Deletion from a B-tree

2C

(We assume search keys in B-tree are unique).

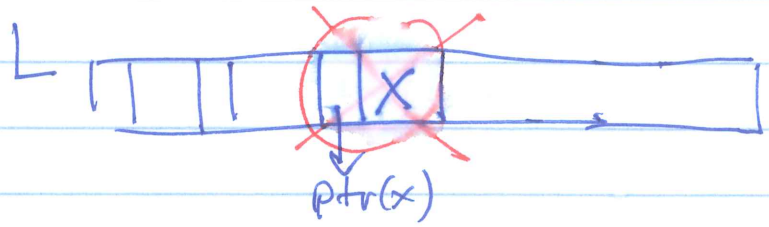
Delete Algorithm (delete x) → (x, pointer to x)

search key

(1) Find the leaf node L where search key x belongs

If not found: done

If found: delete (x, pointer(x))



(2) If after deletion:

L has $\geq \lceil \frac{(n+1)}{2} \rceil$ (min) keys

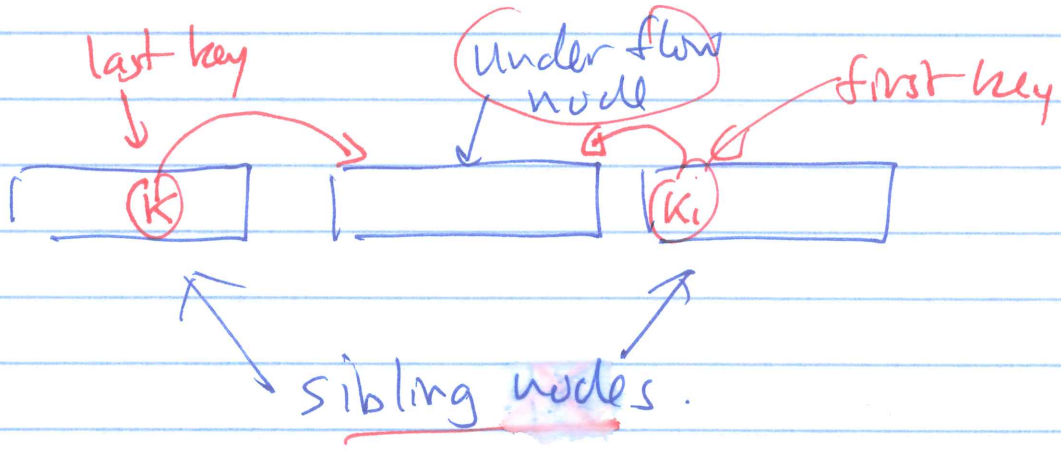
⇒ Done

(3) If L underflow: transfer / merge

Underflow: a node contains less than the min. # keys / pointers

Methods to solve underflow:

(1) preferred (1st choice) solution: transfer



If either sibling has > min. # keys

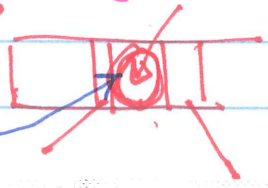
then:

transfer one (key, pointer)

to the underflow node.

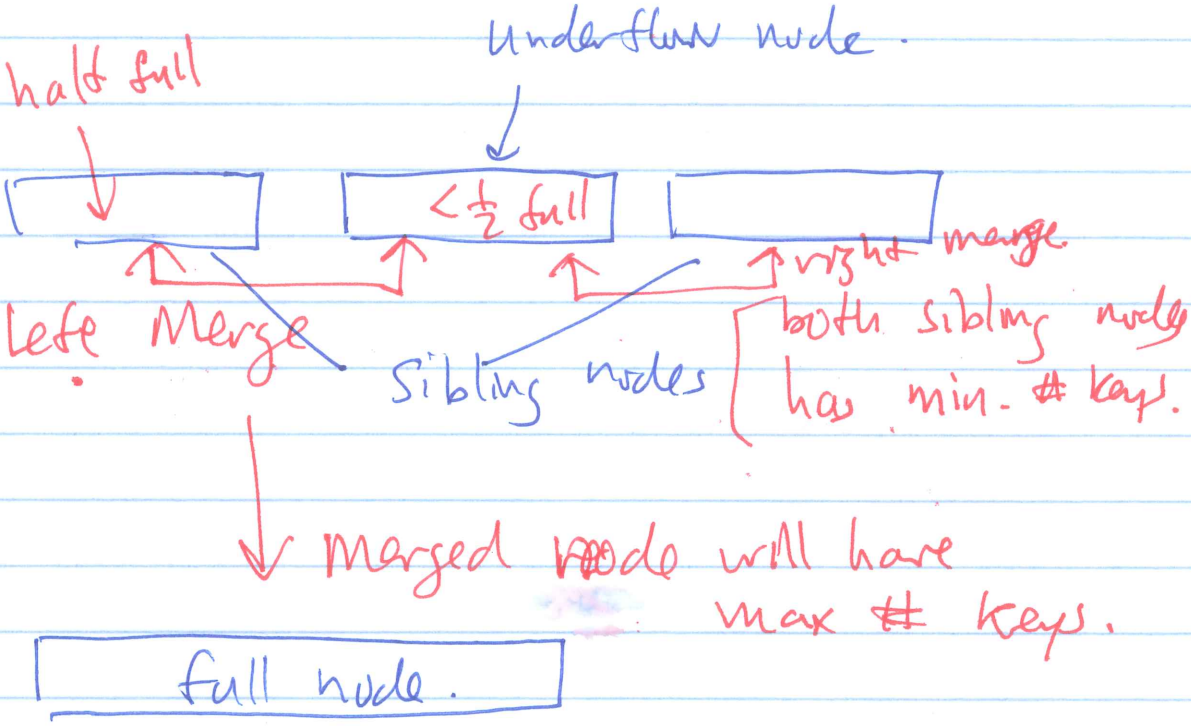
☆ Parent node need to adjust a key search value : P

(transfered key is stored in parent)



Untined: methods to solve underflow

(2) method of last resort: Merge with one sibling.



