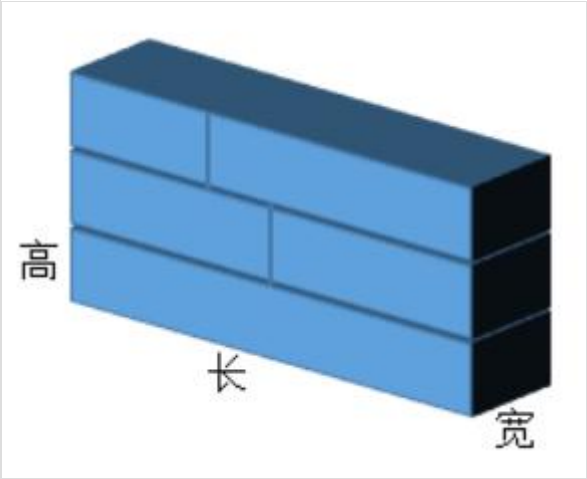
题目描述：

有一堆长方体积木，它们的宽度和高度都相同，但长度不一。小橙想把这堆积木叠成一面墙，墙的每层可以放一个积木，也可以将两个积木拼接起来，要求每层的长度相同。若必须用完这些积木，叠成的墙最多为多少层？

如下是叠成的一面墙的图示，积木仅按宽和高所在的面进行拼接。



输入描述：

输入为一行，为各个积木的长度，数字为正整数，并由空格分隔。积木的数量和长度都不超过 5000。

输出描述：

输出一个数字，为墙的最大层数，如果无法按要求叠成每层长度一致的墙，则输出 -1。

示例 1

输入：

```
3 6 6 3
```

输出：

```
3
```

说明：

可以每层都是长度 3 和 6 的积木拼接起来，这样每层的长度为 9，层数为 2；也可以其中两层直接用长度 6 的积木，两个长度 3 的积木拼接为一层，这样层数为 3，故输出 3。

示例 2

输入：

```
1 4 2 3 6
```

输出：

```
-1
```

说明：

无法用这些积木叠成每层长度一致的墙，故输出 -1。

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String[] s = in.nextLine().split(" ");
        int n = s.length, sum = 0;
        if (n == 0) {
            System.out.println(-1);
            return;
        }
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = Integer.parseInt(s[i]);
            sum += arr[i];
        }
        Arrays.sort(arr);
        f : for (int h = sum; h > 0; h--) {
            if (sum % h != 0 || arr[n - 1] > sum / h) continue;
            int l = sum / h;
            for (int i = n - 1, j = 0; i > j; i--) {
                if (arr[i] == l) continue;
                if (arr[i] + arr[j] == l) {
                    j++;
                } else {
                    continue f;
                }
            }
```

```
                System.out.println(h);
                return;
            }
            System.out.println(-1);
        }
    }
```