Binary Subarrays with Sum / Count subarray sum equals K

arr = [1 0 1 0 1] goal = 2

Binary Subarrays with Sum / Count subarray sum equals K

arr = [1 0 1 0 1] goal = 2

Binary Subarrays with Sum / Count subarray sum equals K

↳ [+, -]

arr = [1 0 1 0 1] goal = 2

Hashing

TC → O(N)

SC → O(N)

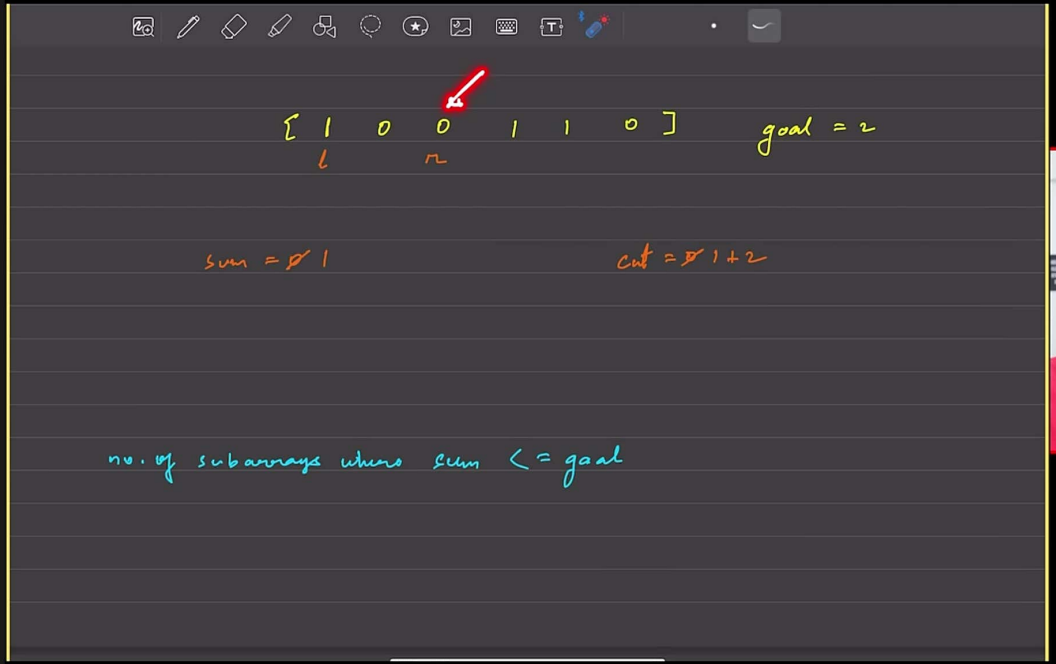
[1 0 0 1 1 0] goal = 2

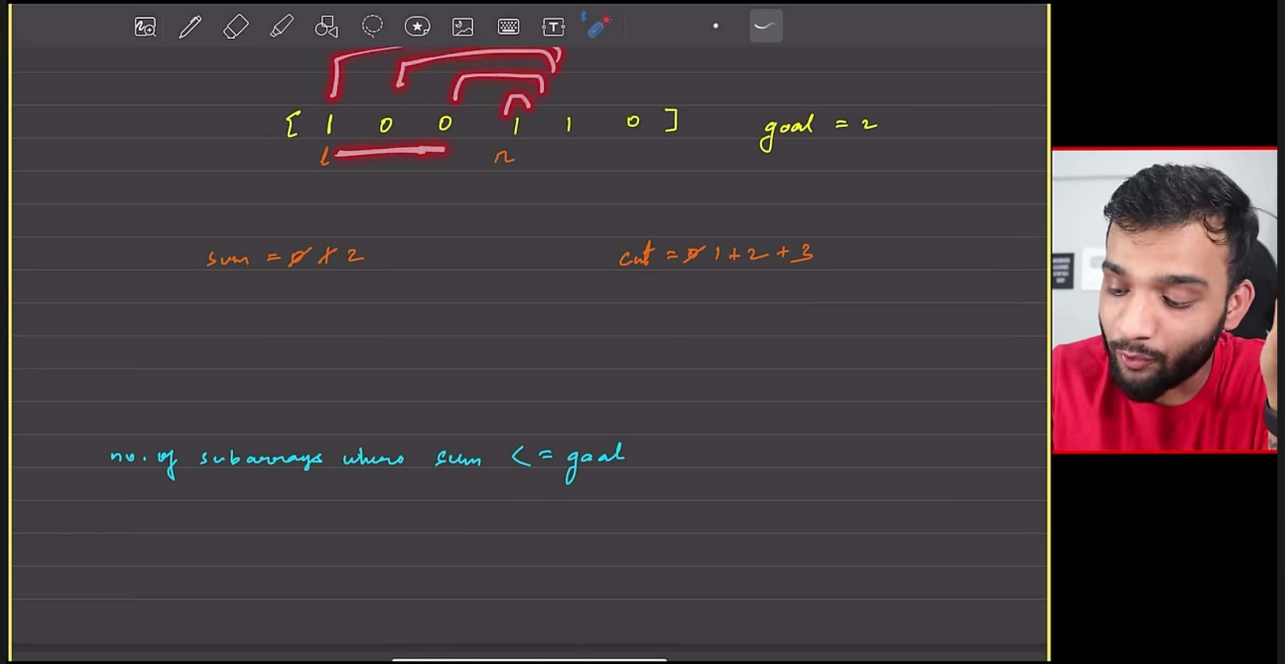
l r

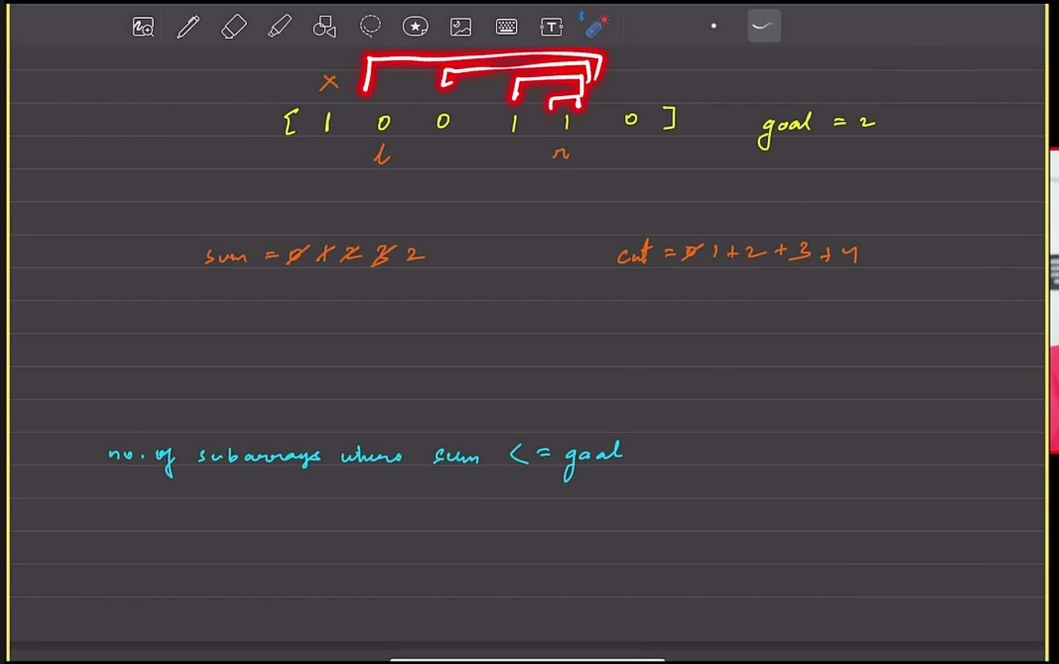
sum = 1

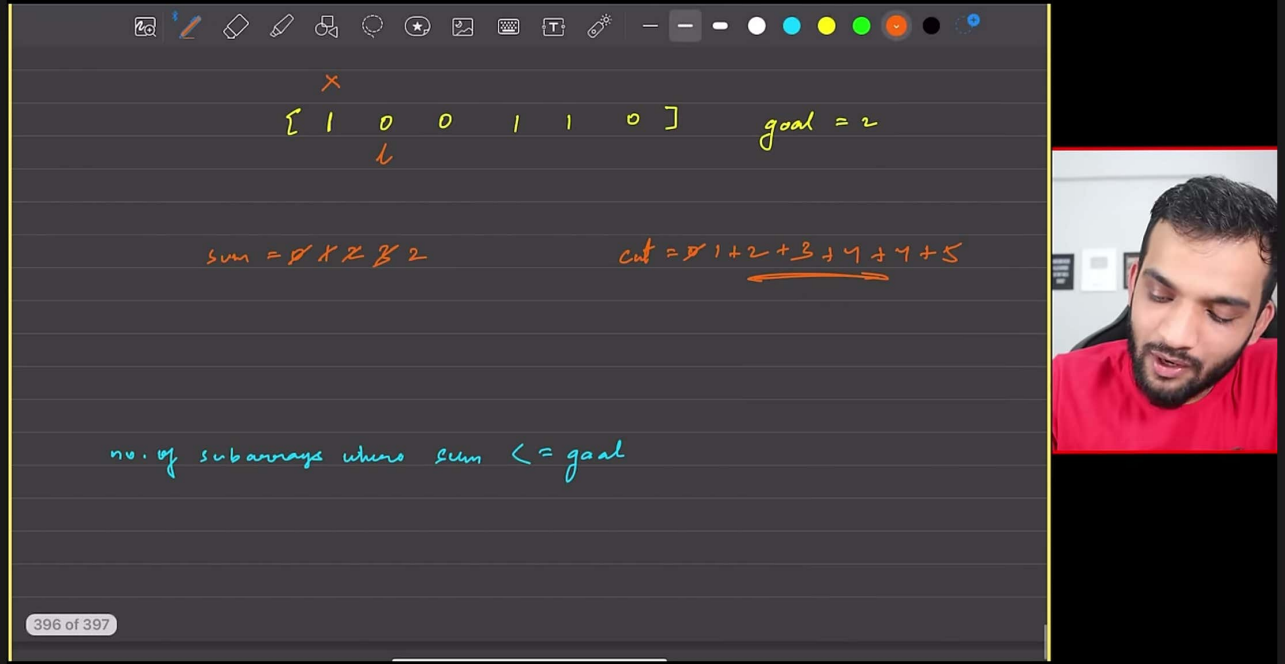
cnt = 1

no. of subarrays where sum <= goal


[1 0 0 1 1 0]
goal = 2
l
r
sum = 1
cnt = 1 + 2
no. of subarrays where sum <= goal


[1 0 0 1 1 0]
goal = 2
l
r
sum = 0 1 2
cnt = 0 1 2 3
no. of subarrays where sum <= goal


[1 0 0 1 1 0]
goal = 2
l
r
sum = 0 + 1 + 2 + 2
cnt = 0 + 1 + 2 + 3 + 4
no. of subarrays where sum <= goal


[1 0 0 1 1 0]
goal = 2
sum = 1 + 2 + 3
cut = 1 + 2 + 3 + 4 + 5
no. of subarrays where sum <= goal
396 of 397

```
func ( list <int> nums , goal)