☆ The Parent node will LOSE one (key, pointer) pair (because it has one less child node !!!).

Propagated Merge/delete

Note: Because the parent node will lose one (key, pointer)

⇓

Parent node can underflow in a merge operation.

⇓

(Parent node undergoes a deletion)

⇓

Handle underflow in parent with

- → transfer
- or: another merge

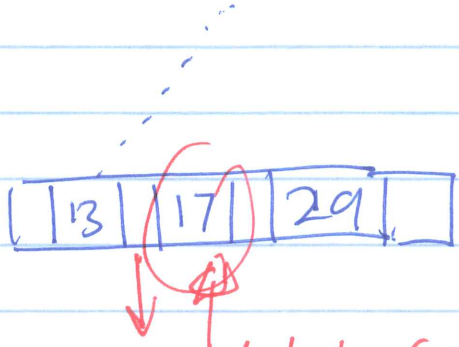
⇓

Can propagate all the way to the root!!!

Example: delete - simple case

Delete in a node that does not result in underflow.

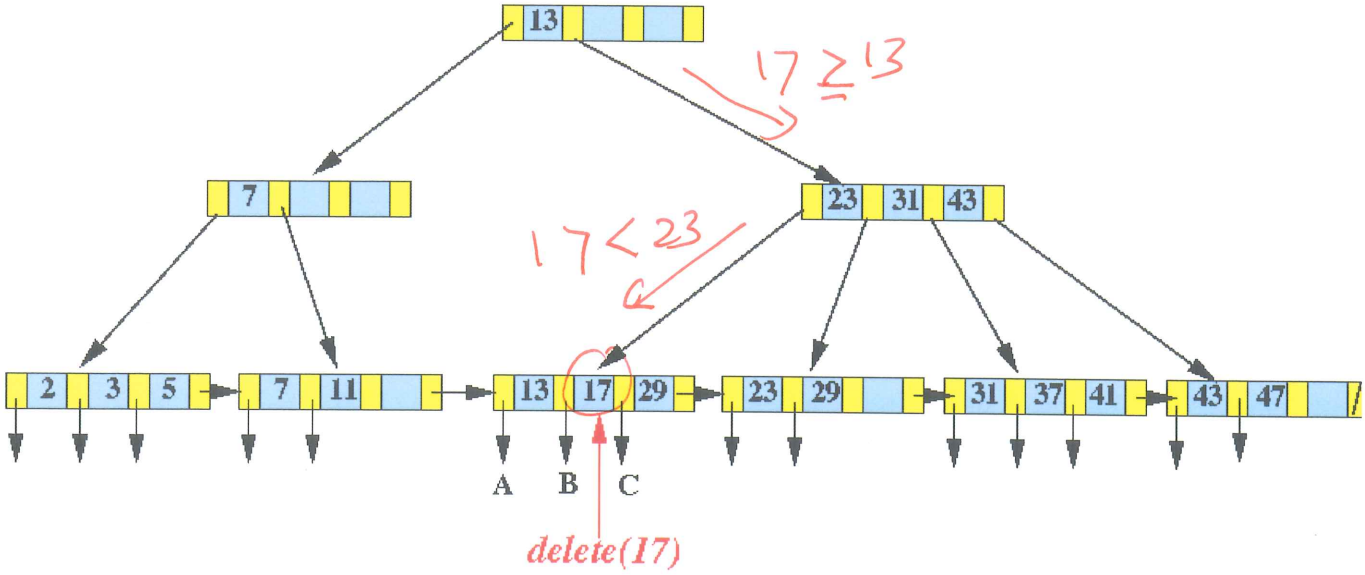
eg:



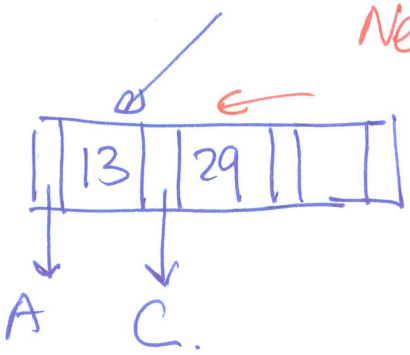
delete(17)

shift keys + pointers!

☆ You must use the SEARCH alg to find the key in the leaf FIRST (prep step).

