27 February 2018 17:43

Quiz 1

Part 1.1 Disk Organization

Question 1.1.1

Disk Organisation :

Q 1.1.1 no. of tracks per surface = 2000
no. of sectors per track = 50
5 double-sided platter .
avg. seek time = 10 msec.
block size = 1024 bytes
file contain 100,000 records of 100 bytes

1. How many records fit onto a block ?

Sol^n: No. of records fit onto the block

$$= \frac{1024}{100}$$

$$\simeq 10$$

∴ We can have atmost 10 records in a block .

2. How many blocks are required to store the entire file

Sol^n: There are total 100,000 records and each block can hold 10 records

∴ No. of blocks needed $= \frac{100,00\cancel{0}}{1\cancel{0}}$

$= 10,000$ Ans

3. If the file is arranged sequentially on the disk, how many surfaces are needed?

Sol^n:

∵ Sector size = 512 bytes and block size = 1024 bytes

∴ 1 block contains 2 sectors.

and no. of sectors per track = 50 (given)

So, no. of blocks per track = $\frac{50}{2}$

= 25 blocks/track

Since there are 5 double-sided platter

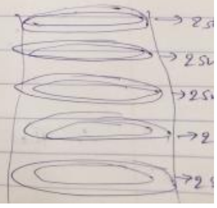
Hence total no. of surfaces = 5×2 = 10

No. of blocks per cylinder = 25 × 10

= 250 blocks

File contains 10,000 blocks, therefore we need more than one cylinders to store this file (10000/250 = 40 cylinders)

ie, 10 surfaces to store this file.



Q 4. How many records of 100 bytes each can be stored using this disk?

Sol^n

capacity of disk = bytes / disk

= ~~bytes / track *~~

Now, bytes / track = bytes / sector * Sector / track

= 512 * 50.

= 25600 bytes

bytes / surface = bytes / track * track / surface

= 25600 * 2000 = 51200000

bytes / disk = bytes / surface * surface / disk

= 51200000 * 10

= 512000000 bytes

∴ capacity of disk = 512 000 000 bytes

≈ 500000 kbytes

≃ 500 000 blocks

1 block contain 10 records

∴ 500000 blocks ——— 10 × 500000

= 5,000,000 records

∴ The disk can store atmost 5,000,000 records.

Q5. If pages are stored sequentially on disk, with page 1 on block 1 of track 1. What page is stored on block 1 of track 1 on the next disk surface?

Sol^n: capacity of track = 25600 bytes
≃ 25000 bytes
≃ 25 K bytes
≃ 25 blocks

∴ Each ~~block~~ track contains 25 blocks.
Therefore it is ~~block~~ page 26 on block 1 of track 1 on the next disk surface

Q6. How would your answer change if the disk were capable of reading and writing from all heads in parallel?

Sol^n: Total no. of surfaces = 10
Since the disk is capable of reading and writing in parallel, we can keep first 10 pages on the block 1 of track 1 of 10 surfaces.
Therefore it is ~~block~~ page 2 on block 1 of track 1 on the next disk surface.

Part 1.2 SQL

Consider the following relations:

• Suppliers(sid:integer, sname:string, address:string)

• Parts(pid:integer, pname:string, color:string)

• Catalog(sid:integer, pid:integer, cost:real)

The key fields are underlined, and the domain of each field is listed after the field name. Therefore sid is the key for Suppliers, pid is the key for Parts, and sid and pid together form the key for Catalog. The Catalog relation lists the prices charged for parts by Suppliers. Write the following queries in SQL statements and in relational algebra expression.

Question 1.2.1. Find the names of suppliers who supply some red part.

SELECT S.sname FROM Suppliers S, Parts P, Catalog C WHERE P.color='red' AND C.pid=P.pid AND C.sid=S.sid

Question 1.2.2. Find the sids of suppliers who supply some red or green part.

SELECT C.sid FROM Catalog C, Parts P WHERE (P.color = 'red' OR P.color = 'green') AND P.pid = C.pid

Question 1.2.3. Find the sids of suppliers who supply some red part or are at 10 West 31st Street.

SELECT S.sid FROM Suppliers S WHERE S.address = '10 West 31st Street' OR S.sid IN (SELECT C.sid FROM Parts P, Catalog C WHERE P.color='red' AND P.pid = C.pid)

Question 1.2.4. Find the sids of suppliers who supply some red part and some green part.

SELECT C.sid FROM Parts P, Catalog C WHERE P.color = 'red' AND P.pid = C.pid AND EXISTS (SELECT P2.pid FROM Parts P2, Catalog C2 WHERE P2.color = 'green' AND C2.sid = C.sid AND P2.pid = C2.pid)

Question 1.2.5. Find pairs of sids such that the supplier with the first sid charges more for some part than the supplier with the second sid.

SELECT C1.sid, C2.sid FROM Catalog C1, Catalog C2 WHERE C1.pid = C2.pid AND C1.sid ≠ C2.sid AND C1.cost > C2.cost

Question 1.2.6. Write only an SQL query that find the pids of parts supplied by at least two different suppliers.

SELECT C.sid FROM Catalog C WHERE EXISTS (SELECT C1.sid FROM Catalog C1 WHERE C1.pid = C.pid AND C1.sid 6≠ C.sid)

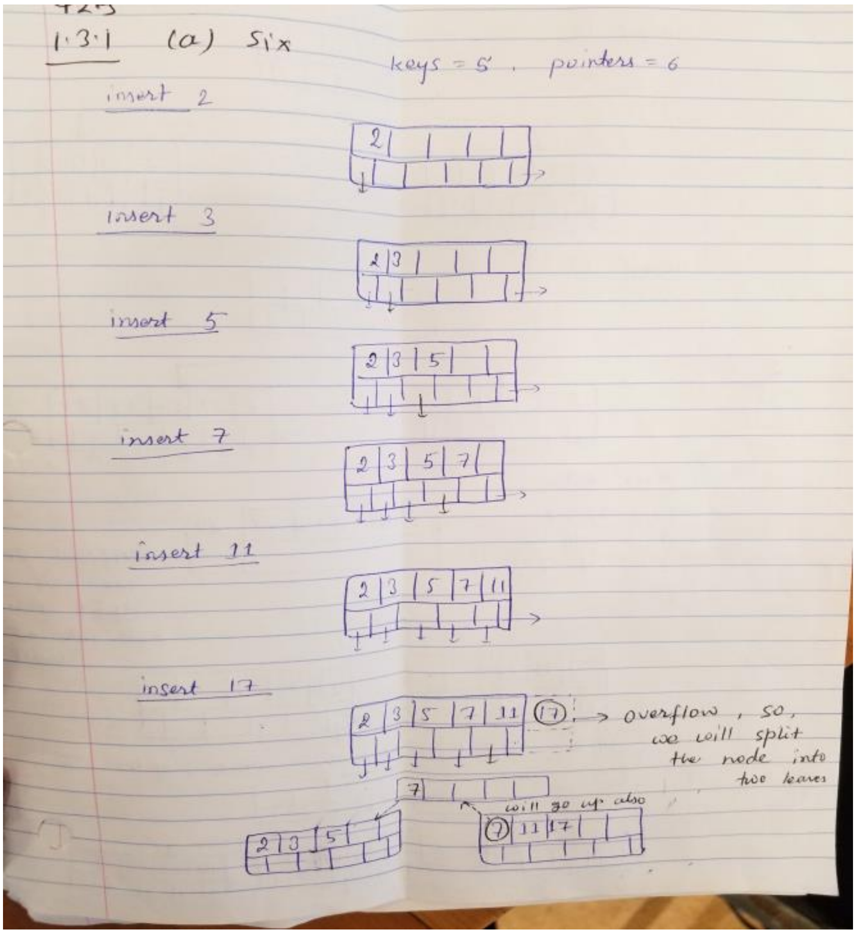
Question 1.2.7. Write only an SQL query that find the pids of the most expensive parts supplied by suppliers named Yosemite Sham.

