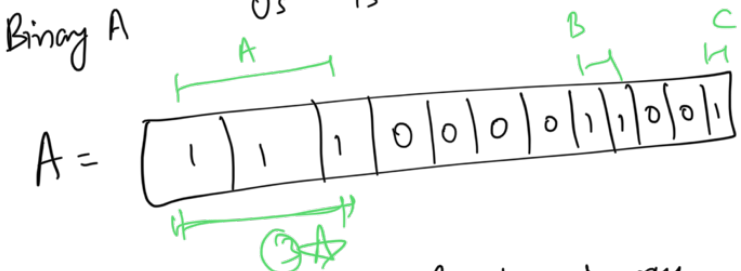
Arrays

0

Amazon
Goldman
Microsoft

Binary A 0s 1s



(0) Return the size of the longest subarray with only 1s

☆ (3)

```
global_answer = 0 //
present_answer = 0
for(i=0, i<N, i++)
{
    if (A[i] == 1)
    {
        pa++
    }
    else if (A[i] == 0)
        pa = 0
}
if (pa > ga)
```

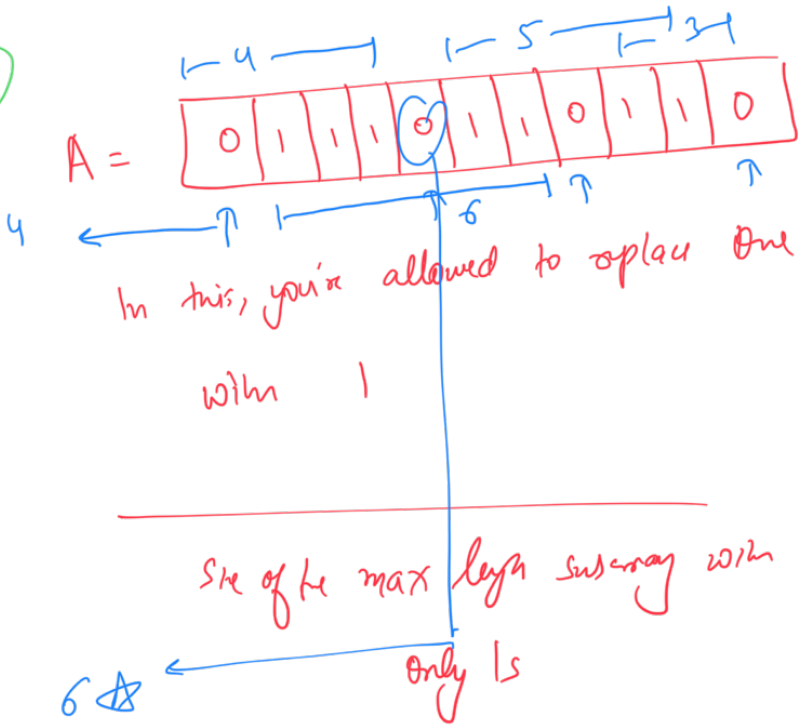
$ga = pa$

}

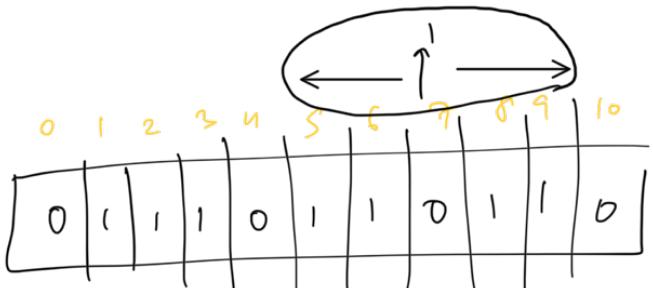
$T.C = O(n)$

$S.C = O(1)$

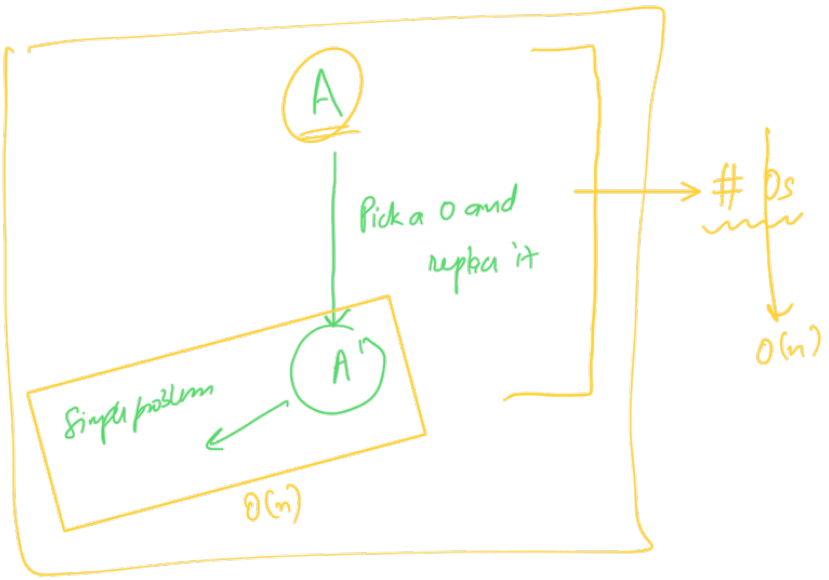
Q2



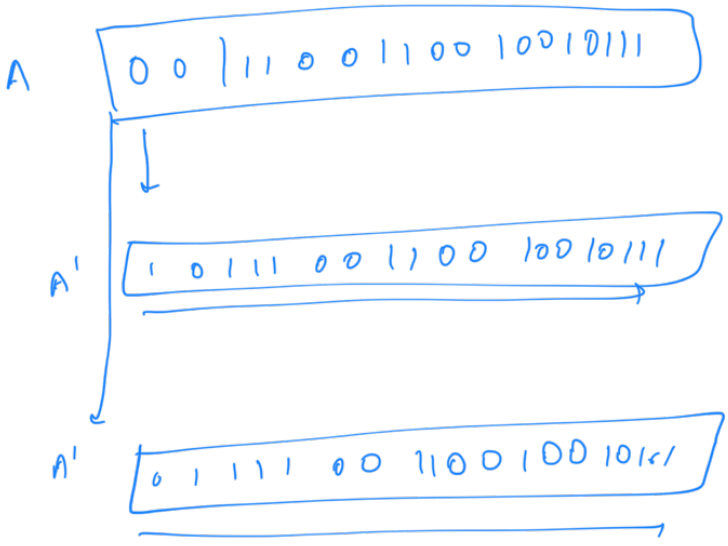
20/April/2022



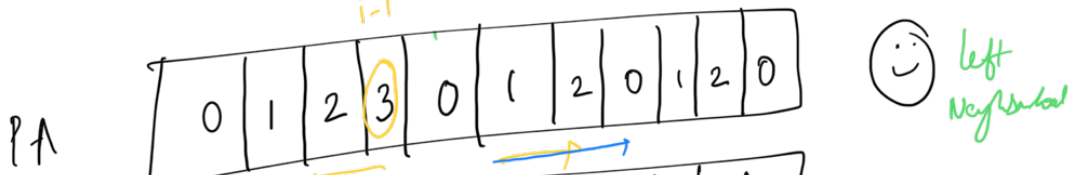
|A| = n

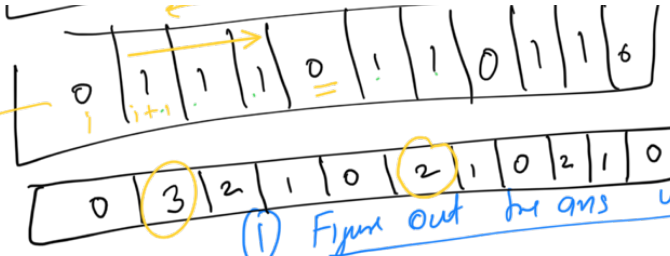


T.C : $O(n^2)$



Q) can we carry forward some boiled info.





SA

O(n)

(1) Figure out the ans without applying any 0

1 + 3 = 4

3 + 1 + 2 = 6

(2) Move to any 0 and try all

Replace 0 → 1

Right most

☆ 3

☆ 4

☆ 6

☆ 5

☆ 3

(2) T.C (P.A) = O(N)

S.C (P.A) = O(N)

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

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

(4) Traverse array to find 0's O(N)

inside O(1)

O(N)

T.C = O(n) + O(n) + O(n) + O(n)

= O(N) ☺

Aux SC = O(N)

h.w Can you do this with auxiliary array

I) ans = get Max Length Whose Replaced (A)

PA = [ ]

II)

```
if (A[0] == 0)
    PA[0] = 0
else
    PA[0] = 1
```

```
for(i=1; i < N; i++)
{   if(A[i] == 0)
        PA[i] = 0
```

else (A(i)==1)

