**作业6**

SA20225085 朱志儒

1. 已知有关系模式R(A, B, C) 和S(B, C, D)，每个属性都占10个字节，请估计下面的逻辑查询计划T(U)，S(U)以及结果关系中每个属性的V值（假设满足“Containment of Value Sets”，并且选择条件中的值都在关系中存在）：

U=πAD [ (σ A=3 ∧ B=5 R)⋈ S)]

相应的统计量如下：

T(R) = 100000, V(R,A) = 20, V(R,B) = 50, V(R,C) = 150

T(S) = 5000, V(S,B) = 100, V(S,C) = 200, V(S,D) = 30

**解：**令 ，则

令 ，则

故

2、 固态硬盘（Solid State Drive，SSD）是一种基于闪存的新型存储器，它与传统磁盘的主要区别之一是：传统磁盘的读写操作的速度相同，而SSD的读速度远快于写速度。同时，SSD的读速度要远高于磁盘，而写速度则比磁盘慢。现在我们想将传统的两阶段多路归并排序算法移植到SSD上。假设SSD上一次读块操作的时间是t，一次写块操作的时间是50t，磁盘上的读/写块时间是30t。对于给定关系R：

• R 包含100000个元组，即T(R) = 100000.

• 一个磁盘块大小为4000 bytes.

• R 的元组大小为400 bytes，即S(R) ＝ 400.

• 关系R在磁盘上非连续存放

• 排序字段的大小为32 bytes.

• 记录指针的大小为8 bytes.

现在我们考虑下面一种改进的归并排序算法。原来的两阶段归并排序的第一阶段是将排序后的整个元组写到chunk中，现在我们仅将排序后的<sortingKey, recordPointer>写出。第一阶段，我们在内存中将记录按<sortingKey, recordPointer>排序，当<sortingKey, recordPointer>记录填满内存时将其写到chunk中。第二阶段，读入各个chunk中的<sortingKey, recordPointer>并在内存中归并。通过记录指针(recordPointer)我们可以读取记录的其它部分(从R的磁盘块中)，并将排好序的记录写回到外存。请回答：

1） 如果R存储在磁盘上，这一改进排序算法的I/O代价（用t的表达式表示，包括最后写出到排序文件中的代价）是多少？并解释该算法性能是否能优于原来的排序算法。

2） 如果R存储在SSD上，这一改进排序算法的I/O代价（用t的表达式表示，包括最后写出到排序文件中的代价）是多少？并解释该算法性能是否能优于原来的排序算法。

**解：**1）如果R存储在磁盘上，由于关系R在磁盘上非连续存放，将元组全部读入内存的时间为

一个磁盘可以装< sortingKey, recordPointer >的数量为

将排序后的< sortingKey, recordPointer >全部写到磁盘的时间为

将各个chunk中的< sortingKey, recordPointer >读入到内存的时间为

通过记录指针(recordPointer)读取记录的其它部分到内存的时间为

所有元组排序后写出到磁盘的时间为

综上可知，改进排序算法的总时间代价为

而原来的排序算法的总时间代价为

显然 ，故改进的排序算法没有优于原来的排序算法。

2）如果R存储在SSD上，将元组全部读入内存的时间为

将排序后的< sortingKey, recordPointer >全部写到SSD的时间为

将各个chunk中的< sortingKey, recordPointer >读入到内存的时间为

通过记录指针(recordPointer)读取记录的其它部分到内存的时间为

所有元组排序后写出到SSD的时间为

综上可知，改进排序算法的总时间代价为

而原来的排序算法的总时间代价为

显然 ，故改进的排序算法优于原来的排序算法。

3、 我们在课本上讨论的归并排序算法是一个两趟算法。设两个连接关系为R1和R2，在基于两趟归并排序的排序连接算法中，我们要求内存M必须满足条件 。现在我们考查关系R的两趟归并排序算法，我们发现当内存M不满足条件 时，我们仍可以采用一种多趟算法来完成归并排序操作。请用自然语言或伪码给出这一多趟归并连接算法的简要描述和步骤，并给出当B(R1)=10000，B(R2)=5000，M = 20时该算法的I/O代价，这里我们假设R1和R2都是连续存放的。

**解：**多趟归并连接算法：

1. 令 ，从磁盘中读取R的M个块到内存的Buffer区，在每个块中采用内排序算法使得元组在块内有序，并且使得第1个块中元组的最小值小于第2个块的最小值，第2个块的最小值小于第3个块的最小值，以此类推，即前一个块的最小值小于后一个块的最小值，再将这M个块作为1个chunk写出到磁盘中，重复上述操作直到R中元组均为块内有序，chunk中以块的最小值有序，进入（2）。
2. 令 ，即现有T个chunk，将该T个chunk作为一个集合S，从S中选M个chunk出来作为chunk数组H，集合S删除选中的chunk。然后分别从chunk数组H的每个chunk中选第1个块组成块数组K，并读入到内存的Buffer区，采用归并算法进行排序，将结果存入1个输出块中。
3. 若块数组K的第i个块中元组全被选完，则从chunk数组H的第i个chunk中读取下一个块到内存中。
4. 若输出块被填满，则将其写回到磁盘中。

本次操作得到的多个输出块组成一个新的chunk，以便下次迭代使用。重复上述操作直到集合S为空，进入（3）。

1. 若 ，则重复（2）进行一次新迭代，直到 ，即最后得到的1个chunk就是排好序的关系R。
2. 使用上述算法将R1和R2排序，最后将R1和R2归并连接。

IO代价分析：

对于关系R1，上述算法的迭代次数为

IO次数为

对于关系R2，上述算法的迭代次数为

IO次数为

将R1和R2归并连接的IO次数为

综上可知，总的IO次数为