100

S.NO: 127

CSCE 5350 SECTION 002

MID TERM 02

NAME: KISHAN KUMAR ZALAVADIA

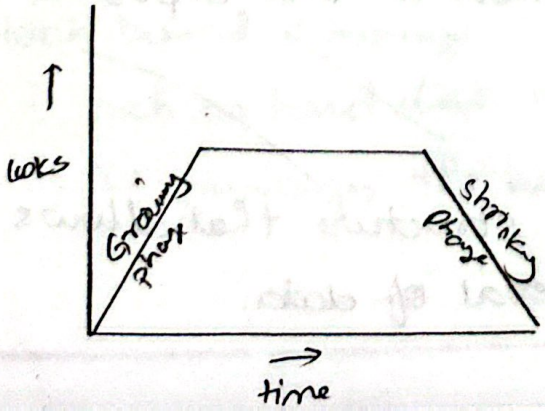
GRADER NAME: VASHISHTA REDDY MUTHIREDDY

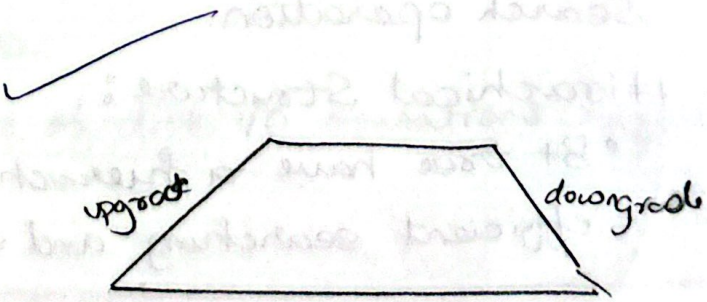
1) a) Growing Phase +5

- In the growing phase a transaction may obtain locks.
- In the growing phase a transaction may not release locks.
- The locking conversion for growing phase is as follows.
- A transaction
  - can acquire a lock-S (shared lock) on item
  - can acquire a lock-x (exclusive lock) on item
  - can convert a lock-s to a lock x (upgrade)

Shrinking Phase

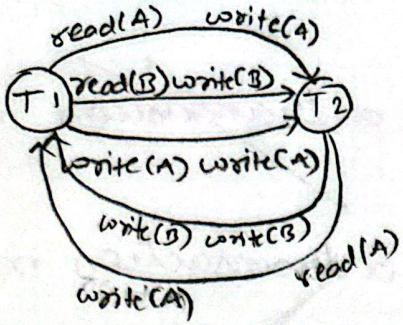
- In shrinking phase a transaction may release locks
- Transcaction may not obtain locks.
- The locking conversion for shrinking phase is as follows
- A transaction
  - can release a lock-S on an item
  - can release a lock-x on an item
  - can convert a lock x to a lock-S (downgrade)





(1C) Conflict - Serializable

+10



Complex ~~wait-for~~ graph
precedence

Simple ~~wait-for~~ graph
~~Precedence~~

The above is a precedence graph for the given transactions T1 and T2.

The graph contains loop which means the ~~trans~~ schedule is NOT SERIALIZABLE.

The arrow from T1 → T2 represents conflict between T1 & T2 where the first operation is done in T1. Similarly with T2 → T1.

(2C) Improves the Search operation on B+ tree.

+10

- B+ tree are balanced tree data structure used for indexing in a RDBMS.
- The below are the justification how B+ tree improves the Search operation.
- Hierachical Structure:
  - B+ tree have a hierachical structure that allows for efficient searching and retrieval of data.

- The tree has multiple levels, with the root node at the top, internal nodes in middle an leaf node containing the data at the bottom.
- Each node in the tree contains a sorted list of key value and pointers to child nodes or data entries.

