合并石子

题目描述

设有N堆石子排成一排，其编号为1，2，3，…，N。

每堆石子有一定的质量，可以用一个整数来描述，现在要将这N堆石子合并成为一堆。

每次只能合并相邻的两堆，合并的代价为这两堆石子的质量之和，合并后与这两堆石子相邻的石子将和新堆相邻，合并时由于选择的顺序不同，合并的总代价也不相同。

例如有4堆石子分别为 1 3 5 2， 我们可以先合并1、2堆，代价为4，得到4 5 2， 又合并 1，2堆，代价为9，得到9 2 ，再合并得到11，总代价为4+9+11=24；

如果第二步是先合并2，3堆，则代价为7，得到4 7，最后一次合并代价为11，总代价为4+7+11=22。

问题是：找出一种合理的方法，使总的代价最小，输出最小代价。

输入

第一行为一个正整数N (2≤N≤100)；

以下N行,每行一个正整数，小于10000，分别表示第i堆石子的个数(1≤i≤N)。

输出

一个正整数，即最小得分。

输入样例

7
13
7
8
16
21
4
18

输出样例

239

解析

我们先来搞懂这道题。很多人一上来都会有这样的疑问，合并得分，那是不是把所有的石子加在一起就好了？答案肯定是否定的。

我们先来看一组简单的数字：1,2,3。

首先，我们合并1,2，此时得到了新的石堆，它的数量为3，当前的得分也为3。然后再来合并剩下的两个石堆，那么新的石堆数量即为6，两次得出的石堆数分别是他们的得分，因此当前总得分为9。

接着，我们变更一下顺序，先来合并后面的两堆2和3，这样操作后的结果将产生一个新的石堆，数量为5，因此，此刻的分数也为5。最后，我们合并剩余两个石堆，新石堆的数量依然为6，但是总得分却是11。很明显，第一种方法更为优秀。

由此，我们明白，在合并的过程中应该尽可能的选择和最小的石堆进行优先合并。那么，如何才能找到这个最小的石堆呢？

根据示例，我们画出待求和的数组

索引	1	2	3	4	5	6	7
数值	13	7	8	16	21	4	18

现在，思考这样一个问题，如果我想要求整个数组的最小值，那么我能不能把数组规模缩小，假设我们目前只求前两个数值的最小和，很明显，答案就是20。

索引	1	2	3	4	5	6	7
数值	13	7	8	16	21	4	18
最值	20						

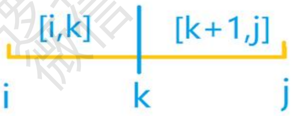
接下来，我们继续扩大规模，来求前三个数的最小和。
我们先将三个数字用不同的颜色分成两组：
首先是13自己一组，7和8一组，则结果为：0+(7+8)+(13+7+8)=43.
这里需要解释一下：
1、因为13自己一组，所以它的合并并不存在代价，因此代价为0.
2、7和8的最小代价就是他们的和，因此为7+8
3、通过对题意的理解，我们明白，求一个范围的最小值，实际上就是要想办法把这个范围拆成两部分，用左边最小代价加上右边最小代价，再加上最后合并动作的最小代价。最后的这个最小合并代价其实就是整个范围的和，无论你的运算顺序如何，都不会对它产生影响。

索引	1	2	3	4	5	6	7
数值	13	7	8	16	21	4	18
最值	43						

继续扩张蓝色范围。现在，我们让13和7一组，8自己一组，根据上面的分析，可以得出，结果为：
(13+7)+0+(13+7+8) = 48

索引	1	2	3	4	5	6	7
数值	13	7	8	16	21	4	18
最值	48						

很明显，前三个石子的最小代价为43。
通过上面这个例子，我们可以发现一个规律,不管集合是如何划分的,最后一定会剩下2堆,然后把这两堆合并成一堆.所以我们可以以最后一次合并的分界线的位置来进行集合的分类，分别尝试不同情况下的最小值。如下图所示,其中i,j是所要求的石子索引范围：



k就是最后两堆石子合
并成一堆石子的分界线

按照这个思路，我们来求四个数的最小代价。同样，我们先来拆分范围。此时，数据被拆分成了独立的13，以及7,8,16三个数字。那么此时的最小代价就是：
0 + 三个数的最小代价 + (13+7+8+16)
其中三个数的最小代价的具体算法，可以参考上文的三个数，在这里我直接给出答案为46.则当前最小值为：0+46+44=90

索引	1	2	3	4	5	6	7
数值	13	7	8	16	21	4	18
最值	90						

继续调整蓝色范围，最小值为：(13+7)+(8+16)+(13+7+8+16)=88

索引	1	2	3	4	5	6	7
数值	13	7	8	16	21	4	18
最值	88						

继续调整，最小值为：前三个数的最小值+0+(13+7+8+16)
其中，前三个数的最小值我们恰好在上文中求过，结果为43，带入得到结果：
43+0+44=87,这就是前四个数的最小代价。

索引	1	2	3	4	5	6	7
数值	13	7	8	16	21	4	18
最值	87						

因为，我们在合并的过程中，右边必须存在一堆，因此，蓝色的扩展就到此为止。
以上，就是合并石子求最小代价的基本流程。
到这里，很多同学可能还有很多疑问。
首先，每次都去计算整个范围的和（上文的红色部分）似乎十分的不妥，有没有办法解决呢？答案是肯定的，这就是前缀和。

什么是前缀和？前缀和是一个数组的某项下标之前(包括此项元素)的所有数组元素的和。对于当前数据，则有如下前缀和：

索引	1	2	3	4	5	6	7
数值	13	7	8	16	21	4	18
和	13	20	28	44	65	69	87

有了前缀和，我们便可以很轻松的求出指定范围的数据和，例如我们想求3-6之前的范围和，那么它就等于：69(1-6的和)-20(1-2的和)=8+16+21+4=49，是不是非常的方便快捷？

其次，就是7,8,16的最小值，难道我们每次都是现用现算吗？实际上，上面的文章只是为了将流程给大家讲解清楚，因而忽略了很多步骤。在真正的编码时，我们还需要做一系列的初始化工作。其中就包括求7,8,16的最小代价。
具体做法就如同文章最初的那样，我们会从范围为2的数字开始，即计算1-2,2-3,3-4等等两个数字的最小代价。然后，我们将增加长度，开始求解1-3,2-4,3-5等等范围的最小代价。以此类推，直到求出整个范围的最小代价。

编码

```
#include<iostream>


using namespace std;


const int N = 1010;
int s[N]; // 前缀和

int f[N][N];

int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> s[i];
    }

    // 计算前缀和
    for (int i = 1; i <= n; i++) {
        s[i] += s[i - 1];
    }
```

```cpp
    // 枚举所有状态
    // 长度从小到大来枚举所有状态
    // 区间长度为1时合并不要代价,所以区间长度从2开始
    for (int len = 2; len <= n; len++)
        // 枚举完长度枚举一下起点
        for (int i = 1; i <= n - len + 1; i++) {

            int l = i, r = i + len - 1;
            // 因为是取min值,所以先将f[l][r]置为无穷
            f[l][r] = 0x3f3f3f3f;
            // 枚举一下分界点,构造状态转移方程
            for (int k = l; k < r; k++) // k从l到r-1
            {
                int value = f[l][k] + f[k + 1][r] + s[r] - s[l - 1];
                f[l][r] = min(f[l][r], value);
            }
        }
    cout << f[1][n] << endl;
    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。