# 分段双调排序补充报告

最终结果：完成所有挑战。

**1.错误原因**

在得知自己的程序有错误之后，我立刻进行了仔细的检查和调试，然而因为测试样本太少，随机性太强，恰好看不出任何问题。重新读了几遍代码也没有发现问题的原因。于是我决定用python随机生成大量数据集进行测试。

经过大量数据测试以及和网上已经实现的代码结果进行比对后，发现了问题所在。我对简书中（<https://www.jianshu.com/p/ea4a62fdaae9>）这篇文章中所说的任意双调排序思想理解有误。

在我的上一版程序中，我在应对非标准长度数组时采用了三次排序来达到整体排序的目的。当时的想法是这样的：

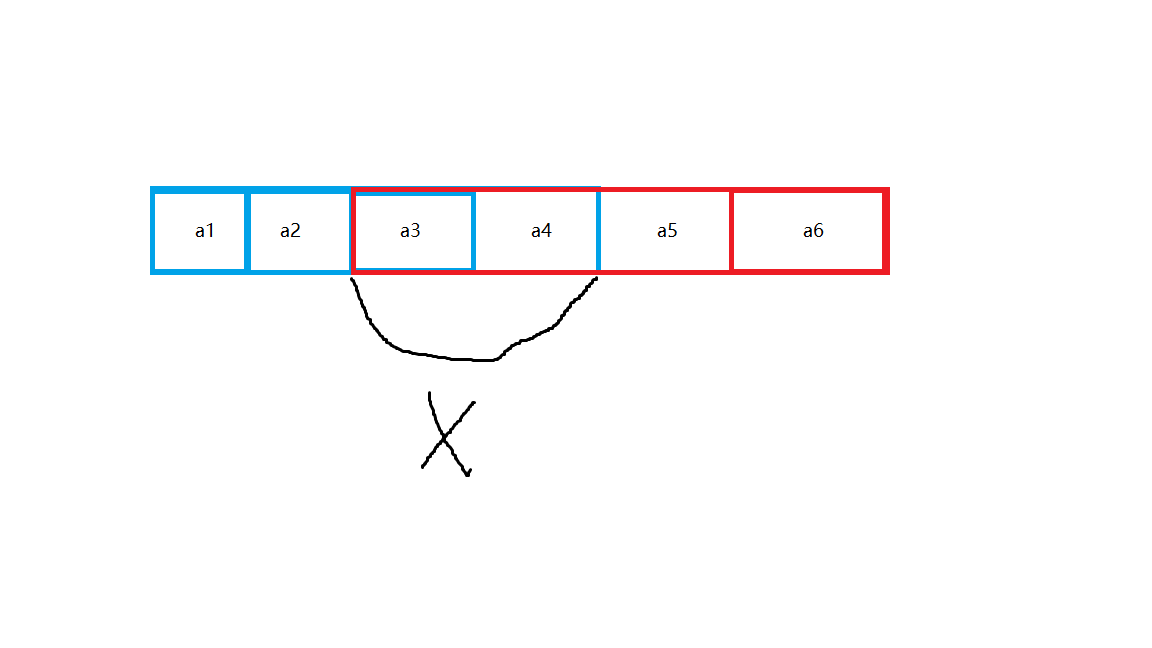


Figure 1错误的排序思路

如图所示，假设图中为一个长度为6的输入数组，不满足2的整数次幂条件。因此无法用标准的算法排序（递归方法可以，但是为了满足挑战内容不予使用）。此时，取为小于的最大的2的整次幂，然后将数组分为前项和后项顺次进行排序。图中，a3,a4位置既包括在前4项中，又包括在后4项中，因此我称这段区间为交换区。显然，经过第一次排序后，前4项中最大的两项处在中，再对后4项进行排序，这样可以保证序列中**最大的两项**处在a5,a6位置，即正确位置。最后再对a1到a4进行排序即可得到完整的升序数组。

