题目描述

给定一个含有N个正整数的数组, 求出有多少个连续区间（包括单个正整数），它们的和大于等于x。

输入描述

第一行两个整数N x（0 < N <= 100000, 0 <= x <= 10000000）

第二行有N个正整数（每个正整数小于等于100）。

输出描述

输出一个整数，表示所求的个数。

注意：此题对效率有要求，暴力解法通过率不高，请考虑高效的实现方式。

用例

输入	3 7 3 4 7
输出	4
说明	第一行的3表示第二行数组输入3个数，第一行的7是比较数，用于判断连续数组是否大于该数；组合为 3 + 4, 3 + 4 + 7, 4 + 7, 7, 都大于等于指定的7；所以共四组。

输入	10 10000 1 2 3 4 5 6 7 8 9 10
输出	0
说明	所有元素的和小于10000，所以返回0。

题目解析

区间和，最快的计算方式就是利用：一维前缀和，关于一维前缀和请看：

因此，我们只需要计算出第二行 输入数组[1] 的前缀和，即可快速计算出任意区间范围的和，比如求解arr数组的[L,R]范围的元素之和，只需要计算 preSum[R] - preSum[L-1]即可。

本题中说区间可以是单个元素，因此preSum需要初始化为 arr.length + 1 长度，其中preSum[0] = 0，因为这样的话，才能基于preSum描述出第一个元素单独作为区间时的区间和，即preSum[1] - preSum[0]。

本题描述第二行输入是：N个正整数的数组。

即arr数组元素都是正整数，因此preSum数组必然是一个升序的数组。

这意味着，如果preSum[R] - preSum[L] >= x的话，则必然成立：preSum[i] - preSum[L] >=x ，其中 i >= R。

这样的话，如果preSum[R] - preSum[L] >= x，那么对于区间左边界固定为L的，且区间和大于等于x的区间个数就有 arr.length - R + 1 个，此时只需要O(1)时间。

下一次，我们继续枕左边界固定为L+1的情况。注意保证R>L。

由于R必须大于L，因此当R越界时，即R>arr.length时，结束。

Java算法源码

```java
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();
    int x = sc.nextInt();

    int[] arr = new int[n];
    for (int i = 0; i < n; i++) arr[i] = sc.nextInt();
    System.out.println(getResult(n, x, arr));
  }

  public static int getResult(int n, int x, int[] arr) {
    int[] preSum = new int[n + 1];

    for (int i = 1; i <= n; i++) {
      preSum[i] = preSum[i - 1] + arr[i - 1];
    }

    int l = 0;
    int r = 1;
    int ans = 0;

    while (r <= n) {
      if (preSum[r] - preSum[l] >= x) {
        ans += n - r + 1;
        l++;
        r = l + 1;
      } else {
        r++;
      }
    }

    return ans;
  }
}
```

JS算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

let lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    const [n, x] = lines[0].split(" ").map(Number);
    const arr = lines[1].split(" ").map(Number);

    console.log(getResult(n, x, arr));

    lines.length = 0;
  }
});

function getResult(n, x, arr) {
  const preSum = new Array(n + 1);

  preSum[0] = 0;
  for (let i = 1; i <= n; i++) {
    preSum[i] = preSum[i - 1] + arr[i - 1];
  }

  let l = 0;
  let r = 1;
  let ans = 0;

  while (r <= n) {
    if (preSum[r] - preSum[l] >= x) {
      ans += n - r + 1;
      l++;
      r = l + 1;
    } else {
      r++;
    }
  }

  return ans;
}
```

Python算法源码

```python
# 输入获取
n, x = map(int, input().split())
arr = list(map(int, input().split()))


# 算法入口
def getResult():
    preSum = [0]*(n+1)

    for i in range(1, n+1):
        preSum[i] = preSum[i-1] + arr[i-1]

    l = 0
    r = 1
    ans = 0

    while r <= n:
        if preSum[r] - preSum[l] >= x:
            ans += n - r + 1
            l += 1
            r = l + 1
        else:
            r += 1

    return ans


# 算法调用
print(getResult())
```