PA(i) = 1 + PA(i-1)

}

S.A

S.A = [ ]

if(A(n-1) == 0)

S.A(n-1) = 0

else S.A(n-1) = 1

for(i = N-2 ; i ≥ 0 ; i--)

{

S.A(i) = S.A(i+1) + 1

}

for(i=0 ; i < N ; i++)

{

if (A(i)==0)

{

LN = 0

if(i != 0)

LN = PA(i-1)

RN = 0

if (i != N-1)

S.A(i+1)

RN = ...

if (LN + 1 + RN > ans)

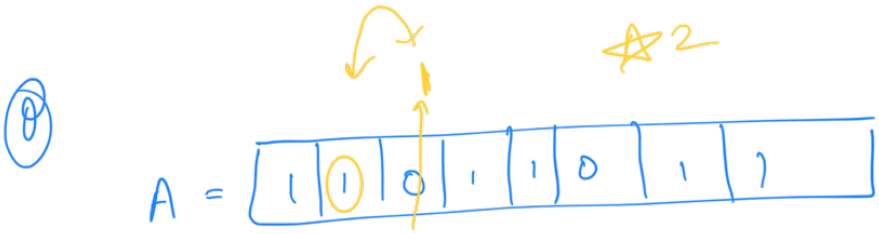
ans = LN + RN + 1

}

}

return ans

}



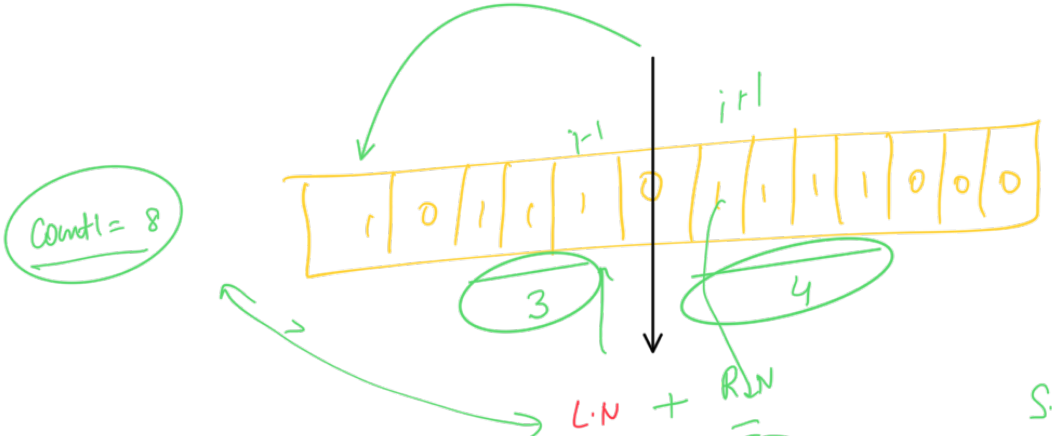
Count1 = 6

Left of the longest subarray with only 1s

Swap

0

swap

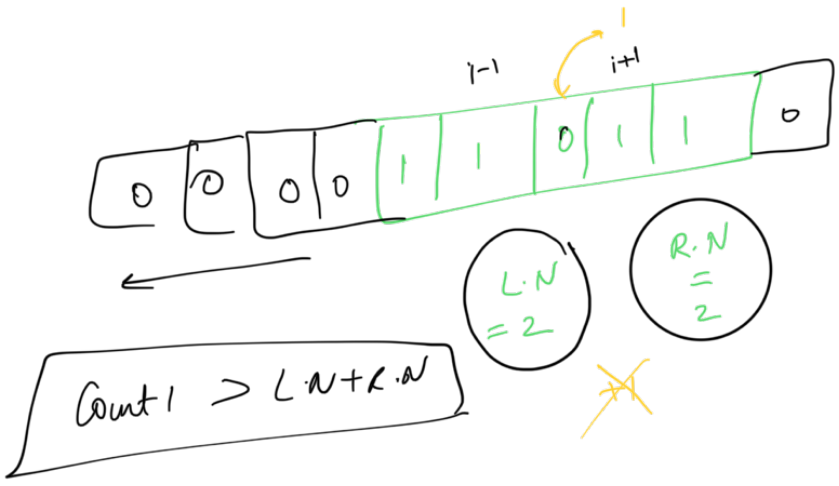


Count1 = 8

S.A

PA

Best swap = swap beyond my CN & Beyond PN



Count1 = Δ

O(N) ← ans = get MaxLen Without Replacement (A)