- Ordered key Storage:
  - The key values in each node are stored in sorted order, which allows for efficient binary searches within each node.
- Leaf Node Data Storage:
  - All the data records are stored in the leaf node, which means the desired data can directly accessed or searched from the leaf node.
  - Efficient Tree traversal due to hierarchical structure.
- Leaf Node Linkage:
  - All leaf nodes are linked together in a sequential order, forming a doubly-linked list.
- The tree is balanced which ensure the height preserve preserving the efficient of seach operation
- Block based storage:
  - such as hard disk or SSD.
  - It minimizes the number of disk I/O operations required

(2d) B+ tree

$n = 4$

1) 40

2) 40 | 43

3) 25 | 40 | 43

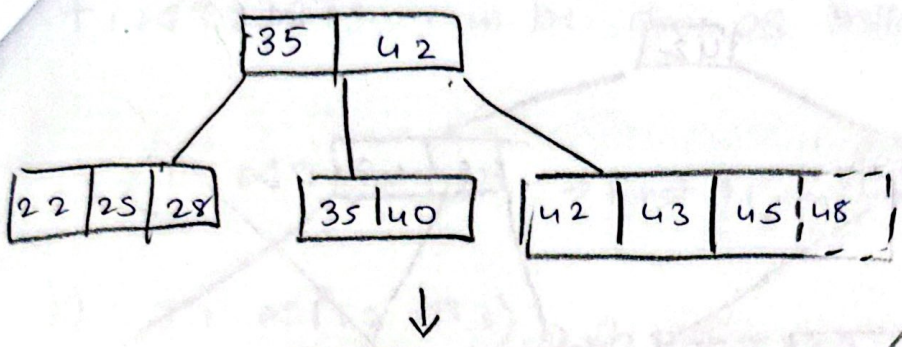
4) 22 | 25 | 40 | 43 → 40 ; 22 | 25 ; 40 | 43

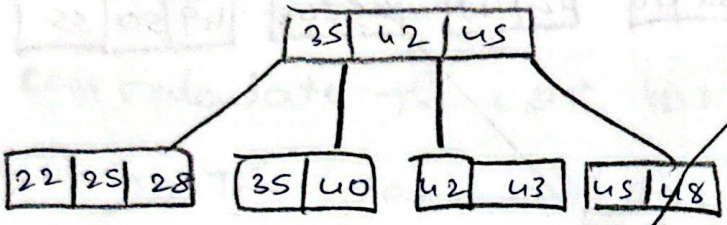
5) 40 ; 22 | 25 | 28 ; 40 | 43

6) 35 ; 22 | 25 | 28 ; 35 | 40 | 43

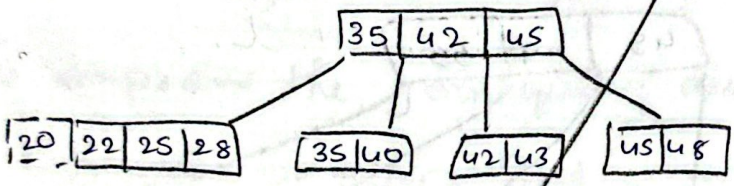
7) 35 ; 22 | 25 | 28 ; 35 | 40 | 42 | 43 → 35 | 42 ; 22 | 25 | 28 ; 35 | 40 ; 42 | 43

8) 35 | 42 ; 22 | 25 | 28 ; 35 | 40 ; 42 | 43 | 45



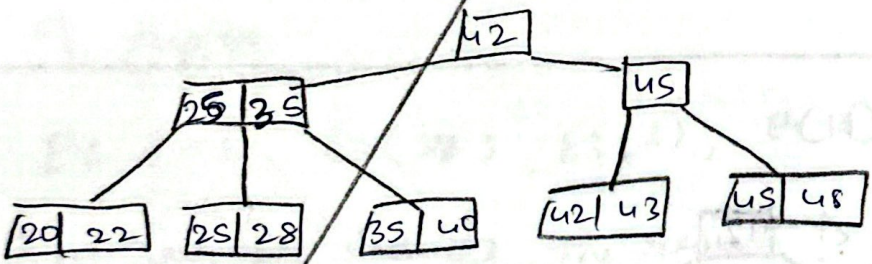


10)