{
    l = 0 , r = 0 ,  sum = 0 , cnt = 0

    while (r <= nums.size())
    {
        sum += nums[r];

        while (sum > goal)
        {
            sum = su
```

```
func ( list <int> nums , goal)
{
    l = 0 , r = 0 ,  sum = 0 , cnt = 0

    while (r <= nums.size() )
    {
        sum += nums[r] ;

        while ( sum > goal)
        {
            sum = sum - nums[l];
            l = l + 1 ;
        }
        cnt = cnt + ( r - l + 1) ;
        r = r + 1 ;
    }
```

```
func ( list <int> nums , goal)
{
    if (goal < 0) return 0;
    l = 0 , r = 0 ,  sum = 0 , cnt = 0

    while (r <= nums.size() )
    {
        sum += nums[r] ;

        while ( sum > goal)
        {
            sum = sum - nums[l];
            l = l + 1 ;
        }
        cnt = cnt + ( r - l + 1) ;
        r = r + 1 ;
```

sum == goal

no. of subarrays where | sum <= goal

func(nums, goal)
-
func(nums, goal-1)

```
func(list<int> nums, goal)
{
    if(goal < 0) return 0;
    l=0, r=0, sum=0, cnt=0

    while(r <= nums.size())
    {
        sum += nums[r];
```

sum == goal

no. of subarrays where | sum <= goal |

func(nums, goal)
-
func(nums, goal-1)

```
func( list<int> nums, goal)
{
    if (goal < 0) return 0;
    l = 0, r = 0, sum = 0, cnt = 0

    while (r <= nums.size())
    {
        sum += nums[r];
```

```
func ( list<int> nums, goal)
{
    if (goal < 0) return 0;
    l = 0, r = 0, sum = 0, cnt = 0

    while (r <= nums.size())          → O(N)
    {
        sum += nums[r];

        while (sum > goal)            → O(N)
        {
            sum = sum - nums[l];
            l = l + 1;
        }
        cnt = cnt + (r - l + 1);
```

TC → O(2N)
SC → O(1)

sum == goal

no. of subarrays where | sum <= goal

func(nums, goal) - func(nums, goal - 1)

TC → O(2 × 2N)
SC → O(1)

func(list<int> nums, goal)
{
if (goal < 0) return 0;
l = 0, r = 0, sum = 0, cnt = 0

while (r <= nums.size()) → O(N)
{
sum += nums[r];