Assignment 8
Jashjeet Singh Madan

Part 1

Question 6

Consider the following parameters:

block size = 4096 bytes
block-address size = 9 bytes
block access time = 10 ms (micro seconds)
record size = 200 bytes
record key size = 12 bytes

Assume that there is a B+-tree, adhering to these parameters, that indexes 108 million records on their primary key values.

(a) Specify (in ms) the minimum time to retrieve a record with key *k* in the B+-tree provided that there is a record with this key.

We need to find out the largest integer n, such that

n <= block size – (|block-address size)/ (block-address size + record key size)

n <= 4096 – (9 / (9 + 12))

⇨ n=194
⇨ new n= previous n + 1 (extra block access for recording data file)

Minimum time:
$([\log_{195}(10^8)] + 1)$* 10 ms = (4+1) * 10 ms

(b) Specify (in ms) the maximum time to retrieve a record with key *k* in the B+-tree.

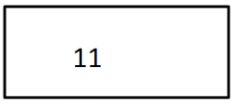
Maximum time will be taken when the branching factor is minimum, i.e. 2.

[194/2] + 1 = 98
The height of such a tree will be $[\log_{98}(10^8/2)]$. Which is equal to 4.

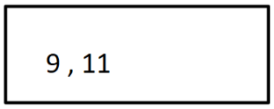
We need an additional block to access the record. So,the total time becomes (4 + 1 + 1)*10 ms = 60 ms.