然而，这种逻辑存在适用范围，即 。例如当时，，经过两次排序后，仅能保证最大值处在a7位置上，而a6位置的值是否正确不能保证。这就是我之前出现错误结果的原因。

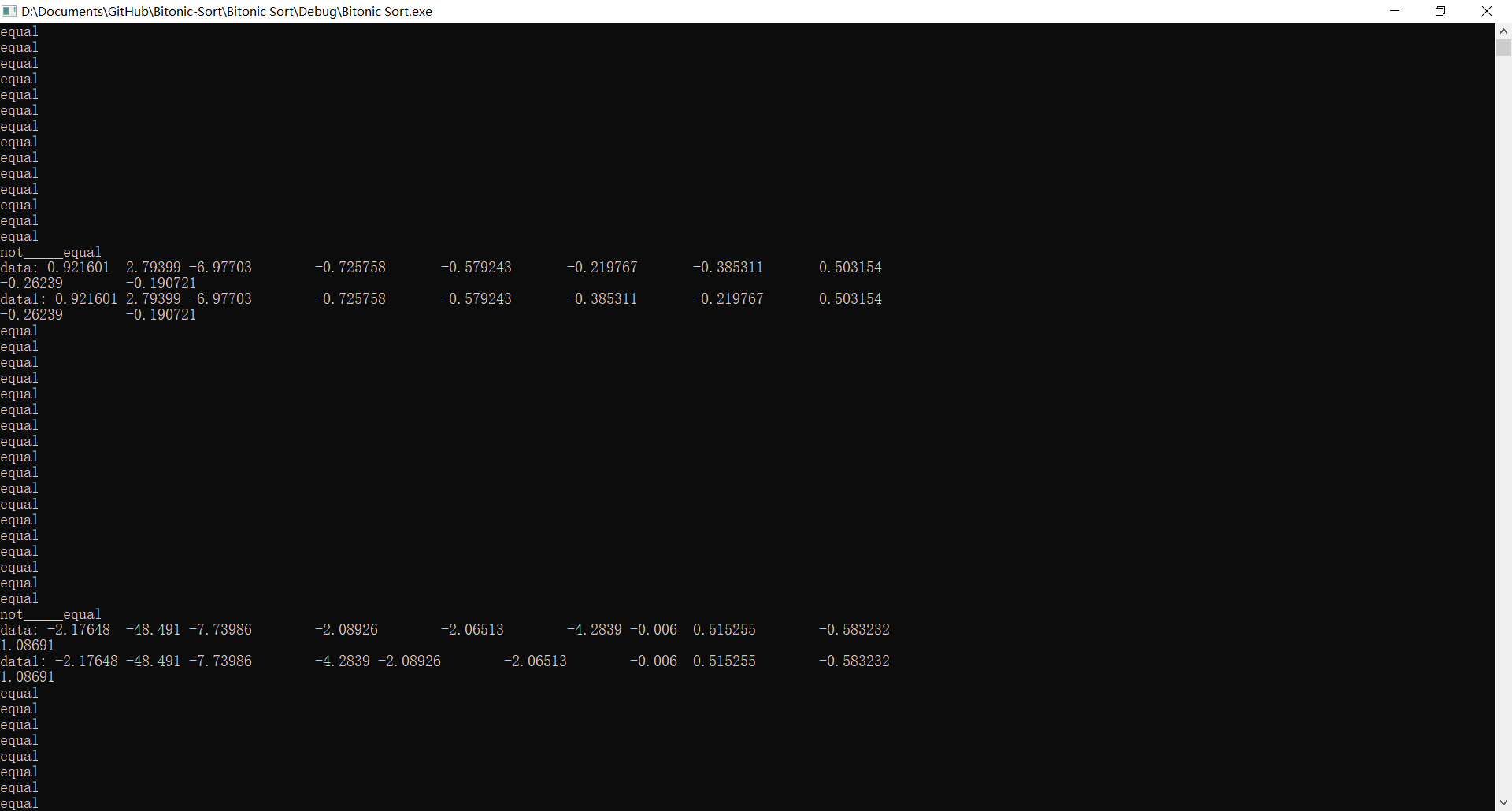


Figure 2错误情况

**2.解决过程**

1. 解决思路一 迭代求解

发现问题后，我直接想到的方法是在现有的基础上以同样的方式进行**有限次地**迭代来得到最终解。例如刚才的情况，进行第一轮排序后可以保证a7的正确性，然后再对a1-a6进行排序就可以得到正确的结果。然而这种算法的时间复杂度过高，会随着数组长度的增加指数型增长，想了想我决定放弃这种方法。