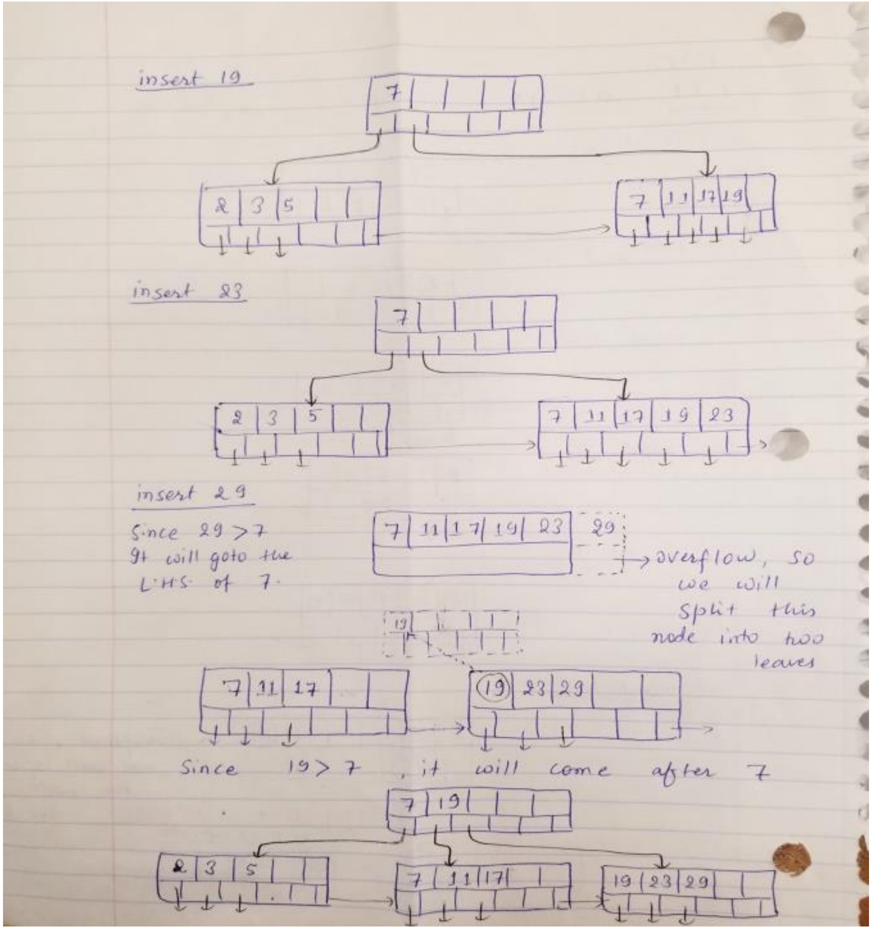
SELECT C.pid FROM Catalog C, Suppliers S WHERE S.sname = ‘Yosemite Sham’ AND C.sid = S.sid AND C.cost ≥ ALL (Select C2.cost FROM Catalog C2, Suppliers S2 WHERE S2.sname = ‘Yosemite Sham’ AND C2.sid = S2.sid)

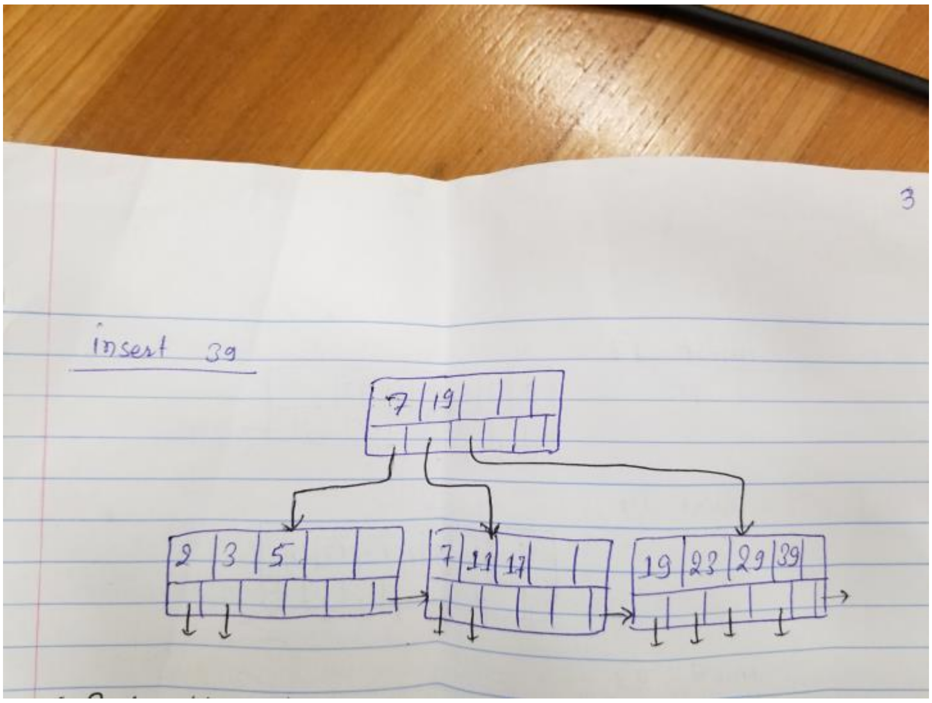
Part 1.3 Index Structures

Question 1.3.1 B+-tree Construction

a. Six




insert 19
7
2 | 3 | 5
7 | 11 | 17 | 19
insert 23
7
2 | 3 | 5
7 | 11 | 17 | 19 | 23
insert 29
Since 29 > 7
It will goto the
L.H.S. of 7.
7 | 11 | 17 | 19 | 23 | 29
overflow, so
we will
split this
node into two
leaves
19
7 | 11 | 17
19 | 23 | 29
Since 19 > 7 , it will come after 7
7 | 19
2 | 3 | 5
7 | 11 | 17
19 | 23 | 29

