COMPSCI 751 S1 C – Assignment 4

Database Systems

Name: Yixuan Li

UPI: yil845

1. A relation, a collection of tuples, is physically stored in a file, a sequence of blocks, on the disk. To execute "order by StudentID" on a relation Student with 33 student records, the database engages external memory sorting algorithm. Originally, all tuples of Student are stored on the disk with their StudentID values sequentially 48, 84, 23, 74, 46, 82, 49, 8, 39, 103, 75, 118, 7, 115, 113, 114, 107, 33, 102, 98, 21, 85, 34, 80, 93, 78, 19, 117, 86, 109, 116, 61, 65, and each block (except for the last one) accommodates 2 tuples. The system has allocated a buffer pool of 4 frames to carry out the sorting. Please answer the following questions (Hint: you may write your python code to save time in manual sorting).

(a) Show the resulting runs of every pass of the sorting, including pass 0. [7 marks]

n= ceiling(33/2) = 17 (blocks)

48, 84	23, 74	46, 82	49, 8	39, 103	75, 118	7, 115	113, 114	
107, 33	102, 98	21, 85	34, 80	93, 78	19, 117	86, 109	116, 61	65

Each Buffer pool has 4 frames,

Runs: ceiling(17/4) = 5 runs, each run has <= 4 pages, records sorted.

Pass 0

R1	R2	R3	R4	R5
8,23,46,48,49, 74,82,84	7,39,75,103, 113,114,115,118	21, 33, 34, 80, 85, 98, 102, 107	19,61,78, 86, 93, 109, 116, 117	65

Pass 1

Merge R1 R2	Merge R3 R4	R5
7,8,23,39,46,48,49,74, 75, 82, 84, 103, 113,114,115,118	19, 21, 33, 34, 61, 78, 80, 85, 86, 93, 98, 102, 107, 109, 116, 117	65

Pass 2

7, 8, 19, 21, 23, 33, 34, 39, 46, 48, 49, 61, 65, 74, 75, 78, 80, 82, 84, 85, 86, 93, 98, 102, 103, 107, 109, 113, 114, 115, 116, 117, 118

(b) Compute the number of I/Os consumed by the above external memory sorting. [3 marks]

m=4, n=17, celing (n/m) = 5, celing(Log3(5)+1) =3 passes; 2n* passes = 102 I/Os.

The above external memory sorting consumes 102 I/Os.

2. Assume that each block has B = 1000 Bytes, and the buffer pool has m = 1001 frames. What is the exact size of the largest file external memory sorting can sort using 3 passes (pass 0 - pass 2; pass 2 produces only 1 run)? What is the scale of the above file, 1KB, 1MB, 1GB, 1TB, or 1PB? [5 marks]

Because $\lceil \log_{m-1} \lceil n/m \rceil \rceil + 1)$ = 3,

for M= 1001, n= 1000^2 * 1001 = 1,001,000,000 blocks. If each block is 1000 bytes, the scale of file is roughly in 1TB.

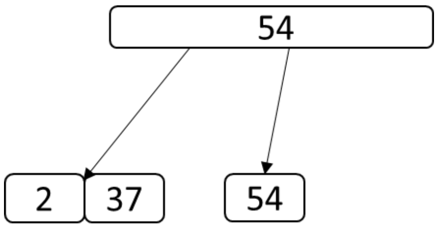
3. Please explain the benefit of distributing the parity blocks of multiple files over different disks under RAID Level 5. [5 marks]

RAID Level 5 is Block-Interleaved Distributed Parity. In this level, every blocks of binary data are stored in such a way that an equivalent parity data (error correction bit) generated by XOR function in order to improve the reliability of data recovery. Parity data of different blocks are stored at different disks. Since for every data modification, parity data will be modified, storing parity data on different disk can balance the I/O of each disk.

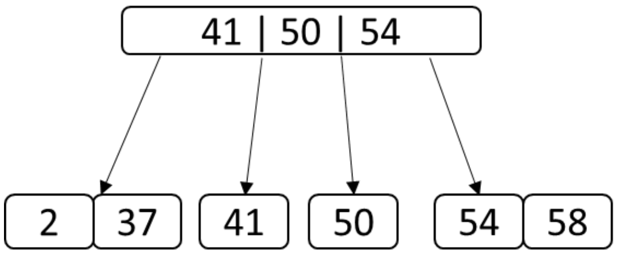
4. Consider the structure of B+-tree introduced in the class1. Each leaf/internal node of a B+-tree is physically stored on the disk as a block. Tuples are stored only on leaves while each internal node holds only interleaved key values and pointers: in each internal node, the # of points is always 1 more than the # of key values. For relation Student, each leaf node can accommodate up to two tuples; each internal node can hold up to 3 keys and 4 pointers. Relation Student is initially empty and its B+-tree has been constantly changing when the following 12 records with keys 37, 2, 54, 50, 41, 58, 56, 19, 67, 69, 63, 21 are inserted sequentially to the relation. Please draw the snapshots of the B+-tree of Student after the insertion of 54, 58, 56 and 21, respectively. [12 marks]

fi = 4, fl = 2,

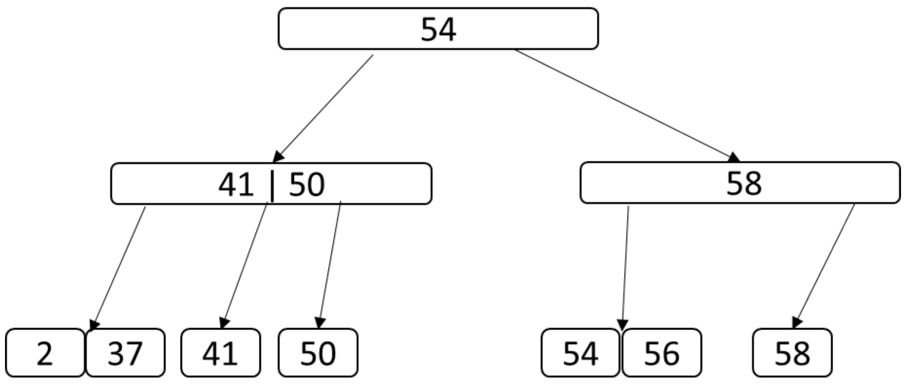
After insertion of 54:



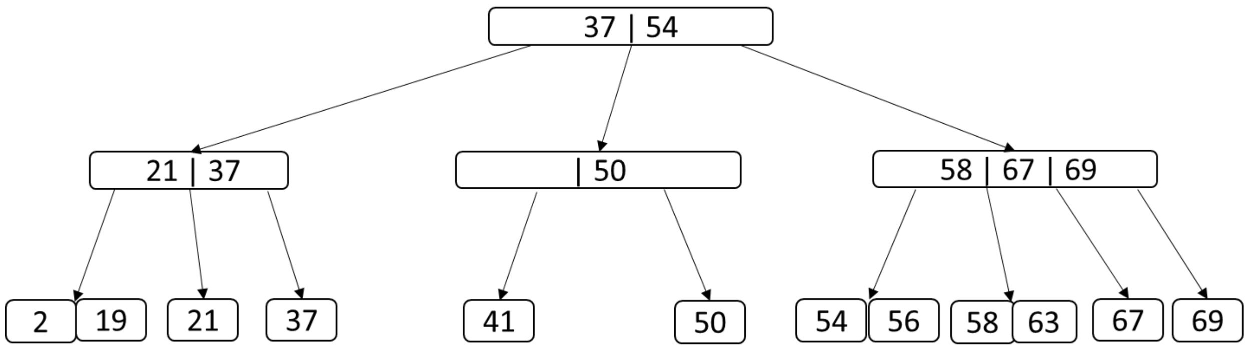
After insertion of 58:



After insertion of 56:



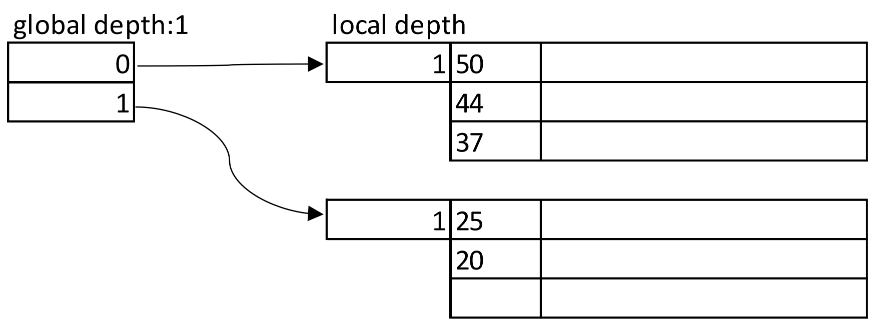
After insertion of 21:



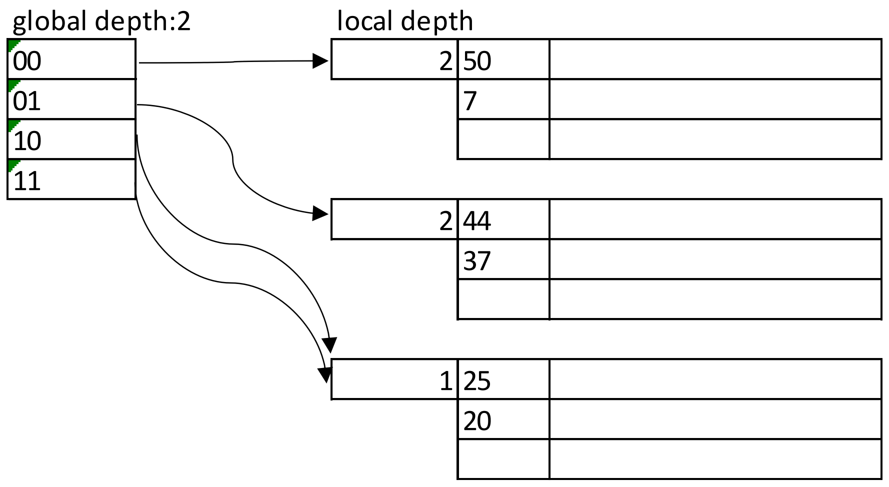
5. Suppose that we are using extendible hashing to manage an initially empty relation Student. Records with the following key values 50, 44, 25, 20, 37, 7, 51, 49, 18, 69 are inserted to the relation sequentially. The binary code of each of the above keys can be found in the following table. Each block can hold up to three records. Show the extendible hashing structure after the insertion of 37, 7, 69, respectively. Please include the global and local depths in the snapshots. [10 marks]

Key value	Hash code
50	001
44	010
25	100
20	110
37	010
7	000
51	010
49	000
18	100
69	110

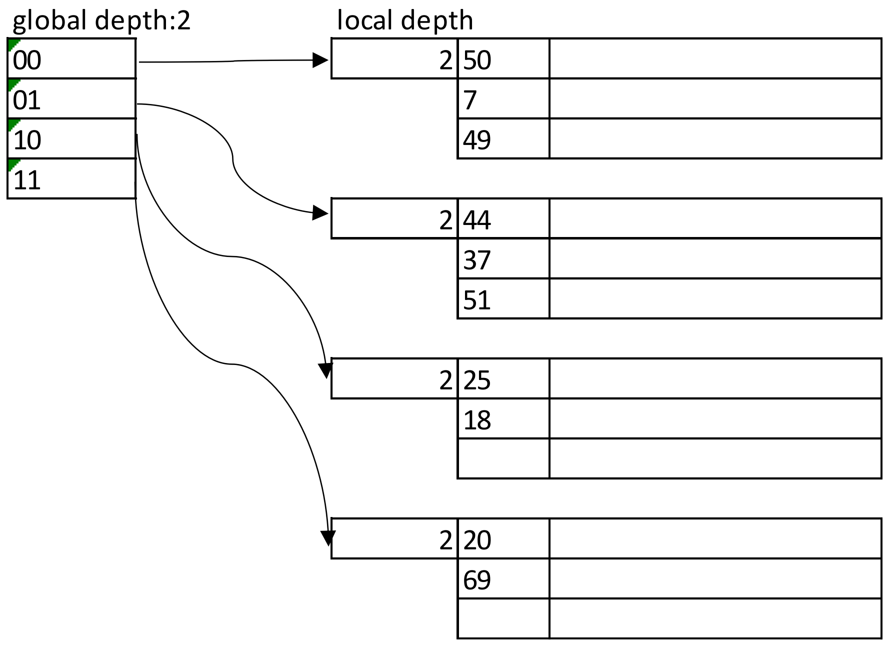
After insertion of 37:



After insertion of 7:



After insertion of 69:



6. Consider the join r ▷◁ s of two relations r and s whose common attribute set is {A}. Physically, r is stored on 25 blocks and s on 21 blocks on the disk, tuples in both relations are unordered. Assume that the buffer pool allocated for carrying out the join has 3 frames. Compare block nested-loop join against merge join in facilitating r ▷◁ s by analyzing their I/O costs. The I/Os for exporting the final joined results to the disk is called the reporting cost, which shall be excluded from the calculation of the I/Os of r ▷◁ s, because the reporting costs of both nested-loop join and merge join cancel each other out in the comparison. Specifically,

(a) Compute the # of I/Os, excluding the reporting cost, engaged by block nested-loop join. [3 marks]

Nr = 25, Ns = 21, m=3,

Because Nr>Ns, we use r as inner relation and s as the outer relation. Total I/O spent will be

Ns + ceiling(Nr/(3-2))*Ns = 21+ 25*21= 546.

(b) Compute the # of I/Os, excluding the reporting cost, engaged by merge join in the worst-case and best-case scenarios, respectively. A scenario is an instantiation of the tuples in r and s, especially on attribute A.

Merge join can be done only on sorted lists. Therefore, we need to sort relation r and s based on attribute A first. The total I/Os spent is 2N * runs. According to $\lceil \log_{m-1} \lceil n/m \rceil \rceil + 1)$, runs of sorting r= $\lceil \log 2 \lceil 25/3 \rceil \rceil$ +1=5, runs of sorting s= $\lceil \log 2 \lceil 21/3 \rceil \rceil$ +1=4. The total I/O for sorting r is 2*25*5=250 I/Os; and 2*21*4 = 168 I/Os

The worst-case is that attributes A in r and s is not the primary key, we need to go back to the scenario of block nested-loop join. According to (a) the total I/O spent for block nested-loop join is 546, plus sorting I/Os for both r and s, in total 546+250+168 = 964 I/Os.

The best-case is attributes A in r is the primary key and unique, we only need to scan s to find the matching tuples. The I/O for scanning will be 21 (# of Ns), plus sorting I/Os for both r and s, in total 21+250+168 = 439 I/Os.