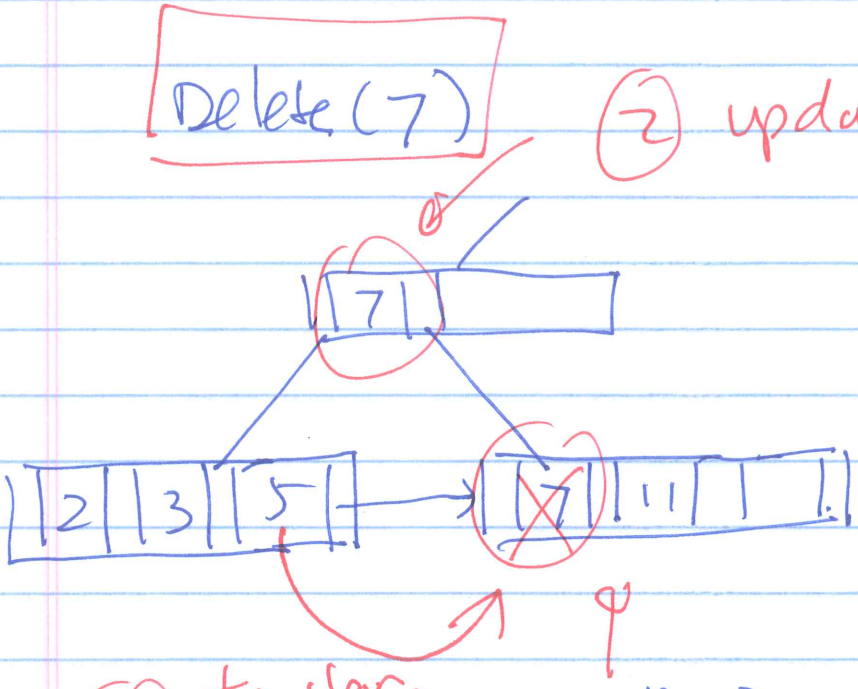


result



Need to move keys + pointers.

Delete: medium-hard case: transfer.



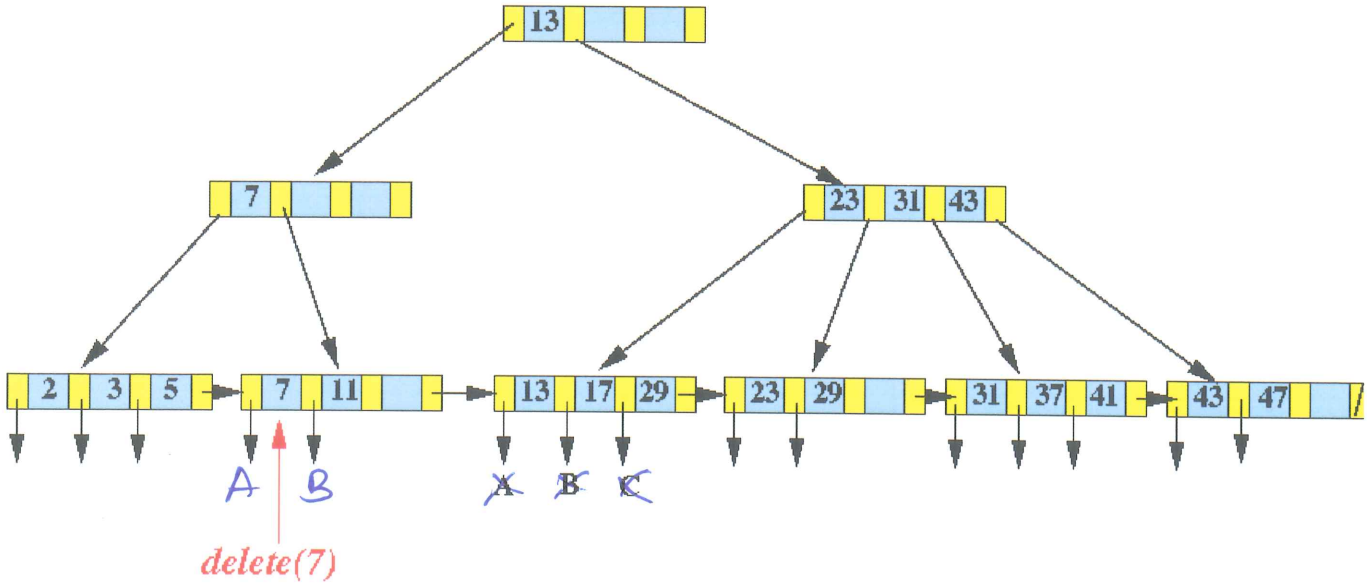
1 key left., $\geq$ 2 keys min

$\Rightarrow$ underflow.

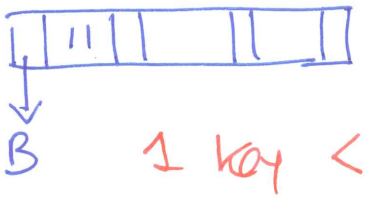
Sibling has 3 keys > 2 keys

$\Rightarrow$ transfer.

