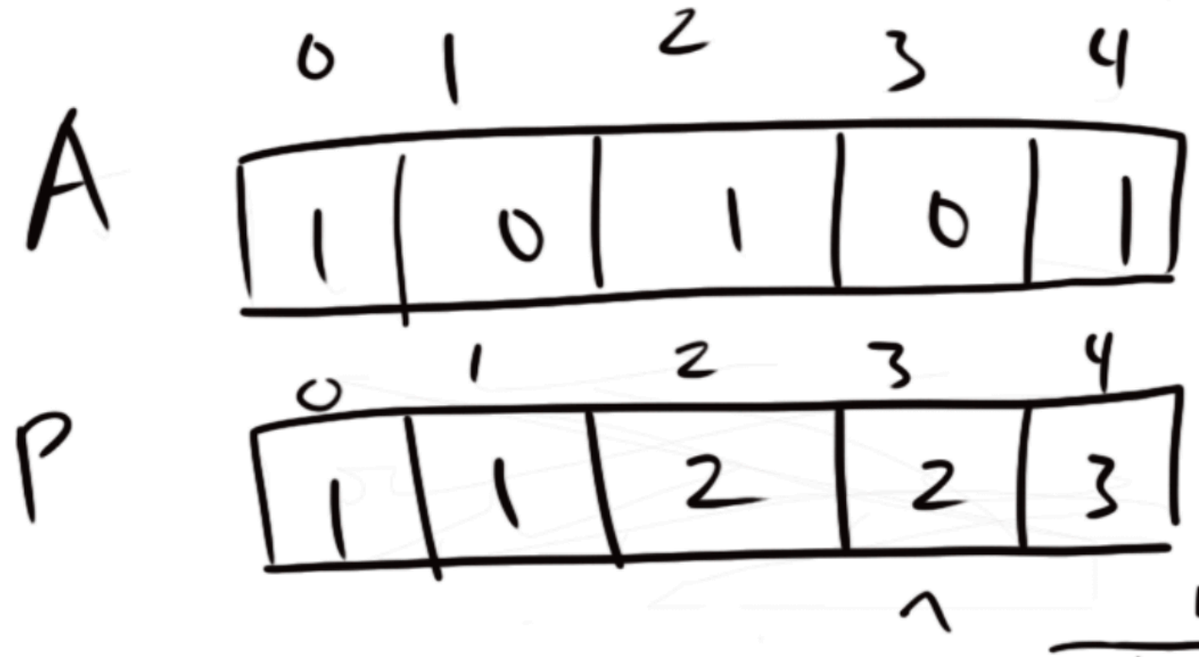
0 → 1

Recall

P[j] - P[i-1] = s

P[j] - s = P[i-1]

Prefix Sums Walkthrough



Assume
P[-1] = 0

① m[P[j-1]]++

m

i	m[i]
0	1
1	2
2	1
3	

② ans += m[P[j]-s]

m[2-2] = 1

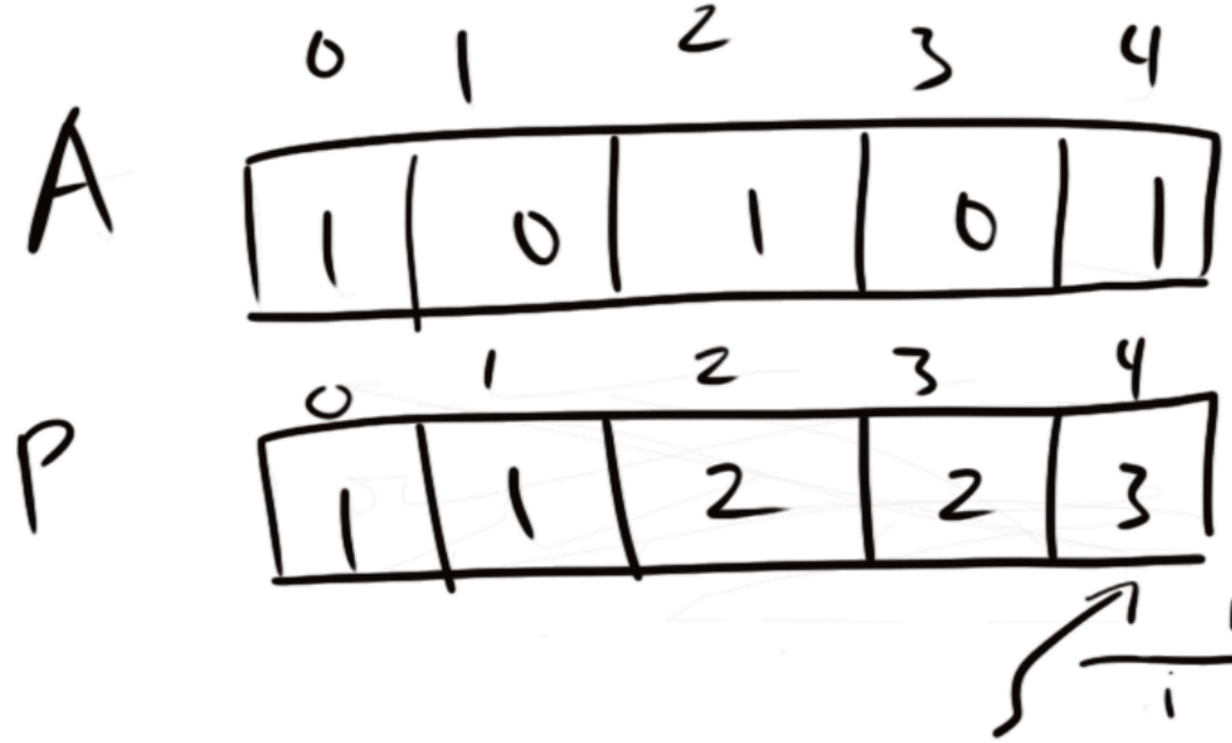
1 → 2

Recall

P[j]-P[i-1]=s

P[j] - s = P[i-1]

Prefix Sums Walkthrough



Assume
P[-1] = 0

① m[P[j-1]]++

m

i	m[i]
0	1
1	2
2	2
3	

j=4

② ans += m[P[j]-5]

m[3-2] = 2

2 → 4 (final answer)

Recall
P[j]-P[i-1]=s

P[j]-s=P[i-1]

Problems

- LC 477: Total Hamming Distance
- LC 134: Gas Station
- LC 974: Subarray Sums Divisible by K

LC 477: Total Hamming Distance

keep prefix counts for # of #s with bit i ($0 \le i \le 31$)

either 0 or 1.

32 bits total

① Count # of previous #s with mismatched bit.

② By incrementing current count, future mismatches also accounted for.

LC 477: Total Hamming Distance

```
# prefix counts
ones = [0 ... 0]  # |ones| = 32

zeros = [0 ... 0] # |zeros| = 32

ans = 0

for bit = 0 ... 32
    if (bit == 1) ones[bit_index]++
    else  ans += zeros[bit_index]
for bit = 0 ... 32
    if (bit == 0) zeros[bit_index]++
    else  ans += ones[bit_index]
return 0
```

Thank You