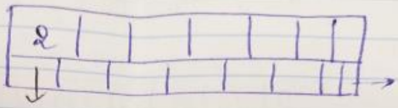

3
insert 39
7 | 19
2 | 3 | 5
7 | 11 | 17
19 | 23 | 29 | 39

b. Eight

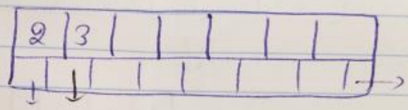
1·3·1 (b) eight

keys = 7
pointer = 8

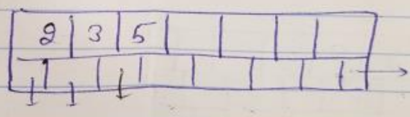
insert 2



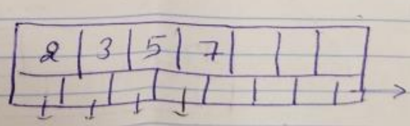
insert 3



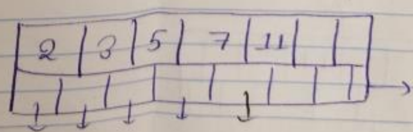
insert 5



insert 7

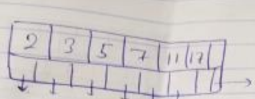


insert 11

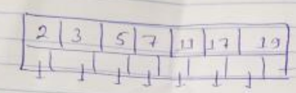


4

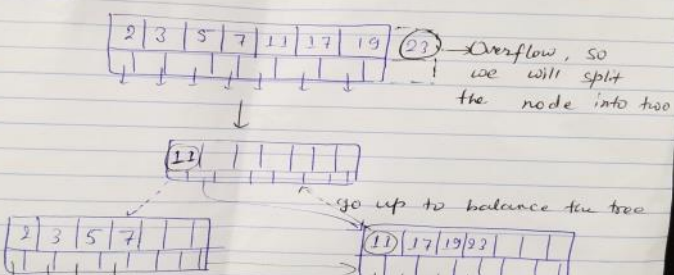
insert 17



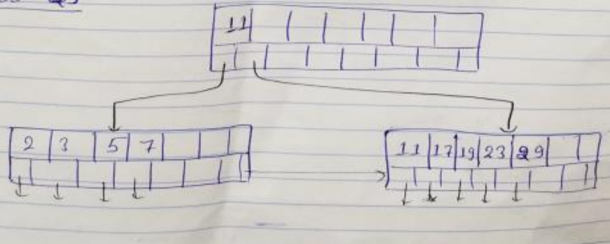
insert 19

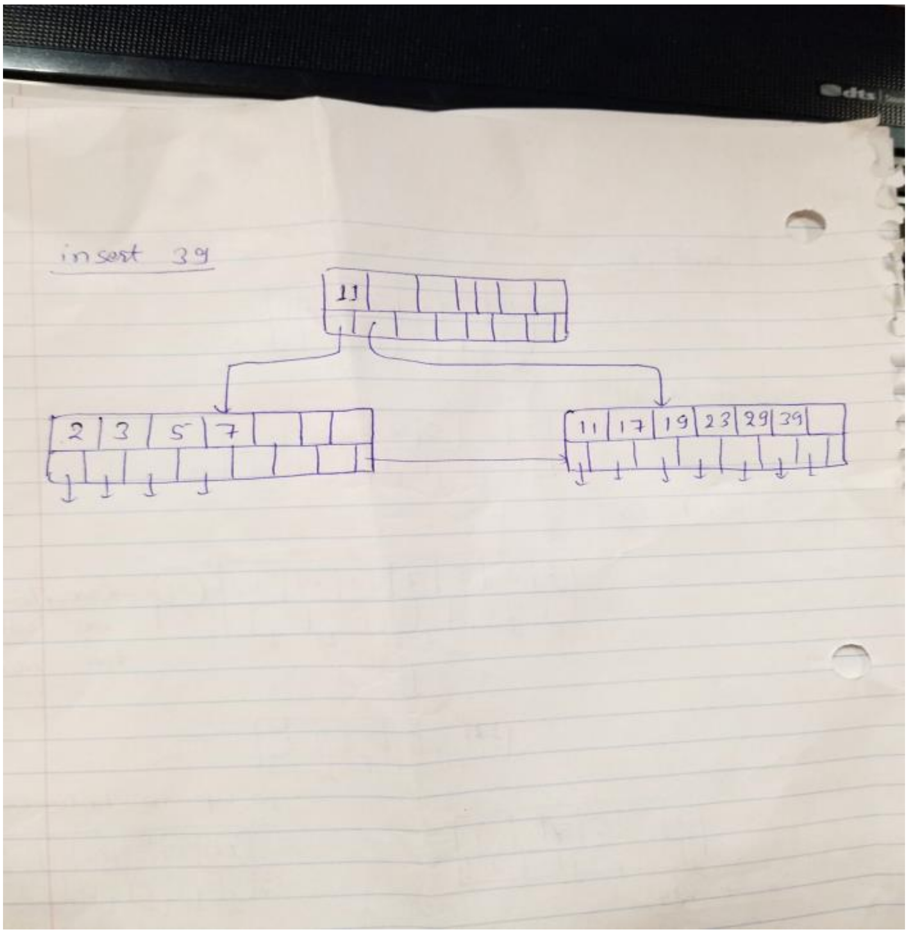


insert 23



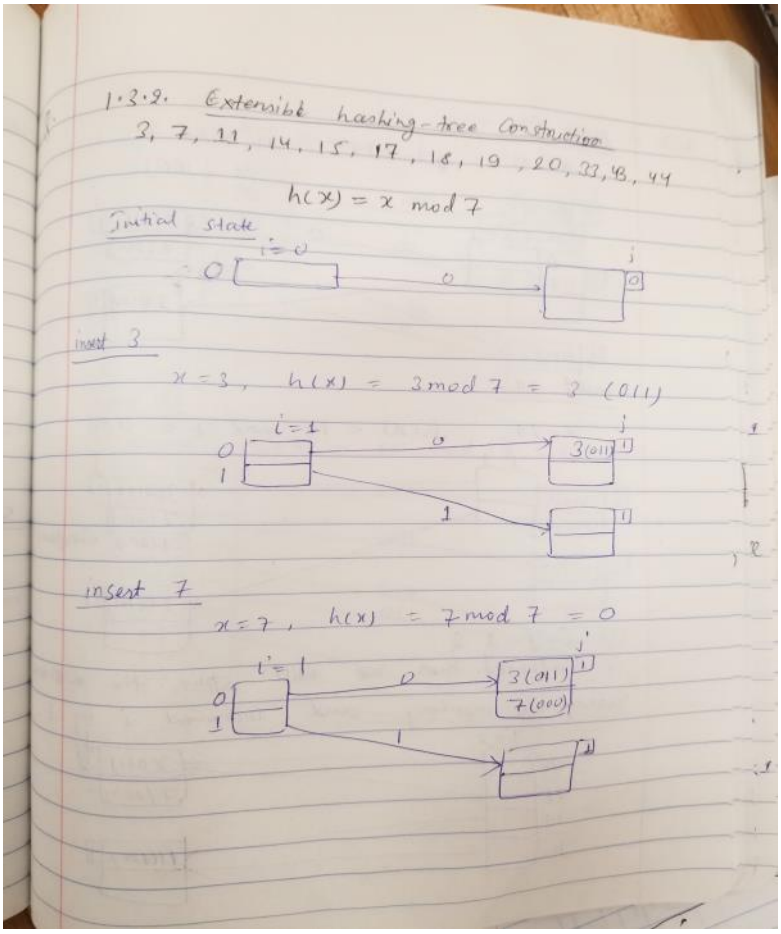
insert 29





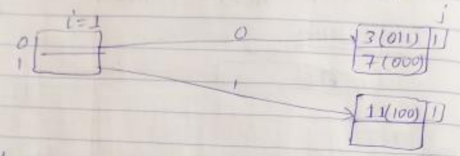
Question 1.3.2 Extendable Hashing-tree Construction