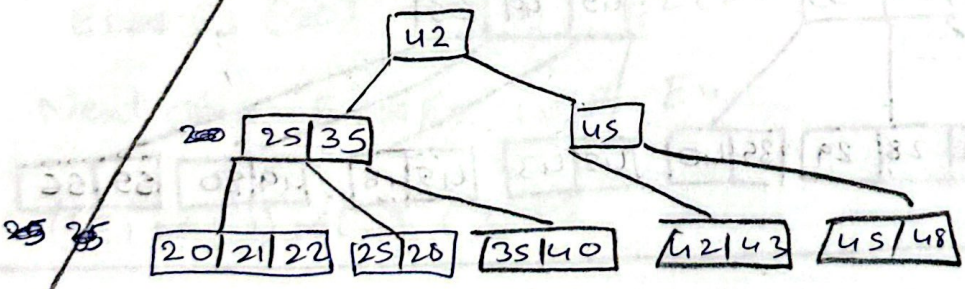


~~20~~ ~~22~~

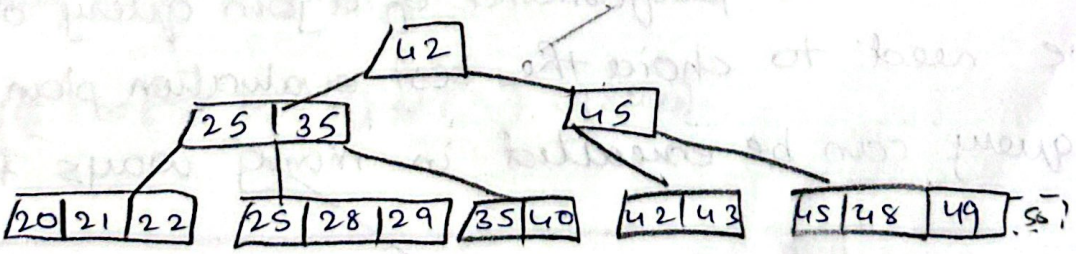
11)



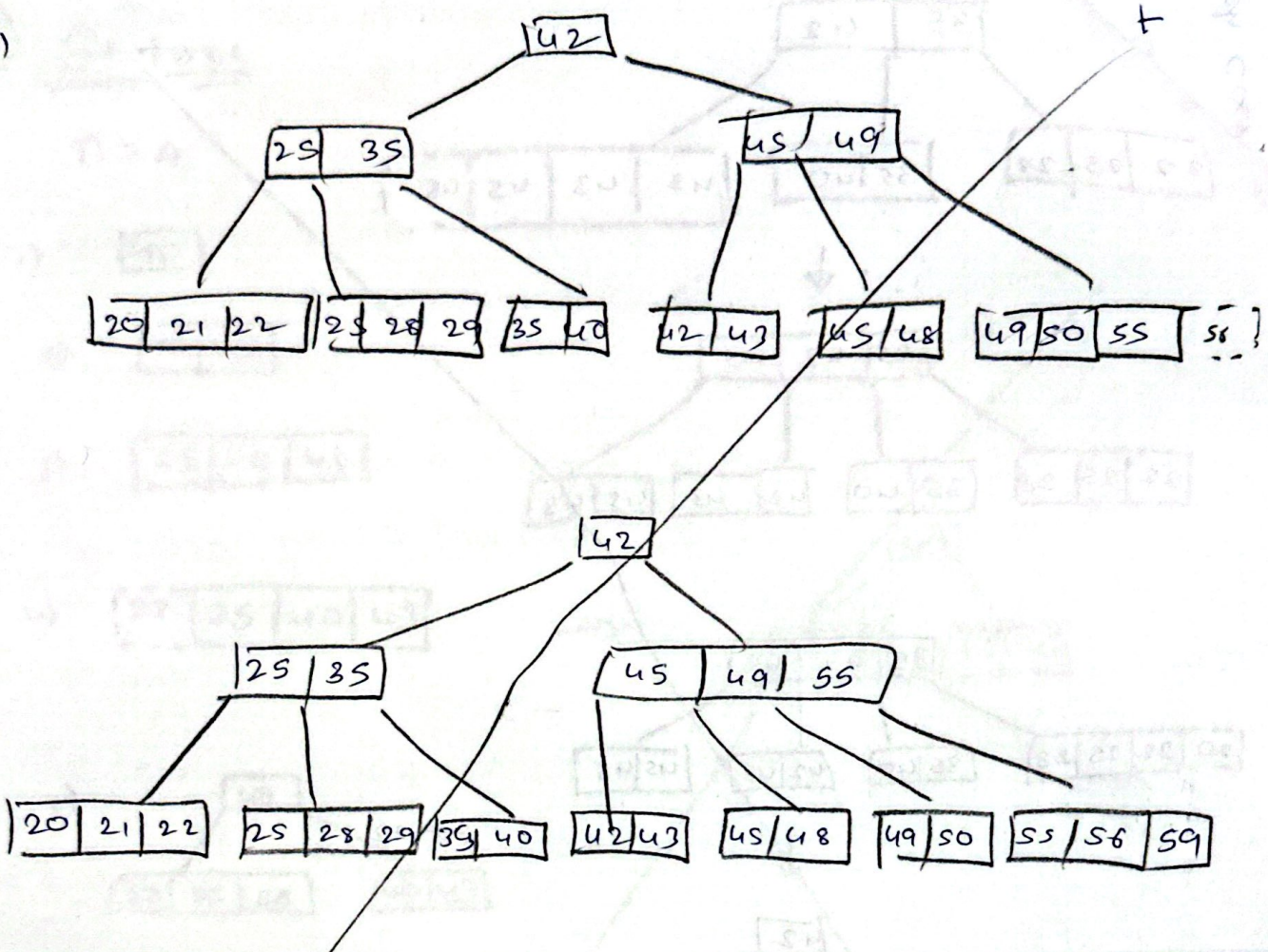
12)



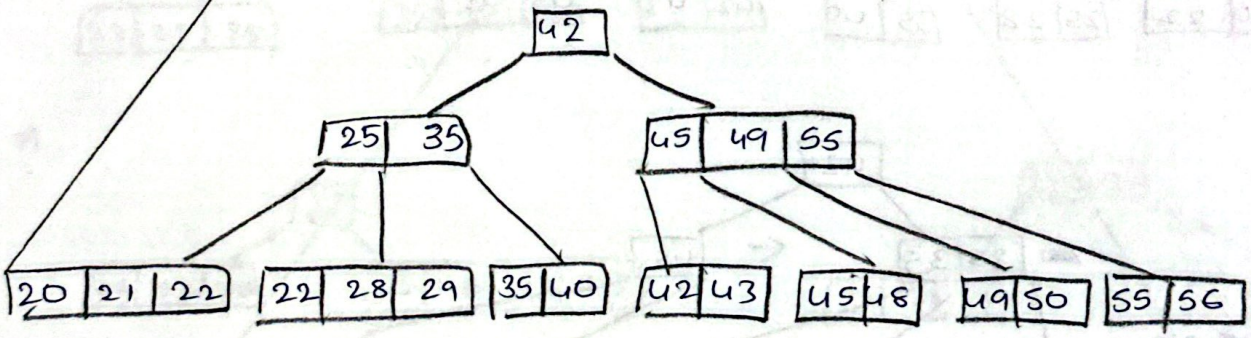
13)



14)



(c) Delete 69



(2b) To measure the performance of a join query or any query we need to choice the best evaluation plan. +5

A query can be executed in many ways for example

$T_1 \bowtie T_2 \bowtie T_3$ can be done as follows

i) $(T_1 \bowtie T_2) \bowtie T_3$ → first T_1 join T_2 & result is join with T_3

ii) $T_1 \bowtie (T_2 \bowtie T_3)$ → first T_2 join T_3 & result join with T_1

- We can calculate the cost for each execution plan and choice the plan which has the least cost.
- The cost depends on size of the table.
- To improve the join query alway join the table with less data or tuples first.
- for example consider the following tables with given numbes of tuples.

E_1 (10), E_2(22), E_3(2), E_4(11)

- First join E_1 and E_3 the result is.

$E_1 \bowtie E_3$ (20), E_4(11), E_2(22)

- Next join $E_1 \bowtie E_3$ with E_4

$((E_1 \bowtie E_3) \bowtie E_4)$ (220)

- Finally join it with E_2

$(((E_1 \bowtie E_3) \bowtie E_4) \bowtie E_2)$ (4840)

(2a) Query Retrieval with several records. §

- The index is a unique attribute defined which can identify each tuple uniquely.
- If a database stores tables which contain a key the records can be easily fetched.
- The database table is traversed to search for the tuple with a particular key and it returns when the key is found.
- The database index can be clusted Index where the file is sorted based on the search key which is the key or non-clustered which is not sorted based on key
- If we do not have an index defined on the query field we can get more than one tuple as the result.
- For example consider the following Student table

