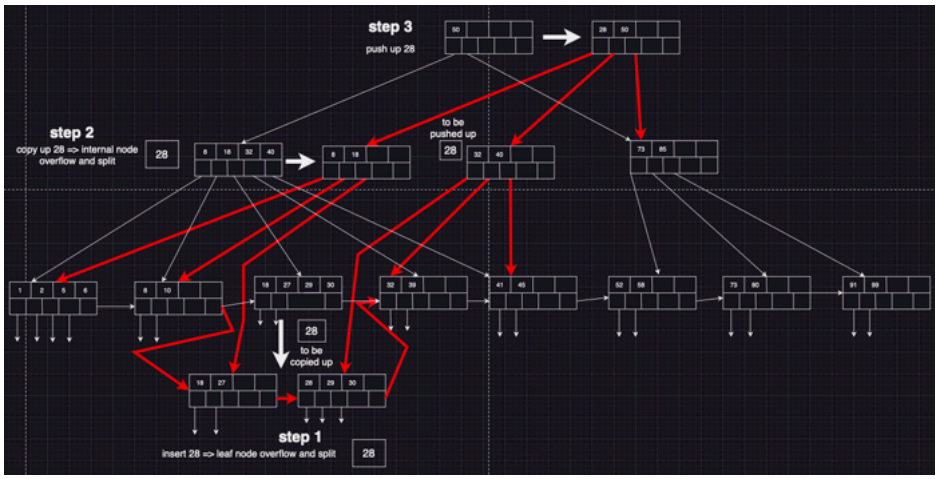
HW6 （曾益銘， 408410081）

tags: 資料庫系統

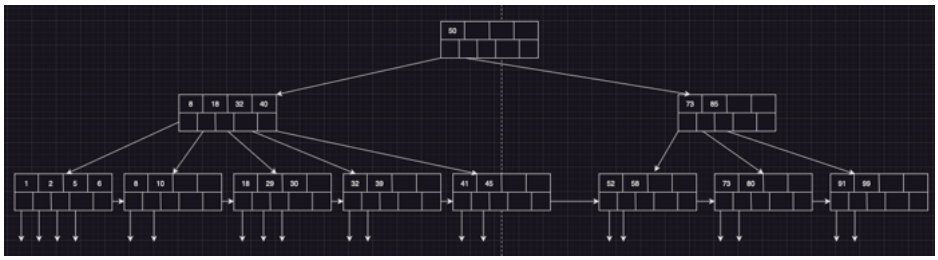
Part A

1. insert key 28 leaf node 發生 overflow，這邊選擇取 middle key copy up 上去當新的 index key，剛好是 key 28。接著 internal node 發生 overflow, 一樣選擇取 middle key 當作新的 index key 但是是用 push up 的方式，剛好也是 key 28。最後 index key 28 push up 到 root 的時候就沒有發生 overflow 了，因此結束整個 insert key 28 的過程。

   Note: 紅色箭頭是新增或修改後的 pointer，白色粗箭頭是舊的 node 發生變化後的樣子，從 step 1 到 step 2 最後到 step 3 完成新增 key 28



2. delete key 27 沒有發生 underflow, 所以直接刪除即可。



Part B

1.

$$\Pi_{sname}((\sigma_{grade=10}(Enrolled) \bowtie Student)$$

2.

$$\Pi_{age}((\sigma_{credits=3}(Courses) \bowtie Student \bowtie Enrolled)$$

3.

$$\Pi_{sname}((\sigma_{cname='Calculus'}(Courses) \bowtie Student \bowtie Enrolled)$$

4.

$$\Pi_{sname}((\sigma_{credits<4}(Courses) \bowtie Student \bowtie \sigma_{grade\geq 8}(Enrolled))$$