a. 2 records



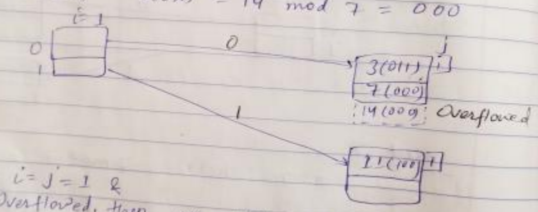
insert 11

$x = 11$, ~~y=~~ $h(x) = 11 \bmod 7$
$= 4 \ (100)$

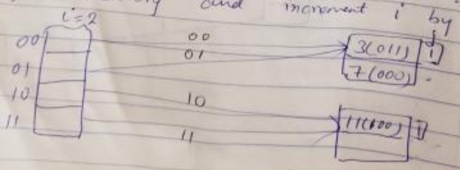


insert 14

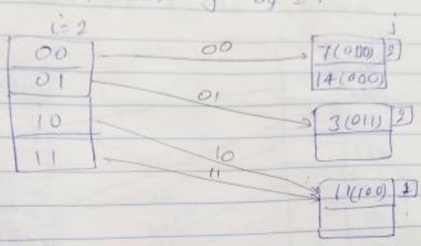
$x = 14$, $h(x) = 14 \bmod 7 = 000$



∵ $i = j = 1$ &
∵ Overflowed, then we will split the disk block array in memory and increment i by 1

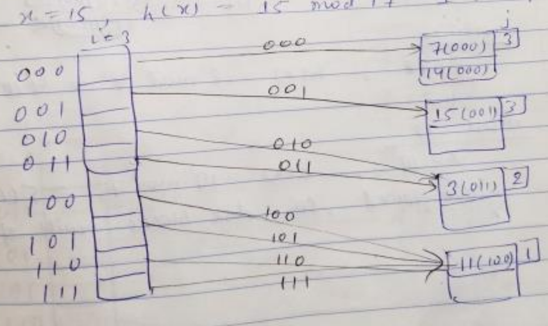


As $j < i$, we will split the block in disk and increase j by 1.



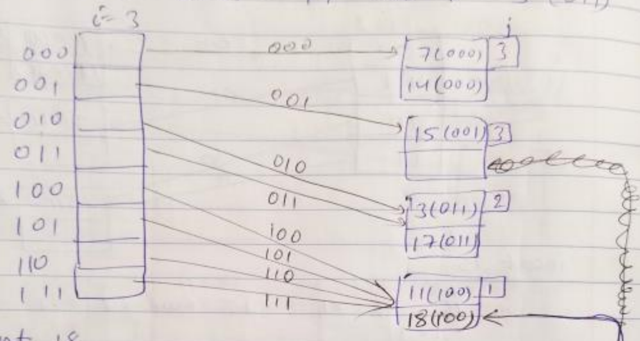
insert 15

$x = 15$, $h(x) = 15 \bmod 17 = 1 = 001$



insert 17

$x = 17$, $h(x) = 17 \bmod 7 = 3\ (011)$



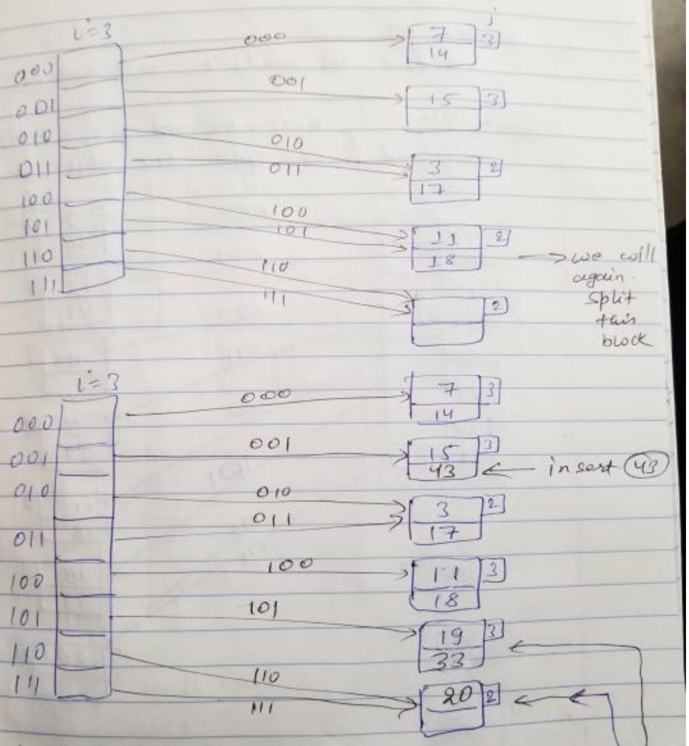
insert 18

$x = 18$, $h(x) = 18 \bmod 7 = 4\ (100)$

insert 19

$x = 19$, $h(x) = 19 \bmod 7 = 5\ (101)$

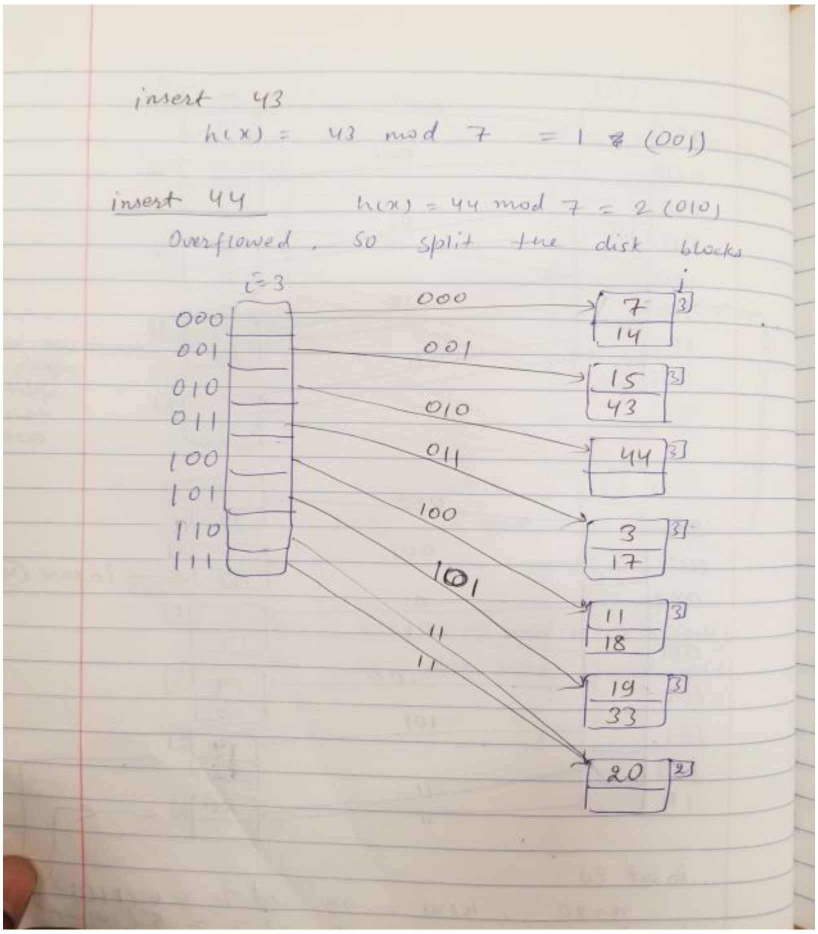
overflowed, so disk blocks will split.