Student ID	name	Dept
1125	Depak	Comp.Sci
1126	Bhaskar	Data Sci
1127	Uday	Comp.Sci

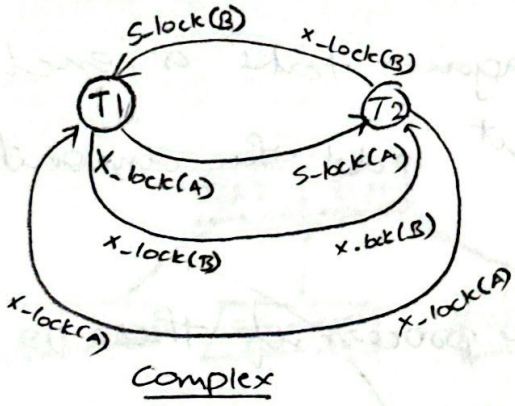
- If we need to search data with StudentID as 1127 we get the last tuple as output.

Consider the following Department table

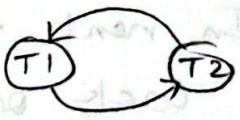
Dept Name	University	Number of faculty
Comp Sci	UNT	117
Data Sci	UNT	100
Comp Sci	UTD	200
AI	UTA	70

- If we want to find dept no tuples with Dept Name as Comp Sci, we get 2 tuples as output because Dept Name is not a key.

(1d) Wait-for-graph +15



Complex



simple

- T2 is requesting an exclusive lock for B but T1 already has a shared lock on B, here T2 is waiting for T1 which means there is an arrow from T2 to T1 in wait-for-graph.
- T1 is requesting a ~~shared~~ exclusive lock on A but T2 is ~~request~~ having a shared lock on A so an arrow goes from T1 to T2.

- There is a cycle in the graph which means there is a dead lock.

---

(15) +10

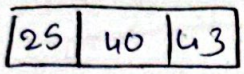
- The transaction must obay ACID properties. to ensure concurrent control.
- When reading the data and writing the data a lock can be made for that ~~too~~ amount and after modifying it the lock can be released.
- For example to withdraw the amount, take a lock on it read the amount and substract the amount then relase a lock from it.
- In next trasaction again take an enclusive lock on the amount add the amount and then relase the lock.
- And comm abort the process if there is any faut to maintaint atomicity which means either the complete transaction is done completely or not.

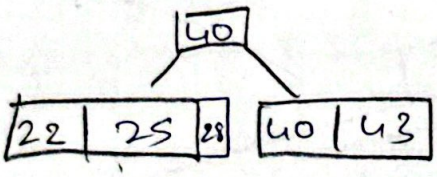
Insert 40

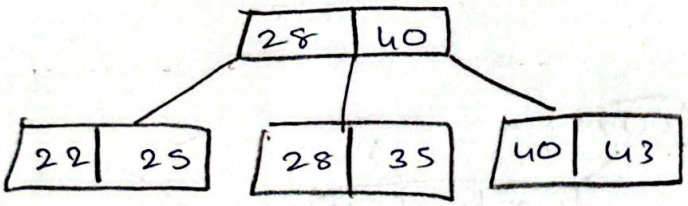
+30

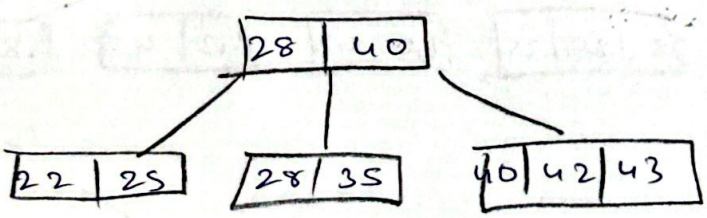
1) 

