CSC 370 A4

1. To sort the relation consisting of 10,000,000, divide that number by the amount of available buffering blocks (320)

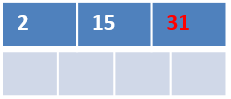
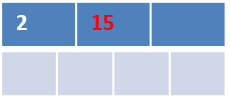
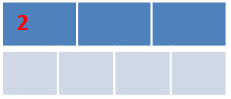
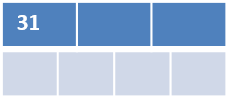
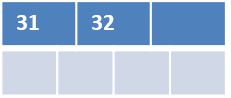
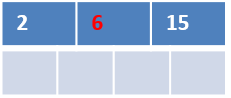
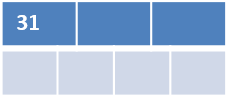
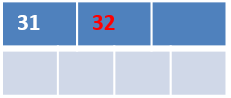
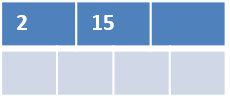
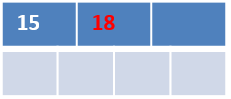
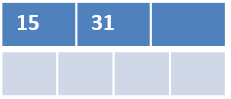
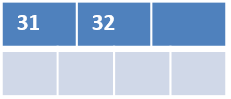
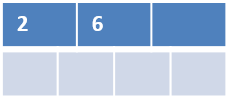
10,000,000/320 = 31,250 sorted sublists at Phase 1

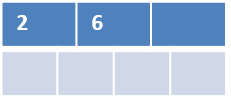
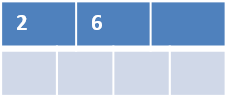
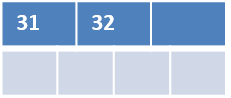
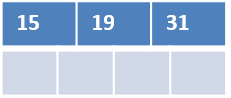
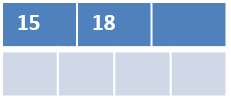
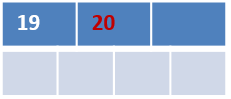
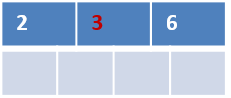
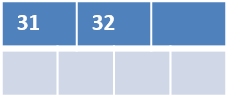
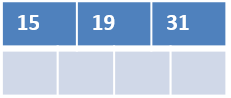
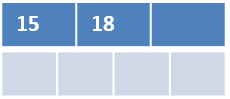
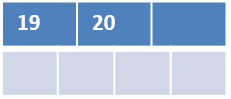
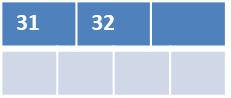
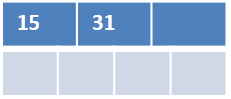
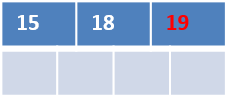
We check if we can merge, but there are too many sorted sublists to merge them all with one pass. Take the 31,250 sorted sublists from Phase 1, and sorted those using (320 - 1 output buffer)

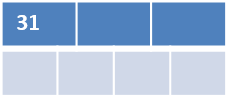
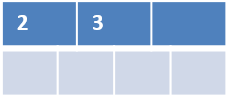
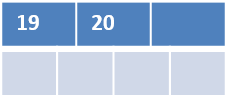
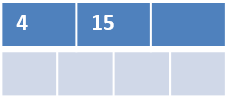
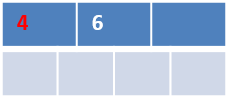
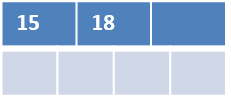
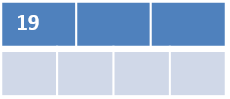
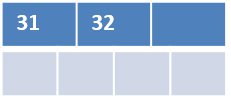
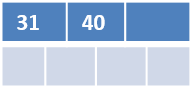
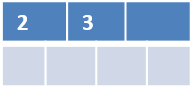
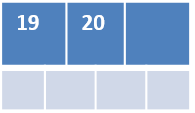
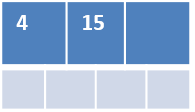
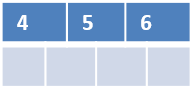
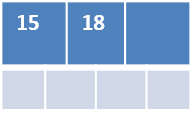
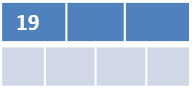
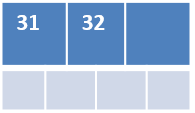
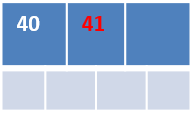
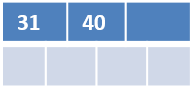
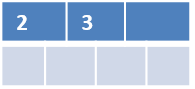
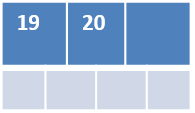
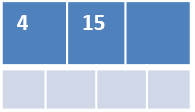
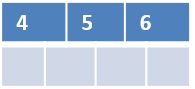
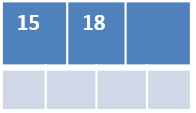
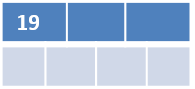
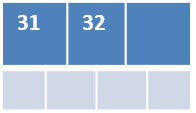
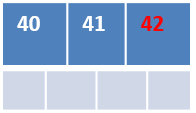
31,250/319 = 98

The 98 sorted sublists above are merged in Phase 3. Therefore 3 phases are needed

In each phase we read and write 10M blocks. Therefore, the total time cost for I/O is 16ms \* 3 \* 2 \* 10,000,000 = 960,000,000ms







Assuming for Maintenances:

1. A tuple is 160 bytes
2. A block can hold is 16K
3. There are 1000 blocks

Assuming for Employees

1. A tuple is 130bytes
2. A block can hold 120 tuples
3. There are 50 blocks.

**sort-merge**:

Maintenance: 2 \* 2 \* 1000 = 4000 I/O

Employees: 2 \* 2 \* 50 = 200 I/O

Merge-joining requires rescanning both tables, thus the total I/OI for sort-merge is 5250 I/O.

**Projection:**

is done on the fly, thus costs 0 I/O.

**Maintenance:**

takes ~1000 I/O if there are 1000 blocks to search. Keeping in memory requires:

1. Originally scanning the relation
2. Writing to MM
3. Reading from MM when writing result (extremely quick process, if exponentially quicker, will take ~10 I/O)

Therefore scanning **Maintenance** and writing to MM takes ~2010 I/O.

**Ratings:**

Since the query is scanning a file and piping the results, the result should only be the amount of blocks, or 50 I/O.

Adding all of the above, we get 5250 + 0 + 2010 + 50 = 7310 I/O.

1. R(A,B,C,D) with FDs AB->C, B->D, CD -> A, AD -> B

AB -> C: {A,B}+ = {A, B, C, D }. Is superkey, therefore holds

B -> D: {B}+ = {B, D}, is not a superkey, and therefore violates BCNF

B -> BD, therefore:

R1(B, D)

R2(A, B, C)

CD -> A: {CD}+ = {C, D, A, B} is a superkey, and therefore holds

AD -> B: {AD}+ = {A, D, B, C} is a superkey, and therefore holds

1. R(A,B,C,D) with FDs A->B, B->C, C->D, D->A
   1. A -> B: {A}+ = {A, B, C, D} is a superkey, and therefore holds
   2. B -> C: {B}+ = {B, C, D, A} is a superkey, and therefore holds
   3. C -> D: {C}+ = {C, D, A, B} is a superkey, and therefore holds
   4. D ->A: {D}+ = {D, A, B, C} is a superkey, and therefore holds