贪心算法：1125 Chain the Ropes (25分)

这道题我愚蠢地用了二叉堆去做，然后一看柳神的，原来贪心就可以……

下面给出贪心算法正确的证明：证明最短的两条绳子对半融合后仍然是最小的一段绳子

1. 前提：绳子有4段以上
2. 给出前提的理由：绳子端数小于4时，例如3条:a, b, c，ab融合成d，不论$d > c$ or $d < c$,都只剩两条绳子，只能互相结合了。
3. 不妨假设绳子有四条，分别为$a, b, c, d,\ a < b < c < d$

$$\begin{aligned} e &= (a+b)/2 \\ \because a &< c, b < c \\ \therefore a &< d, b < d \\ \therefore a + b &< 2d \\ \therefore e &< d \\ \because a &< c, b < c \\ \therefore a + b &< 2c \\ \therefore e &< c \\ \therefore e &< c < d \end{aligned}$$

可得出结论，最小的两个绳子结合后一定也是最小的一段绳子。证明比较简单。

例题：

Given some segments of rope, you are supposed to chain them into one rope. Each time you may only fold two segments into loops and chain them into one piece, as shown by the figure. The resulting chain will be treated as another segment of rope and can be folded again. After each chaining, the lengths of the original two segments will be halved.

Your job is to make the longest possible rope out of *N* given segments.

Input Specification:

Each input file contains one test case. For each case, the first line gives a positive integer *N* (2≤*N*≤104). Then *N* positive integer lengths of the segments are given in the next line, separated by spaces. All the integers are no more than 104.

Output Specification:

For each case, print in a line the length of the longest possible rope that can be made by the given segments. The result must be rounded to the nearest integer that is no greater than the maximum length.

Sample Input:

```
8
10 15 12 3 4 13 1 15
```

Sample Output:

```
14
```

我的代码注释部分是二叉堆写法，有兴趣可以看看QAQ

```
#include <cstdio>
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
//vector<double> stack(1);
/*void insert(double len)
{
    stack.push_back(len);
    for (int i = stack.size() - 1; i >= 0; i /= 2)
        if (stack[i / 2] > stack[i]) swap(stack[i / 2], stack[i]);
        else break;
}
int pop()
{
    double key = stack[1];
    int parent = 1, child;
    stack[1] = *(stack.end() - 1);
    stack.pop_back();
    for(parent=1;parent*2<=stack.size();parent=child)
    {
        child = parent * 2;
        if(child>=stack.size()) break;
        if (child+1<stack.size() && stack[child + 1] < stack[child]) child++;
        if(stack[parent]<=stack[child]) break;
        swap(stack[parent], stack[child]);
    }
    return key;
}*/
int main()
{
    //stack[0] = -1;
    int number;
    cin >> number;
    int ans;
    vector<int> a;
    a.resize(number);
    for (int i = 0; i < number; i++) cin >> a[i];
    ans = a[0];
    sort(a.begin(), a.end());
    for (int i = 1; i < a.size(); i++)
        ans = (ans + a[i]) / 2;
    /*for(int i=0;i<number;i++)
    {
        int len;
        cin >> len;
        insert(len);
    }
    while (stack.size()>1)
    {
        if (stack.size() == 2)
            ans = pop();
        else
        {
            double a = pop(), b = pop();
            insert((a + b) / 2);
        }
    }*/
    printf("%d", ans);
    return 0;
}
```