**哈夫曼树**

**1544 合并果子**

Time Limit: 1000 ms

Memory Limit: 256 mb

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。

**输入输出格式**

**输入描述:**

输入包括两行，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。

**输出描述:**

输出包括一行，这一行只包含一个整数，也就是最小的体力耗费值。输入数据保证这个值小于2^31。

**输入输出样例**

**输入样例#:**

3

1 2 9

**输出样例#:**

15

**题目来源**

**中南大学2016年机试题**

**1382 哈夫曼树**

Time Limit: 1000 ms

Memory Limit: 256 mb

哈夫曼树，第一行输入一个数n，表示叶结点的个数。需要用这些叶结点生成哈夫曼树，根据哈夫曼树的概念，这些结点有权值，即weight，题目需要输出所有结点的值与权值的乘积之和。

**输入输出格式**

**输入描述:**

输入有多组数据。每组第一行输入一个数n，接着输入n个叶节点（叶节点权值不超过100，2<=n<=1000）。

**输出描述:**

输出权值。

**输入输出样例**

**输入样例#:**

5

1 2 2 5 9

**输出样例#:**

37

**题目来源**

**北京邮电大学/兰州大学2019年机试**

**1562 哈夫曼编码**

Time Limit: 1000 ms

Memory Limit: 256 mb

熵编码器是一种数据编码方法，它通过对删除了“浪费”或“多余”信息的消息进行编码来实现无损数据压缩。换句话说，熵编码首先删除了对消息进行精确编码所不需要的信息。高度的熵意味着一条带有大量浪费信息的消息。以ASCII编码的英文文本是具有非常高的熵的消息类型的示例。已经压缩的消息（例如JPEG图形或ZIP归档文件）的熵很小，因此无法从进一步的熵编码尝试中受益。

用ASCII编码的英文文本具有很高的熵，因为所有字符都使用相同的位数（八位）进行编码。众所周知的事实是，与英语文本中大多数其他字母相比，字母E，L，N，R，S和T的出现频率要高得多。如果可以找到一种仅用4位对这些字母进行编码的方法，则新的编码将更小，包含所有原始信息并且熵更少。ASCII会使用固定位数，这是有原因的：这很容易，因为一个人总是处理固定位数来表示每个可能的字形或字符。对于上述字母使用四位的编码方案如何区分四位代码和八位代码？使用所谓的“无前缀可变长度”编码可以解决这个看似困难的问题。

在这种编码中，可以使用任意数量的位来表示任何字形，并且消息中不存在的字形也不会被简单地编码。但是，为了能够恢复信息，不允许将编码字形的位模式作为任何其他编码位模式的前缀。这允许对编码的比特流进行逐位读取，并且只要遇到代表字形的一组位，就可以对该字形进行解码。如果没有实施无前缀约束，那么这样的解码将是不可能的。

考虑文字“ AAAAABCD”。使用ASCII，对此进行编码将需要64位。相反，如果我们用位模式“ 00”编码“ A”，用“ 01”编码“ B”，用“ 10”编码“ C”，用“ 11”编码“ D”，那么我们只能用16位编码该文本。位;结果位模式将为“ 0000000000011011”。但是，这仍然是固定长度的编码。每个字形使用的是两位，而不是八位。由于字形“ A”以更高的频率出现，我们可以通过用更少的比特对其进行编码来做得更好吗？实际上我们可以，但是为了保持无前缀编码，其他一些位模式将变得比两位长。最佳编码是用“ 0”编码“ A”，用“ 10”编码“ B”，用“ 110”编码“ C”，用“ 111”编码“ D”。（这显然不是唯一的最佳编码，因为很明显，对于B，C和D的编码可以在不增加最终编码消息大小的情况下自由地互换用于任何给定的编码。）使用这种编码，消息可以在“ 0000010110111”只有13位，压缩比为4.9:1（也就是说，最终编码消息中的每个位代表的信息与原始编码中4.9位的信息一样多）。从左到右通读此位模式，您会发现，即使代码的位长不同，无前缀编码也可以很容易地将其解码为原始文本。

作为第二个示例，请考虑文字“帽子里的猫”。在本文中，字母“ T”和空格字符均以最高频率出现，因此在最佳编码中，它们显然具有最短的编码位模式。字母“ C”，“ I”和“ N”仅出现一次，因此它们的编码最长。

有许多可能的无前缀可变长度位模式集，它们可以产生最佳编码，也就是说，允许文本以最少的位数进行编码。一种这样的最佳编码是使用“ 00”，“ A”和“ 100”，“ C”和“ 1110”，“ E”和“ 1111”，“ H”和“ 110”，“ I”和“ 1010”，“ N”和“ 1011”以及“ T”和“ 01”。因此，最佳编码仅需要51位，而使用144位消息以8位ASCII编码（压缩率为2.8到1）进行编码时则需要144位。

**输入输出格式**

**输入描述:**

输入文件将包含文本字符串列表，每行一个。 文本字符串将仅包含大写字母数字字符和下划线（用于代替空格）。 输入结束将通过仅包含单词“ END”作为文本字符串的行来表示。 此行不应被处理。

**输出描述:**

对于输入中的每个文本字符串，输出8位ASCII编码的位长度，最佳无前缀可变长度编码的位长度以及精确到小数点后一位的压缩率。

**输入输出样例**

**输入样例#:**

AAAAABCD

THE\_CAT\_IN\_THE\_HAT

END

**输出样例#:**

64 13 4.9

144 51 2.8