Delete (7)



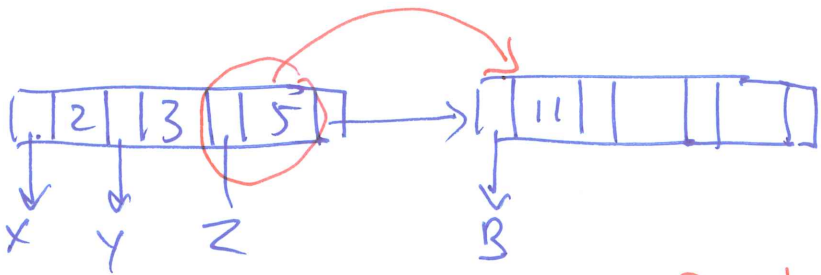
(1) After delete:



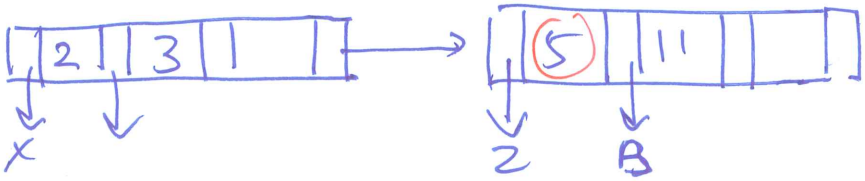
1 key < 2 (m/2).

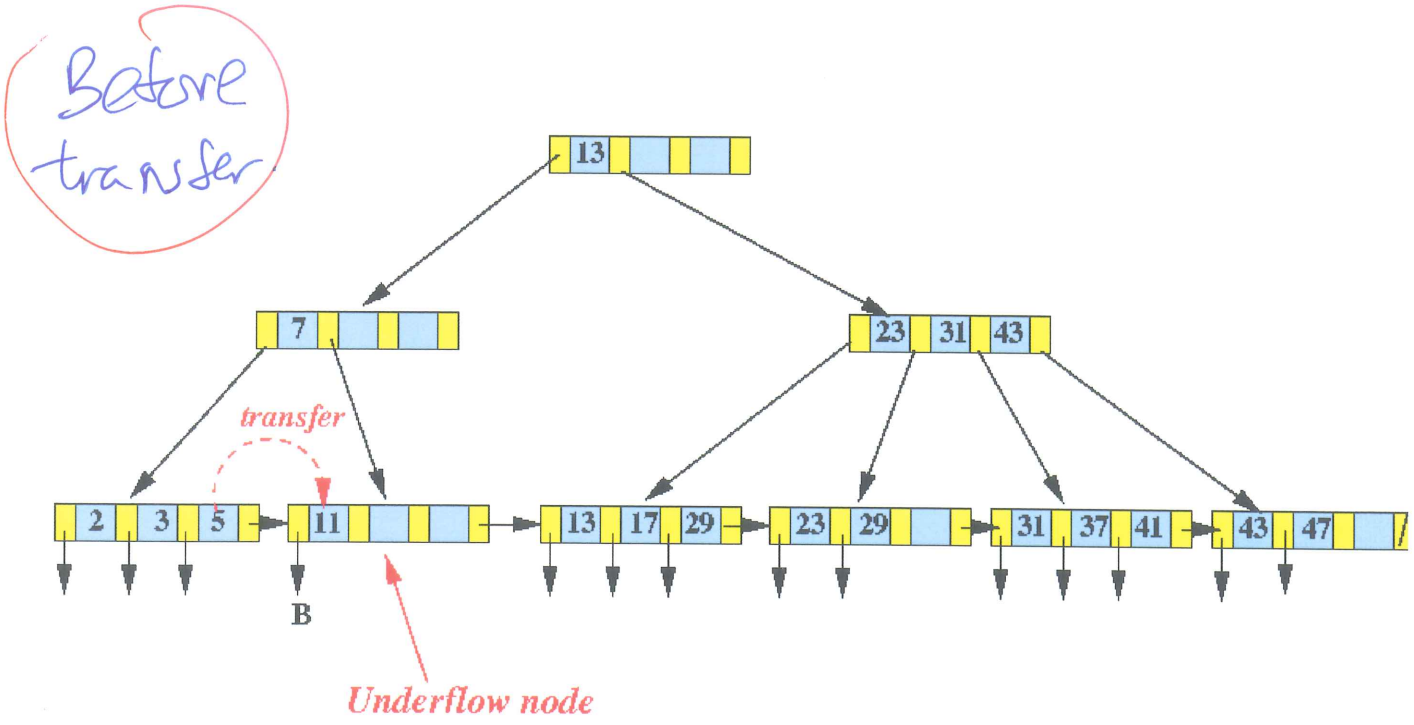
Underflow.

(2) Transfer from sibling:

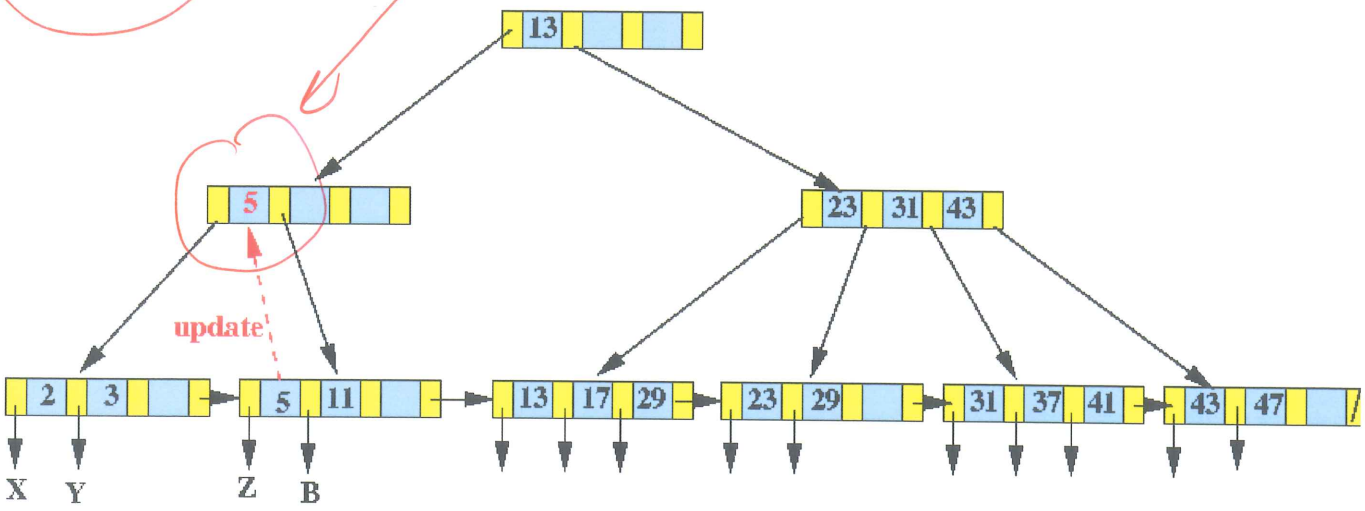


Parent must be updated!




Before transfer
13
7
23 31 43
transfer
2 3 5
11
13 17 29
23 29
31 37 41
43 47
B
Underflow node

