**数据结构与算法专题实践报告**

**Project3**

1、源代码

见code文件夹中的源文件

2、问题描述/需求分析

编写代码实现外部排序（对大量数据进行排序）

需要模拟磁盘-内存操作，~~外部排序使用归并排序实现~~，~~初始顺串用插入排序生成~~

初始顺串改用败者树生成，这样可以生成尽可能长的变长顺串

初始顺串生成时加入多线程，使磁盘交互与败者树操作并行

外部排序使用多路归并排序实现（本代码采用8路归并）

归并时使用最佳归并序

3、系统结构/算法思想

使用vector模拟磁盘和内存缓冲区，用input/outputBufferSize来限制内存缓冲区大小

编写专用的函数来实现模拟的磁盘-内存操作，如内存读入、内存输出

~~使用简单的插入排序生成初始的顺串~~

使用败者树来生成初始的顺串

在败者树生成过程中加入多线程，保证在磁盘交互时败者树能保持运转状态

~~对顺串不断执行二路归并排序来实现最终的排序~~

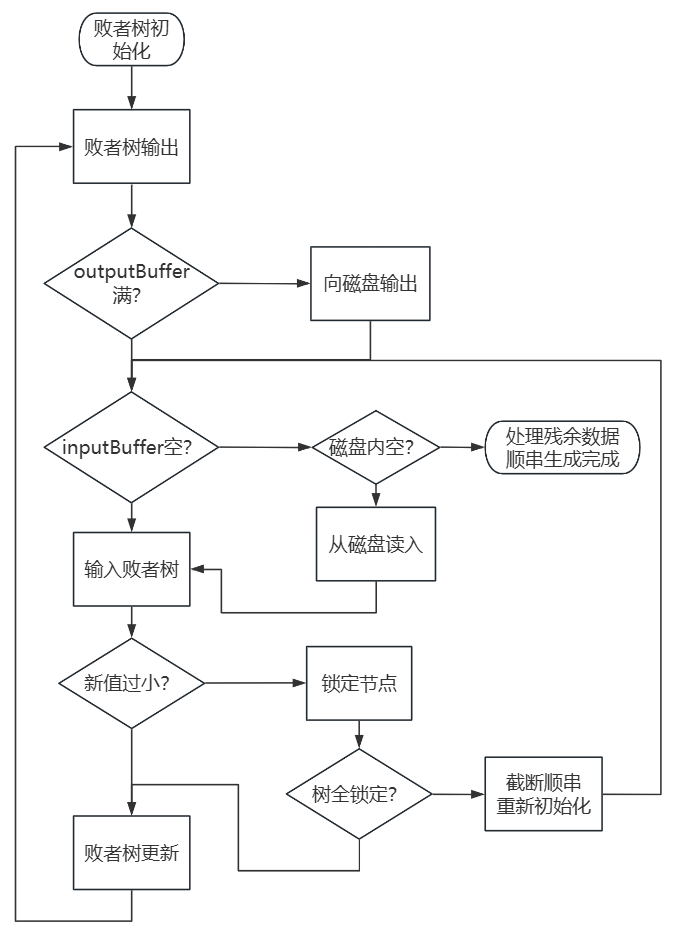
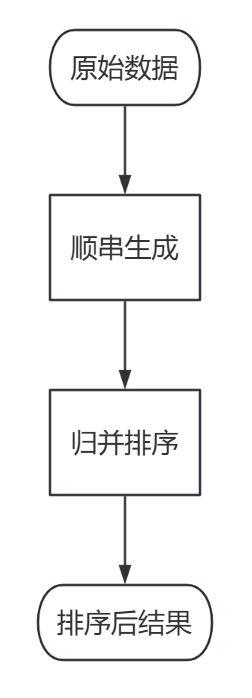
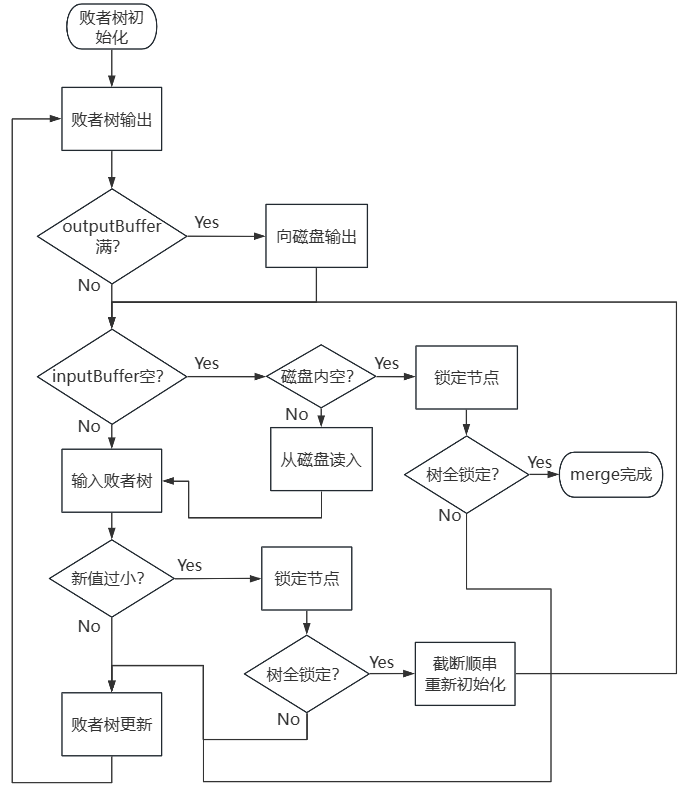
对顺串不断执行八路归并排序来实现最终的排序