1. 解决思路二 将递归算法转化为循环

抛弃了之前的方法后，一时间找不到思路，于是回头重新细读了一遍问题和简书。文章作者是采用递归的方法，将数组不断分解至原子数组（长度为2或者1），再不断的进行双调合并得到最终结果。原理很简单，这里引用作者原话：

“双调排序分治的核心理论：**任意的正整数都能表示成 2 的幂指数和的形式，其实就是所有的正整数都能用二进制表示。**因为序列在不断地拆分成 2 的幂指数的长度。”

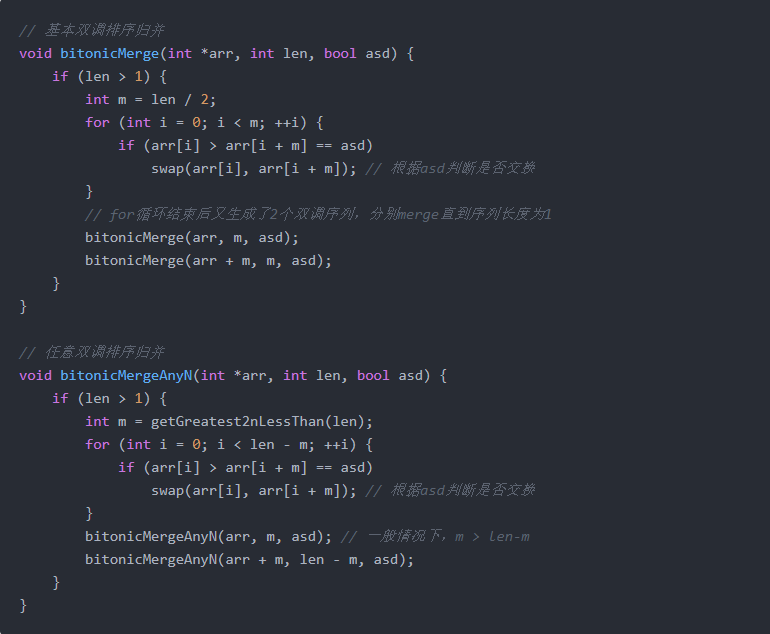


Figure 3任意双调排序归并

这个算法使用递归很容易实现，但是改写成循环却十分困难。递归是自顶而下的一种思想，而循环是自下而上的用法。我在网上查阅了许多资料后了解到。把递归改写成循环的一个**必要**条件是，不同递归分支的**递归深度**需要相同。对于标准数列来说，由于数列的高度对称性，经过有限次分解都可以偶数个原子数组。使用循环实现也很容易。

