Q1

Q1: L=X, M=Y. Then

a) range of items in Leaf: $[\lceil \frac{L}{2} \rceil, L]$

b) range of children in internal Node:

$$[\lceil \frac{M}{2} \rceil, M]$$

range of keys in internal Node:

$$[\lceil \frac{M}{2} \rceil - 1, M-1]$$

c) root Node:

1° if root Node is a leaf Node, it has $[\lceil \frac{L}{2} \rceil, L]$ children, then $[\lceil \frac{L}{2} \rceil - 1, L-1]$ keys

2° if root Node isn't a Leaf Node, it has $[2, M]$ children, then $[1, M-1]$ keys

---

Q2

Q2: insert into L=M=3 B+ Tree some datas as follows: 3, 20, 18, 4, 9, 6, 10, 23, 25, 27, 40, 13, 14.

Answer:

According to Q1:

root node $\begin{cases} \text{is a Leaf Node, then } [2, 3] \text{ children} \\ \text{isn't a Leaf Node, then } [2, 3] \text{ children} \end{cases}$

internal node $\begin{cases} [2, 3] \text{ children} \\ [1, 2] \text{ keys} \end{cases}$

leaf node: $[2, 3]$ items

1° first 3 steps:

$\boxed{3 \quad 18 \quad 20}$

2° insert 4

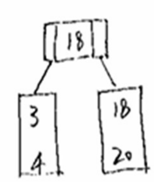
step1: $\boxed{3 \quad 4 \quad 18 \quad 20}$

it's the case that leaf node needs to be split and its parent node is not full

Step2: Split into $\begin{cases} X_L: \lfloor \frac{L+1}{2} \rfloor = 2 \text{ keys}, (3, 4) \\ X_R: \lceil \frac{L+1}{2} \rceil = 2 \text{ keys}, (18, 20) \end{cases}$

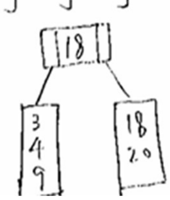
their new parent J = minimum key in X_R

∴ J = 18. that is:



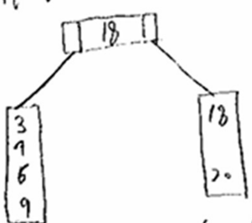
3° insert 9

it is the case that leaf node is half-full, but not full, insert it into the right place without doing anything else



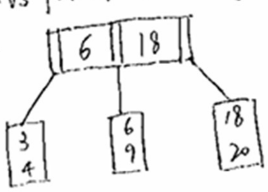
4° insert 6

step1:



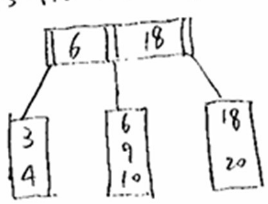
it is the case the leaf node needs to be split and its parent is not full. same as above

step2:



5° insert 10

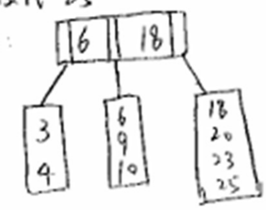
it is the case of inserting directly.



6° insert 23. it is the case of inserting directly

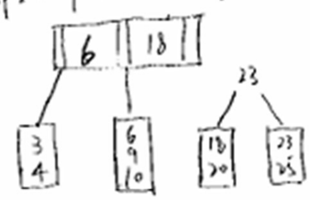
6° insert 25

step1:

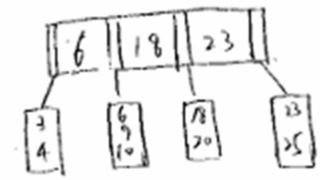


it's the case that both the leaf and its parent are full

step2: split the leaf node



step3:



step4: the parent is also full, then it needs to be split. the parent = $X = (6, 18, 23)$

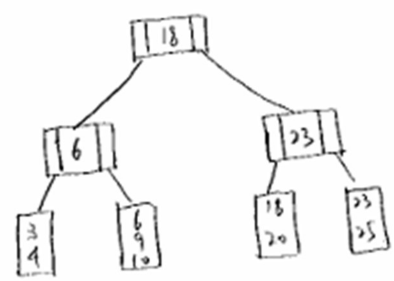
split into

X_L: $\lceil \frac{M}{2} \rceil - 1 = 1$ smallest key. (6)

J: = the $\lceil \frac{M}{2} \rceil$th key = the second key = 18

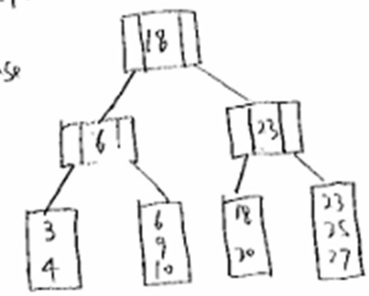
X_R: $\lfloor \frac{M}{2} \rfloor = 1$ largest key. (23)

So,



7° insert 27.

it is the case that insert directly



8° insert 40.