5. 所有 Enrolled 減掉所有 grade 10 的 Enrolled，留下的都是至少有修其他課但沒拿到 grade 10 的 sid，接著用所有 Students sid 去 set difference 減掉這些 sid 就會得到修課成績全部都是 grade 10 的學生的 sid。最後拿 sid 去 join Students

$$\rho(S, Students)$$
$$\rho(E1, Enrolled)$$
$$\rho(E2, Enrolled)$$

$$\rho(t, \Pi_{E1.sid,E1.cid}(E1) - \Pi_{E2.sid,E2.cid}(\sigma_{E2.grade=10}(E2)))$$

$$\rho(AllGra10Sid, \Pi_{sid}(S) - \Pi_{sid}(t))$$

$$\Pi_{sname}(AllGra10Sid \bowtie Students)$$

6. 這題理解為要找出只修一門 3 學分的課的以及在一些課拿過 grade 10 的學生，所以這題是用了和 B9 一樣的概念找出只修一門課的學生加上是否該課是 3 學分的課。最後把

只修一門 3 學分的課以及在一些課拿過 grade 10 的學生 union 起來

$$\rho(E, Enrolled)$$
$$\rho(E1, Enrolled)$$
$$\rho(E2, Enrolled)$$
$$\rho(SingleEnrSid, \Pi_{sid}(E) - \Pi_{E1.sid}(E1 \bowtie_{E1.sid=E2.sid \wedge E1.cid \neq E2.cid} E2))$$

$$\rho(t1, \Pi_{sname}(SingleEnrSid \bowtie Students \bowtie Enrolled \bowtie \sigma_{credits=3}(Courses)))$$
$$\rho(t2, \Pi_{sname}((\sigma_{grade=10}(Enrolled)) \bowtie Student \bowtie Courses))$$

$$\Pi_{sname}(t1 \cup t2)$$

7. 用修過 'Calculus' 的學生 set difference 減掉所有修至少一門 4 學分的課的學生，得到的就會是題目要求的學生

$$\rho(t1, \Pi_{sname,age}(\sigma_{cname='Calculus'}(Courses) \bowtie Student \bowtie Enrolled))$$
$$\rho(t2, \Pi_{sname,age}(\sigma_{credits=4}(Courses) \bowtie Student \bowtie Enrolled))$$
$$\Pi_{age}(t1 - t2)$$

8. 讓 Enrolled table self join 得到 t1 和 t2 然後保留兩個 col(grade 和 sid)，第一個 col 比第二個 col 來得小，然後拿第一個 col - 第二個 col 就會得到最小值，Note: 這裡的 - 是 set difference，如果找最大值就拿第二個 col - 第一個 col。最後拿最小值 grade 所匹配的 sid 去 join Students

$$\rho(E1, Enrolled)$$
$$\rho(E2, Enrolled)$$
$$\rho(t1, \Pi_{E1.grade,E1.sid}(\sigma_{E1.grade<E2.grade}(E1 \bowtie E2)))$$
$$\rho(t2, \Pi_{E2.grade,E2.sid}(\sigma_{E1.grade<E2.grade}(E1 \bowtie E2)))$$
$$\rho(LowestGraWithSid, t1 - t2)$$
$$\Pi_{sname}(\Pi_{sid}(LowestGraWithSid) \bowtie Students)$$

9. 讓 Enrolled table self join，主要會有四個 col: (sid1, cid1, sid2, cid2)。假設有修兩門課的學生的 records 是(1, 3)，(1, 5) ， 那 self join 結果會是 (1, 3, 1, 5)，(1, 5, 1, 3)，只要 cid1 和 cid2 不一樣就是修等於或超過兩門課。假設只修一門課的學生的 record 是 (3, 8)，那 self join 不會有結果因為是 （3，8，3，8） 。最後把所有 Enrolled set difference 減掉修超過至少兩門課的學生就會得到只修一

門課的學生。最後拿只修一門課學生的 sid 去 join Students

$$\rho(E, Enrolled)$$
$$\rho(E1, Enrolled)$$
$$\rho(E2, Enrolled)$$

$$\rho(SingleEnrSid, \Pi_{sid}(E) - \Pi_{E1.sid}(E1 \bowtie_{E1.sid=E2.sid \wedge E1.cid \neq E2.cid} E2))$$

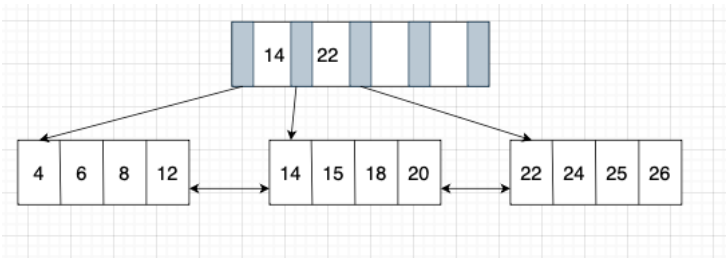
$$\Pi_{sname}(SingleEnrSid \bowtie Students)$$

10. 讓 Enrolled table self join 得到 t1 和 t2 然後保留兩個 col(credits 和 cid)，第一個 col 比第二個 col 來得小，然後拿第二個 col - 第一個 col 就會得到最大值，Note: 這裡的 - 是 set difference，如果找最小值就拿第一個 col - 第二個 col。最後拿最大值 credits 所匹配的 cid 去 join Enrolled 和 Students

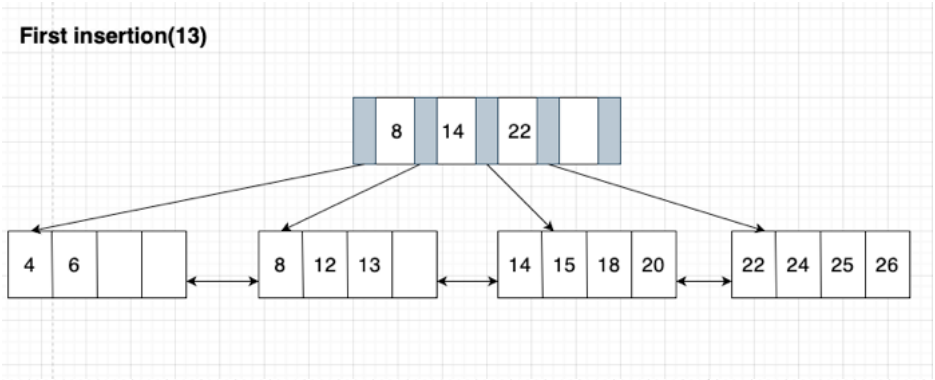
$$\rho(C1, Courses)$$
$$\rho(C2, Courses)$$
$$\rho(t1, \Pi_{C2.credits, C2.cid}(\sigma_{C1.credits < C2.credits}(C1 \bowtie C2)))$$
$$\rho(t2, \Pi_{C1.credits, C1.cid}(\sigma_{C1.credits < C2.credits}(C1 \bowtie C2)))$$
$$\rho(HighestCredWithCid, t1 - t2)$$
$$\Pi_{sname}(\Pi_{cid}(HighestCredWithCid) \bowtie Enrolled \bowtie Students)$$

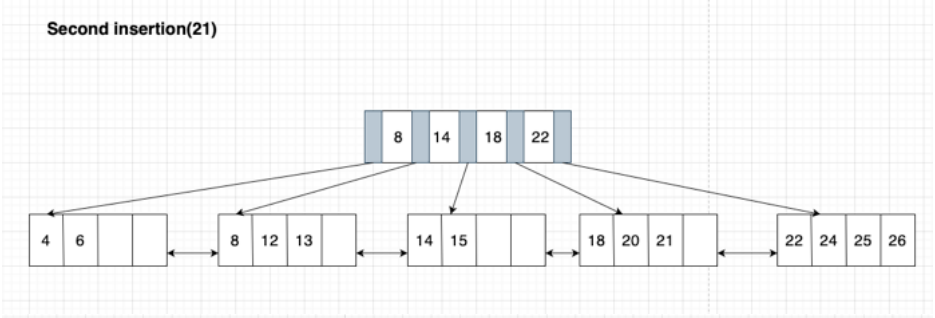
Part C

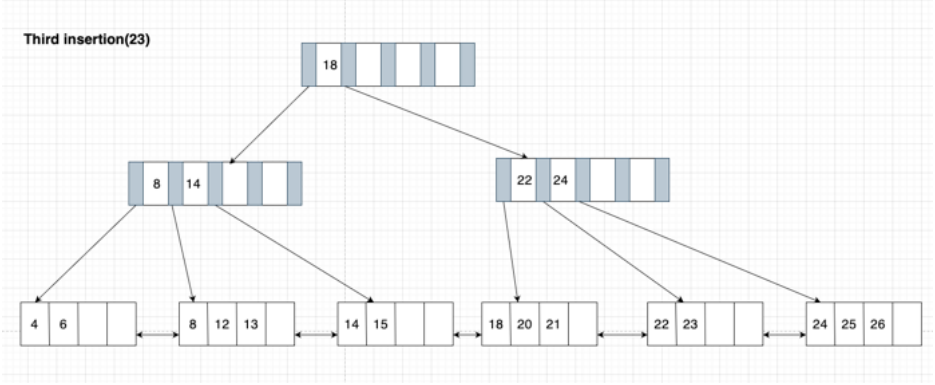
1.



2. The sequence is 13, 21, 23。insert 13 後會導致 (4, 6, 8, 12) leaf node split 然後 key 8 會 copy up 放到 root，接著 insert 21 後會導致 (14, 15, 18, 20) leaf node split 然後 key 18 會 copy up 放到 root，注意這時候 root 已經滿了，最後 insert 23 會導致 (22, 24, 25, 26) leaf node split 然後 key 24 會 copy up 放到 root，這時候 root 裝不下 key 24 了所以必須 split，然後 key 18 會 push up 到新的 root。



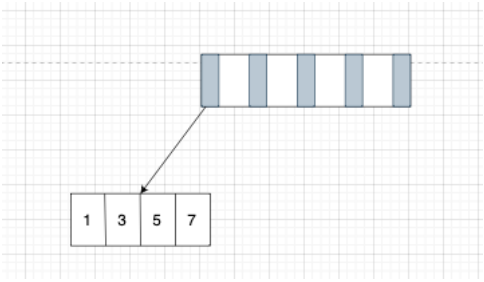




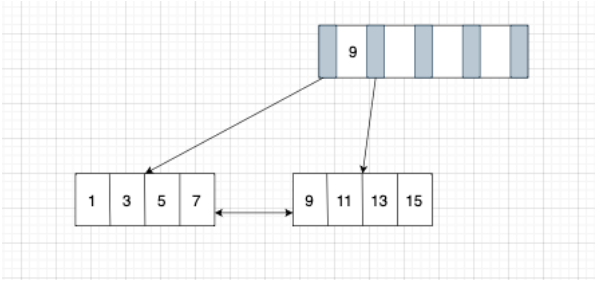
Part D

1.

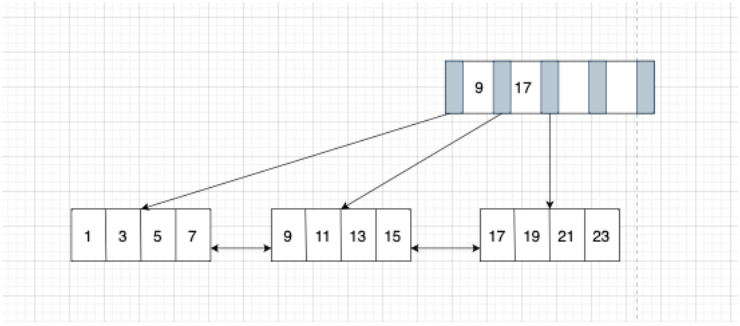
步驟1:



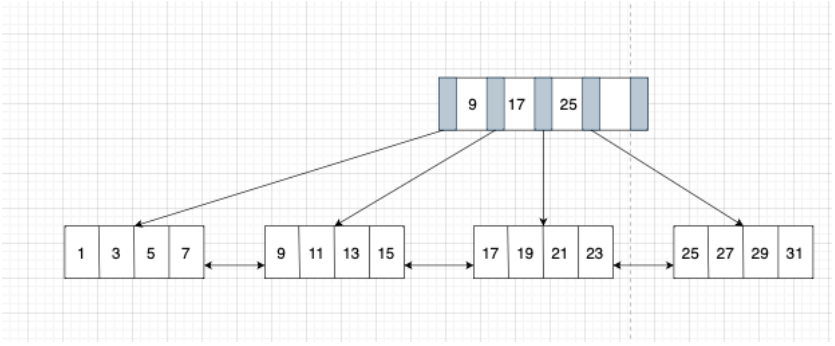
步驟2:



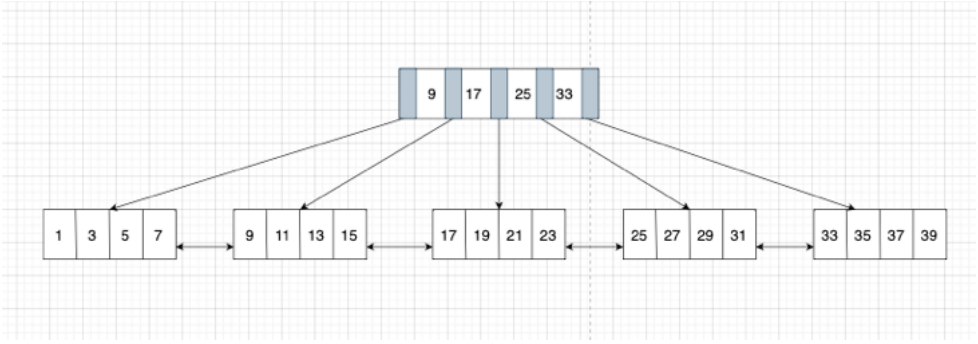
步驟3:



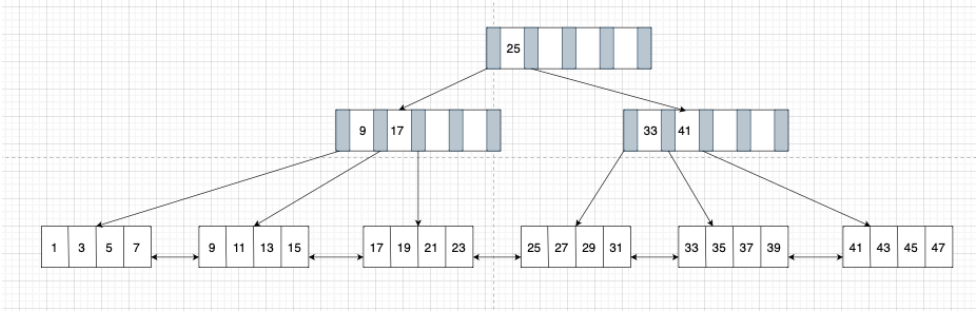
步驟4:



步驟5:

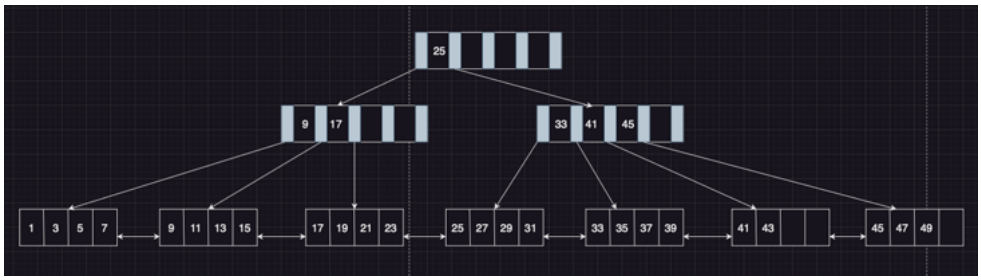


步驟6: bulk loading leaf node（41, 43, 45, 47) 導致 internal node 發生 overflow, internal node split 然後選擇 middle key 25 push up 當作新的 index key

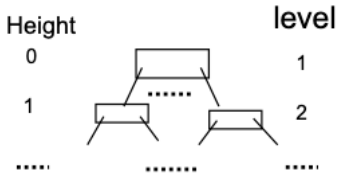


步驟7: 最後一個要 bulk loading 的 leaf node 只有一個 key 49，所以用一般 split insert 的方式 insert key 49 發生 overflow 所以 split 新的 leaf node 接著新增 index key 45 到

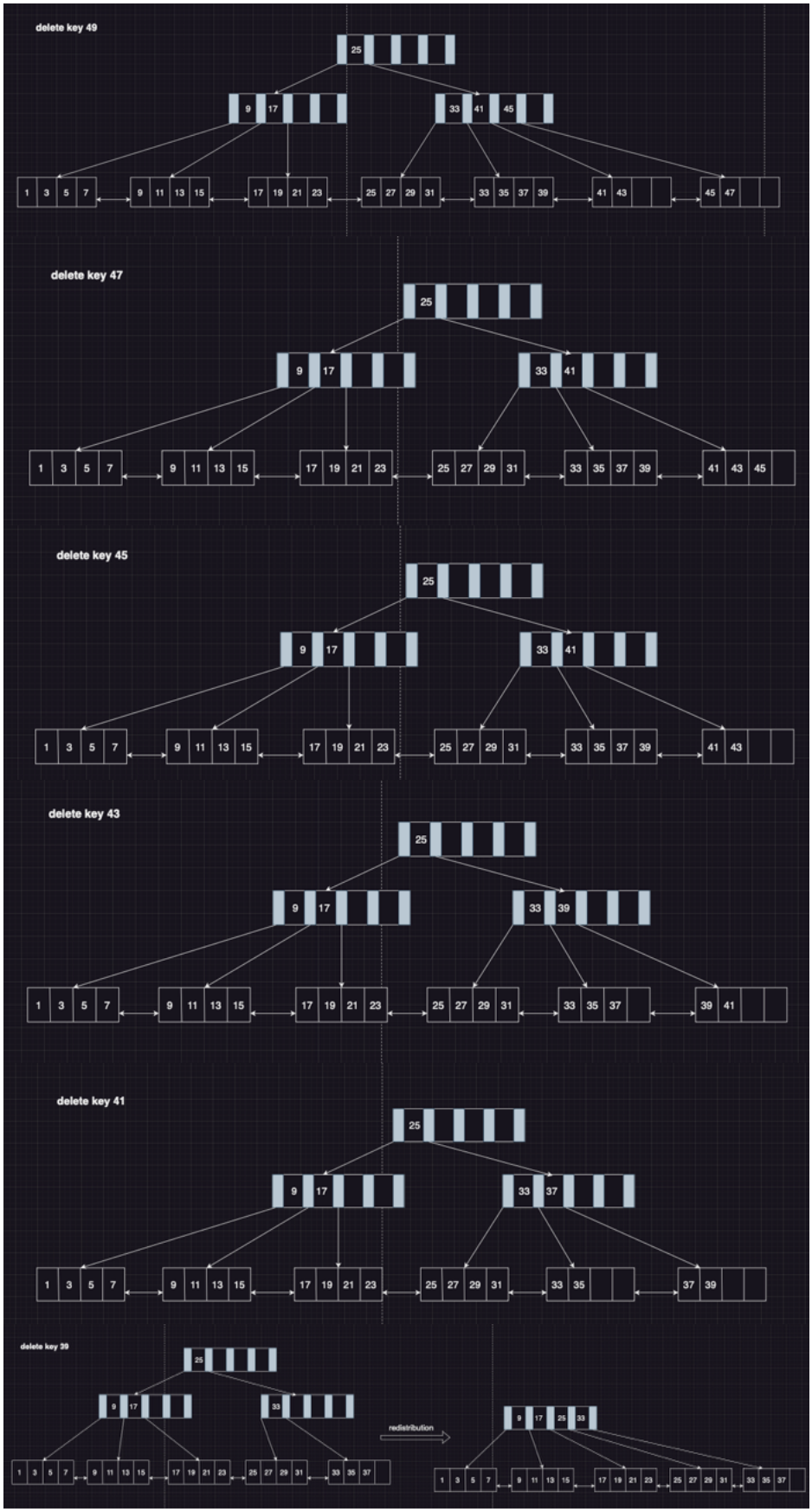
internal node (33, 41) 即可。



2. 2，參考了課堂上課講義第 20 頁的這張圖

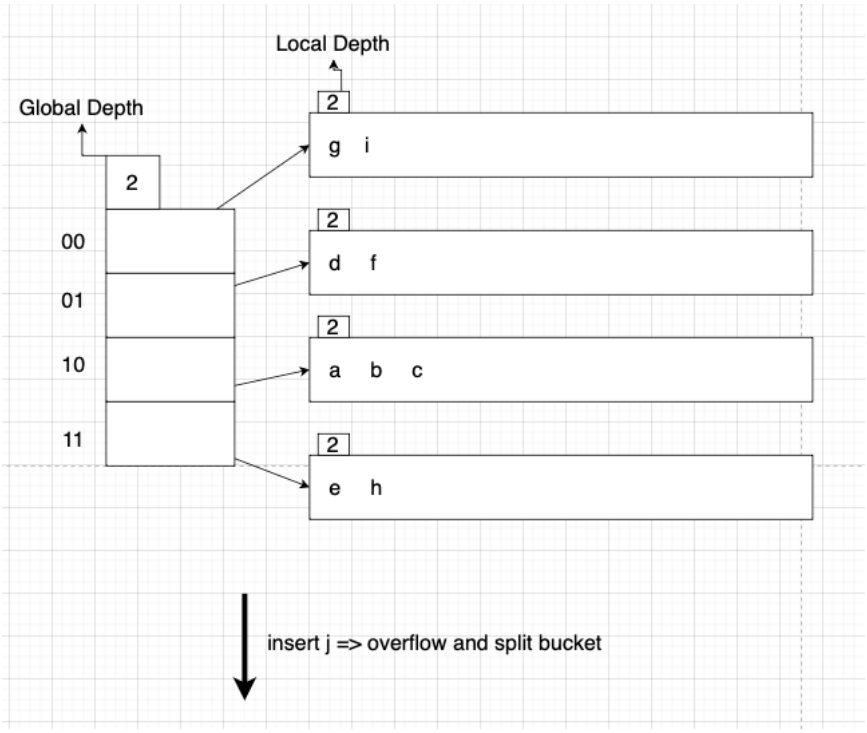


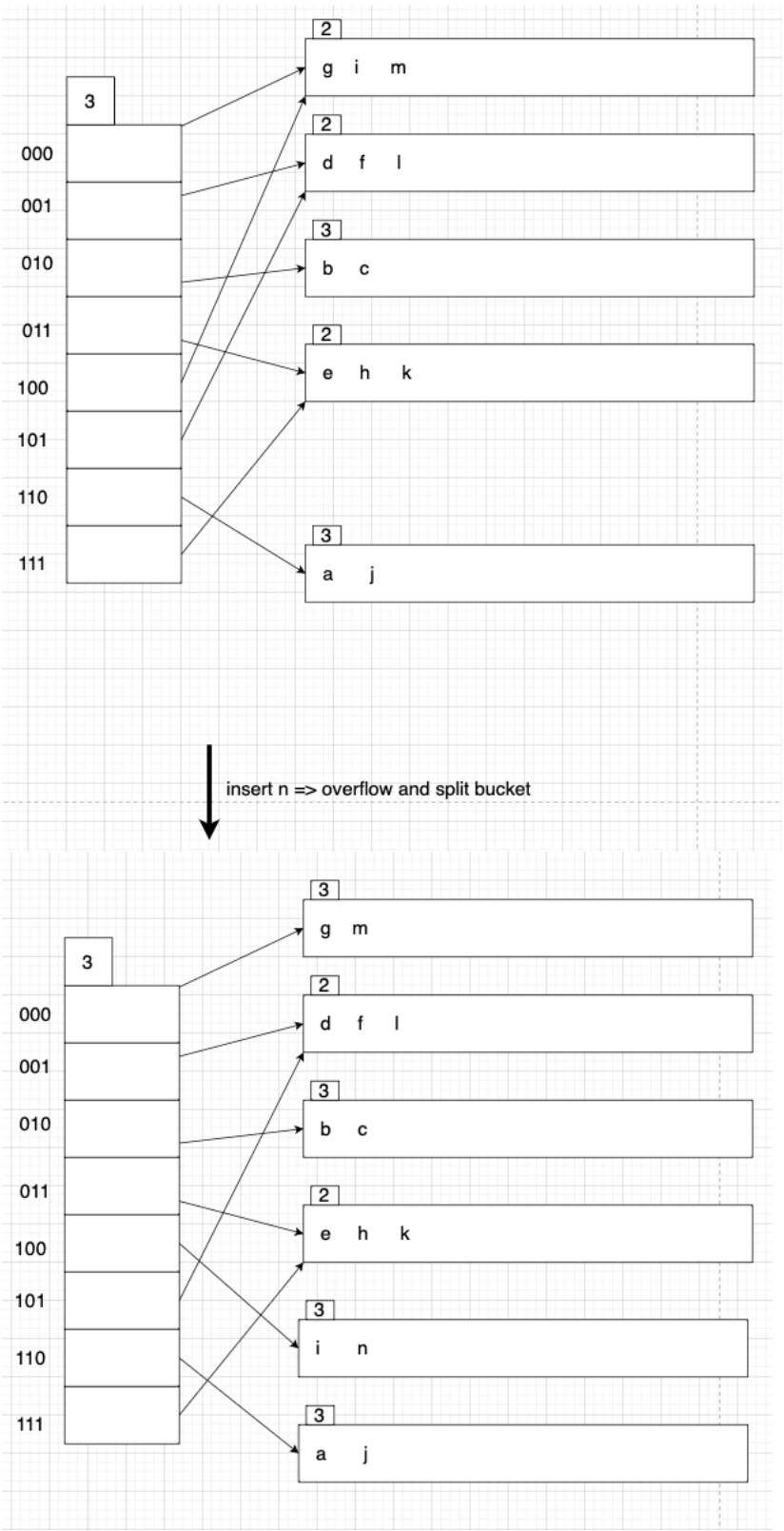
3. The minimum number of keys that must be deleted is 6 which are (49, 47, 45, 43, 41, 39)。刪掉 key 49 沒有發生 underflow 直接刪掉即可。接著刪掉 key 47 發生 underflow 加上也不能和左邊 sibling (41, 43）借 key 所以要跟 sibling (41, 43) merge 變成 (41, 43, 45)，index key 45 會被刪掉。接著刪除 key 45 沒有發生 underflow 直接刪掉即可。