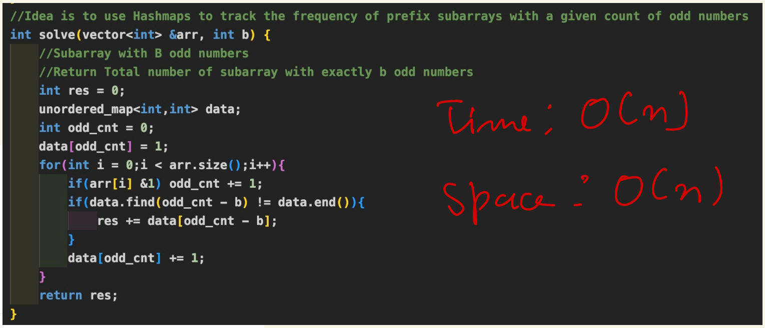
⇒ Subarray with B odd numbers

Time: O(n) Space: O(n)

Hashing we can do

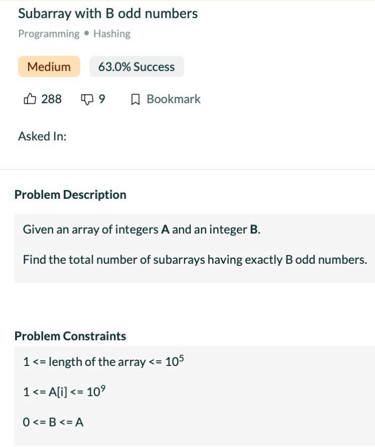
Use hashmap to track frequency of prefix subarrays with given cnt of odd numbers



```
//Idea is to use Hashmaps to track the frequency of prefix subarrays with a given count of odd numbers
int solve(vector<int> &arr, int b) {
    //Subarray with B odd numbers
    //Return Total number of subarray with exactly b odd numbers
    int res = 0;
    unordered_map<int,int> data;
    int odd_cnt = 0;
    data[odd_cnt] = 1;
    for(int i = 0;i < arr.size();i++){
        if(arr[i] &1) odd_cnt += 1;
        if(data.find(odd_cnt - b) != data.end()){
            res += data[odd_cnt - b];
        }
        data[odd_cnt] += 1;
    }
    return res;
}
```

Time: O(n)

Space: O(n)



Subarray with B odd numbers

Programming • Hashing

Medium 63.0% Success

288 9 Bookmark

Asked In:

Problem Description

Given an array of integers **A** and an integer **B**.

Find the total number of subarrays having exactly B odd numbers.

Problem Constraints

1 <= length of the array <= 10^5

1 <= A[i] <= 10^9

0 <= B <= A

Best Approach Time: O(n)

Space: O(1)

2 Pointers Sliding Window

We can find :