八路归并时使用一个按顺串长度排序的优先队列来提供最佳归并序（类似哈夫曼树）

八路归并的具体实现可以重用顺串生成中的败者树结构

使用简单的随机数生成器来生成需要排序的数据，按上述顺序进行操作即可实现

4、功能模块设计

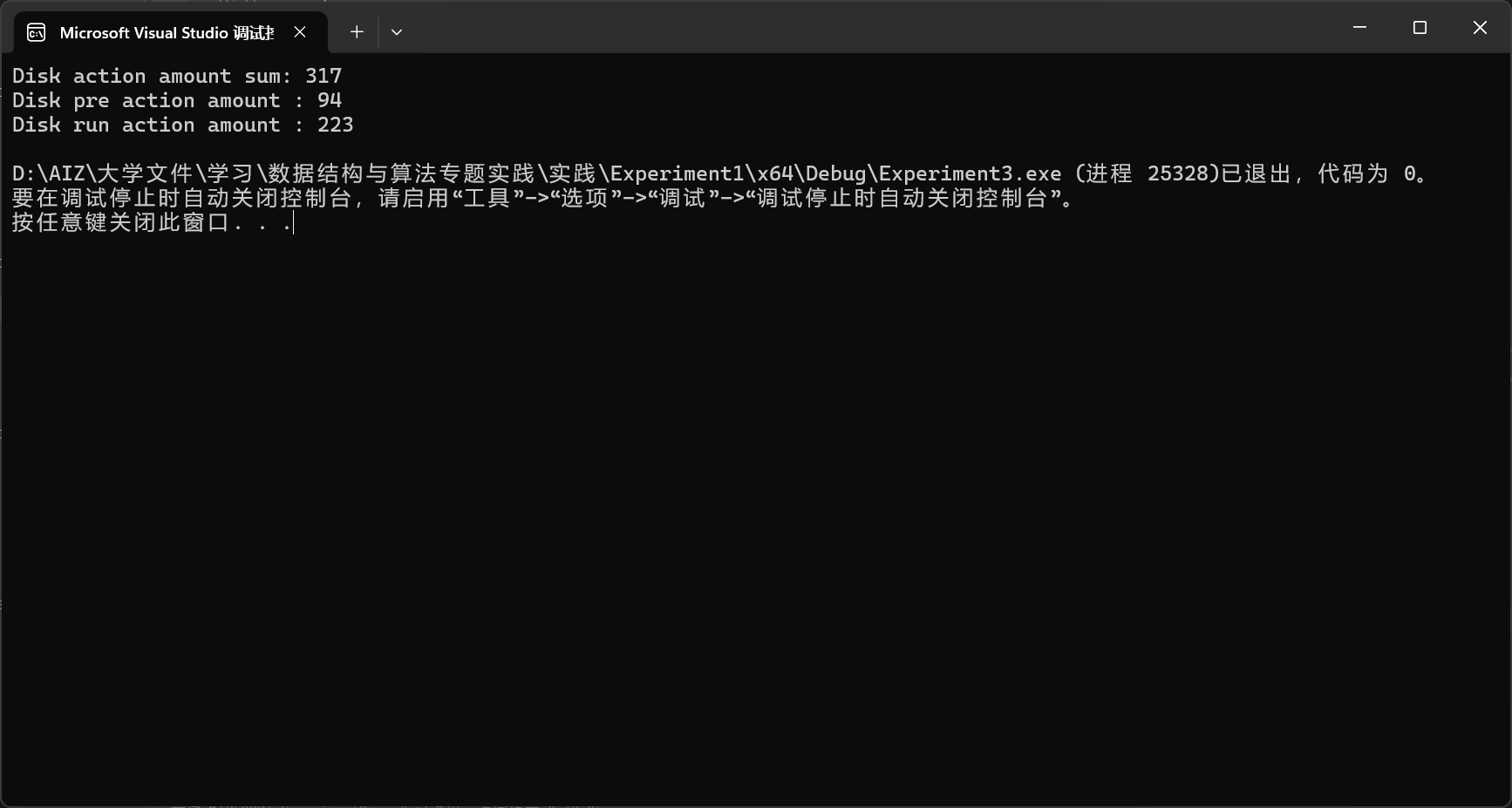
 

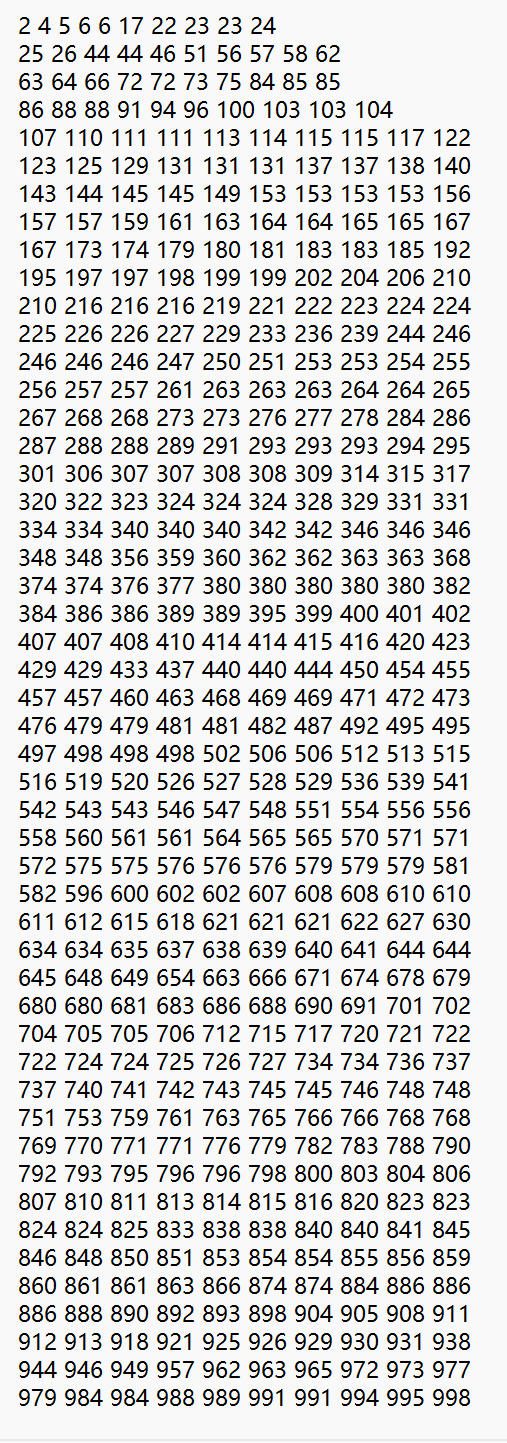
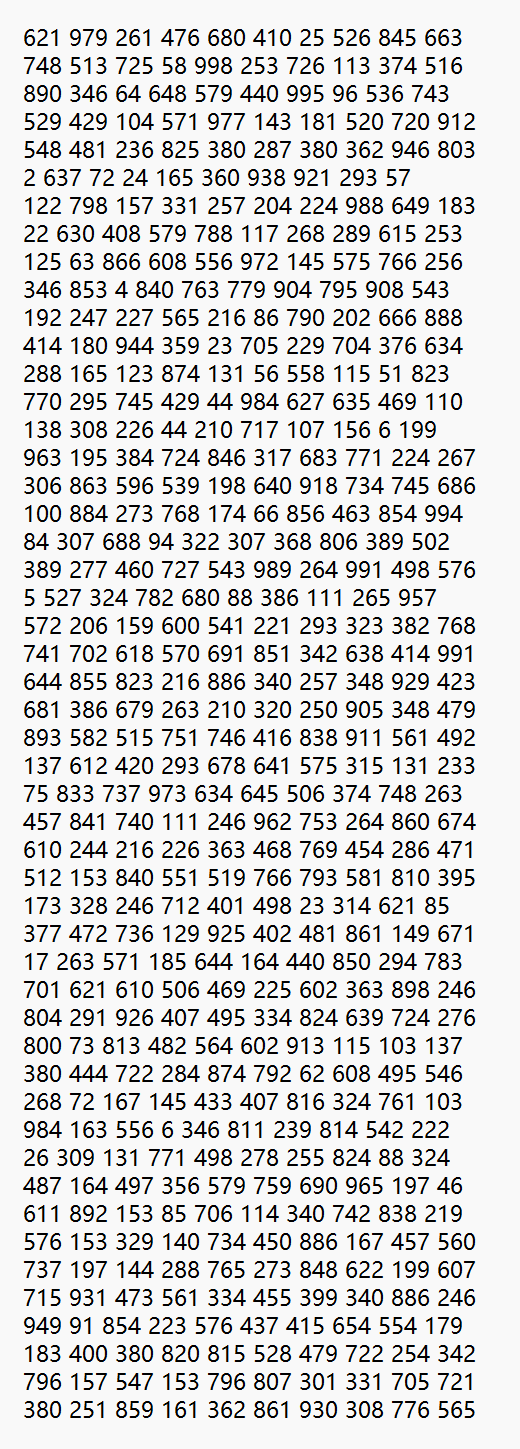
上图中，最右侧图即为归并排序过程中的败者树流程图，其中菱形框右侧为判断为Yes的情况，下方为判断为No的情况

