【题意】：乔治拿来一组等长的木棒，将它们随机地砍断，使得每一节木棍的长度都不超过50个长度单位。然后他又想把这些木棍恢复到为裁截前的状态，但忘记了初始时有多少木棒以及木棒的初始长度。请你设计一个程序，帮助乔治计算木棒的可能最小长度。每一节木棍的长度都用大于零的整数表示。  
  
【随笔】：这道题很早之前就过了，这几天看回之前的代码，觉得以前的代码写得太烂了，而且那个代码风格也不好看，于是决定再重写一边，顺便把uva的也过了。这次重写比之前的代码加多了一个重要的剪枝，最后在uva上以0.2s过了。  
  
【题解】：不得不说，这道题出得非常好，特别是uva那里的大数据（poj的数据太水了），对于剪枝能力要求很高。下面说下几个重要的剪枝：  
1.把所有木棍的长度从大到小排列，组合木棒时优先使用长的木棍，这样可以加快组合速度，并且对后面的剪枝有帮助。  
2.木棒的长度一定是大于等于最长木棍的长度并且小于等于所有木棍长度的和，这个很容易证明。  
3.木棒的长度一定是所有木棍长度的和的约数，这个也很容易证明。  
4.在某一个木棒的组合过程中，对于当前的木棍stick[i]，如果stick[i-1]没有被组合并且stick[i] == stick[i-1]，那么不用考虑stick[i]，显然stick[i]最终也不会被组合。  
5.如果此次是在尝试第i个木棒的第一段，假设stick[j]为当前可以被使用的最长的木棍，如果此次组合失败，直接退出搜索，即退回到对第i-1个木棒的搜索。试想：失败说明现在使用stick[j]是不可行的，那么以后无论什么时候使用stick[j]都是不可行的，因为以后再处理stick[j]时可使用的木棍一定是当前可使用的木棍的子集，在更多木棍选择的情况下都不能组合成功，那么，在更少木棍选择的情况下一定不能组合成功。  
  
【代码】：

 1 #include "iostream"  
 2 #include "cstdio"  
 3 #include "cstring"  
 4 #include "algorithm"  
 5 #include "functional"  
 6 using namespace std;  
 7 #define maxn 65  
 8 int n, sum, goal;  
 9 int stick[maxn];  
10 bool visit[maxn];  
11   
12 bool cmp(const int &a, const int &b) {  
13     return a > b;  
14 }  
15   
16 bool dfs(int now, int index, int cnt) {  
17     if(goal \* cnt == sum) return true;  
18     for(int i = index; i < n; i++) {  
19         if(visit[i] || (i && !visit[i-1] && stick[i] == stick[i-1])) continue;  
20         if(now + stick[i] == goal) {  
21             visit[i] = true;  
22             if(dfs(0, 0, cnt + 1)) return true;  
23             visit[i] = false;  
24             return false;  
25         } else if(now + stick[i] < goal) {  
26             visit[i] = true;  
27             if(dfs(now + stick[i], i + 1, cnt)) return true;  
28             visit[i] = false;  
29             if(now == 0) return false;  
30         }  
31     }  
32     return false;  
33 }  
34   
35 int solve() {  
36     sort(stick, stick + n, cmp);  
37     for(goal = stick[0]; goal < sum; goal++) {  
38         if(sum % goal != 0) continue;  
39         memset(visit, false, sizeof(visit));  
40         if(dfs(0, 0, 0)) break;  
41     }  
42     return goal;  
43 }  
44   
45 int main() {  
46     while(~scanf("%d", &n)) {  
47         if(!n) break;  
48         sum = 0;  
49         for(int i = 0; i < n; i++) {  
50             scanf("%d", &stick[i]);  
51             sum += stick[i];  
52         }  
53         printf("%d\n", solve());  
54     }  
55     return 0;  
56 }