O(N) ←

PA = [3

O(N) ← { //code

S.A = [3

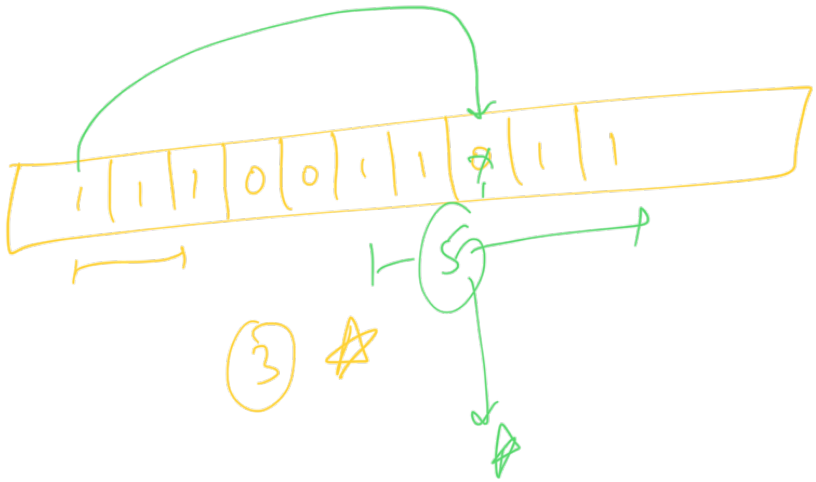
O(N) ← { //code

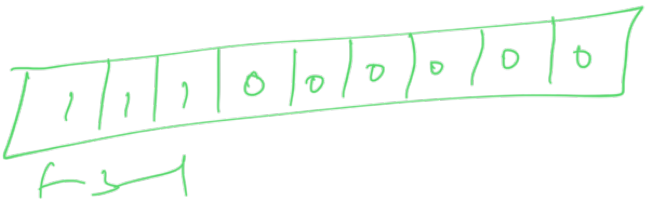
```
(
for(i=0, i<N, i++)
{
    if (A[i]==0)
    {
    // check if you can
      swap with a 1
      beyond my neighborhood
    (best option 😊😊)
      if (count1 > L·N + R·N)
      {
          if( L·N + R·N + 1 > ans)
          ans = L·N + R·N + 1
      }
    else
    {
        if(L·N + R·N > ans)
        ans = L·N + R·N
    }
}
```

beyond your neighborhood

from the wilderness

No 1 left out of your own neighbor





T.C = O(N)

Aux S.C = O(N)

A = [ ]

Find the number of triplets

i j k ≡ indexes

s.t

$i < j < k$

&

$A(i) < A(j) < A(k)$

Ex

0	1	2	3	4
3	4	6	9	2

IAFS

0 1 2 ✓.

0 1 3 ✓. 0 2 3 ✓

0 1 4 X 0 2 4 X

1 2 3 ✓

1 2 4 ✗

2 3 4 ✗

0 3 4 ✗

1 3 4 ✗

T.C = O(N3)

sup S.C = O(1)

count=0

for i=0, i<N

for j=i+1, j<N

for k=j+1, k<N

if (A(i)<A(j)<A(k)

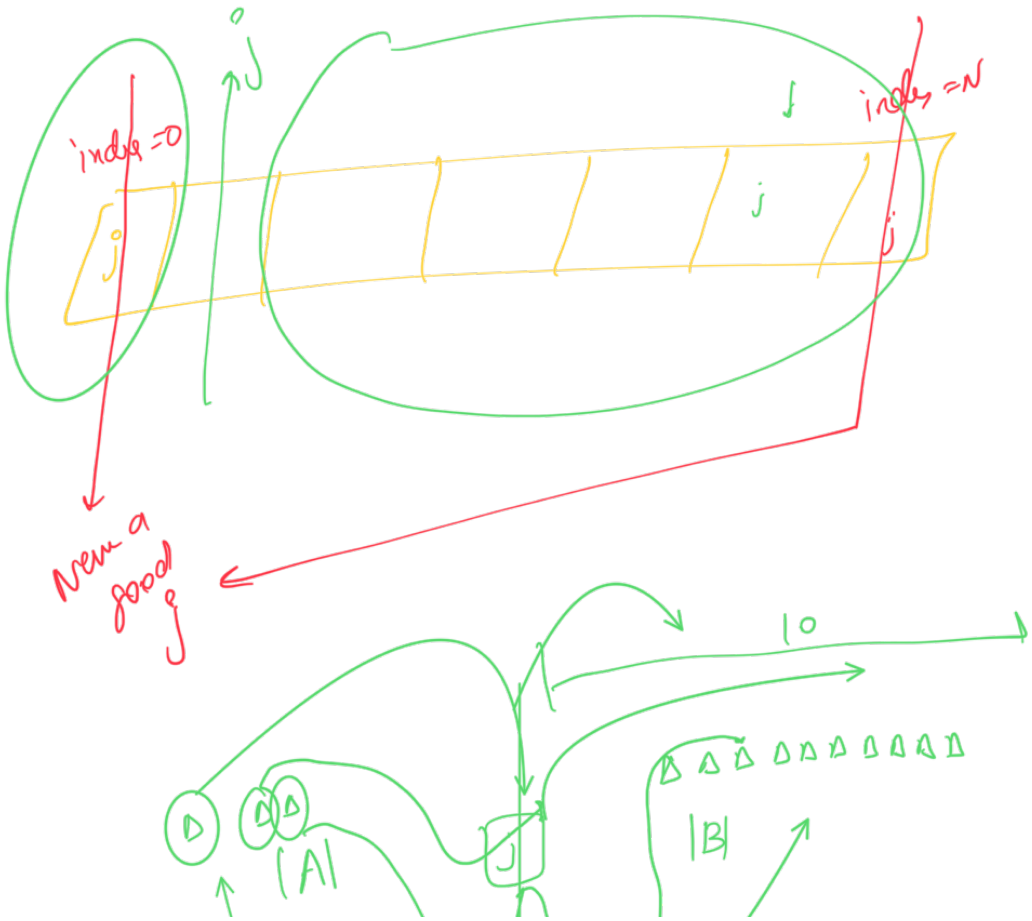
count++

return count

$A < E(j)$

$B > E(j)$

$A \subset B$



Obs 3

|A| x |B|

3 x 10 = 30

"good" triplets with a fixed j

=

No of elems of A left smaller than E(j)

X

No of elems on A right with are bigger than E(j)

E = [ ]

ans = 0

for (j = 1, j < N-1, j++)

O(N) ←

{

E(j)

Use a BST O(log N)

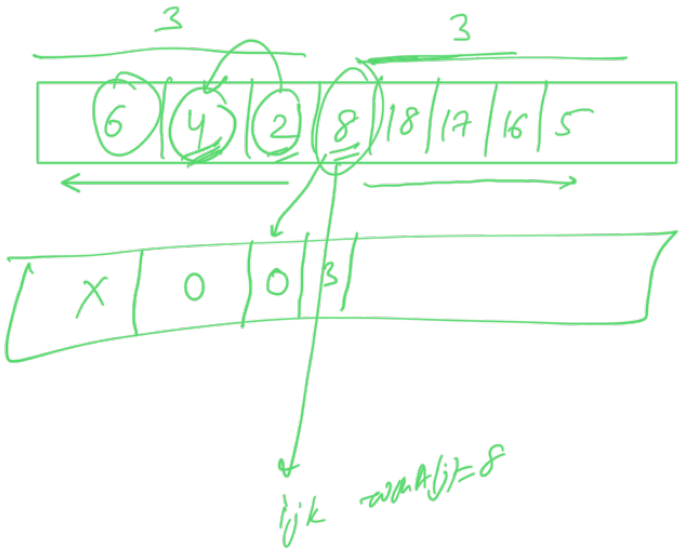
ans = ans + # elm < E(j) X # elm > E(j)

}

B.S.T. T.C = N log N

:) waiti

Binary Search Trees



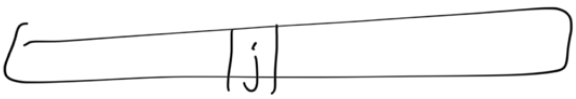
Prefix and Suffix Array want help us.

BST would

$O(n^3)$ Brute force

$O(n^2)$ ← ugly Array

$O(n \log n)$ ← buildinfo + BST




```
ans = 0
for (j = 1 ; j < N-1 ; j++)
{
    CountLeftSmaller = 0
    for (i = 0 ; i < j ; i++)
        if ( A(i) < A(j))
            CLS++

    CountRightBigger = 0
    for (k = j+1 ; k < N ; k++)
        if (A(k) > A(j))
            CRB++

    ans = ans + CLS X CRB
```

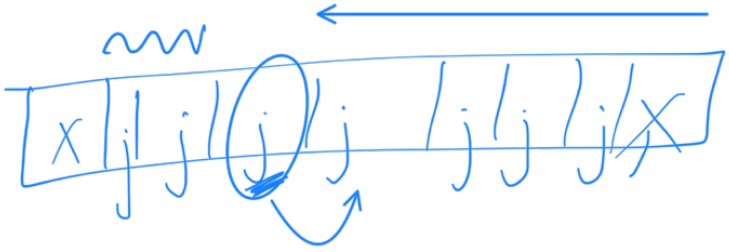
$O(N)$ ← for (j = 1 ; j < N-1 ; j++)

$O(N)$ ← CountLeftSmaller = 0; for (i = 0 ; i < j ; i++) if (A(i) < A(j)) CLS++

$O(N)$ ← CountRightBigger = 0; for (k = j+1 ; k < N ; k++) if (A(k) > A(j)) CRB++

$O(n)\ [O(N)+O(N)]$

$T.C = O(n^2)$



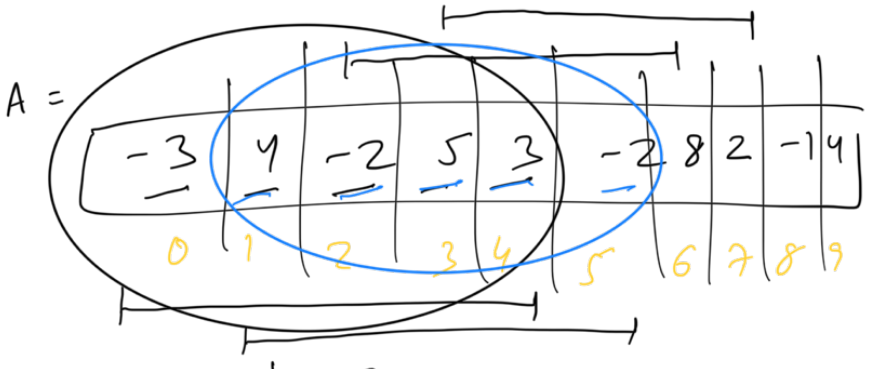
(0 5 0)

(0 6 0)

(0 7 0)

---

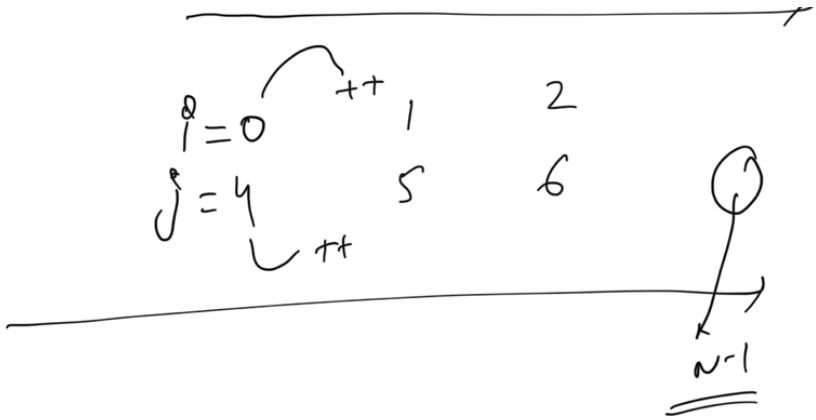
(8)



subarray size = $k = 5$

$Sum_{SA1} = 7$

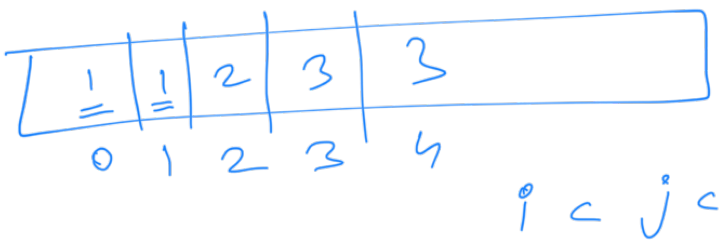
$Sum_{SA2} = 8$



O(N) subarrays

Find the map sum

subarray of size = 5 = k



0 2 3 ✓

1 2 3 ✓