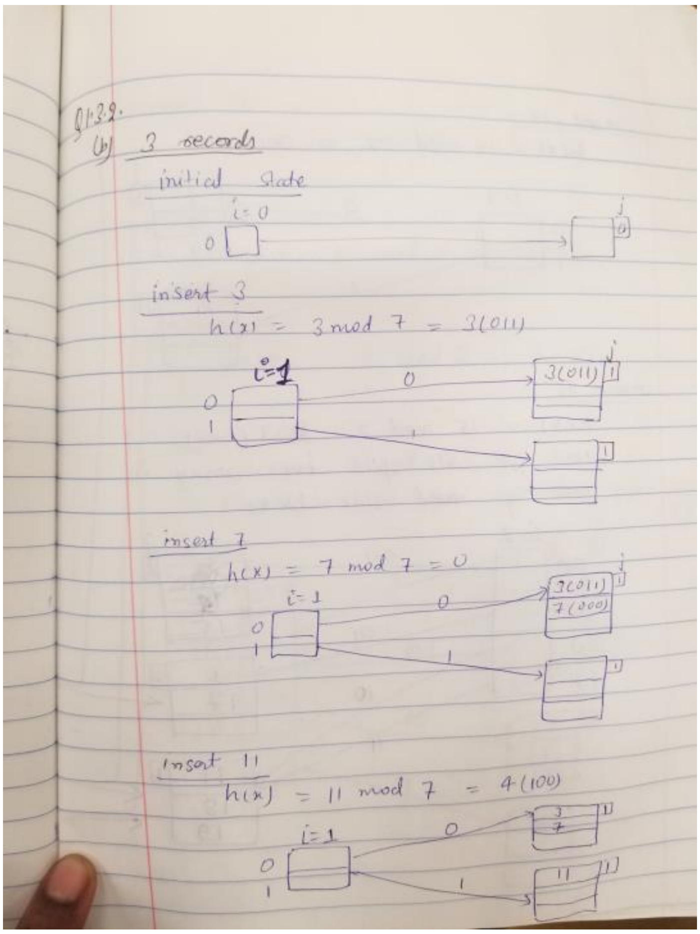
insert 20

$x = 20$, $h(x) = 20 \bmod 7 = 6\ (110)$

insert 33, $h(x) = 33 \bmod 7 = 5\ (101)$

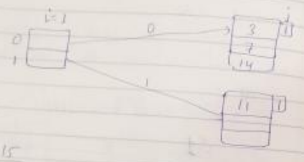

insert 43
h(x) = 43 mod 7 = 1 (001)
insert 44
h(x) = 44 mod 7 = 2 (010)
Overflowed, so split the disk blocks
i=3
000
001
010
011
100
101
110
111
7
14
15
43
44
3
17
11
18
19
33
20

b. 3 records


Q1.3.2.
(b) 3 records
initial state
i=0
insert 3
h(x) = 3 mod 7 = 3(011)
i=1
3(011)
insert 7
h(x) = 7 mod 7 = 0
3(011)
7(000)
insert 11
h(x) = 11 mod 7 = 4(100)
3
7
11

insert 14

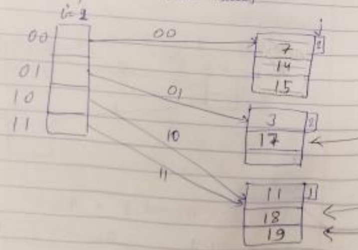
$h(x) = 14 \bmod 7 = 000$



insert 15

$h(x) = 15 \bmod 7 = 1\ (001)$

Now, we will split the array in memory and disk blocks



insert 17

$h(x) = 17 \bmod 7 = 3\ (011)$

insert 18

$h(x) = 18 \bmod 7 = 4\ (100)$

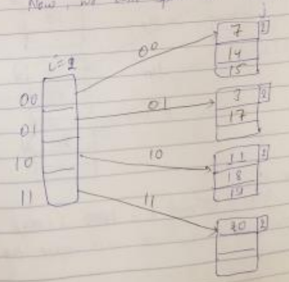
insert 19

$h(x) = 19 \bmod 7 = 5\ (101)$

insert 20

$h(x) = 20 \bmod 7 = 6\ (110)$

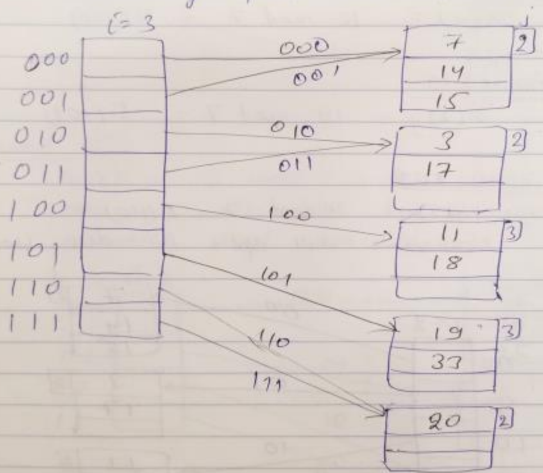
Now, we will split the disk block since (7)



insert 33

$h(x) = 33 \bmod 7 = 5\ (101)$

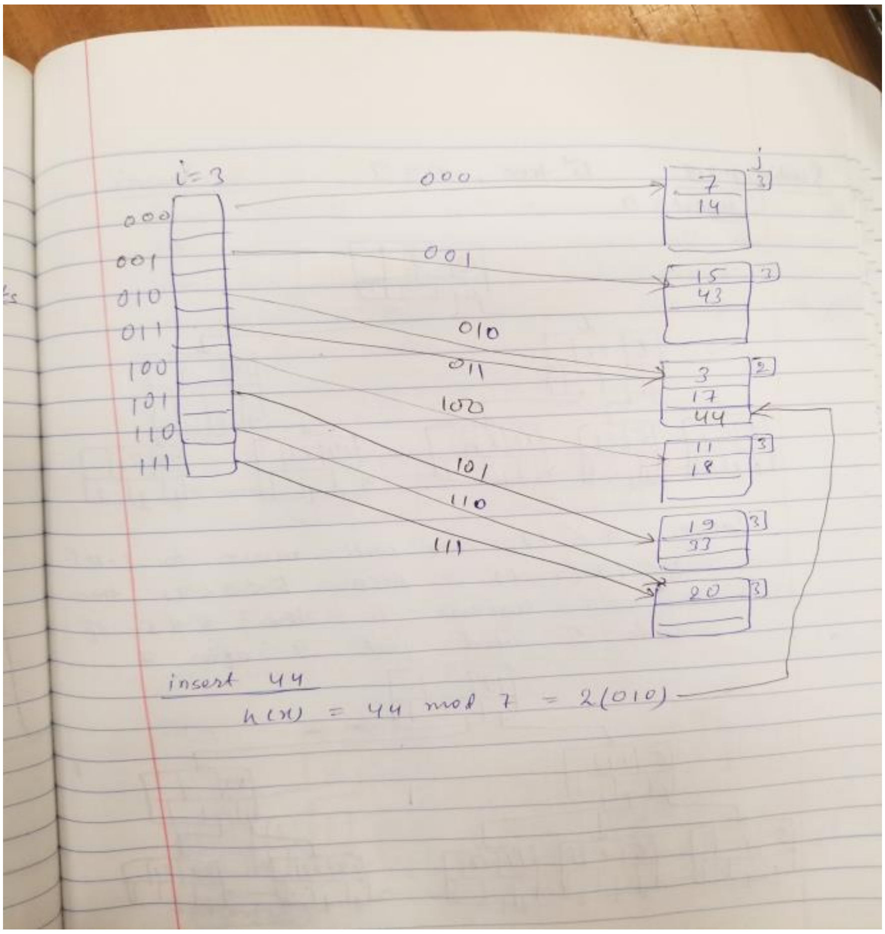
No space, so we will split array in memory first and then disk blocks



