

Count Nice Subarrays

Exactly K odd numbers — the atMost trick again ■

● MEDIUM

■ Hook

Count subarrays containing exactly **K odd numbers**. A 'nice' subarray has exactly K odds. This is the same mathematical trick as Binary Subarrays: $\text{exact}(k) = \text{atMost}(k) - \text{atMost}(k-1)$. But now instead of sum, we count odd numbers. The pattern is identical — just the 'condition' changes.

■ Core Intuition — Reusing the Pattern

Treat each number as 1 if odd, 0 if even. Now 'count odd numbers in window' = 'sum of transformed array'. This maps the problem directly to Binary Subarrays with Sum! Same code, different variable name.

```
nums = [1, 1, 2, 1, 1]    K = 3
odds = [1, 1, 0, 1, 1]    (1=odd, 0=even)

Now: count subarrays with sum(odds) = 3
    = atMost(odds, 3) - atMost(odds, 2)

Subarrays with exactly 3 odds:
[1,1,2,1], [1,2,1,1] → answer = 2
```

■ Code Implementation (Java)

```

public class CountNiceSubarrays {

    public int numberOfSubarrays(int[] nums, int k) {
        return atMost(nums, k) - atMost(nums, k - 1);
    }

    private int atMost(int[] nums, int k) {
        int left = 0, oddCount = 0, count = 0;

        for (int right = 0; right < nums.length; right++) {
            if (nums[right] % 2 == 1) oddCount++; // it's odd

            while (oddCount > k) {
                if (nums[left] % 2 == 1) oddCount--;
                left++;
            }

            count += (right - left + 1);
        }

        return count;
    }
}

```

■ Time & Space Complexity

■ TIME COMPLEXITY

$O(n)$

■ SPACE COMPLEXITY

$O(1)$

■ Patterns & Connections

The atMost subtraction pattern appears in all these problems:

- Binary Subarrays with Sum (K = sum of 0s and 1s)
- Count Nice Subarrays (K = count of odds)
- Subarrays with K Different Integers (K = distinct count)
- Any 'exactly K ' counting problem on subarrays → try atMost trick

■ Practice Problems

#	Problem	Difficulty	Pattern
1	Count Number of Nice Subarrays (LC 1248)	Medium	atMost trick
2	Binary Subarrays With Sum (LC 930)	Medium	atMost trick

3	Subarrays with K Different Integers (LC 992)	Hard	atMost trick
4	Count Subarrays Where Max Element Appears K Times	Medium	SW
5	Number of Substrings Containing All 3 Chars (LC 1358)	Medium	SW

■ Quick Revision

- ✓ $\text{exact}(k) = \text{atMost}(k) - \text{atMost}(k-1) \leftarrow$ universal counting formula
- ✓ Odd number check: $\text{num} \% 2 == 1$
- ✓ atMost counts subarrays with $\leq k$ odd numbers
- ✓ Each right contributes $(\text{right} - \text{left} + 1)$ valid windows
- ✓ Works identically to Binary Subarrays With Sum
- ✓ Time: $O(n)$ | Space: $O(1)$

■ MENTAL MODEL — Remember This Forever

Every 'exactly K' subarray counting problem has this pattern hiding inside it: $\text{atMost}(K) - \text{atMost}(K-1)$. Once you see it, you'll apply it in 30 seconds. The only thing that changes is HOW you count: sum, odds, distinct chars, etc.