L1532：赌徒 ☆☆

题目描述

有n个赌徒打算赌一局。规则是：
每人下一个赌注，赌注为非负整数，且任意两个赌注都不相同。
胜者为赌注恰好是其余任意三个人的赌注之和的那个人。
如果有多个胜者，我们取赌注最大的那个为最终胜者。
例如，A，B，C，D，E分别下赌注为2、3、5、7、12，最终胜者是E，因为12=2+3+7。

输入输出格式

输入：输入包含多组测试数据。每组首先输入一个整数n（1¡=n¡=1000），表示赌徒的个数。

接下来n行每行输入一个非负整数b（0¡=b¡32768），表示每个赌徒下的赌注。

当n=0时，输入结束。

输出：对于每组输入，输出最终胜者的赌注，如果没有胜者，则输出no solution。

输入示例	输出示例
5 2 3 5 7 12 5 2 16 64 256 1024 0	12 no solution

样例解释

1. **第一组数据：**赌注为2、3、5、7、12
   12 = 2 + 3 + 7，满足条件，所以输出12。

2. **第二组数据：**赌注为2、16、64、256、1024
   没有任何一个赌注等于其他三个赌注之和，所以输出no solution。

算法分析

1. **问题分析**
   我们需要找到一个赌注x，使得x恰好等于其他三个不同赌注的和。如果有多个这样的x，取最大的那个。

2. **高效算法思路**

   （1） **预处理两数之和：** 先计算所有可能的两个赌注之和，并用数组s记录每个和出现的次数。

   （2） **枚举胜者候选：** 对于每个赌注x，检查是否存在另外两个赌注a,b，使得x = a + b + c，其中c是第三个赌注。

   （3） **优化检查过程：** 通过数学变换，将问题转化为检查x - a - b是否存在于赌注中。

3. **关键公式推导**

   - 设我们要找的胜者赌注为x，那么需要满足：x = a + b + c
   - 可以变换为：x - a = b + c
   - 进一步变换为：x - a - b = c
   - 因此，对于每个x和a，我们需要检查是否存在b和c使得上述等式成立

AC代码实现

```
```

代码解释

1. **预处理阶段：**

   - 使用数组cnt记录每个赌注出现的次数
   - 使用数组s记录所有两数之和的出现次数

2. **查找胜者阶段：**

   - 对于每对赌注(a[i], a[j])，计算ns = a[i] - a[j]
   - 检查ns是否在预处理的两数之和数组中
   - 通过t = a[i] - 2*a[j]排除包含a[j]的无效组合
   - 更新最大满足条件的赌注

3. **数学原理：**

   - 如果a[i]是胜者，那么a[i] = a + b + c
   - 令a = a[j]，那么b + c = a[i] - a[j] = ns
   - 如果s[ns] ¿ 0，说明存在b和c使得b + c = ns
   - 需要排除b或c等于a[j]的情况

算法复杂度分析

- **时间复杂度：** $O(n^2)$ - 两重循环预处理加上两重循环检查
- **空间复杂度：** $O(MAX)$ - 使用固定大小的数组存储两数之和
- **优势：** 相比直接三重循环的$O(n^3)$方法，效率显著提升

关键技巧

- **空间换时间：** 使用大数组预存储所有可能的两数之和
- **数学变换：** 将三数之和问题转化为两数之和问题
- **边界处理：** 仔细处理各种特殊情况，确保算法正确性

拓展思考

- 如果赌注可以相同，算法需要如何修改？
- 如果要求找出所有满足条件的胜者，而不仅仅是最大的，该如何实现？
- 如果赌注范围更大，如何优化空间复杂度？