insert 43

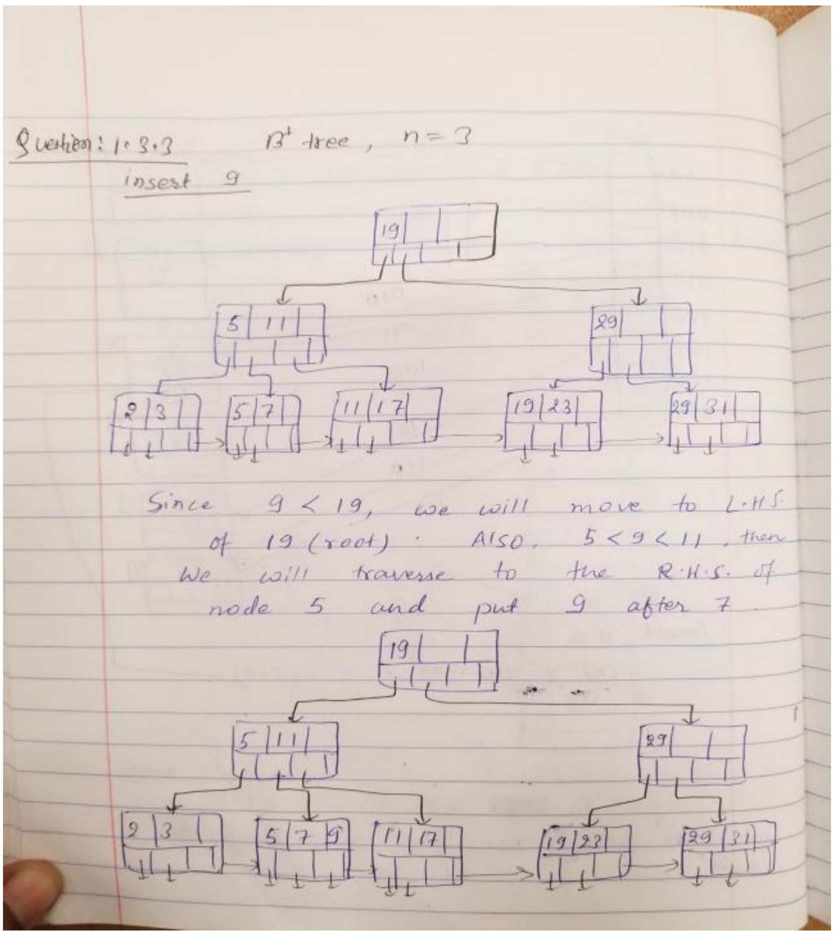
$h(x) = 43 \bmod 7 = 1\ (001)$

No space, so we will split the disk blocks.

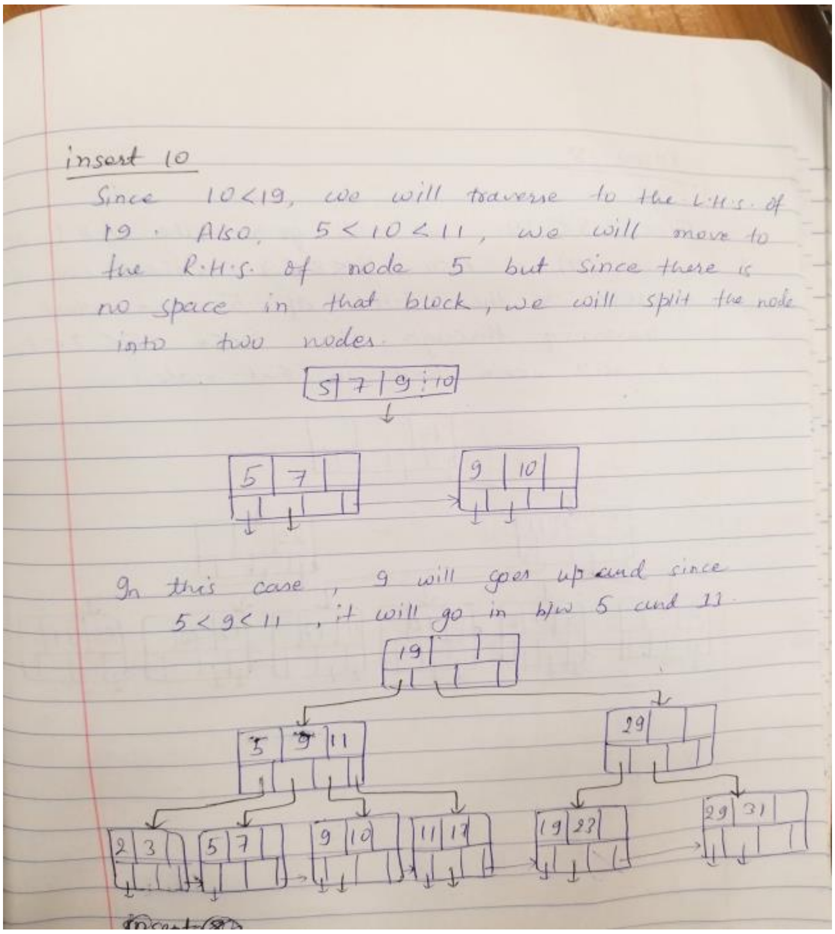


Question 1.3.3 Operations

insert(9)



insert(10)

