P1090 合并果子

题目描述

在一个果园里，多多已经将所有的果子打了下来，而且按果子的不同种类分成了不同的堆。多多决定把所有的果子合成一堆。

每一次合并，多多可以把两堆果子合并到一起，消耗的体力等于两堆果子的重量之和。可以看出，所有的果子经过 n-1 次合并之后， 就只剩下一堆了。多多在合并果子时总共消耗的体力等于每次合并所耗体力之和。

因为还要花大力气把这些果子搬回家，所以多多在合并果子时要尽可能地节省体力。假定每个果子重量都为 1 ，并且已知果子的种类 数和每种果子的数目，你的任务是设计出合并的次序方案，使多多耗费的体力最少，并输出这个最小的体力耗费值。

例如有 3 种果子，数目依次为 1 ， 2 ， 9 。可以先将 1 、 2 堆合并，新堆数目为 3 ，耗费体力为 3 。接着，将新堆与原先的第三堆合并，又得到新的堆，数目为 12 ，耗费体力为 12 。所以多多总共耗费体力 =3+12=15 。可以证明 15为最小的体力耗费值。

输入格式

共两行。

第一行是一个整数 n(1≤n≤10000) ，表示果子的种类数。

第二行包含n个整数，用空格分隔，第i个整数 ai(1≤ai≤20000) 是第i种果子的数目。

输出格式

一个整数，也就是最小的体力耗费值。输入数据保证这个值小于 2^{31}。

输入样例

```
3
1 2 9
```

输出样例

```
15
```

逻辑航线 微信号：luojihangxian

解析

我们先来对比一下不同的组合方式。首先是题目中的组合：

1和2合并，总体力消耗为3

再和3合并，总体力消耗为3+12 = 15

再来尝试一下1和3先合并：

总体力消耗为10

再和第2堆合并，总体力消耗为10+12=22

很明显，这种合并方式不是最优解。

通过分析题意，我们发现，越是提前合并的果子数之和，越会被多次累加。

比如方案1中，第一堆和第二堆果子之和就被累加了两次，由此我们大胆推测：是不是数量越小的和就应该越先加呢？

如何证明呢？在上面，其实我们已经使用了反证法。先用两个较大的数字进行合并，发现它们的结果并不是最优的。因此，贪心策略成立。

此外，本题还有另外一个问题，就是如何选出两个数字的最小和？

在这里我们使用双数组来进行实现，原理如下：

1、初始化双数组，第一个数组存储排序后的原始数据，第二个数组全部存储无穷大的数据。

原始数据

0	1	2	3
5	10	13	14

无穷大数据

0	1	2	3
max	max	max	max

累计体力值：0

逻辑航线 微信号：luojihangxian

2、因为原始数组是排序过的，所以，前两个数据的和必然是最小的。我们将其计算完毕后，存入无穷大数组中的第一位。如下所示，我用红色表示这两个数据不再使用：

第一次合并

原始数据

0	1	2	3
5	10	13	14

无穷大数据

0	1	2	3
15	max	max	max

累计体力值：15

3、现在，我需要在三个绿色数据中找到两个加和最小的数字，很明显是13和14（分别在两个数组中找到两个数值最小的数字进行加和），结果为27，同样的，我们需要将它存入无穷大数组。

第二次合并

原始数据

0	1	2	3
5	10	13	14

无穷大数据

0	1	2	3
15	27	max	max

累计体力值：15+27 = 42

4、最后，我们再来合并剩下的两个数据。

第三次合并

原始数据

0	1	2	3
5	10	13	14

无穷大数据

0	1	2	3
15	27	42	max

累计体力值：42+42 = 84

需要注意的是，我们应该使用一个循环来控制合并计算，n个数字总共进行n-1次循环。

编码

```
#include <bits/stdc++.h>

using namespace std;
const int MaxN = 0x3f3f3f3f;
int n; //果子种类
int nums[10001]; //存储果子的数量
int sums[10001]; //每次合并后的体力消耗
```

```
int sum = 0; //总的体力消耗
int rIndex; //合并后体力消耗的索引

int main() {
    cin >> n;
    //因为后面需要比较最小值
    //所以最初时刻应该把数组设置为最大值
    memset(nums, MaxN, sizeof(nums));
    memset(sums, MaxN, sizeof(sums));
    //循环读入果子的数量
    for (int i = 0; i < n; ++i) {
        cin >> nums[i];
    }
    //按照果子的数量，进行升序排列
    sort(nums, nums + n);
    //定义i，j双指针，用来寻找最小值
    int i = 0, j = 0;
    //当前两个数字的和
    int w;
    //总共n个数字，所以需要合并n-1次
    for (int k = 1; k < n; k++) {
        //使用双指针找到最小的两个数字进行组合
        //选择第一个加数
        int plusA = nums[i] < sums[j] ? nums[i++] : sums[j++];
        //选择第二个加数
        int plusB = nums[i] < sums[j] ? nums[i++] : sums[j++];
        //将加和放到第二数组中
        sums[rIndex++] = plusA + plusB;
        //累加总的体力消耗
        sum += plusA + plusB;
    }
    cout << sum;
    return 0;
}
```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。

，
，
，

逻辑航线 微信号：luojihangxian