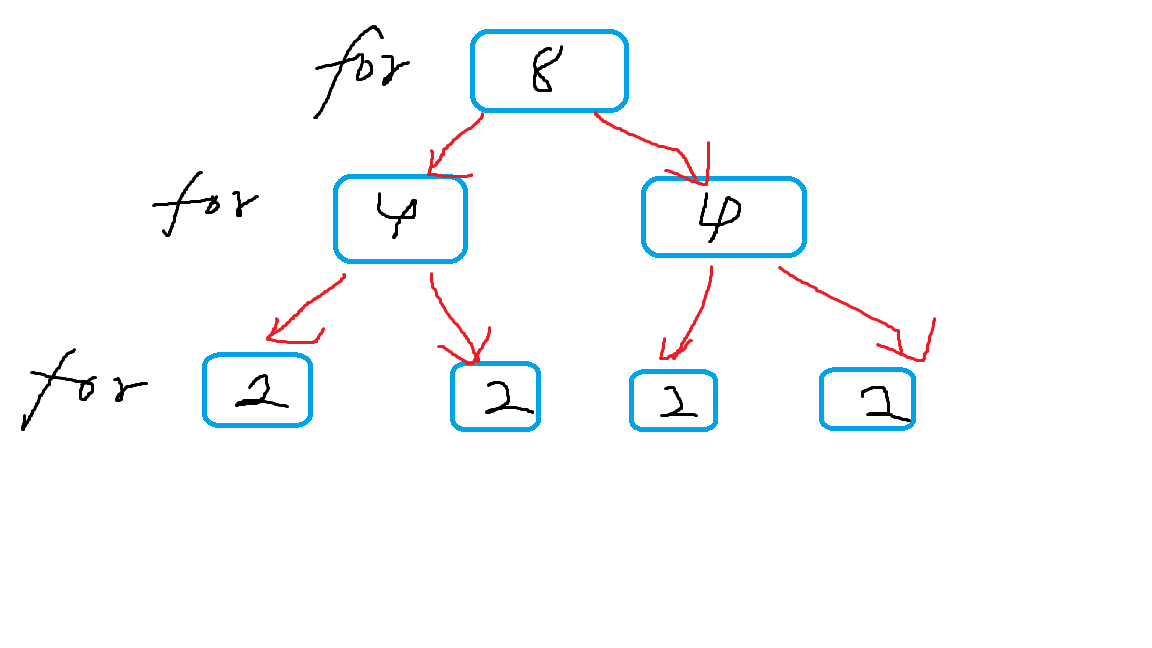


Figure 4标准数组分解

但是对于任意数组情况，由于分解是**不对称的**，导致同一级循环中步长不一致，分支的递归深度也各不相同，因此循环条件十分难以控制。

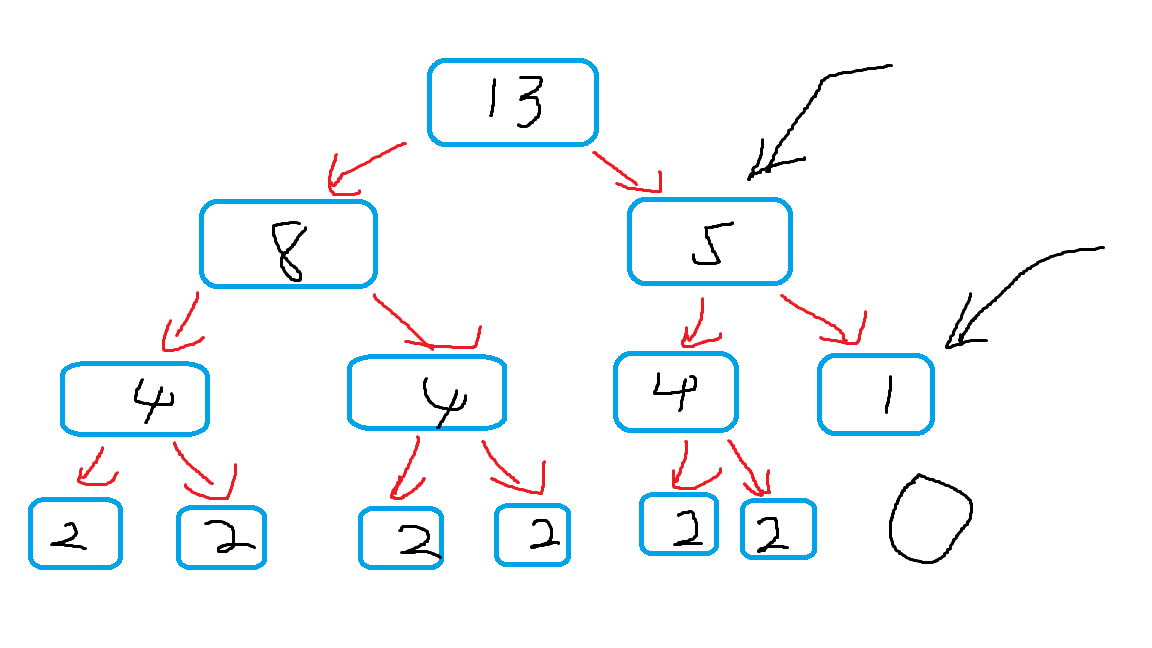


Figure 5任意数组分解

虽然很难直接改写成循环，但是不难看出其中还又一定的规律。左半部分都可以标准分解，因此可以独立拿出来进行排序。对于右边，每一分支的递归深度可以在循环前计算得到。也许可以利用得到的递归深度，合理地控制循环条件。但是我思考了一个上午没找到合适的方法，以后有机会再研究。

1. 解决思路三 利用冗余输入项扩展数组

双调排序的非递归算法其实已经非常成熟了，唯一的问题在于当数组长度不是标准长度时候要进行数组的填充，而根据挑战要求**不能动态分配内存**，导致无法填充数组。我在不断的审题过程中发现这样一句话：



重新审视了一遍冗余输入信息后，我发现了一个一直没有注意到的输入项。在程序中的作用仅仅在程序开始时候用来核对输入项是否合法。

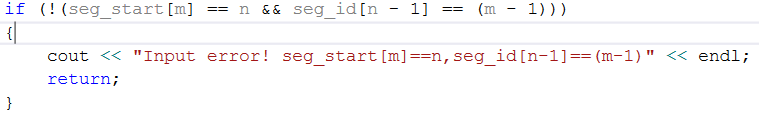


Figure 6输入合法检测

有很多优秀的特征，首先在循环中没有被使用，其次他的长度足够长与输入的数据数组长度相同，最重要是的和变量在内存中都占用**四个字节**，即与在内存中拥有**完全相同的两块内存**（不需要关心数据类型，比较时可以进行类型转换）。想到这里我顿时恍然大悟。需要用来填充数组的空间找到了，接下来我要做的只是将需要的内存块拼接在一起（进行动态的地址转换）。