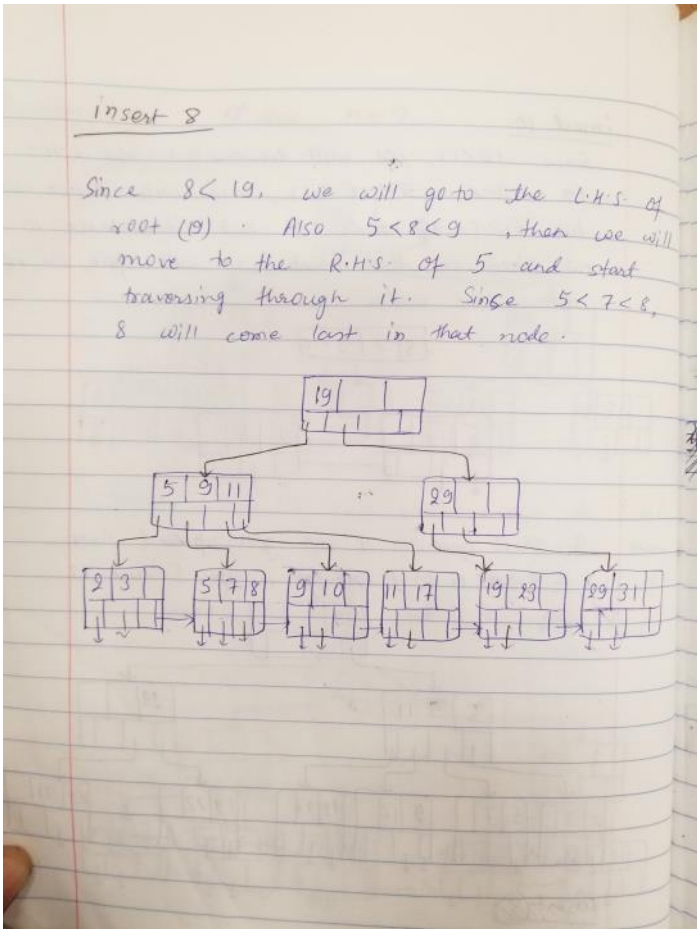


insert 10

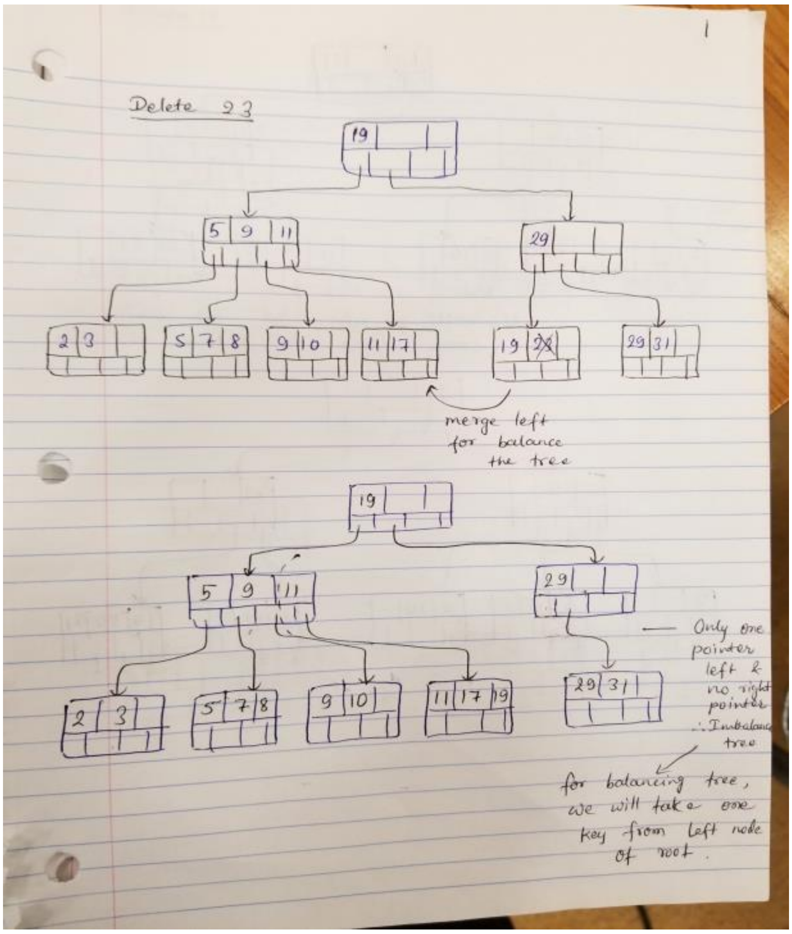
Since $10 < 19$, we will traverse to the L.H.S. of 19. Also, $5 < 10 < 11$, we will move to the R.H.S. of node 5 but since there is no space in that block, we will split the node into two nodes.

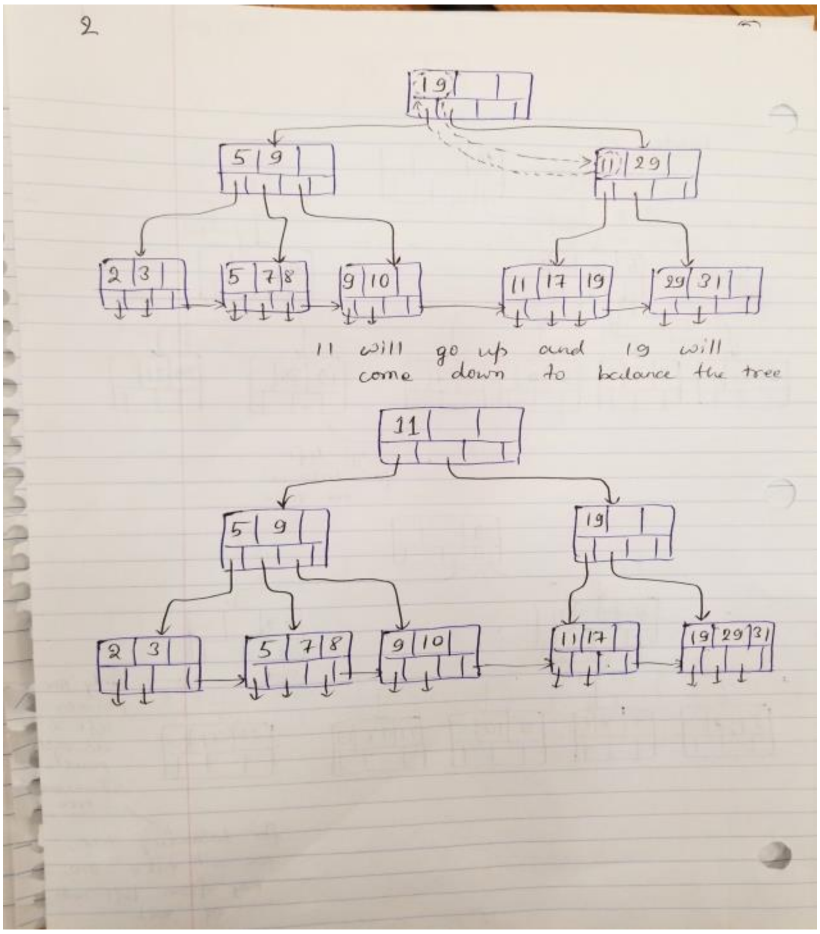
In this case, 9 will goes up and since $5 < 9 < 11$, it will go in b/w 5 and 11.

insert(8)

