题目描述

有N个正整数组成的一个序列。给定整数sum，求长度最长的连续 子序列，使他们的和等于sum，返回此子序列的长度，

如果没有满足要求的序列，返回-1。

输入描述

第一行输入是：N个正整数组成的一个序列

第二行输入是：给定整数sum

输出描述

最长的连续子序列的长度

备注

- 输入序列仅由数字和英文逗号构成，数字之间采用英文逗号分隔
- 序列长度：1 <= N <= 200
- 输入序列不考虑异常情况

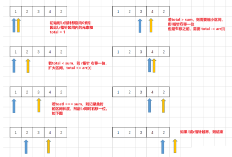
用例

输入	1,2,3,4,2 6
输出	3
说明	1,2,3和4,2两个序列均能满足要求，所以最长的连续序列为1,2,3，因此结果为3。

输入	1,2,3,4,2 20
输出	-1
说明	没有满足要求的子序列，返回-1

题目解析

本题可以利用 双指针 解决。



同时，有一种特殊情况

JavaScript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,
});

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

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

    console.log(getMaxLen(arr, sum));

    lines.length = 0;
  }
});

function getMaxLen(arr, sum) {
  let ans = -1;
  let l = 0;
  let r = 0;
  let n = arr.length;

  let total = arr[l];

  while (true) {
    if (total > sum) {
      l++;
      total -= arr[l - 1];
      if (l < n && r < l) {
        r = l;
        total += arr[r];
      }
    } else if (total < sum) {
      r++;
      if (r < n) total += arr[r];
      else break;
    } else {
      ans = Math.max(ans, r - l + 1);
      l++;
      r++;
      if (r < n) total += arr[r] - arr[l - 1];
      else break;
    }
  }

  return ans;
}
```

Java算法源码

```java
import java.util.Arrays;
import java.util.Scanner;

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

    Integer[] arr =
        Arrays.stream(sc.nextLine().split(",")).map(Integer::parseInt).toArray(Integer[]::new);

    int sum = Integer.parseInt(sc.nextLine());

    System.out.println(getResult(arr, sum));
  }

  public static int getResult(Integer[] arr, int sum) {
    int ans = -1;
    int l = 0;
    int r = 0;
    int n = arr.length;

    int total = arr[l];
    while (true) {
      if (total > sum) {
        l++;
        total -= arr[l - 1];
        if (l < n && r < l) {
          r = l;
          total += arr[r];
        }
      } else if (total < sum) {
        r++;
        if (r < n) total += arr[r];
        else break;
      } else {
        ans = Math.max(ans, r - l + 1);
        l++;
        r++;
        if (r < n) total += arr[r] - arr[l - 1];
        else break;
      }
    }

    return ans;
  }
}
```

Python算法源码

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


# 算法入口
def getResult():
    ans = -1
    l = 0
    r = 0
    n = len(arr)

    total = arr[l]

    while True:
        if total > sumV:
            l += 1
            total -= arr[l - 1]
            if r < l < n:
                r = l
                total += arr[r]
        elif total < sumV:
            r += 1
            if r < n:
                total += arr[r]
            else:
                break
        else:
            ans = max(ans, r - l + 1)
            l += 1
            r += 1
            if r < n:
                total += arr[r] - arr[l - 1]
            else:
                break

    return ans


# 算法调用
print(getResult())
```