【华为OD机考 统一考试机试C卷】分解连续正整数组合/ 分解正整数（C++ Java JavaScript Python）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

2023年11月份，华为官方已经将 华为OD机考：OD统一考试（A卷 / B卷）切换到 OD统一考试（C卷）和 OD统一考试（D卷） 。根据考友反馈：目前抽到的试卷为B卷或C卷/D卷，其中C卷居多 ，按照之前的经验C卷D卷部分考题会复用A卷/B卷题，博主正积极从考过的同学收集C卷和D卷真题，可以查看下面的真题目录。

真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境 华为OD机考B卷C卷华为OD机考华为OD机考B卷华为OD机试B卷华为OD机试C卷华为OD机考C卷华为OD机考D卷题目华为OD机考C卷/D卷答案华为OD机考C卷/D卷解析华为OD机考C卷和D卷真题华为OD机考C卷和D卷题解

题目描述

给定一个正整数 n，如果能够分解为 m（m > 1）个连续正整数之和，请输出所有分解中，m最小的分解。

如果给定整数无法分解为连续正整数，则输出字符串"N"。

输入描述

输入数据为一整数，范围为 (1, 2^30]

输出描述

比如输入为：

> 21

输出：

> 21=10+11

用例

输入	21
输出	21=10+11
说明	21可以分解的连续正整数组合的形式有多种： 21=1+2+3+4+5+6 21=6+7+8 21=10+11 其中 21=10+11，是最短的分解序列。

解题思路

解题思路：

1. 问题转化为寻找一个起始整数x和连续整数的个数m，使得x到x+m-1的和等于n。
2. 连续整数求和可以用等差数列求和公式表示：m*x* + *m*(m-1)/2 = n。
3. 通过调整m的值，尝试找到一个合适的x，使得上述等式成立。x必须是整数，即(n - m*(m-1)/2) % m == 0。
4. 由于题目要求m最小，因此从m=2开始逐渐增加m的值，直到m*(m+1)/2 > n为止。
5. 如果找到了符合条件的x和m，就构建输出字符串并返回；如果没有找到，则返回"N"。

数学知识：

- 等差数列求和公式：S = n*(a1+an)/2，其中n是项数，a1是首项，an是末项。
- 在本题中，连续整数构成等差数列，首项是x，末项是x+m-1，项数是m，因此公式变为S = m*(2*x+m-1)/2。
- 通过等式变形，可以求得x的值，进而判断是否存在符合条件的连续整数序列。

C++

```
#include <iostream>
#include <string>

```

```cpp
using namespace std;

// 寻找连续整数和的分解
string findMinConsecutiveNumbersSum(int n) {
    // 从2开始尝试每个可能的m值，m代表连续整数的个数
    for (int m = 2; m * (m + 1) / 2 <= n; ++m) {
        // 判断是否存在一个起始整数x，使得从x开始的m个连续整数之和等于n
        if ((n - m * (m - 1) / 2) % m == 0) {
            // 计算起始整数x
            int x = (n - m * (m - 1) / 2) / m;
            // 构建输出字符串
            string result = to_string(n) + "=";
            for (int i = 0; i < m; ++i) {
                result += to_string(x + i);
                // 在每个整数后面添加加号，除了最后一个整数
                if (i < m - 1) {
                    result += "+";
                }
            }
            // 返回构建好的字符串
            return result;
        }
    }
    // 如果没有找到符合条件的连续整数序列，返回"N"
    return "N";
}

int main() {
    // 读取一个整数n
    int n;
    cin >> n;

    // 调用函数并输出结果
    cout << findMinConsecutiveNumbersSum(n) << endl;

    return 0;
}
```

Java

```java


import java.util.*;

public class Main {
    public static void main(String[] args) {
        // 创建一个扫描器来读取控制台输入
        Scanner scanner = new Scanner(System.in);
        // 读取一个整数n
        int n = scanner.nextInt();
        // 关闭扫描器
        scanner.close();

        // 调用findMinConsecutiveNumbersSum方法来寻找连续整数和的分解
        String result = findMinConsecutiveNumbersSum(n);
        // 输出结果
        System.out.println(result);
    }

    private static String findMinConsecutiveNumbersSum(int n) {
        // 从2开始尝试每个可能的m值，m代表连续整数的个数
        // 当m*(m+1)/2（连续整数求和公式）大于n时，停止循环
        for (int m = 2; m * (m + 1) / 2 <= n; m++) {
            // 判断是否存在一个起始整数x，使得从x开始的m个连续整数之和等于n
            // 即找到一个x使得x+(x+1)+...+(x+m-1)=n
            // 连续整数求和公式可以转换为等差数列求和公式：m*x + m*(m-1)/2 = n
            // 解这个方程，得到x = (n - m*(m-1)/2) / m
            // 如果x是整数，说明找到了一个符合条件的连续整数序列
            if ((n - m * (m - 1) / 2) % m == 0) {
                // 计算起始整数x
                int x = (n - m * (m - 1) / 2) / m;
                // 使用StringBuilder来构建输出字符串
                StringBuilder sb = new StringBuilder();
                // 首先添加n=
                sb.append(n).append("=");
                // 然后添加连续整数序列
                for (int i = 0; i < m; i++) {
                    sb.append(x + i);
                    // 在每个整数后面添加加号，除了最后一个整数
```

```
                    if (i < m - 1) {
                        sb.append("+");
                    }
                }
                // 返回构建好的字符串
                return sb.toString();
            }
        }
        // 如果没有找到符合条件的连续整数序列，返回"N"
        return "N";
    }
}
```

javaScript

```
const readline = require('readline');

// 创建readline接口实例
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});


rl.on('line', (answer) => {
    const n = parseInt(answer);
    const result = findMinConsecutiveNumbersSum(n);
    console.log(result);
    rl.close();
});

// 寻找连续整数和的分解
function findMinConsecutiveNumbersSum(n) {
    // 从2开始尝试每个可能的m值，m代表连续整数的个数
    for (let m = 2; m * (m + 1) / 2 <= n; m++) {
        // 判断是否存在一个起始整数x，使得从x开始的m个连续整数之和等于n
        if ((n - m * (m - 1) / 2) % m === 0) {
            // 计算起始整数x
            let x = (n - m * (m - 1) / 2) / m;
            // 构建输出字符串
```

```
            let result = `${n}=`;
            for (let i = 0; i < m; i++) {
                result += (x + i).toString();
                // 在每个整数后面添加加号，除了最后一个整数
                if (i < m - 1) {
                    result += '+';
                }
            }
            // 返回构建好的字符串
            return result;
        }
    }
    // 如果没有找到符合条件的连续整数序列，返回"N"
    return 'N';
}
```

Python

```
# 寻找连续整数和的分解
def find_min_consecutive_numbers_sum(n):
    # 从2开始尝试每个可能的m值，m代表连续整数的个数
    for m in range(2, n):
        # 当m*(m+1)/2（连续整数求和公式）大于n时，停止循环
        if m * (m + 1) // 2 > n:
            break
        # 判断是否存在一个起始整数x，使得从x开始的m个连续整数之和等于n
        if (n - m * (m - 1) // 2) % m == 0:
            # 计算起始整数x
            x = (n - m * (m - 1) // 2) // m
            # 构建输出字符串
            result = f"{n}="
            result += '+'.join(str(x + i) for i in range(m))
            # 返回构建好的字符串
            return result
    # 如果没有找到符合条件的连续整数序列，返回"N"
    return "N"

# 读取一个整数n
n = int(input())

```

```
# 调用函数并输出结果
print(find_min_consecutive_numbers_sum(n))
```

文章目录