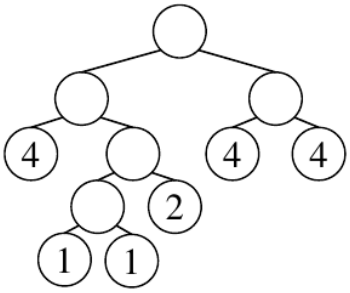
题意

二叉树上每个叶子结点都赋有一个整数作为权值。如果每个非叶子结点的左右子树的权值之和相同，那么这棵树就认为是平衡的，下图就是一个例子：



如果一个平衡树从左到右把所有叶子结点的权值全部列出来形成的序列是序列 A 的子序列(不一定连续)，那么我们就说这棵树隐藏在 A 中。上图的树就隐藏在序列 3 4 1 3 1 2 4 4 6 中，因为 4 1 1 2 4 4 是其子序列。

给出一个长度为 N(1≤N≤1000)的整数序列 $A_1A_2...A_N$(1≤A_i≤500)，找出隐藏在其中的叶子个数最多的平衡树并输出其叶子个数。上图中的树就是上述序列中隐藏的叶子个数最多的平衡树。

题解

我们首先探究答案可能有的性质。可以发现答案树的权值存在一个或多个最小值，记为 base，且答案树的其他权值都是 base 的 2^k 倍。提供一个直观理解如下：

1. 假设 base 所在子树有 k 层，那么易得权和 = base * 2^{k-1}。这对于答案树也成立，所以答案树里的点权都是 base 的倍数。
2. 假设有些点权不是 base 的 2^k 倍，且它们已经合成了另一棵子树，其最小权值为 base1。由上面计算子树权和的公式，和 base1 / base 不是 2 的幂，可知这棵子树一定不能和 base 所在的子树合并。

于是对于每一个可以成为 base 的 A[i]（即没有 A[j]使 A[i] / A[j] = 2^k，k>0 的 A[i]），我们在 A 里依次取出其他权值，把它们和 base 的比值存到另一个容器 B 里。接下来只需要对 B 考虑合并问题，并更新最终答案。

我们定义 dp[i,j]为只用 B 中前 i 个元素，且强制使用 B[i]，所得树权值和为 j 的情况下的最大叶子个数。为叙述方便，定义 B 的前缀和数组为 sum。由上面计算子树权和的公式可知，对 B 而言答案树权值和为 2 的幂，故答案是：

$$\max(dp[B.size(),2^x]),\ 2^x <= sum[B.size()],\ x >= 0$$

我们可以采用打表和刷表两种方法。状态转移方程：

$$dp[i,j] = \max(dp[i,j],dp[i-1,j-B[i]]+1),\ j-B[i]\text{是 }B[i]\text{的倍数},\ x >= 0$$

记得还有一种方案是只选 B[i]：

$$dp[i,B[i]] = \max(dp[i,B[i]],1)$$

j 的枚举次数是复杂度的瓶颈，它的一个上界是 sum[i]，且只需枚举 B[i]的倍数，因此我们实现时采用的上界是 sum[i] / B[i] * B[i]。

由于只需要用到 i - 1 的状态，我们可以借鉴 01 背包进行滚动优化。我们也发现这个算法和 01 背包的算法有比较多的相似性。

因为权值小于等于 500，所以在 B 中每个元素最大为 256。共有 1000 个元素，考虑极端情况是 512 个 256 合并，所以有效状态数不超过 $256 * 512 = 2^{17}$，其实远远小于此上界。但 dp 数组需要的空间大一些，256 * 1000。

总结

上面对答案特性的探讨，使得以上 dp 过程、从 dp 寻找答案的过程由不可行变为可行。以上 dp 和 01 背包 dp 有一定相似性，比如，权值和和叶子数分别对应重量和价值。

时空复杂度

时间复杂度：$O(N * 2^{17})$
空间复杂度：$O(2^{17})$

代码

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <string>
#include <ctime>
#include <vector>
#include <cmath>
#include <queue>
#include <map>
#include <set>
#include <stack>
#include <algorithm>
//#include <bits/stdc++.h>
using namespace std;

typedef long long LL;

const int SZ = 1005,SZ0 = 256005;

int n,a[SZ],dp[SZ0];
bool bas[SZ];bool vis[SZ];
int sl,B[SZ],sum[SZ];

template<typename Type>inline void read(Type &xx){
    Type f = 1;char ch;xx = 0;
    for(ch = getchar();ch < '0' || ch > '9';ch = getchar()) if(ch == '-') f =
-1;
    for(;ch >= '0' && ch <= '9';ch = getchar()) xx = xx * 10 + ch - '0';
    xx *= f;
}
```

```
int DP(){
    for(int i = 1;i <= sl;i++) sum[i] = sum[i - 1] + B[i];
    dp[0] = 0;
    for(int i = 1;i <= sl;i++){
        int mn = B[i];
        for(int j = sum[i] / mn * mn;j >= (mn << 1);j -= mn)
            dp[j] = max(dp[j],dp[j - mn] + 1);
        dp[mn] = max(dp[mn],1);
    }
    int ans = 0;
    for(int i = 1;i <= sum[sl];i <<= 1) ans = max(ans,dp[i]);
    memset(dp,0x8f,sizeof dp);
    return ans;
}

bool is_least(int idx){
    if(vis[a[idx]]) return false;
    vis[a[idx]] = true;
    for(int j = 1;j <= n;++j)
        if(a[idx] % a[j] == 0 && bas[a[idx] / a[j]] && a[idx] > a[j])
            return false;
    return true;
}

int main(int argc, char** argv) {
    for(int i = 1;i < SZ;i <<= 1) bas[i] = true;
    while(scanf("%d",&n) != EOF && n){
        memset(dp,0x8f,sizeof dp);memset(vis,0,sizeof vis);
        for(int i = 1;i <= n;i++) read(a[i]);
        int ans = 0;
        for(int i = 1;i <= n;++i){
            if(!is_least(i)) continue;
            sl = 0;
            for(int j = 1;j <= n;++j)
                if(a[j] % a[i] == 0 && bas[a[j] / a[i]])
                    B[++sl] = a[j] / a[i];
            ans = max(ans,DP());
        }
        printf("%d\n",ans);
    }
    return 0;
}
```