数据库实验2报告

’  
题目1：多路归并排序算法的实现

1：自己设计记录格式，至少包括2个属性（A和B），其中A为数值型，B作为记录的内容类型不限。

2：随机生成足够数量的记录，并存储为外存文件（尽量选择2进制格式）。

3：基于数值型属性A，用高级语言实现多路归并排序算法，并分析性能（时间和空间）。

# 1：描述实验环境的构建（记录和文件的准备）；

**说明：**该实验模拟外部排序，*考虑内存有限的条件，*设定条件:内存<总计Record大小<外存

*内存每次能装下的最大Record的数量loadMaxCountInMemory = 2500*

*文件中一共含有的Record的数量allRecordCount = 10000*

## Record类

设计了Record记录格式的*记录类，包含一个整数型属性A和一个字符串型属性B*

private int a;  
private String b;

实现了Serializable以便于序列化存入外存

实现了方法compareTo，通过属性a来比较类的大小

## RecordManager类

这是记录生成器类，用于生成随机记录并将其写入外部文件

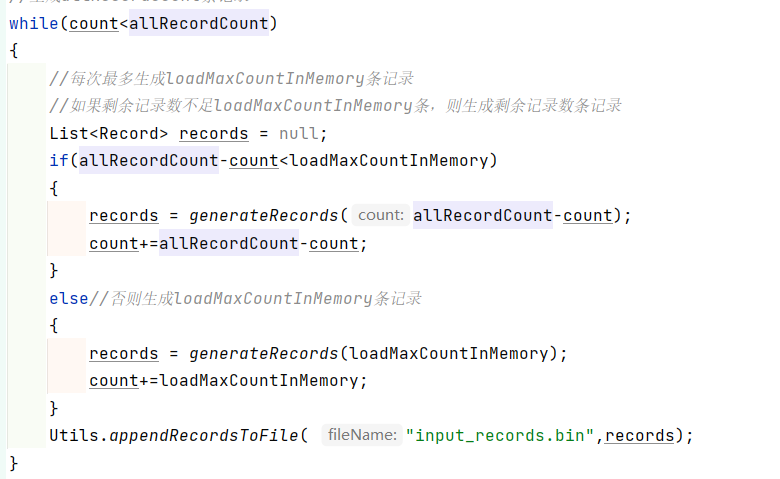
### public static List<Record> generateRecords(int count)

通过该方法可以生成指定数量的随机记录

### public static int generateRecords\_simulation(int allRecordCount,int loadMaxCountInMemory)

*模拟内存和外存的交互，考虑内存有限的条件，生成allRecordCount条随机记录并写入文件*

*由于内存有限，每次只能生成内存能运行的最大随机记录数量，利用循环多次生成并以二进制形式写入文件input\_record.bin*



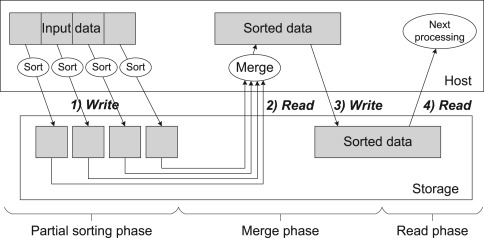
## Chunk类

说明：chunk实际上在模拟期间代表的是某个初始归并段的按照一定内存缓存区大小读入的流，chunk的大小可以写入内存，并且chunk将以流形式读入读出，chunk每次读出一个元素，将会从对应的初始归并段中读入一个元素。

Chunk实现了Serializable序列化方便从外存中读入读出

# 2：给出基本算法设计的伪代码或流程图；

## 外部排序算法设计的流程图：



## 外部排序算法设计的流程概述：

1. **第一步-生成初始归并段：**
   1. 将外存大文件input\_record作为流输入，利用置换-选择排序算法或者其他算法，将原文件输出为多个小的已排序好的chunk（初始归并段）
2. **第二步-归并：**
   1. 不断对之前的初始归并段进行k路归并排序，按照一定的内存缓存区大小读入，进行归并排序，当缓冲区满时，写入中间临时文件
   2. 当归并完成时候，排序结束，此时output\_record是已经排序好的文件

## 用败者树归并的思路概述

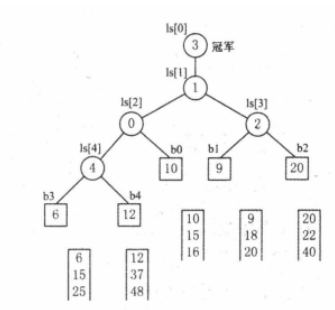
说明：chunk实际上在模拟期间代表的是某个初始归并段的按照一定内存缓存区大小读入的流，chunk的大小可以写入内存，并且chunk将以流形式读入读出，chunk每次读出一个元素，将会从对应的初始归并段中读入一个元素。

**通过败者树将初始归并段chunk流进行归并：**

1. 利用败者树选择所有chunk中头部最小的元素，并把它放进外存，即ls[0]对应的bq
2. 更换刚才被选出来的元素pop，换成它的chunk的下一个元素放在bq，调整败者树
3. 如果某个chunk空了，则剔除该chunk，然后调整败者树
4. 循环以上1~3直到全部chunk为空

# 3：描述主要函数的设计过程；

## 败者树LoserTree的设计：

​

败者树示意图

* 按照这个图顺序定义b0,b1~b4的顺序，b0必须从第一个最小叶子结点作为b0，才符合b\_parnet = (b\_index+b\_size)/2
* 叶子节点对应着初始归并段的流chunk
* 每次都是记录败者节点，拿着胜者去向上比较

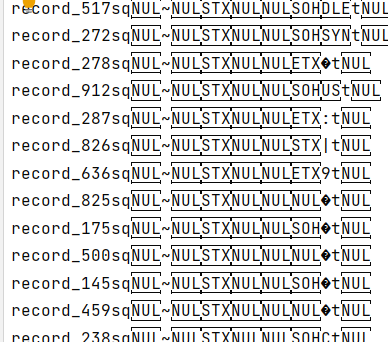
### 败者树的属性：

int[] l*s以顺序存储方式保存所有非叶子结点*int size 记录叶子节点个数  
ArrayList<Chunk> leaves = null;*// 叶子节点*

### 败者树的主要方法：

# 4：程序运行结果；

* 1. 随机生成10000条乱序数据记录存入input\_record.bin二进制文件



2.进行第一趟归并，生成4