败者树中的核心为败者树初始化函数、败者树更新函数、败者树节点锁定和败者树输入输出，实验3中对实验2的代码进行了模块化整合，将败者树初始化和败者树更新重写成了函数，同时修改了代码使其能够接受2的n次方规模的败者树操作

多线程的实现与实验2一致

5、测试结果与分析





对与实验1相同的数进行模拟外部排序（所以排序前后的文件内内容一致），结果显示在归并过程中进行了317次磁盘-内存操作，其中顺串生成时进行了94次，归并时进行了223次

由于顺串生成与实验2代码一致所以这里统计的次数基本一致（误差可能是由于顺串生成中的磁盘-内存操作数是总数减去归并时的操作数，而归并时的操作数是人工找到每次操作的代码并在前面加上一次操作数++操作实现导致的，所以可能有少量误差，但不影响整体结论）

可以看出八路归并相比二路归并明显减少了磁盘-内存操作数，这应该是由于八路归并相比二路归并大大减少了归并操作数，所以大大减少了整体的磁盘-内存操作数

6、实验总结

实验3中的多路归并可以使用修改后的实验2的败者树结构实现，所以在实验3中对实验2的败者树代码进行了优化，这个过程中感受到了代码鲁棒性的重要性，所以在代码整合的同时修改了代码使其能支持定制化的败者树操作

整体3个实验做下来，不光锻炼了我的代码能力，也让我对代码注释和代码管理有了更深入的理解，在一步步优化代码的过程中也有一种成就感