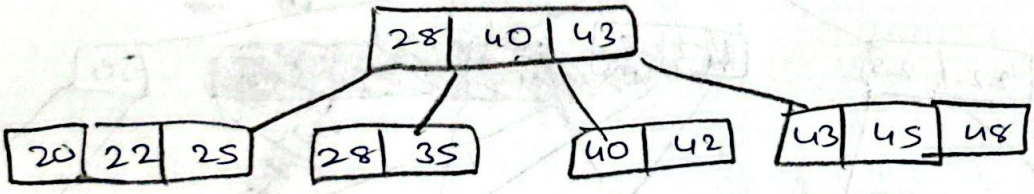
~~Insert~~ 43

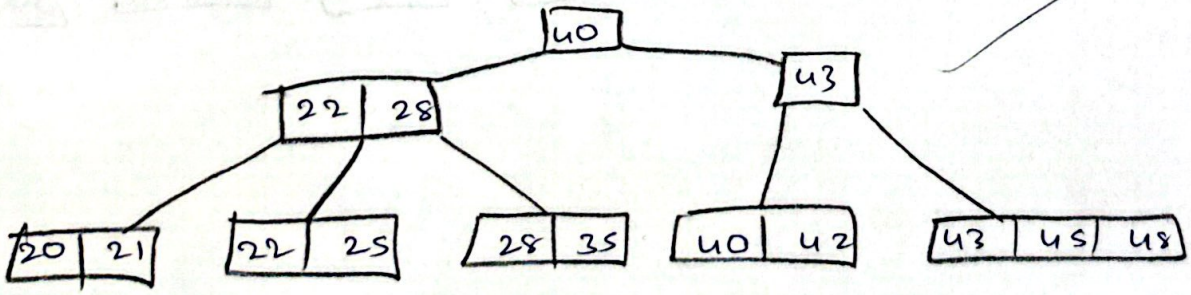
2) 

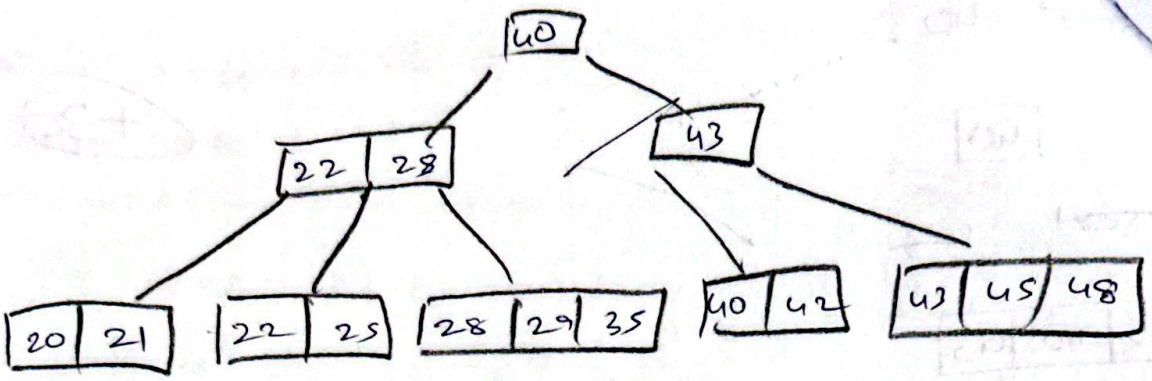
3) 

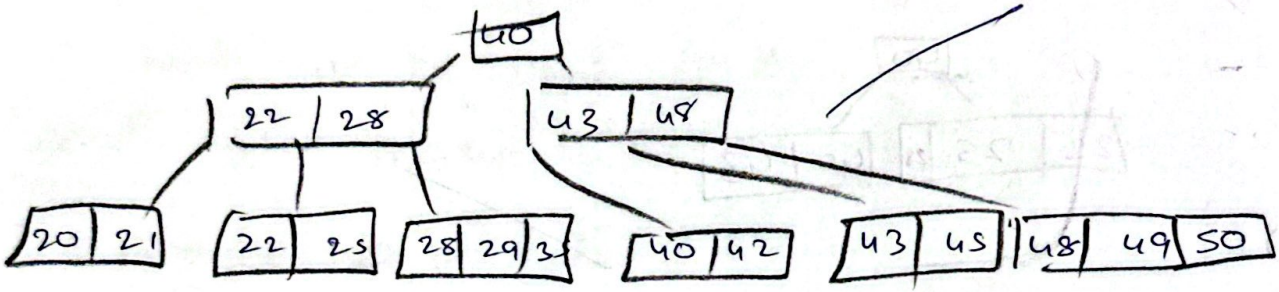
4) 

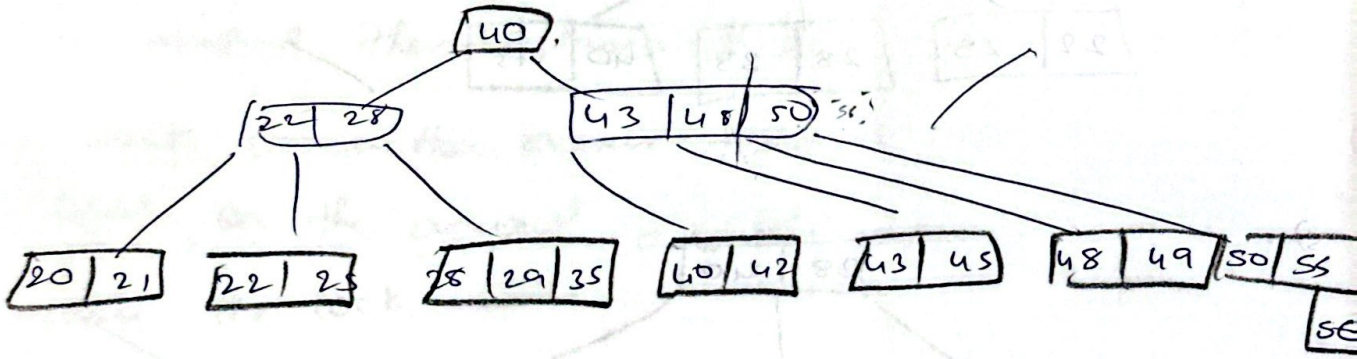
5) 

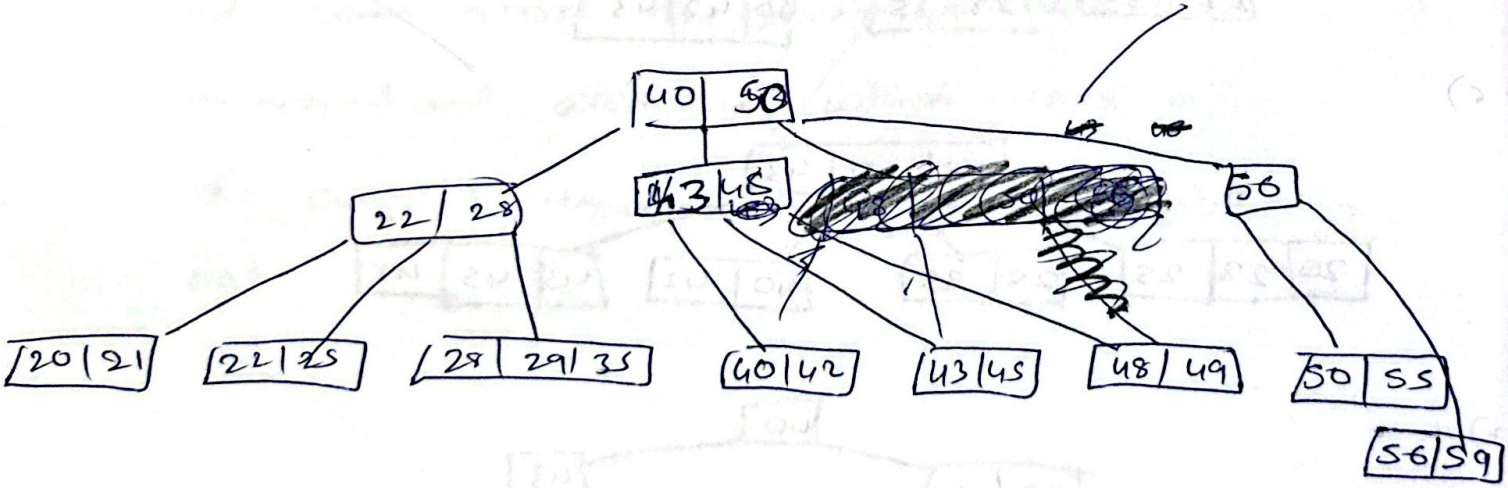
6) 

7) 









re 59

+10

(5)