Homework #5 (12.5)

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的 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）：

$$U=\pi_{AD}[(\sigma_{A=3 \wedge B=5} R) \bowtie S)]$$

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

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。

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 $M \geq \max\{\sqrt{R1}, \sqrt{R2}\}$。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 $M \geq \sqrt{R}$ 时，我们仍可以采用一种多趟算法来完成归并排序操作。请用伪码给出这一多趟归并连接算法的简要描述，并估计当 **T(R1)**=500000，**T(R2)**=100000，**S(R1)= S(R2)=**1/10 **block**， **M** = 100 时该算法的 I/O 代价，这里我们假设 **R1** 和 **R2** 都不是连续存放的。