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。

• 最多64个木棍

**剪枝**

看到《短码之美》上有，那就直接看书吧。

如果是简单地输入，即使做了一些无谓的搜索，也能够求出结果。但是，由于切出来的段数最多有64根，单纯计算所有组合就需要很长时间。这次用一个稍微复杂一点的例子，首先手动求组合看看。

首先，准备下面这样的测试数据。

8 3 8 15 2 11 4 8 1

这9个数值的合计（sum)是60，最大值（max)是15，60的约数中比15大的有15、20、 30、60(=length)这4个，检查切出来的全部片段是不是能够拼出这些长度的棒子。

首先考虑length=15的情况。

将这9个数字降序排列：

15 11 8 8 8 4 3 2 1

从这个清单的开头开始，分割成合计为15的部分。

15

11，4

8,…

15保持原状，下一个11+4=15,再下一个8加上8就是16 了，超出了 15。即使依次加上3、 2、1还是等于14,凑不上15。所以知道原本棒子的长度不是15。那么下面这组又怎么样呢？

15

11，3, 1

8,…

同前面那组一样无法凑成15,似乎原本棒子的长度不是15。 20能行吗？

15, 4, 1

11，8,…

最初选择{15, 4, 1}就不能拼出第2根棒子了，但是这就说20不行有些太早了。

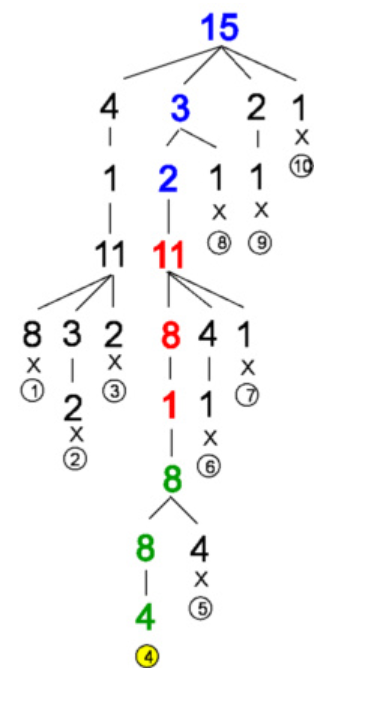
15,3,2

11，8, 1

8, 8, 4

如果最开始的一组用{15, 3, 2}，就能顺利凑齐长度是20的棒子。接下来把这样的搜索方法用程序代码来实现就可以了。怎样做才好呢？

前面是手动来检査搜索方法，现在可以通过DFS算法来实现。DFS会考虑将输入值（片段的长度）作为顶点，从最幵始的节点开始顺序地建立树形结构（清单）。用图表示就像下面这样。



在编写程序之前，先整理一下重点。

1、将数椐按降序排序。

2、从数值最大的开始（=从数组开头幵始），依次减去K度值，直到棒子的长度（length) 变成0为止，同时标记用到的数值。

3、如果长度变成0,则把length恢复原值，再执行②。

4、如果失败，则返回到前一个状态。

5、如果做出sum/length个长度是length的棒子就成功了。

#include <iostream>

#include<stdio.h>

#include<cstring>

using namespace std;

int in[51]; //长度为i的片段个数

int length; //棒子的原始长度

int finish; //终止标志（非0即终止）

/\*\*

\* 遍历

\* @param count 棒子的总数（如果为0，则结束）

\* @param len 目前要凑的棒子长度的剩余量，就是还剩多少长度就能满足凑出一根棒子来了

\* @param plen 现在检查的片段长度

\*/

void check(int count, int len, int plen)//check(sum / length, length, max);

{

if(count == 0)

{

finish=1;

return;

}

in[plen]--; //取一个长度是plen的片段来用

if (finish==0)

{

len-=plen;

if(len!=0)

{

int next\_plen = min(len,plen); //剪枝策略：用plen去凑len，next\_plen必须是较小者

for( ;next\_plen>0;--next\_plen)

if(in[next\_plen]>0) //有剩余的这个长度的棒子片段

check(count,len,next\_plen);

}

else//如果len为0，也就是说已经凑出一根来了

{

//片段长度最大为50

int max = 50;

while (in[max]==0)

--max;//找当前剩余地最大片段

check(count - 1, length, max);//已经凑出一根来了，凑下一根

}

}

++in[plen]; //恢复原始状态

}

int main()

{

int n;

while (cin>>n&& n)

{

memset(in, 0, sizeof(in));

int sum = 0;

int max = 0;

finish = 0;

while (n--)

{

int b;

scanf("%d", &b);

++in[b];

sum += b;

if (max < b)

max = b;

}

length = max;

while (true)

{

if (sum % length == 0)

check(sum / length, length, max);

if (finish>0)

break;

++length;

}

cout<<length<<endl;

}

}