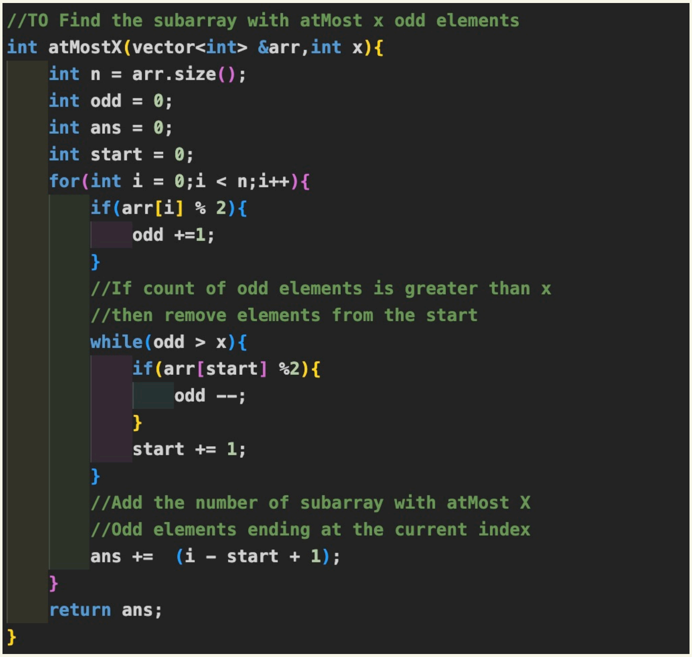
1) At Most X

−

2) At Most X -1

Exactly X

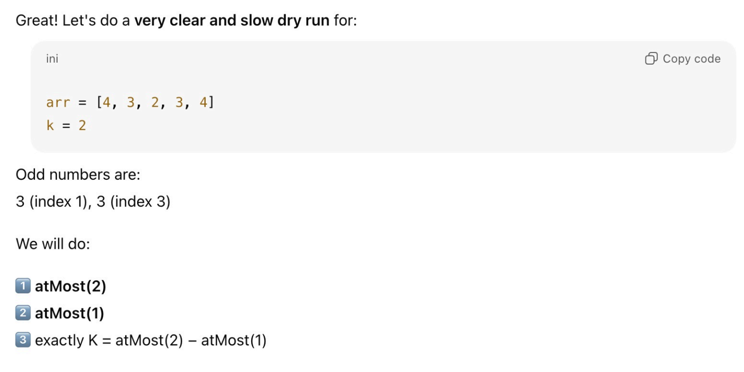
At Most X odd numbers



```cpp
//TO Find the subarray with atMost x odd elements
int atMostX(vector<int> &arr,int x){
    int n = arr.size();
    int odd = 0;
    int ans = 0;
    int start = 0;
    for(int i = 0;i < n;i++){
        if(arr[i] % 2){
            odd +=1;
        }
        //If count of odd elements is greater than x
        //then remove elements from the start
        while(odd > x){
            if(arr[start] %2){
                odd --;
            }
            start += 1;
        }
        //Add the number of subarray with atMost X
        //Odd elements ending at the current index
        ans +=  (i - start + 1);
    }
    return ans;
}
```

Example :

arr: | 4 | 3 | 2 | 3 | 4 | K = 2



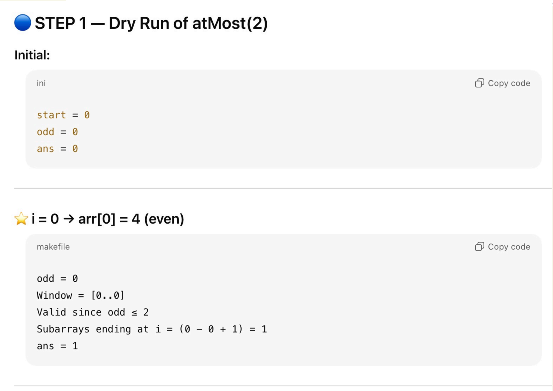
Great! Let's do a **very clear and slow dry run** for:

```ini
arr = [4, 3, 2, 3, 4]
k = 2
```

Odd numbers are:
3 (index 1), 3 (index 3)

We will do:

1️⃣ **atMost(2)**
2️⃣ **atMost(1)**
3️⃣ exactly K = atMost(2) – atMost(1)



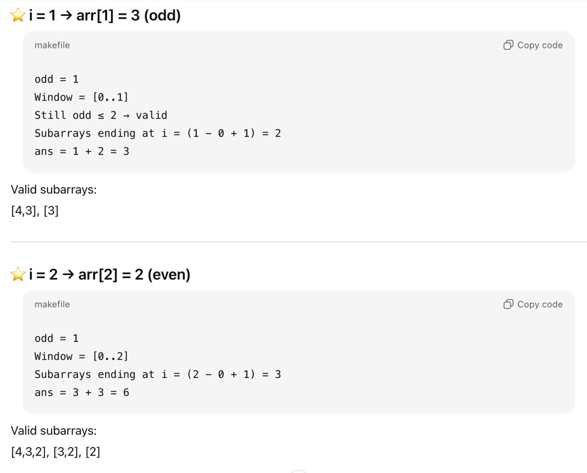
🔵 STEP 1 — Dry Run of atMost(2)