After transfer + update key in parent node



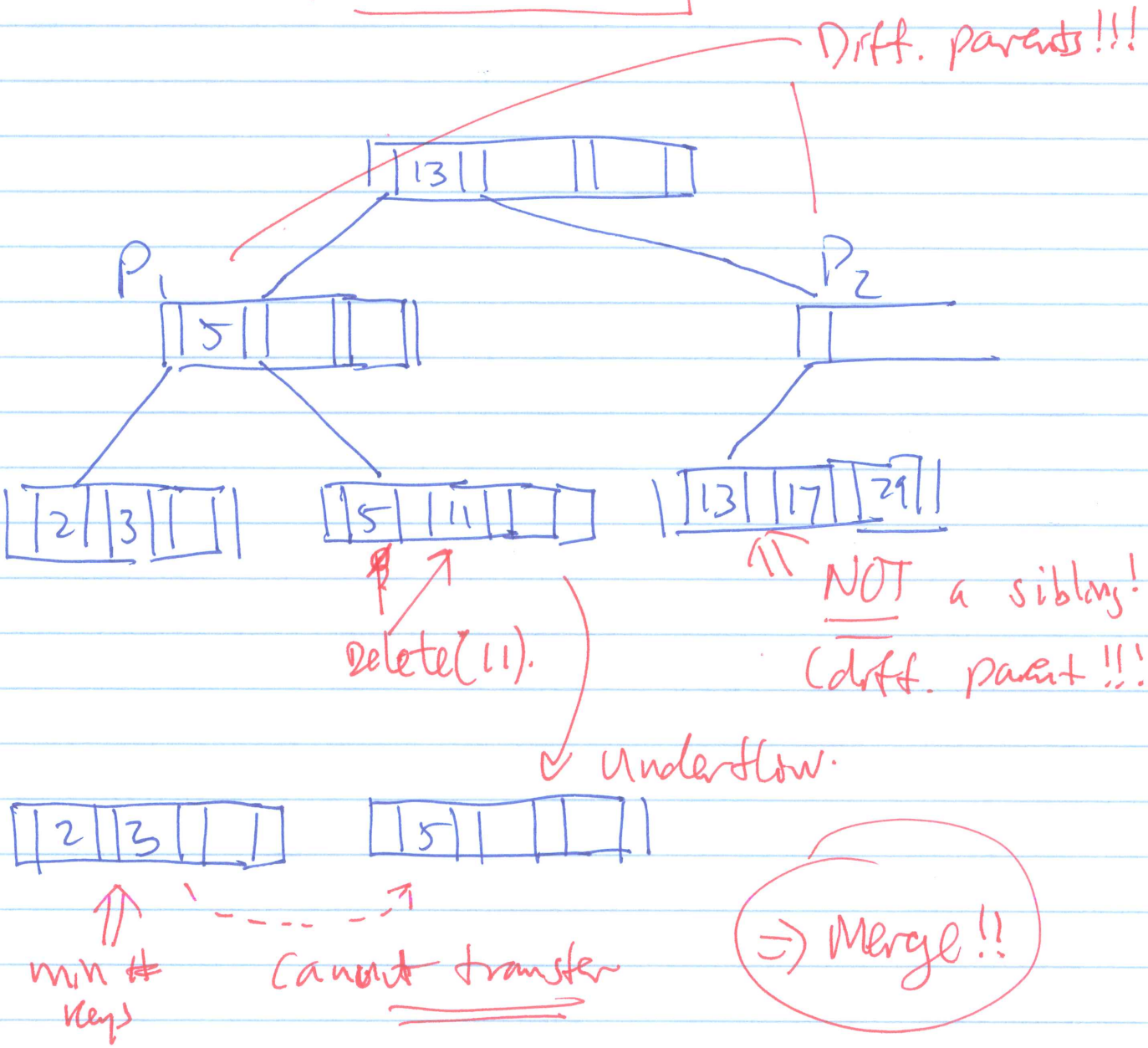
Done

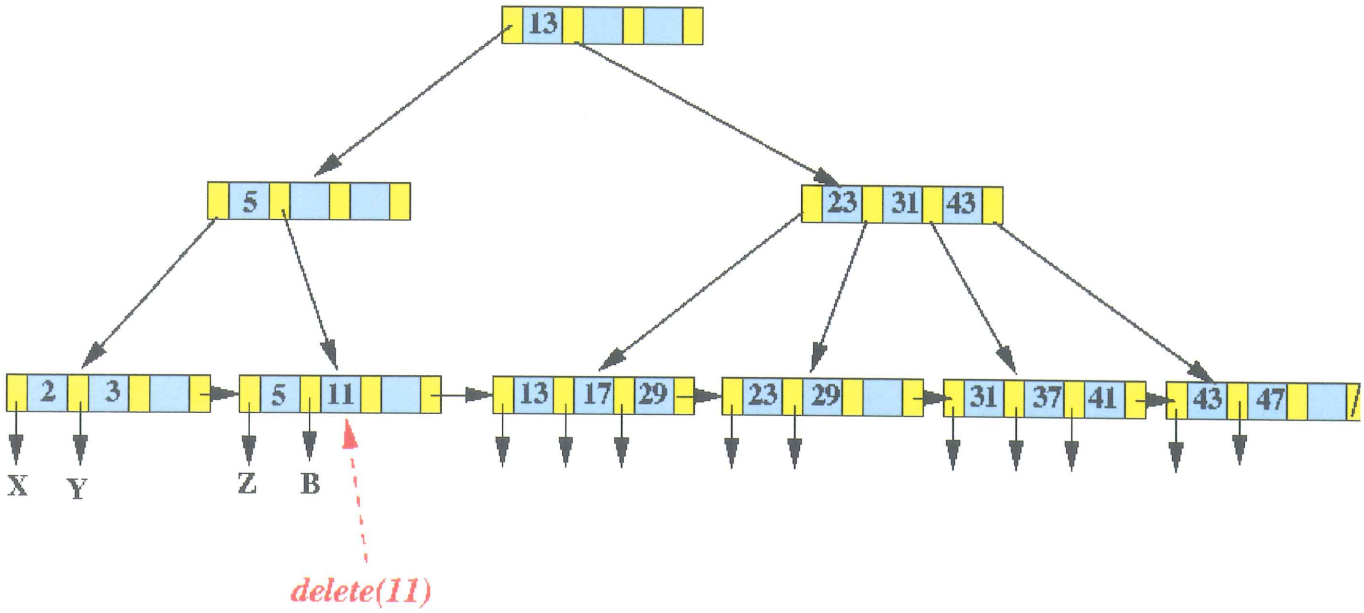
Delete: hard case : merge

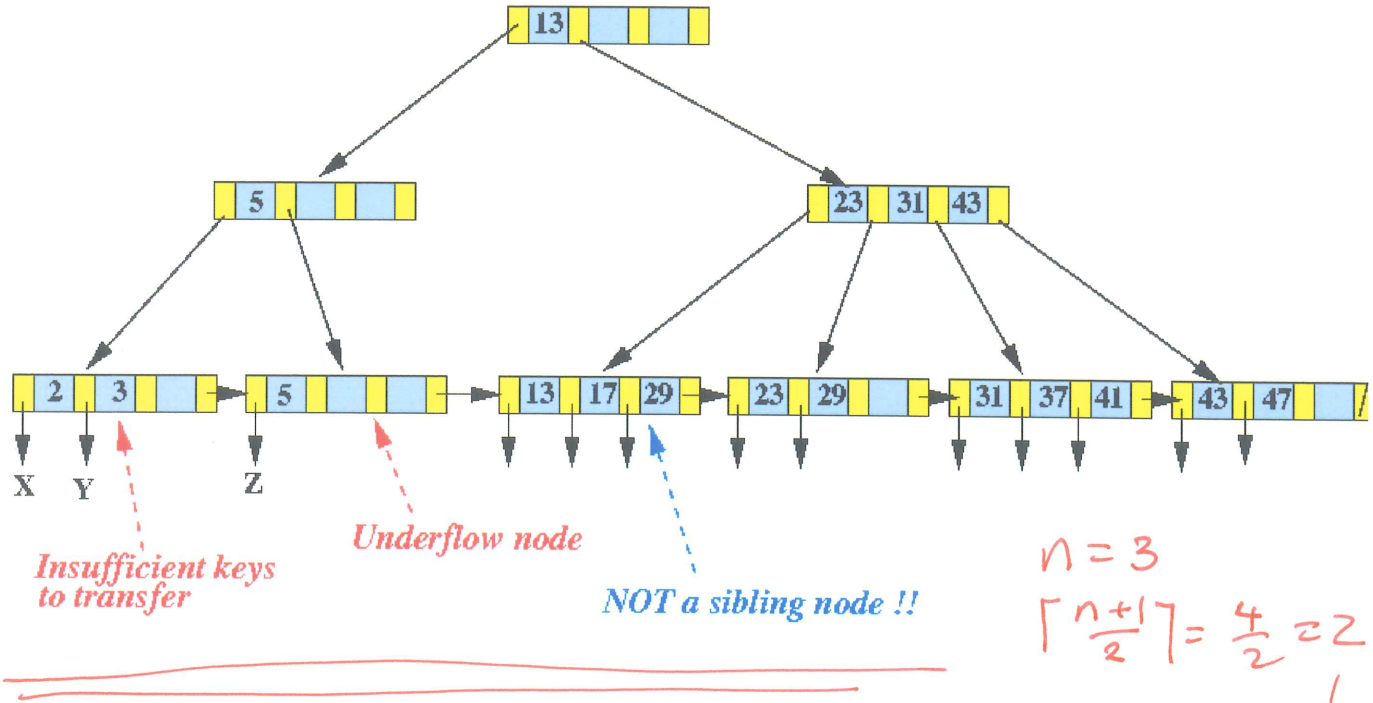
- Start with tree after delete(7)