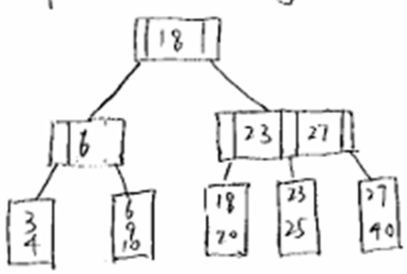
it is the case that leaf node is full but it's parent isn't full

step1 (23, 25, 27, 40) split into

$$\begin{cases} X_L: \lfloor \frac{L+1}{2} \rfloor = 2 \text{ smallest keys } (23, 25) \\ X_R \lceil \frac{L+1}{2} \rceil = 2 \text{ Largest keys } (27, 40) \end{cases}$$

new parent J = min key in X_R = 27

=>



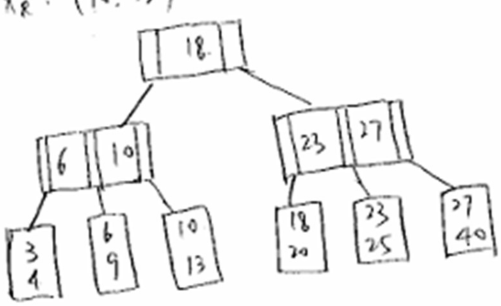
9° insert 13

it's the case that leaf node is full but it's parent isn't full.

(6, 9, 10, 13) split into

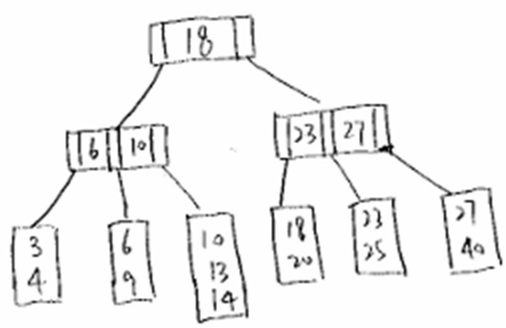
$$\begin{cases} X_L: (6, 9) \\ X_R: (10, 13) \end{cases} \quad J = 10$$

=>



10° insert 14

it's the case of inserting directly



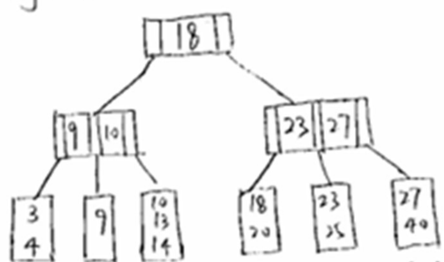
Q3

Q3 delete ~~nodes~~ items 6, 9, 10, 13 in order of B+ tree in Q2
18, 14

1° delete 6

it's the case that the key in internal nodes to be deleted

step1. delete the old key, and update with a new key in the internal nodes



step2: To check if the deleted leaf is half full or not. If it's not, we need to consider to borrow from its sibling (if the sibling can borrow) or merge with its sibling (if the sibling cannot borrow.

If a sibling can borrow items to other, it means it has $[\lceil \frac{L}{2} \rceil + 1, L]$ items

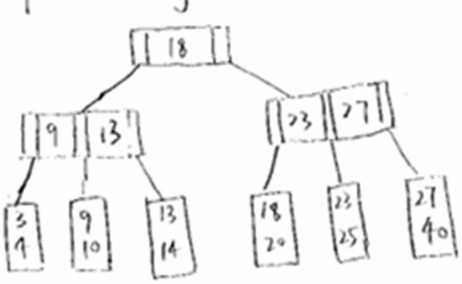
In this case, we need to borrow.

⇒

incorrect B+ Tree, just a step

⇓

step3. update the keys in internal nodes



2° delete 9

it's the case that the key in internal nodes to be deleted

Step1: delete the old key, and update with a new key in the internal nodes

step2: As above, the new leaf is not half anymore and two sibling can't lend any item to it either

Then we need to merge.

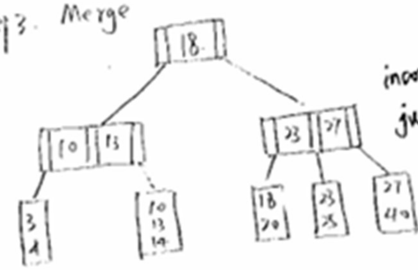
In this case, we can merge (3, 4) with 10.

or merge (10) with (13, 14). they are both okay.

Let's choose to merge (10) with (13, 14)

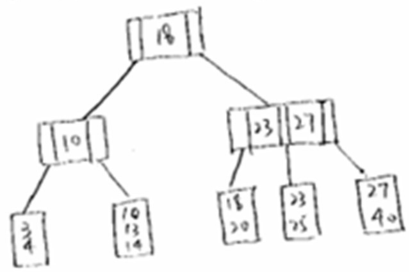
Let u = (10) v = (13 14). So

step3. Merge



incorrect B+ Tree, just a step

step4. delete the old key in the internal nodes

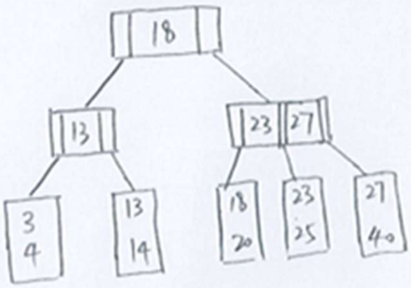


step5. check if the internal nodes are half full or not

Yes, it's half full, and no need to do any changes

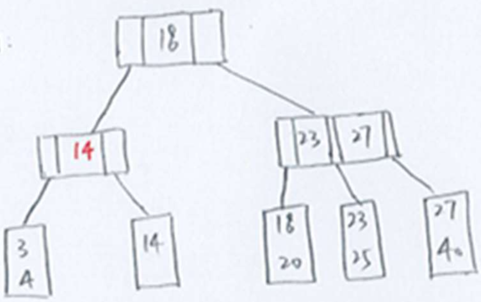
3° delete 10

it's the case of deleting directly. remeber to update the key in the internal nodes

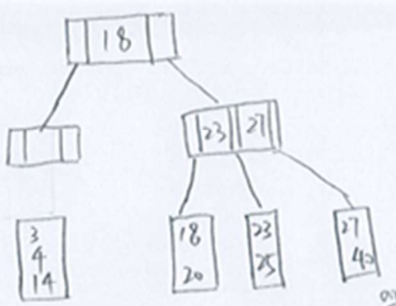


4° delete 13

step1.

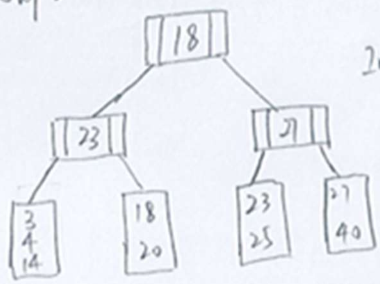


step2: Merge (3.4) with 14. and delete old key



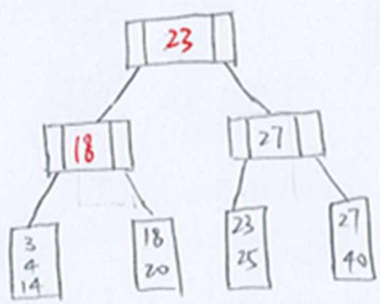
it's the case that the internal nodes are not half full any more. Similarly. we need to borrow or merge. If its sibling has more than $\lceil \frac{M}{2} \rceil - 1$ key. then it can borrow; if not. we can only choose the merge method In this case. the sibling (23.27) can borrow, So

step3:



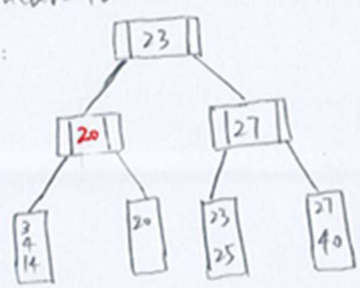
Incorrect B+ Tree just a step

step4: update the keys in internal nodes and root node.

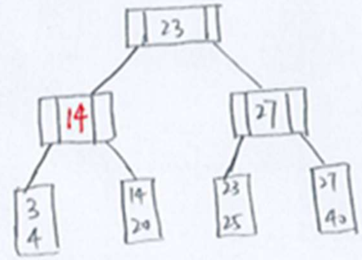


5° delete 18.

step1:



step2. borrow



Q4

Answer: 3

Explanation: Maximum number of pointers in a node is 7, i.e. the order of the B+ -tree is 7. In a B+ tree of order n each leaf node contains at most n – 1 key and at least $\lceil (n-1)/2 \rceil$ keys. Therefore, a minimum number of keys each leaf can have = $\lceil (7-1)/2 \rceil = 3$.

Q5

Answer: O(1)
Explanation: In a B+ -tree finding the next recored (successor) involves accessing an additional leaf at most. So, the efficiency of finding the next record is O(1).

