Tutorial 2 – CSC443 - Notes

Example 11.7 :
Records: 10,000,000
Record size: 160 bytes
Records/Block = 100
blocks = 100,000
Main Memory: 100 MB
Blocks Main Memory can hold: 6400

Exercise 11.4.1: Using TPMMS, how long would it take to sort the relation of Example 11.7 if the Megatron 747 disk were replaced by the Megatron 777 disk described in Exercise 11.3.1, and all other characteristics of the machine and data remained the same?

Answer 11.4.1 : From Example 11.7 We know:
10,000,000 records. 100 records/ block -> 100,000 blocks.
6400 blocks fit in main memory.
1st Phase: Data is read, 16 times in total. 15 times in chunks of 6400 blocks, and last chunk 4000 blocks.
100,000 I/Os (Read) + 100,000 I/Os (Write) = 200,000 I/Os. 11ms per IO = 37 minutes in total for 1st phase.

2nd Phase : 1 buffer for each of 16 sublists from phase 1. 1 output buffer. Put smallest element in output block and if output block is full -> Write to disk. Repeat until all blocks of 16 sublists processed.
I/O cost: 100,000 I/Os (Read) + 100,000 I/Os(Write) = 200,000 I/Os. 11ms per IO = 37 minutes.

Total time = 74 minutes.
Total I/Os = 200,000 + 200,000 = 400,000 I/Os

Exercise 11.4.2: Suppose we use TPMMS on the machine and relation R of Example 11.7, with certain modifications. Tell how many disk I/O's are needed for the sort if the relation R and/or machine characteristics are changed as follows:

Answer 11.4.2 :
(a) : **The number of tuples in R is doubled (all else remains the same):**
The relation occupies 100,000 blocks, and the sort takes 4 disk I/O's per block, or 400,000 disk I/O's. If the number of tuples were doubled, the relation would occupy twice as many blocks, so we would expect 800,000 disk I/O's. We should check that the 2-phase sort is still possible, but since doubling the size of the relation would require 32 sublists rather than 16, there will still be plenty of room in memory on the second pass.

(b) : **Length of Tuple is doubled from 160 to 320 bytes.**
So, number of records per block reduced to 50 from 100. Relation R now fits into 10,000,000/50 = 200,000 blocks.
1st Phase: Data is read 32 times in total. 32 sorted sublists are created on disk.
I/O = 200,000 (Read) + 200,000 (Write) = 400,000 I/Os.

2nd Phase: 1 buffer for each of 32 sublists. 1 output buffer. Following merging algorithm of Two-Phase Multi way merge sort.
I/O = 200,000(Read) + 200,000 (Write) =400,000 I/Os

Total I/O = (800,000) I/Os. (Doubled overall).

(c) : **The size of blocks is doubled, to 32,768 bytes**
Doubling the size of blocks reduces the number of disk I/O's needed to half, or 200,000. We might imagine that we could keep growing the size of blocks indefinitely, and thus reduce the number of disk I/O's to almost zero. In a sense we can, but there are limitations, including the fact that transfer time will eventually grow to dominate other aspects of latency, in which case counting disk I/O's fails to be a good measure of running time.

(d): **The size of available main memory is doubled to 200 megabytes:**
Number of blocks that fit into memory doubled to 12,800 from 6400. First phase now creates 8 sorted sub lists instead of 16 (originally). 2nd phase performs the merge. But, total I/O cost in both phases is still unchanged. 400,000 I/Os in total.

! Exercise 11.4.3: Suppose the relation R of Example 11.7 grows to have as many tuples as can be sorted using TPMMS on the machine described in that example. Also assume that the disk grows to accommodate R, but all other characteristics of the disk, machine, and relation R remain the same. How long would it take to sort R?

11.4.3:
Total records that can be sorted:
Total memory buffers available = 6400.
Max sorted sublists 1st phase can generate = 6399
1 buffer reserved for output in second phase.
Rec/block = 100.
So, max blocks that can be sorted = 6400 * 6399.
6400 * 6399 * 100 = 4,095,360,000 = (approx 4.1 billion records)
Blocks = records/100 = 40,953,600
1st phase: Num of sorted lists generated = 40953600/6400 = 6399.
2nd Phase: merge.

Total I/O = 40953600 * 4 = 163,814,400 (approx 164 million I/Os)
Time = 11ms per I/O -> 30032.64 minutes

* **Exercise 11.4.4:** Let us again consider the relation R of Example 11.7, but assume that it is stored sorted by the sort key (which is in fact a "key" in the usual sense, and uniquely identifies records). Also, assume that R is stored in a sequence of blocks whose locations are known, so that for any i it is possible to locate and retrieve the ith block of R using one disk I/O. Given a key value K, we can find the tuple with that key value by using a standard binary search technique. What is the maximum number of disk I/O's needed to find the tuple with key K?

11.4.4: Exercise 11.4.4

Binary search requires probing the block in the middle of the 100,000 blocks, then the one in the middle of the first or second half, and so on, for log_2(100,000) = 17 probes, until we can narrow down the presence of the desired record on one block. Thus, we require 17 disk I/O's.

!! **Exercise 11.4.5:** Suppose we have the same situation as in Exercise 11.4.4, but we are given 10 key values to find. What is the maximum number of disk I/O's needed to find all 10 tuples?

11.5.5 : Exercise 11.5.5

In worse case each key value in different block. 10*log_2(100,000) = 170 I/Os.
In best case all key values in same block = 1 I/O.