Initial:

```ini
start = 0
odd = 0
ans = 0
```

⭐ i = 0 → arr[0] = 4 (even)

```makefile
odd = 0
Window = [0..0]
Valid since odd ≤ 2
Subarrays ending at i = (0 − 0 + 1) = 1
ans = 1
```



⭐ i = 1 → arr[1] = 3 (odd)

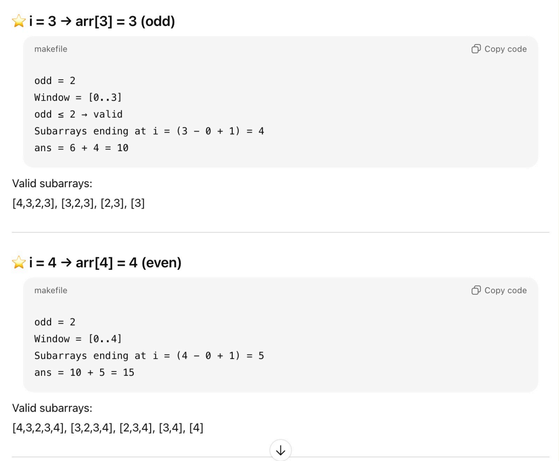
```makefile
odd = 1
Window = [0..1]
Still odd ≤ 2 → valid
Subarrays ending at i = (1 − 0 + 1) = 2
ans = 1 + 2 = 3
```

Valid subarrays:
[4,3], [3]

⭐ i = 2 → arr[2] = 2 (even)

```makefile
odd = 1
Window = [0..2]
Subarrays ending at i = (2 − 0 + 1) = 3
ans = 3 + 3 = 6
```

Valid subarrays:
[4,3,2], [3,2], [2]



⭐ i = 3 → arr[3] = 3 (odd)

```makefile
odd = 2
Window = [0..3]
odd ≤ 2 → valid
Subarrays ending at i = (3 − 0 + 1) = 4
ans = 6 + 4 = 10
```

Valid subarrays:
[4,3,2,3], [3,2,3], [2,3], [3]

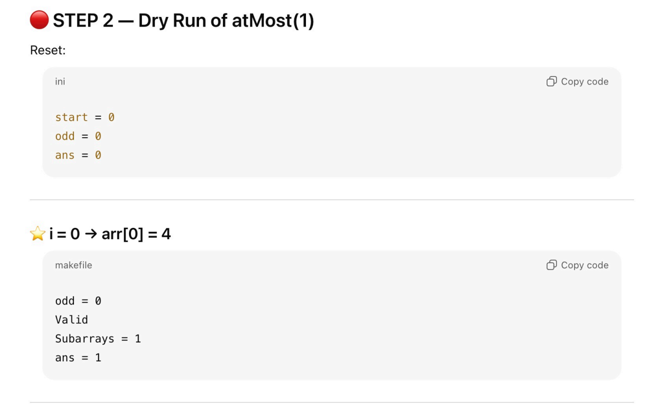
⭐ i = 4 → arr[4] = 4 (even)

```makefile
odd = 2
Window = [0..4]
Subarrays ending at i = (4 − 0 + 1) = 5
ans = 10 + 5 = 15
```

Valid subarrays:
[4,3,2,3,4], [3,2,3,4], [2,3,4], [3,4], [4]

Total : 15

atMost (2) = 15



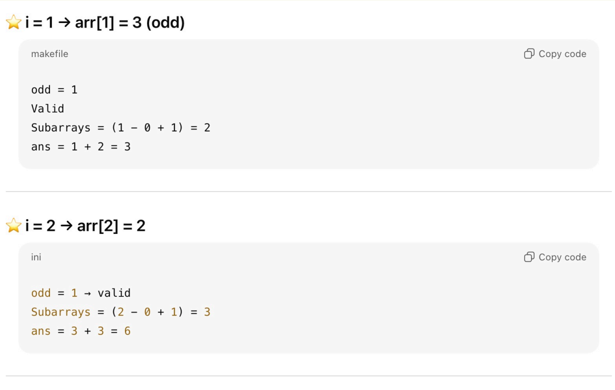
🔴 STEP 2 — Dry Run of atMost(1)

Reset:

```ini
start = 0
odd = 0
ans = 0
```

⭐ i = 0 → arr[0] = 4

```makefile
odd = 0
Valid
Subarrays = 1
ans = 1
```

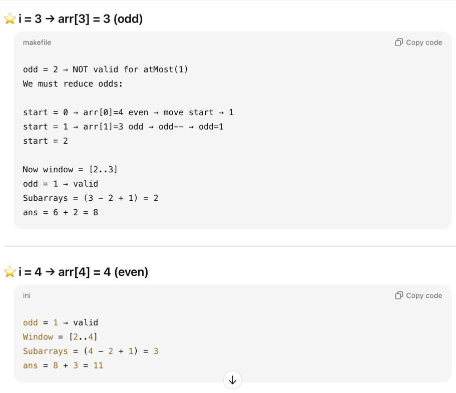


⭐ i = 1 → arr[1] = 3 (odd)

```makefile
odd = 1
Valid
Subarrays = (1 − 0 + 1) = 2
ans = 1 + 2 = 3
```

⭐ i = 2 → arr[2] = 2

```ini
odd = 1 → valid
Subarrays = (2 − 0 + 1) = 3
ans = 3 + 3 = 6
```



⭐ i = 3 → arr[3] = 3 (odd)

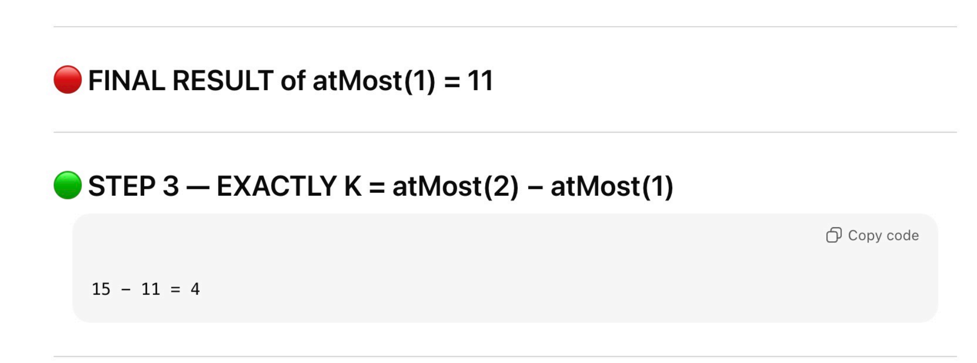
```makefile
odd = 2 → NOT valid for atMost(1)
We must reduce odds:

start = 0 → arr[0]=4 even → move start → 1
start = 1 → arr[1]=3 odd → odd−− → odd=1
start = 2

Now window = [2..3]
odd = 1 → valid
Subarrays = (3 − 2 + 1) = 2
ans = 6 + 2 = 8
```

⭐ i = 4 → arr[4] = 4 (even)

```ini
odd = 1 → valid
Window = [2..4]
Subarrays = (4 − 2 + 1) = 3
ans = 8 + 3 = 11
```



🔴 FINAL RESULT of atMost(1) = 11

🟢 STEP 3 — EXACTLY K = atMost(2) − atMost(1)

```
15 − 11 = 4
```

// Edge case

if k = 0

atMost(0) - atMost(-1)

It can give wrong result

Add condition

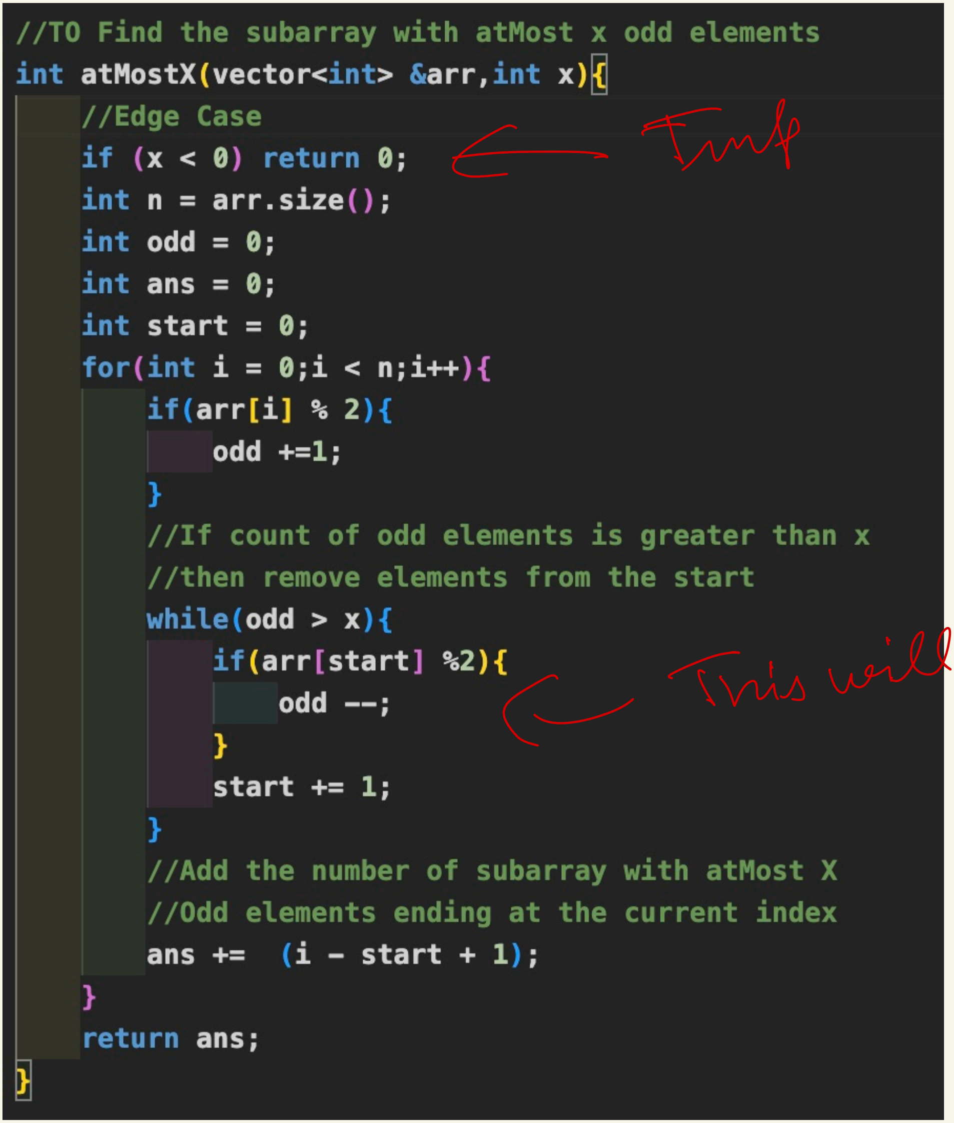
```
if (x < 0)
    return 0;
```

Test Case failed

[68, 35], 0

Run It

Final Code :



```
//TO Find the subarray with atMost x odd elements
int atMostX(vector<int> &arr,int x){
    //Edge Case
    if (x < 0) return 0;
    int n = arr.size();
    int odd = 0;
    int ans = 0;
    int start = 0;
    for(int i = 0;i < n;i++){
        if(arr[i] % 2){
            odd +=1;
        }
        //If count of odd elements is greater than x
        //then remove elements from the start
        while(odd > x){
            if(arr[start] %2){
                odd --;
            }
            start += 1;
        }
        //Add the number of subarray with atMost X
        //Odd elements ending at the current index
        ans +=  (i - start + 1);
    }
    return ans;
}
```

Imp

This will run infinite