Q6

Answer: 26

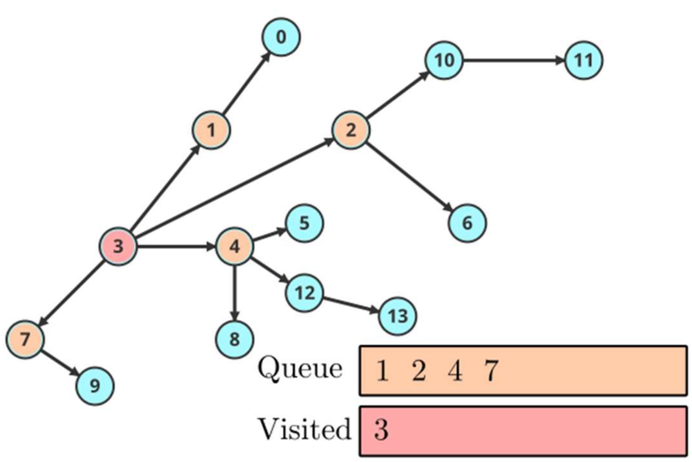
Explanation: A B+ tree of order n and height h can have at most $n^h - 1$ keys. Therefore maximum number of keys = $3^3 - 1 = 27 - 1 = 26$.

Q7

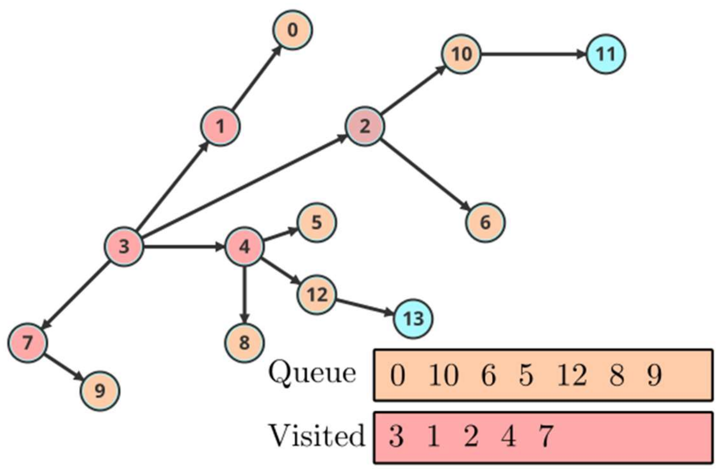
Step1:

Start from source s = 3. Set Q = {3}. Q is the queue of nodes to visit.

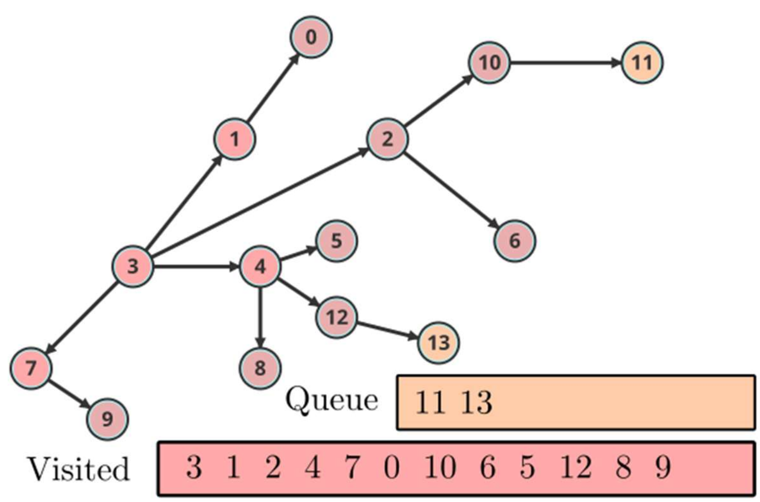
Step2:



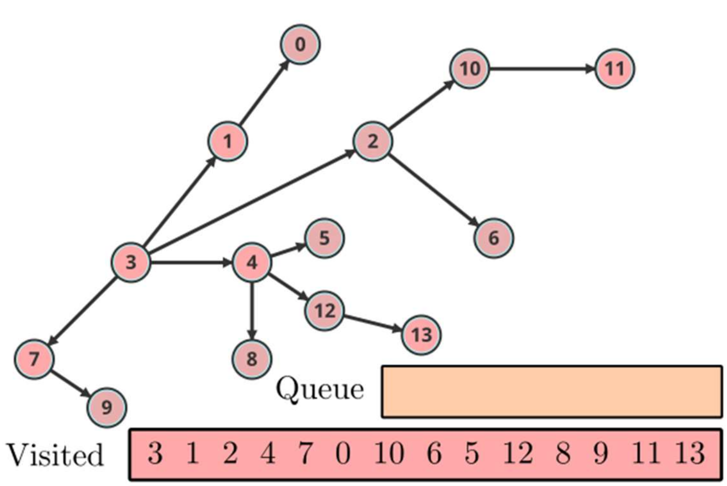
Step3:



Step4:



Step5:



The BFS traversal of the graph is 3, 1, 2, 4, 7, 0, 10, 6, 5, 12, 8, 9, 11, 13