关于冒泡排序的算法改进

冒泡排序是对数列自上而下相邻的两个数进行比较和调整，让大数下沉，小数上浮。每次对数列的遍历可以确定一个最大值放在数列的末尾，下次遍历时，对这个最大值及以后的数不再进行排序。传统冒泡排序的时间复杂度为O（N^2）。

如图1所示，传统的冒泡排序无法对正序数列进行识别，造成多余的运算。在一些情况中，效率低下。



图1

如图2，对冒泡排序算法进行改进，增加一个bool类型变量flag用以标识一次遍历中是否进行数据交换。如进行交换，则将flag置为true，如果未进行数据交换，flag置为false，此时则说明数列已经成为正序数列，可以停止比较排序。



图2

通过使用clock（）函数，测得程序运行整体所需的时间，分别测试多次。获得改进前的冒泡排序所用时间为69.151s、47.485s、71.624s、42.534s、73.938s，如图3所示。改进后的冒泡排序所用时间为71.183s、72.605s、47.928s、51.981s、72.169s，如图4所示。总体来看，改进后冒泡排除所用时间与改进前所用时间相差不大。

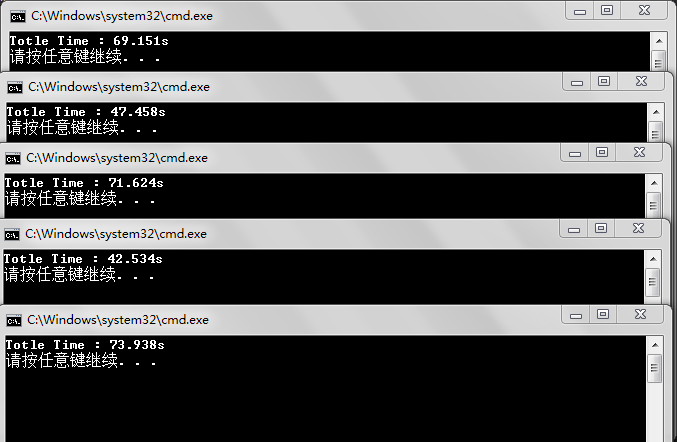


图3

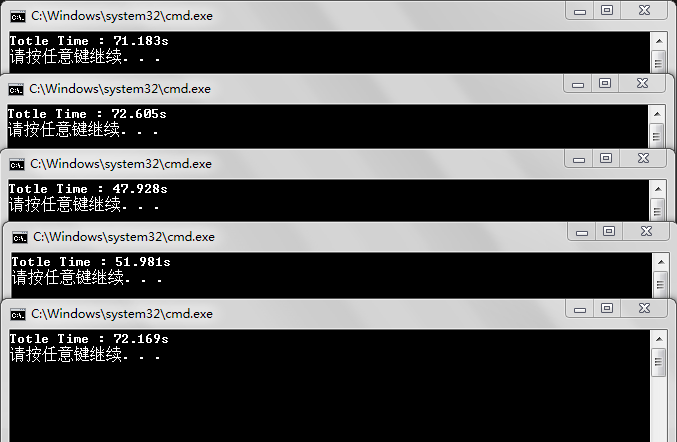


图4

考虑到改进后算法增加了bool类型的flag变量与对flag的值进行的判断，可能消耗更多的时间，同时考虑到测试次数不足，样本缺少代表性的可能，针对数据的特殊情况进行再次测试，在初始数列即为正序数列的情况下测试两个程序（图5）所用的时间。

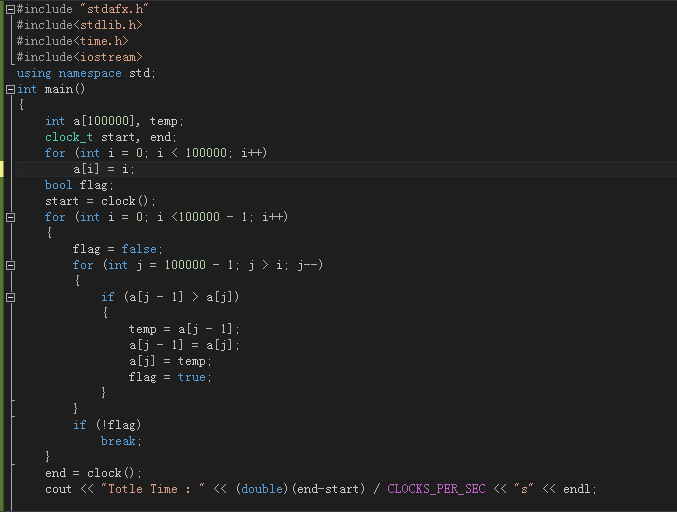


图5

如图7、图8，通过clock（）函数测得程序运行所需平均时间分别为0.001s、31.267s。

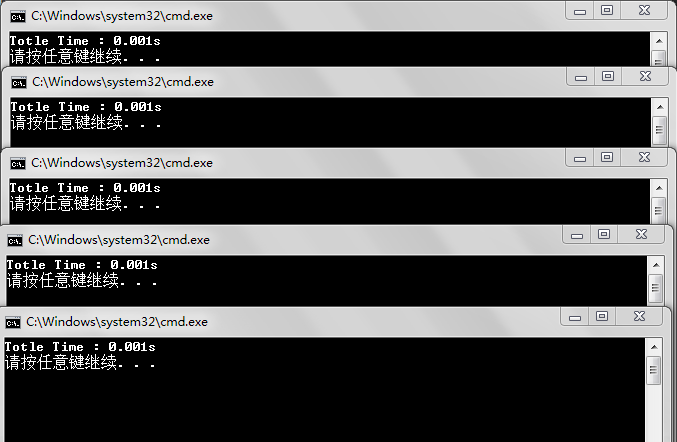


图7

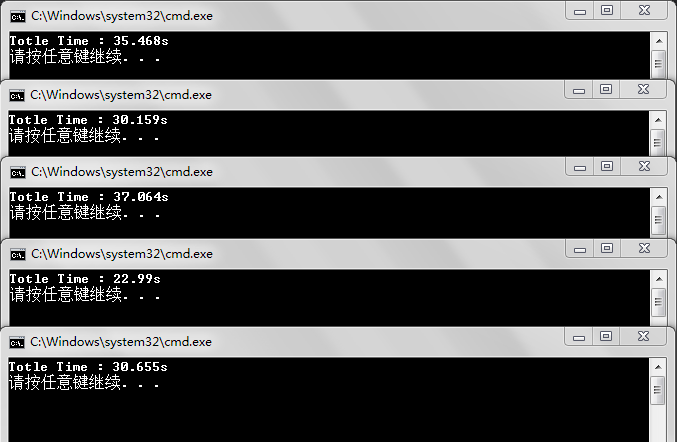


图8

总结：

个人认为改进后的算法能在一些情况下减少比较次数，但在大多数初始数列杂乱无序的情况下，无法明显节约所需时间。改进后的算法是否比原始算法更为省时，取决于初始数列的复杂度。初始数列越接近正序，改进后算法的优越性越明显。