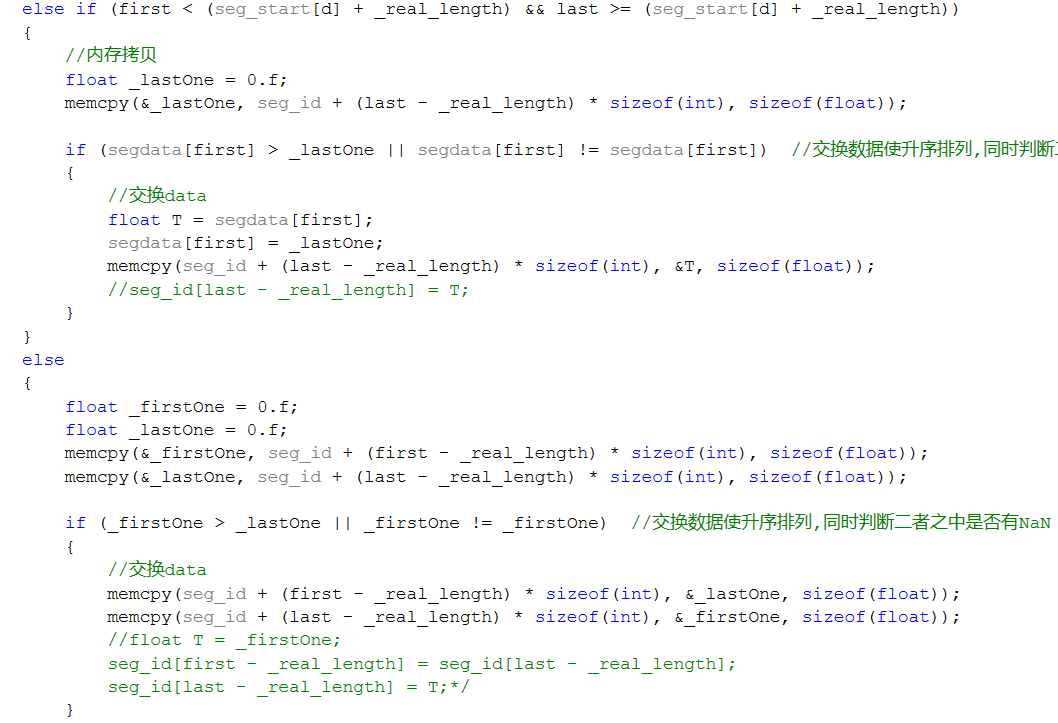


Figure 7动态地址转换

之后的测试非常成功，只是由于没有严格的管理内存，在大量样本测试时偶尔会出现内存冲突导致结果错误，概率很低。

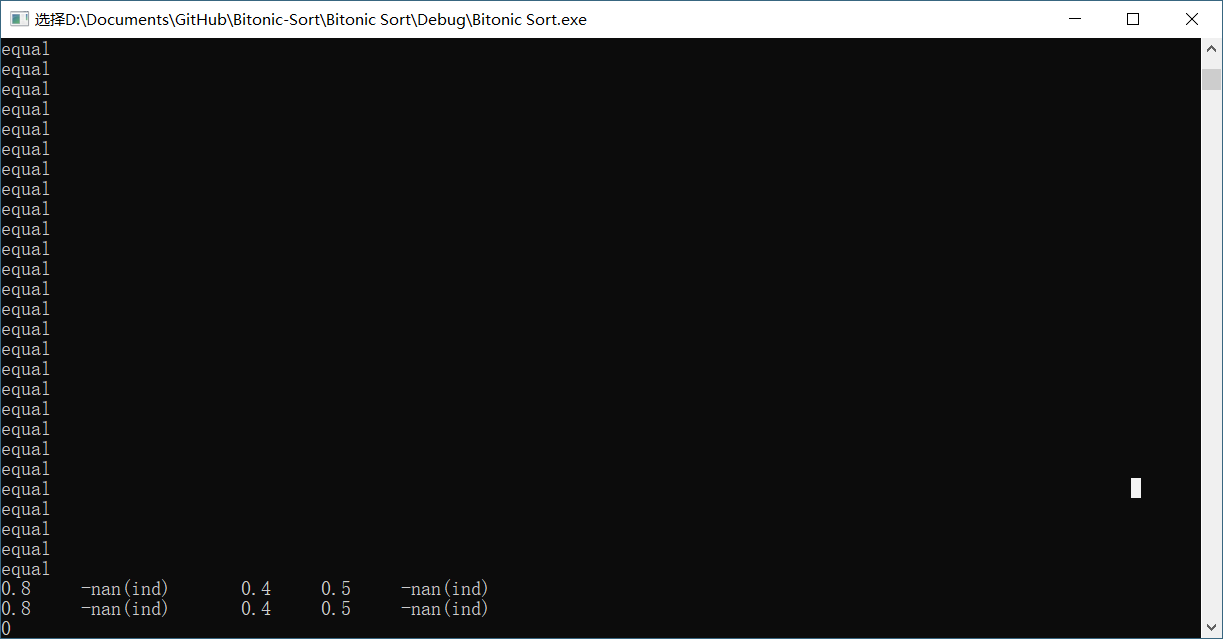


Figure 8正确测试结果

由于和任意数比较都是（包括本身），因此需要单独测试。在程序中，我设置了文件样本测试开关。如果不需要样本测试可直接修改下方data3等数组即可。

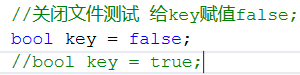


Figure 9样本测试开关

在程序中，我设置的默认填充数是 float Max = 9999999;//作为填充数 ，因此当输入存在更大的浮点数时，应将Max调整到**足够大**。