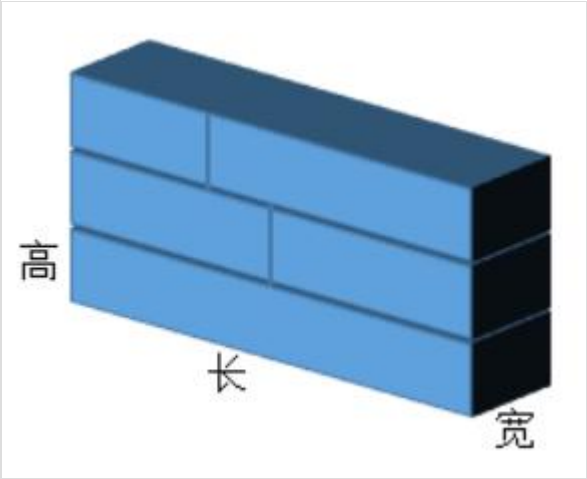
Java-题目描述：

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。

补充说明：

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.*;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        while (in.hasNextInt()) { // 注意 while 处理多个 case
            String str = in.nextLine();
            String[] str1=str.split(" ");
            List<Integer> a = new ArrayList<Integer>();
            List<Integer> b = new ArrayList<Integer>();
            int max=0;
            int min=Integer.MAX_VALUE;
            for(int i=0;i<str1.length;i++)
            {
                a.add(Integer.valueOf(str1[i]));
                b.add(Integer.valueOf(str1[i]));
                int temp =Integer.valueOf(str1[i]);
                if(max<temp)
                max=temp;
                if(min>temp)
                min=temp;
            }
            int a1=A1(a,max,min);
```

```
            int a2=A2(b,max,min);
            if(a1>a2)
            {
                  System.out.println(a1);
            }
            else{
                  System.out.println(a2);
            }



      }
}
public static int A1(List<Integer> a,int max,int min)
{
      int c=0;
      for(int i=a.size()-1;i>=0;i--)
      {
            int temp=a.get(i);
            if(temp == max)
            {
                  a.remove(i);
                  c++;
            }
      }
      while(a.size()>1)
      {
            boolean f=false;
            for(int i= 1;i<a.size();i++)
            {
                  if(a.get(0)+a.get(i)==max)
                  {
                        a.remove(i);
                        a.remove(0);
                        c++;
                        f=true;
                        break;
                  }
            }
            if(!f)
            {
                  return -1;
            }
            if(a.size()==0)
```

```
                {
                    return c;
                }
            }
            return -1;
        }
        public static int A2(List<Integer> a,int max,int min)
        {
            int l = min+max;
            int c=0;
            while(a.size()>1)
            {
                boolean f=false;
                for(int i= 1;i<a.size();i++)
                {
                    if(a.get(0)+a.get(i)==l)
                    {
                        a.remove(i);
                        a.remove(0);
                        f=true;
                        c++;
                        break;
                    }
                }
                if(!f)
                {
                    return -1;
                }
                if(a.size()==0)
                {
                    return c;
                }
            }
            return -1;
        }
}
```