(from last example)

- Now: delete(11).



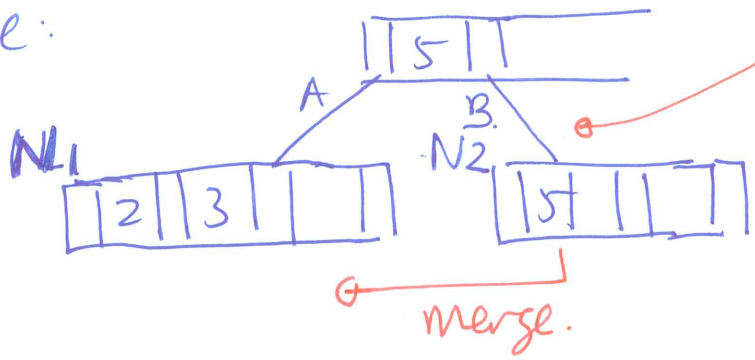

13
5
23 31 43
2 3
5 11
13 17 29
23 29
31 37 41
43 47
X
Y
Z
B
delete(11)



$n = 3$

$\lceil \frac{n+1}{2} \rceil = \frac{4}{2} = 2$

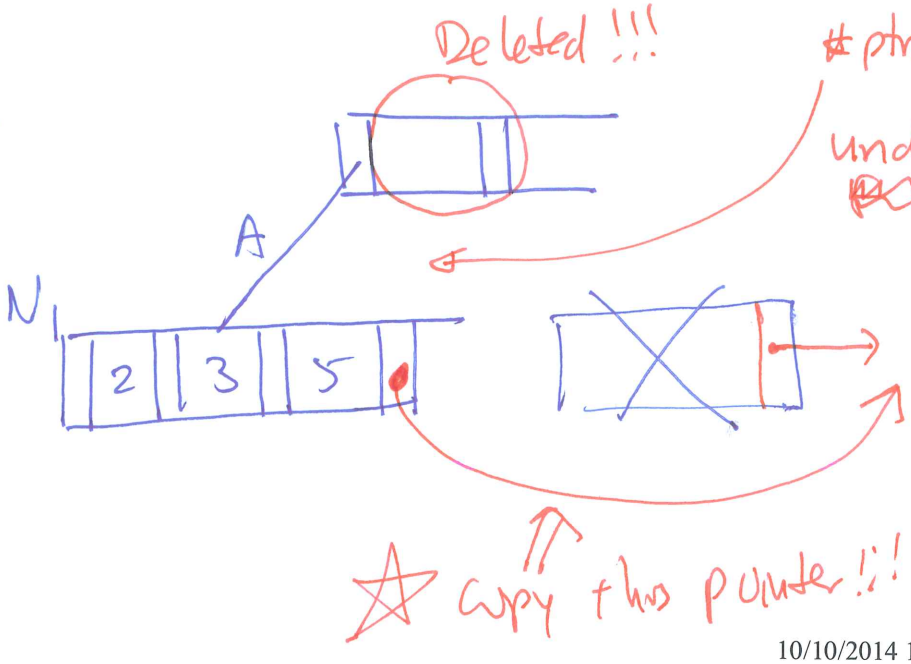
(1) Before merge:



ptrs ≥ 2

OK !!

(2) After merge:

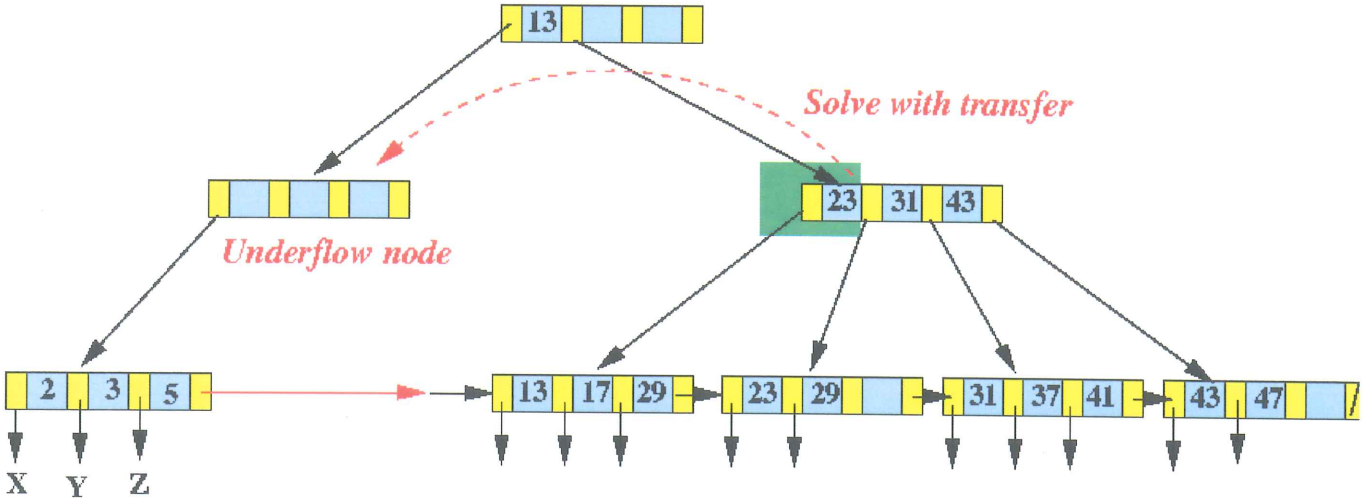


ptrs < 2

underflow

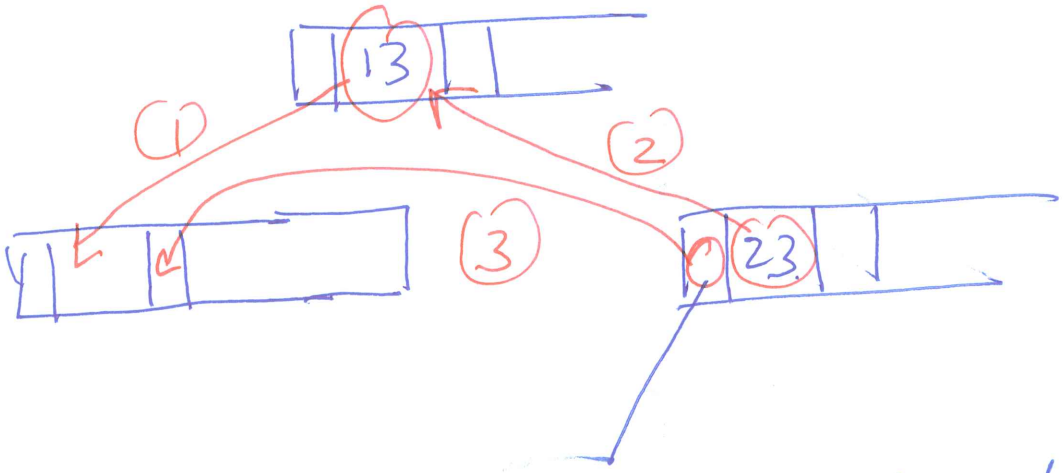
☆ copy this pointer !!!

Situation now:



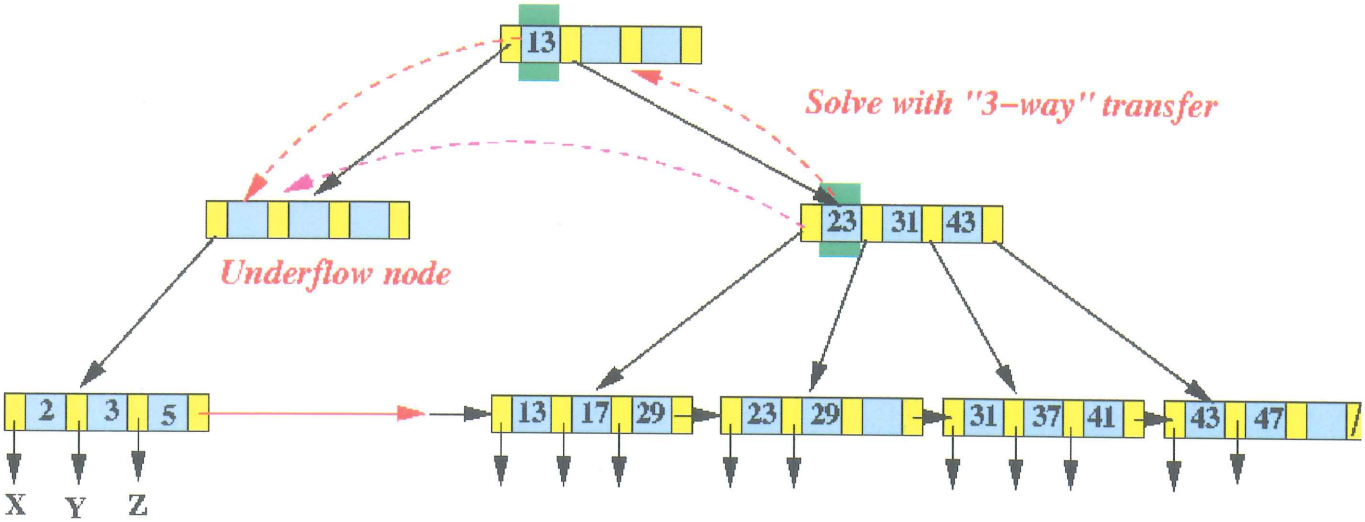
Note: transfer between internal node is different:

It is a 3-way transfer:



(keys + pointers go to different nodes!!!)

this is the "(a,b)-tree" transfer a/s!



This figure shows where the

- keys are transfered
- where a pointer is transfered.

Result: