



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 $\rightarrow O(N)$

SC $\rightarrow O(N)$





[1 0 0 1 1 0] goal = 2
L R

sum = 1

cut = 1

no. of subarrays where sum \leq goal





$[1 \ 0 \ 0 \ 1 \ 1 \ 0]$ goal = 2
 \downarrow \downarrow

sum = 1

cut = 1 + 2

no. of subarrays where sum \leq goal



 $[1 \ 0 \ 0 \ 1 \ 1 \ 0]$ goal = 2

sum = ~~0~~ * 2

cut = ~~0~~ 1 + 2 + 3

no. of subarrays where $\text{sum} \leq \text{goal}$





\times

[1 0 0 1 1 0] goal = 2

\downarrow \downarrow

 1 2

Diagram illustrating a subarray [1, 0, 0, 1, 1, 0] with a goal of 2. Red brackets above the array indicate subarrays starting from index 0: [1, 0, 0, 1, 1, 0], [1, 0, 0, 1, 1], [1, 0, 0, 1], [1, 0, 0], [1, 0], and [1]. Below the array, the indices 1 and 2 are marked under the second and fourth elements respectively.

sum = ~~1~~ + 2 = 2

cut = ~~1~~ + 2 + 3 + 4

no. of subarrays where sum \leq goal





~~x~~

$$[1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0]$$

~~1~~

goal = 2

~~sum = 1 + 2 + 3 + 2~~

~~cut = 1 + 2 + 3 + 4 + 4 + 5~~

no. of subarrays where $\text{sum} \leq \text{goal}$





```
func (list <int> nums, goal)
```

```
{  
    l = 0, r = 0, sum = 0, cnt = 0
```

```
while (r <= nums.size()-1)  
{
```

```
    sum += nums[r];
```

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

```
    sum = sum - nums[l];  
    l++;  
}
```



```
func (list <int> nums, goal)
```

```
{  
    l = 0, r = 0, sum = 0, cnt = 0
```

```
    while (r <= nums.size()-1)  
    {
```

```
        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;
```



no. of subarrays where sum \leq goal

sum == goal

fun(nums, goal)
-
fun(nums, goal - 1)

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

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



no. of subarrays where sum \leq goal

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

TC $\rightarrow O(2N)$

SC $\rightarrow O(1)$



no. of subarrays where sum \leq goal

TC $\rightarrow O(2 \times 2N)$
SL $\rightarrow O(1)$

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];

$O(N)$