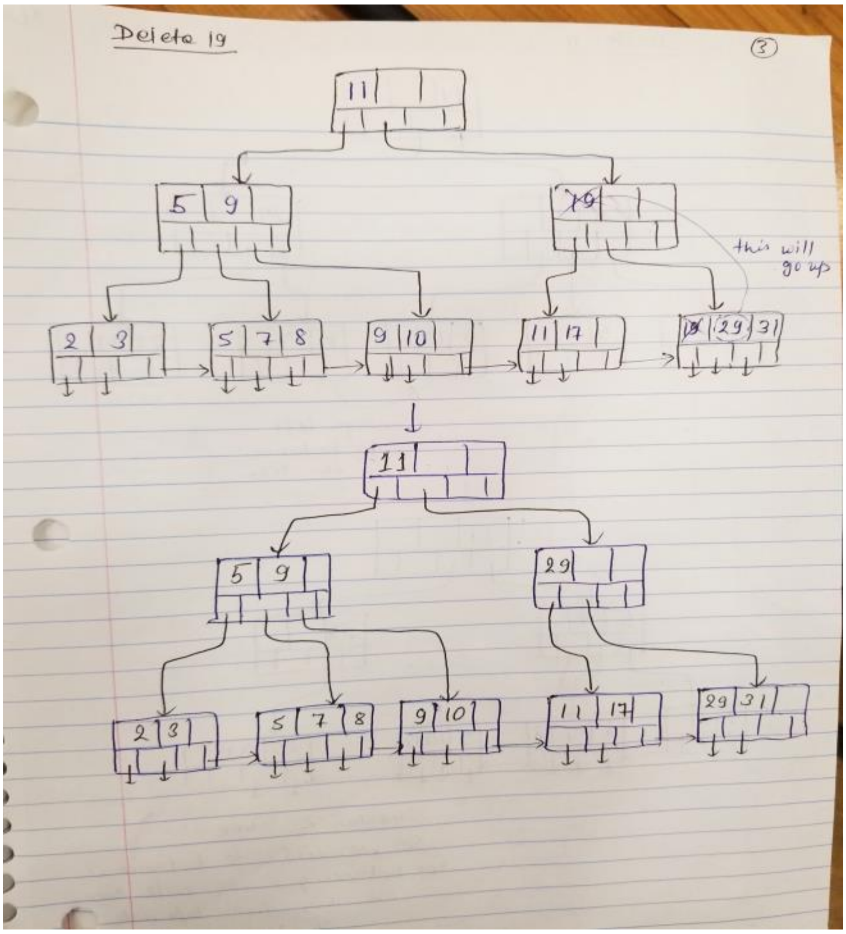

insert 8
Since 8 < 19, we will go to the L.H.S. of root (19). Also 5 < 8 < 9, then we will move to the R.H.S. of 5 and start traversing through it. Since 5 < 7 < 8, 8 will come last in that node.
19
5 9 11
29
2 3
5 7 8
9 10
11 17
19 23
29 31

delete(23)

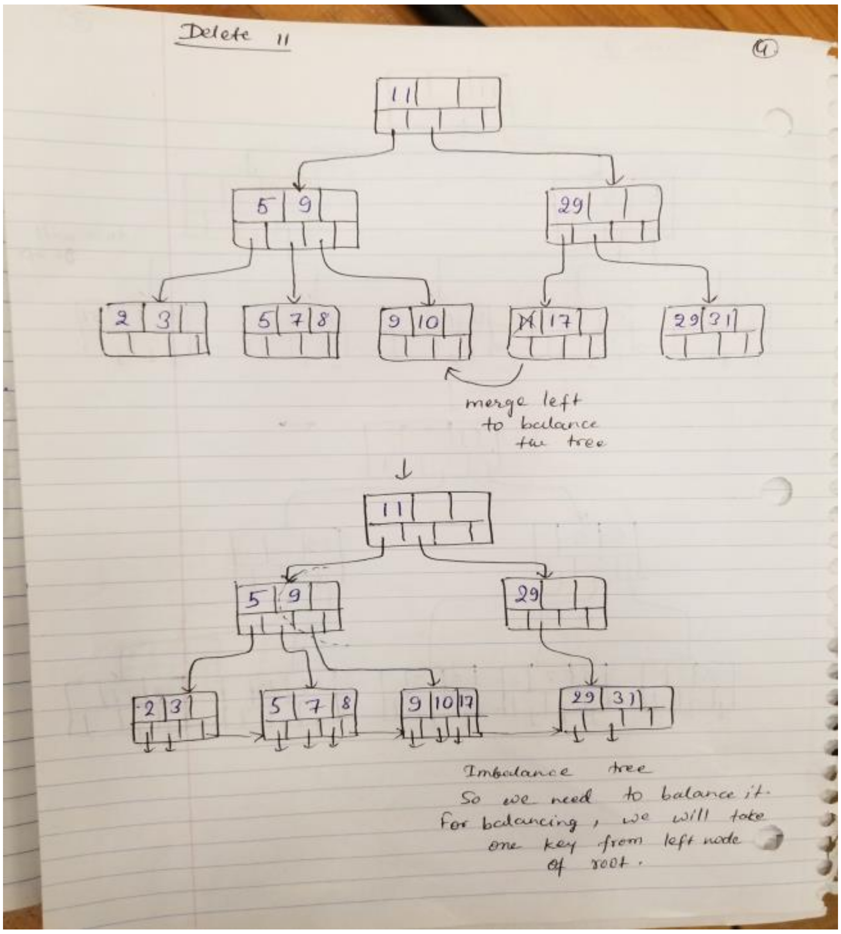

1
Delete 23
19
5 9 11
29
2 3
5 7 8
9 10
11 17
19 23
29 31
merge left for balance the tree
19
5 9 11
29
2 3
5 7 8
9 10
11 17 19
29 31
Only one pointer left & no right pointer ∴ Imbalance tree
for balancing tree, we will take one key from left node of root.

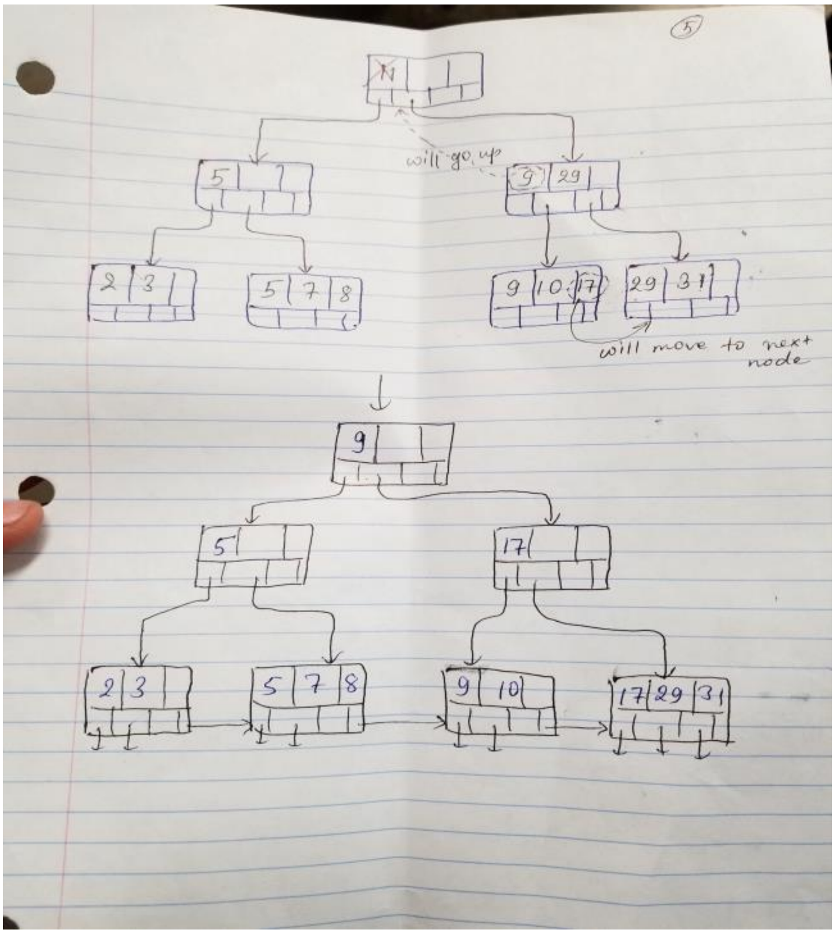


delete(19)



delete(11)


Delete 11
4
11
5 | 9
29
2 | 3
5 | 7 | 8
9 | 10
11 | 17
29 | 31
merge left to balance the tree
11
5 | 9
29
2 | 3
5 | 7 | 8
9 | 10 | 17
29 | 31
Imbalance tree
So we need to balance it.
For balancing, we will take one key from left node of root.


5
11
will go up
5
9 | 29
2 | 3
5 | 7 | 8
9 | 10 | 17
29 | 31
will move to next node
9
5
17
2 | 3
5 | 7 | 8
9 | 10
17 | 29 | 31