Question 7
Drawing the tree given in the question

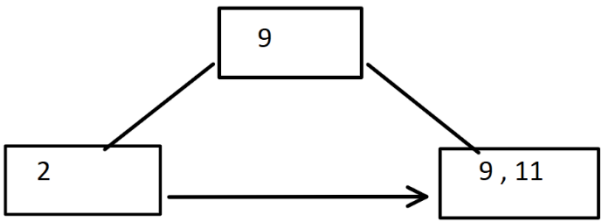
Insert 11



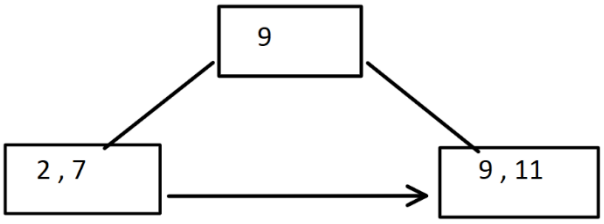
Insert 9



Insert 2

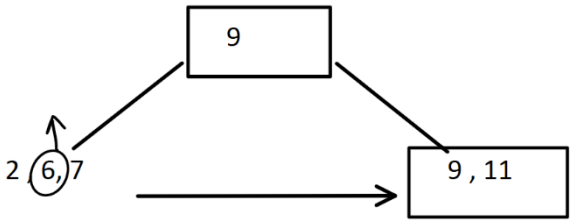


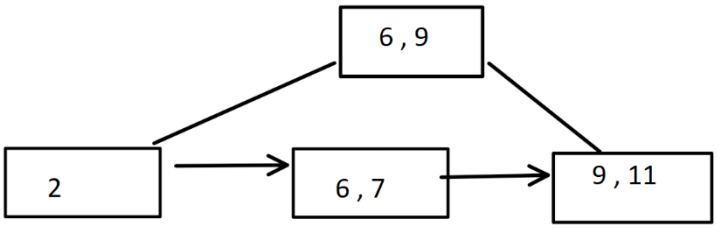
Insert 7



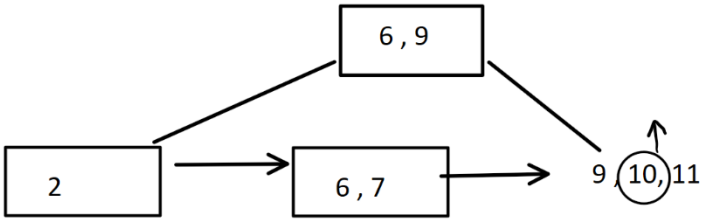
Part a

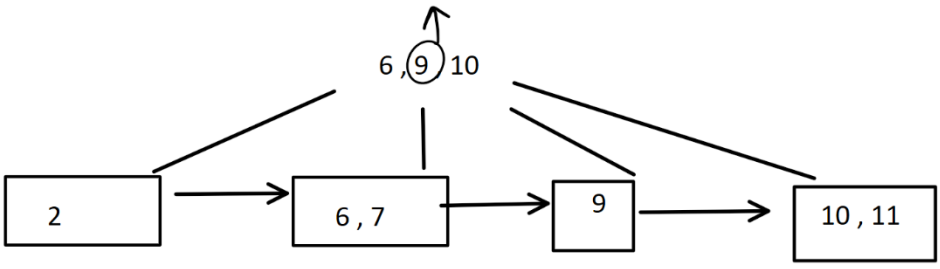
Insert 6

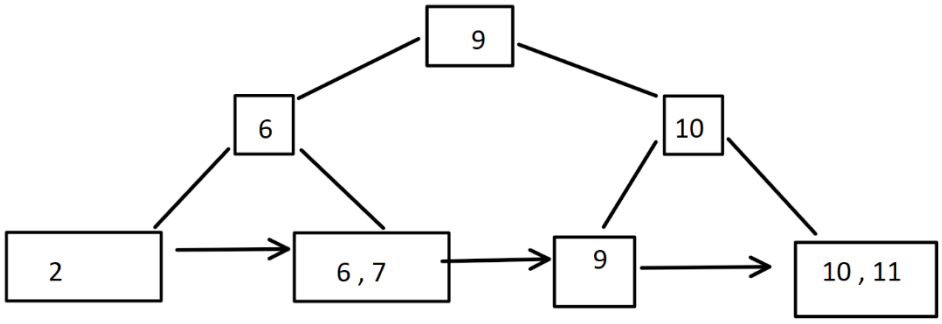




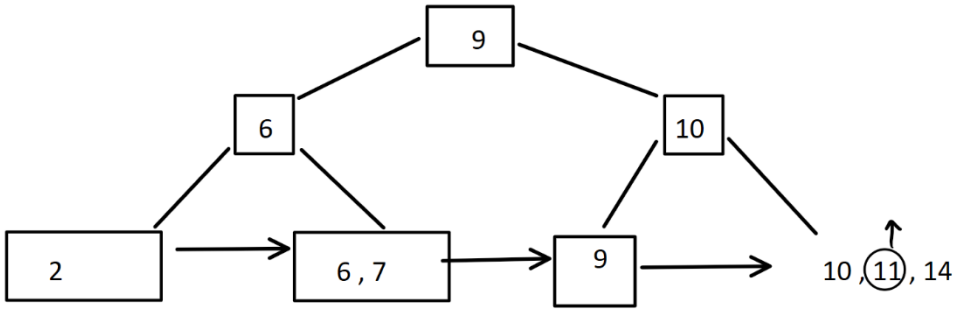
Insert 10

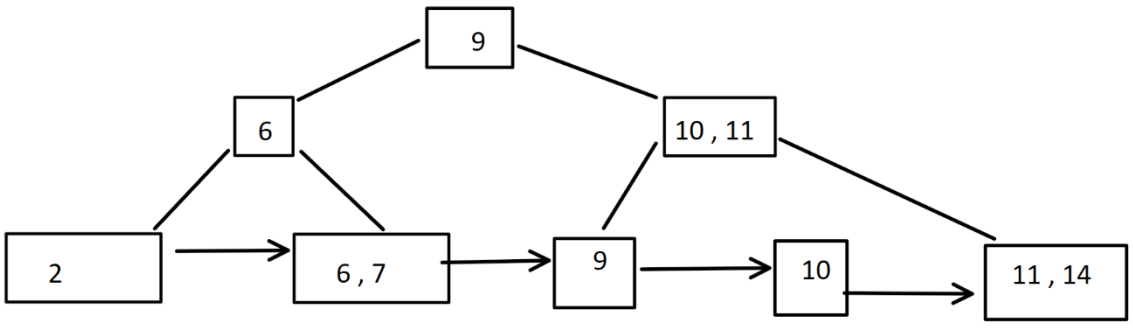




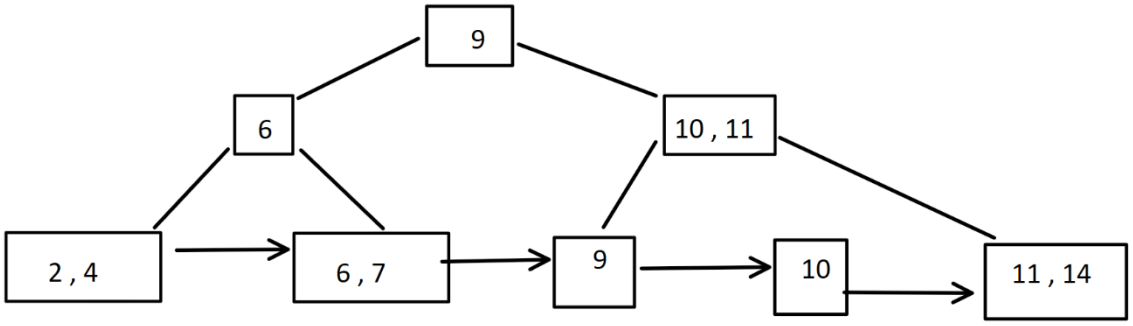


Insert 14



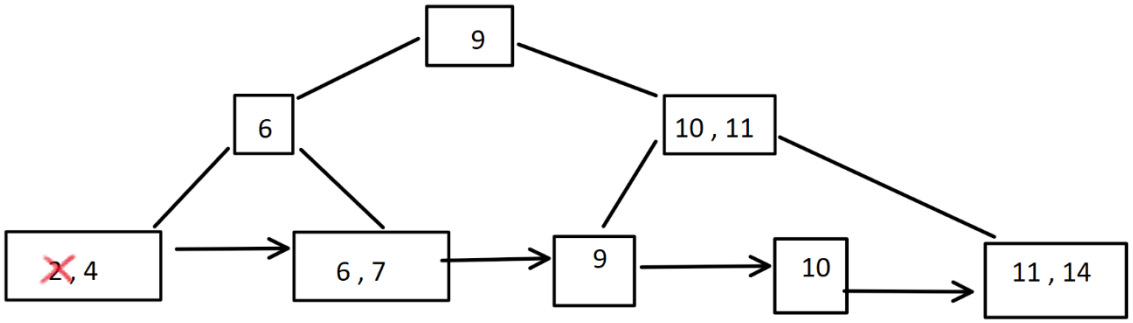


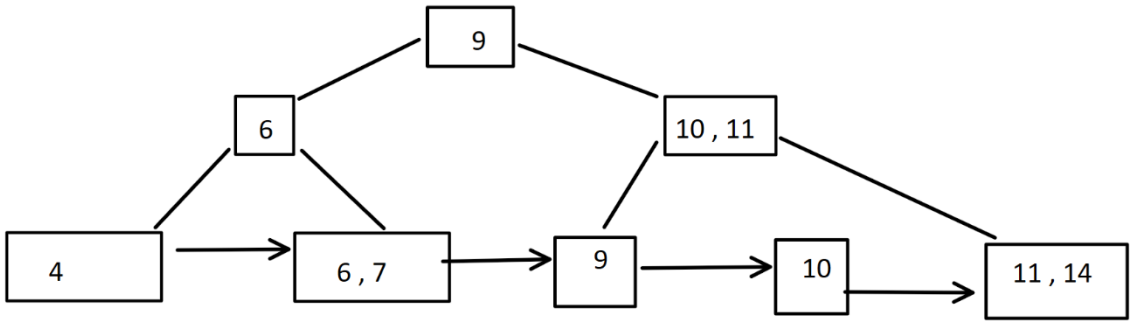
Insert 4



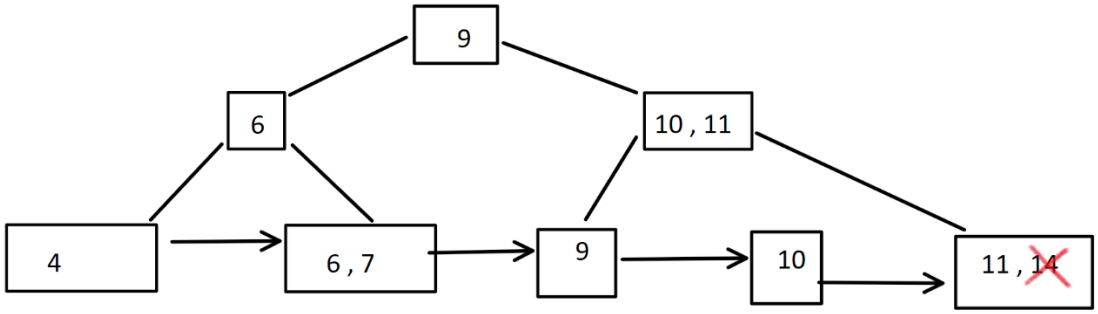
Part b

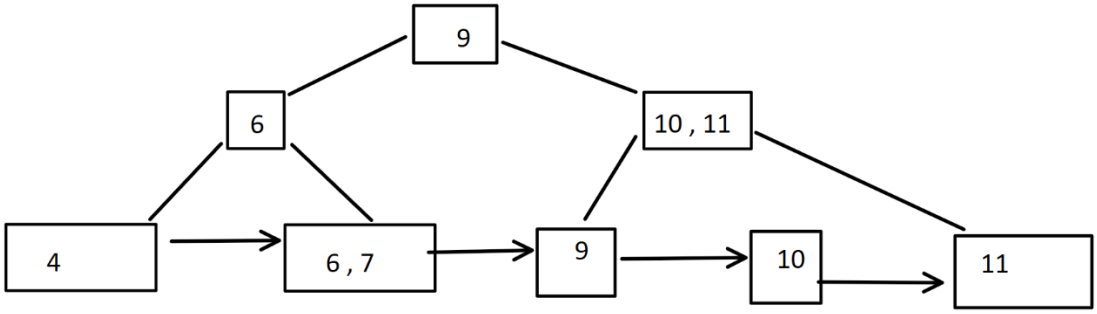
Delete 2



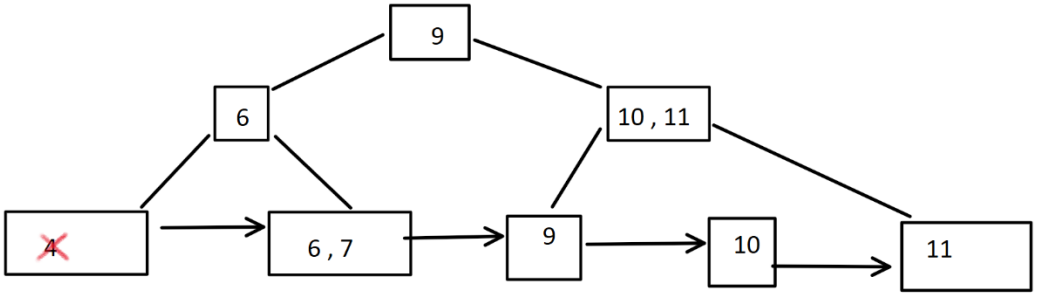


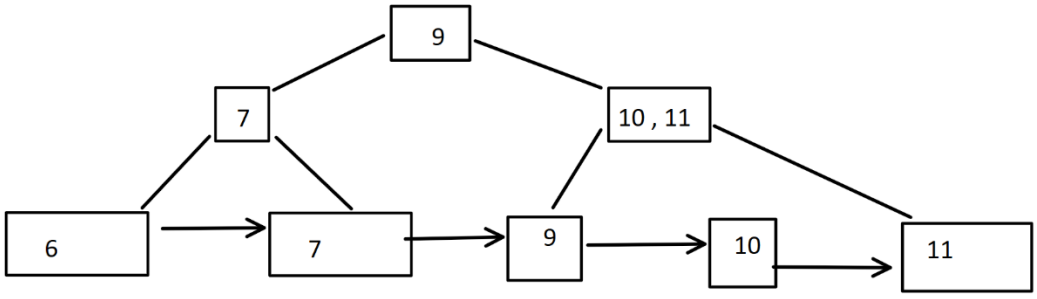
Delete 14



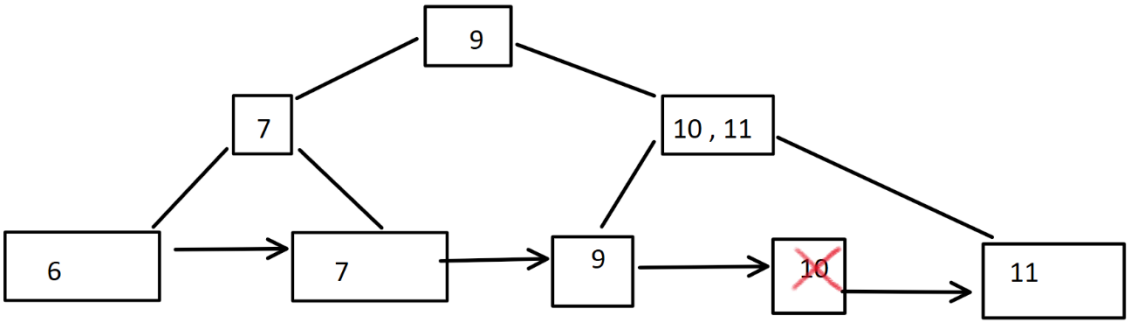


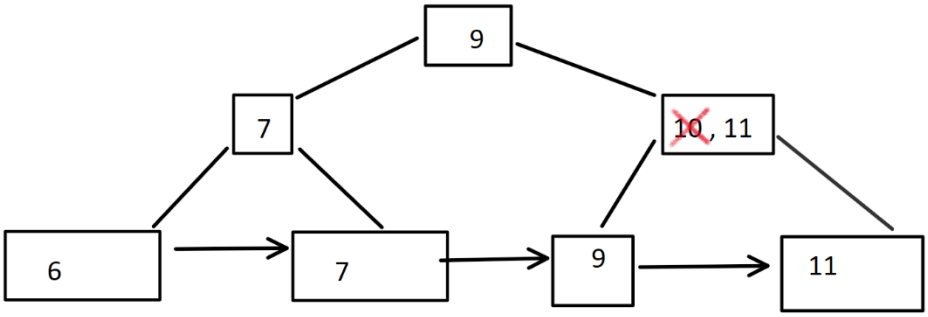
Delete 4

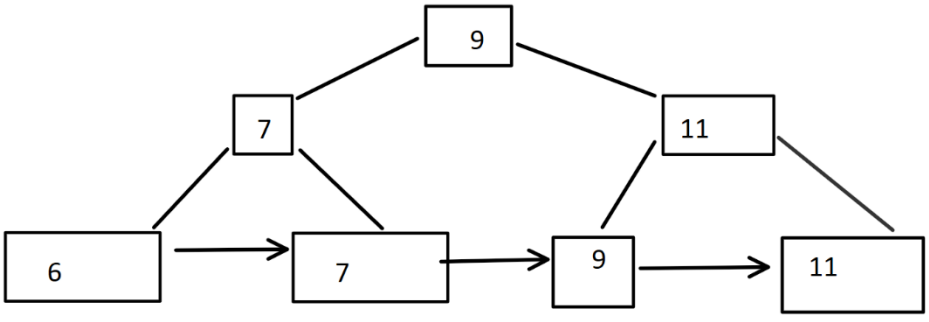




Delete 10





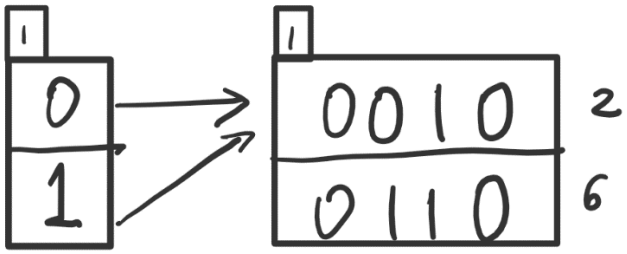


Question 8

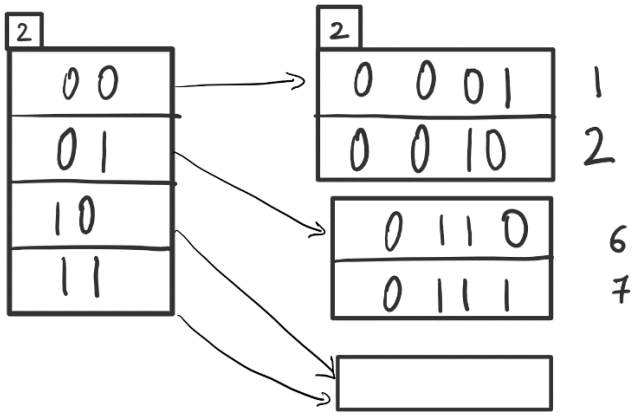
we need 4 bits to represent integers from 0 to 9.

0 -> 0000
1 -> 0001
2 -> 0010
3 -> 0011
4 -> 0100
5 -> 0101
6 -> 0110
7 -> 0111
8 -> 1000
9 -> 1001

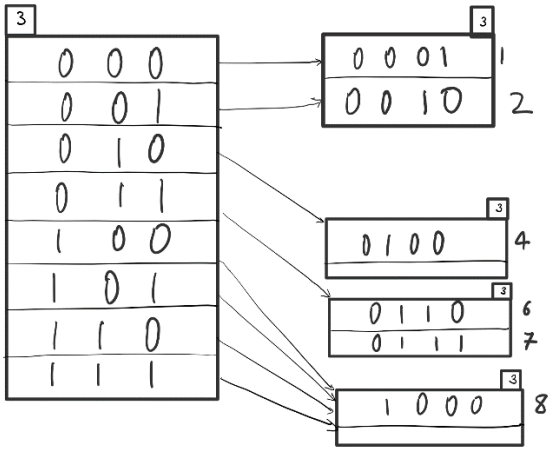
Insert 2,6



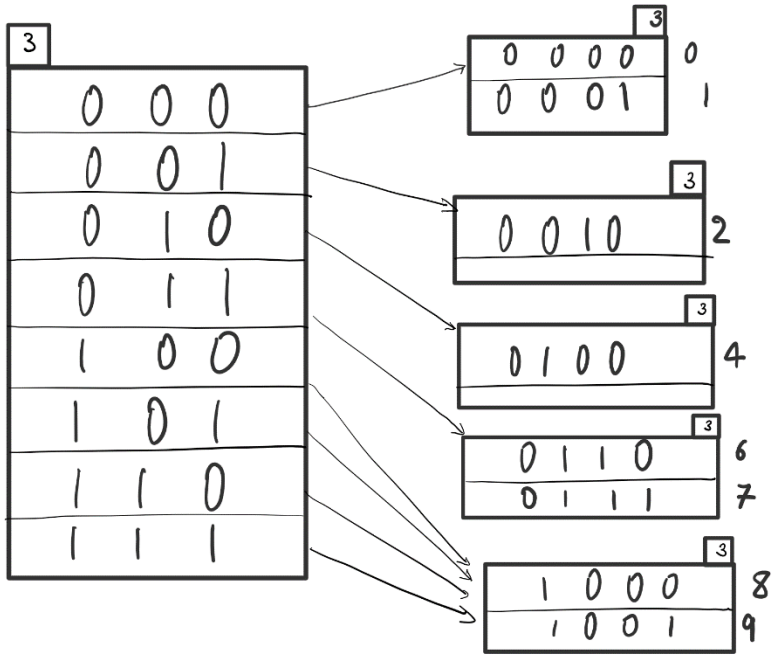
Insert 1,7



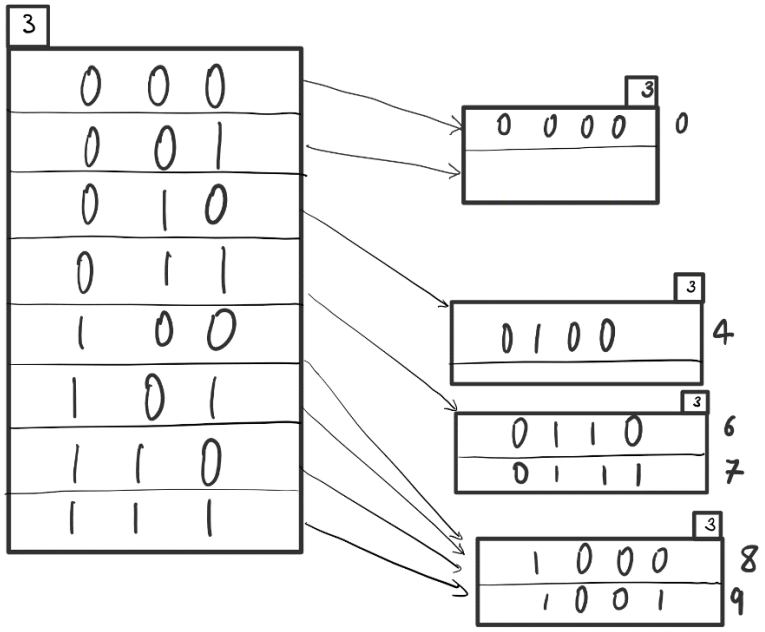
Insert 4,8



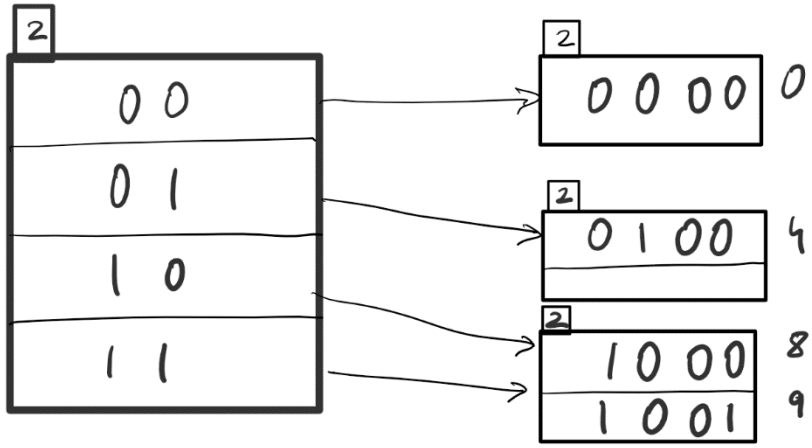
Insert 0, 9



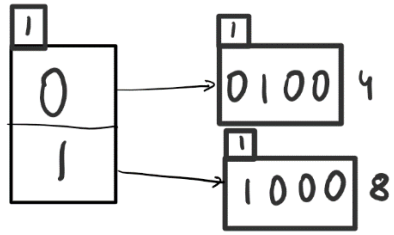
Delete 1,2



Delete 6,7



Delete 0,9



Question 9

Let *R*(*A, B*) and *S*(*B, C*) be two relations and consider their natural join *R ./ S*.

Assume that *R* has 1,500,000 records and that *S* has 5*,* 000 records. Furthermore, assume that 30 records of *R* can fit in a block and that 10 records of *S* can fit in a block. Assume that you have a main-memory buffer with 101 blocks.

(a) How many block IO's are necessary to perform *R ./ S* using the block nested-loops join algorithm? Show your analysis.

R = 1,500,000
S = 5,000.

30 records of R can fit in a block and that 10 records of S can fit in a block, therefore
B(R) = 1500000/30 = 50000
B(S) = 5000/10 = 500

B(S) + (B(S)*B(R))/(100) = 500 + 500*50000/100 = 250500

(b) How many block IO's are necessary to perform *R ./ S* using the sort-merge join algorithm? Show your analysis.

Merge sorting R takes 2*B(R) * ceil($\log_{100}$(B(R))) = 2*50000 * ceil(log(50000)) = 300000

Merge sorting S takes 2*B(S) * ceil($\log_{100}$(B(S))) = 2*500 * ceil(log(500)) = 2000

Merge of these sorted files = B(R) + B(S) = 50000 + 500 = 50500.

Total = 300000 + 50500 + 2000 = 352500.

(c) Repeat question 9b under the following assumptions. Assume that there are *p* different *B*-values and that these are uniformly distributed in *R* and *S*. Observe that to solve this problem, depending on *p*, it may be necessary to perform a block nested-loop join per occurrence of a *B*-value.

Merge sorting R takes 2*B(R) $\log_{100}$(B(R)) = 2*50000 * ceil(log(50000)) = 300000

Merge sorting S takes 2*B(S) $\log_{100}$(B(S)) = 2*500 * ceil(log(500)) = 2000

P = 1
R and S do not fit in the buffer, so we will do a block nested-loop join. S is smaller than R, so it will require B(S) + B(R)B(S) 100 block accesses. So an additional of 250500 block accesses.

P = 2
With 2 B-values, we will need 2 block nested-loop joins for 50000/2 = 25000 and 500/2 = 250 file size. Since, S is smaller, the total time is 250+ 250*25000/100 = 62750. For 2 block accesses, it will be 2*62750 = 125500.

P = 3
With 3 B-values, we will need 3 block nested-loop joins for 50000/3 = 17000 (approx) and 500/3 = 170 (approx) file size. Since, S is smaller, the total time is 170+ 170*17000/100 = 29070. For 3 block accesses, it will be 3*29070 = 87210.

We can further continue for P=4,5,6 . . .

(d) How many block IO's are necessary to perform *R ./ S* using the hash-join algorithm? Show your analysis.

If the buffer is large enough to hold all the records, then the number of blocks required are:
3*(B(R) + B(S)) = 3*(50000+ 50) = 150150.

Part 3

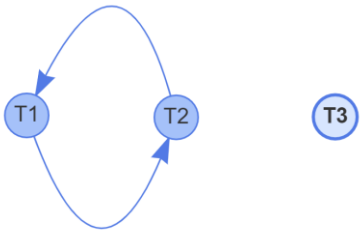
Question 10

State which of the following schedules S1, S2, and S3 over transactions T1, T2, and T3 are conflict-serializable, and for each of the schedules that is serializable, given a serial schedule with which that schedule is conflict equivalent.

(a) S1 = R1(x)R2(y)R1(z)R2(x)R1(y)

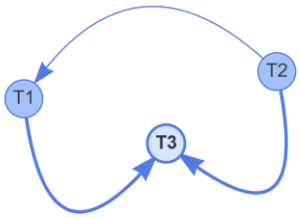
It is conflict serializable, as there are no cycles.
Conflict equivalent schedule: R1(x) R1(z) R1(y) R2(x) R2(y)

(b) S2 = R1(x)W2(y)R1(z)R3(z)W2(x)R1(y).



It is not conflict serializable, as there are cycles.

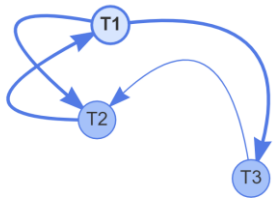
(c) S3 = R1(z)W2(x)R2(z)R2(y)W1(x)W3(z)W1(y)R3(x).



It is conflict serializable, as there are no cycles.
Conflict equivalent schedule: R1(z) W2(x) R2(y) R2(z) W1(x) W3(z) W1(y) R3(x)

Question 11

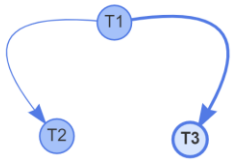
Give 3 transactions T1, T2, T3 and a schedule S on these transactions whose precedence graph (i.e. serialization graph) consists of the edges (T1, T2), (T2, T1), (T1, T3), (T3, T2).



R1(A) R2(B) W3(A) W1(B) W2(A)

Question 12

Give 3 transactions T1, T2, and T3 that each involve read and write operations and a schedule S that is conflict-equivalent with all serial schedules over T1, T2, and T3.



The above figure is represented by: R1(x) R2(x) W2(x) R1(y) W1(y) R3(y) W3(y)

There are two possible conflict-equivalent serial schedules over T1, T2, and T3. As T1 should execute before both T2 and T3, and the order or T2 and T3 does not matter, the two combinations are:

Combination 1: T1 T2 T3
T1: R1(x) R1(y) W1(y)
T2: R2(x) W2(x)
T3: R3(y) W3(y)

Combination 2: T1 T3 T2
T1: R1(x) R1(y) W1(y)
T3: R3(y) W3(y)
T2: R2(x) W2(x)

Since there are 3 transactions, T1, T2, and T3, there are 3!=6 ways to arrange them. But only the above two are conflict equivalent, the other four have conflicts.
T1 T2 T3
T1 T3 T2

T2 T1 T3
T2 T3 T1
T3 T1 T2
T3 T2 T1

Question 13

Consider the following transactions:

T1: read(A);
read(B);
if A = 0 then B := B+1;
write(B).

T2: read(B);
read(A);
if B = 0 then A := A+1;
write(A).

Let the consistency requirement be A = 0 ∨ B = 0, and let A = B = 0 be the initial values.

(a) Show that each serial schedule involving transaction T1 and T2 preserves the consistency requirement of the database.

There are two possible executions: T1 T2 and T2 T1.

Case 1	A	B
Initially	0	0
After T1	0	1
After T2	0	1

A = 0 ∨ B = 0 ≡ T ∨ F = T

Case 2	A	B
Initially	0	0
After T2	1	0
After T1	1	0

A = 0 ∨ B = 0 ≡ F ∨ T = T

In both the cases consistency is met.

(b) Construct a schedule on T1 and T2 that produces a non-serializable schedule.

read1(A);
read2(B);
read2(A);
read1(B);

if A = 0 then B := B+1;
if B = 0 then A := A+1;
write2(A);
write1(B) ;

(c) Is there a non-serial schedule on T1 and T2 that produces a serializable schedule. If so, give an example

No, there isn't a non-serial schedule that produces serializable schedule.

R1(A) conflicts with W2(A) and R2(B) conflicts with W1(B). We will get a cycle whether we start with T1 or T2.

If we begin with R1(A), then we will execute T1, T2.
If we begin with R2(B), then we will execute T2, T1.