接著刪掉 key 43 發生 underflow 所以 key 43 所在的 node 會和左邊 sibling（33, 35, 37, 39）借 key 39 過來達到 half full 變成 (39, 41)，index 41 會換成 index key 39。接著刪掉 key 41 發生 underflow 所以 key 41 所在的 node 會和左邊 sibling（33, 35, 37）借 key 37 過來達到 half full 變成 (37, 39)，index 41 會換成 index key 37。最後，刪掉 key 39 發生 underflow 加上也不能和左邊 sibling (33, 35)借 key 所以要跟 sibling (33, 35) merge 變成 (33, 35, 37)，index key 39 會被刪掉，這時候 （33, 35, 37）的 parent 也就是 (33) 發生 underflow 需要做 key redistribution 或 merge，觀察後可以發現 root 和兩個 child 的 key 總數量剛好等於 4（key 數量的上限），因此 root 的 key 可以拉下來和兩個 child 的 key 做 merging 進而讓 B+tree 的高度減一。


delete key 49
delete key 47
delete key 45
delete key 43
delete key 41
delete key 39
redistribution

Part E

1.


Local Depth
Global Depth
2
2
00
01
10
11
g i
2
d f
2
a b c
2
e h
insert j => overflow and split bucket


3
000
001
010
011
100
101
110
111
2
g i m
2
d f l
3
b c
2
e h k
3
a j
insert n => overflow and split bucket
3
3
g m
2
d f l
3
b c
2
e h k
3
i n
3
a j

2. 4