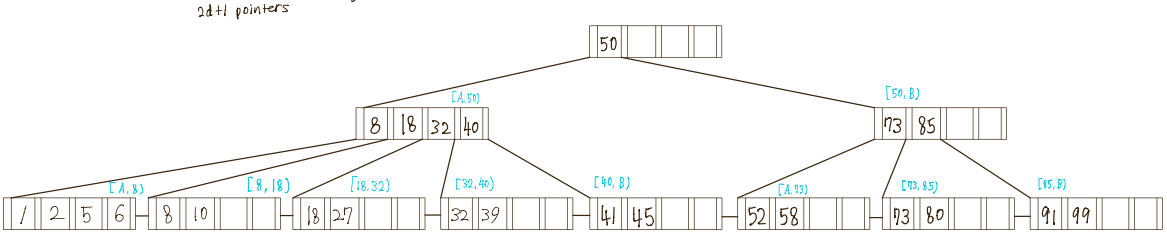
B+ Tree Indexing (Sample Q's)

B+ tree w/ order: 2

min # of keys in a node

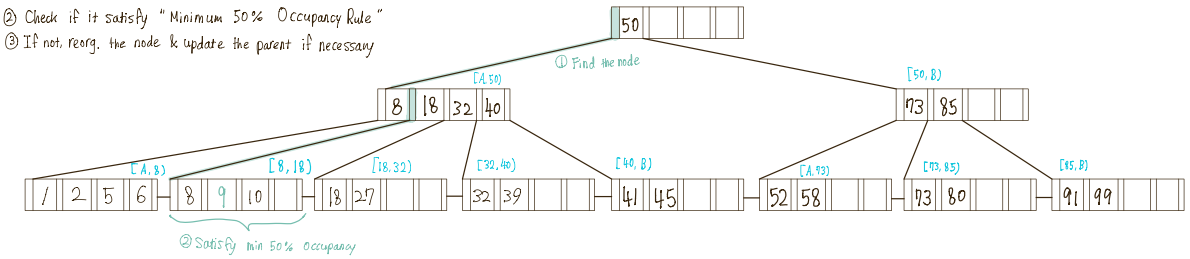
Original B+ Tree:

Each node can have d ~ 2d key values ranging from [0, □)
2d+1 pointers

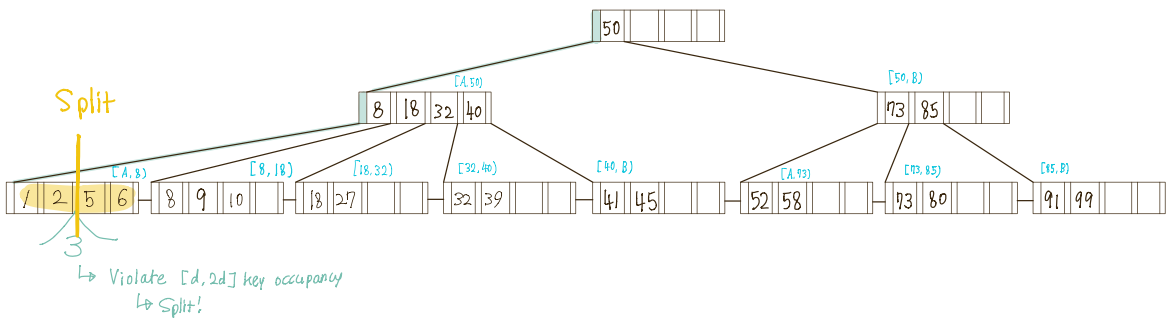


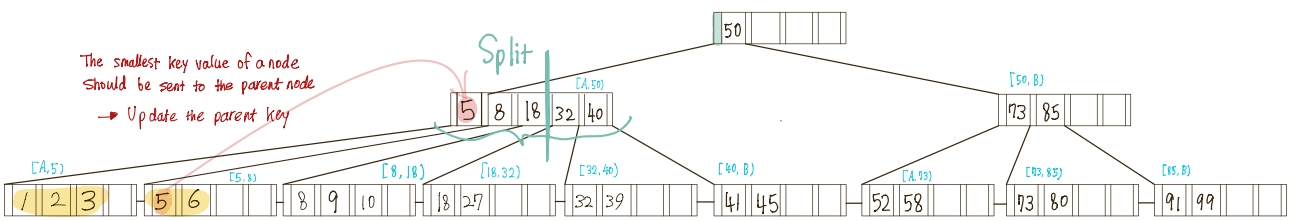
Q1. Insertion: a key "9"

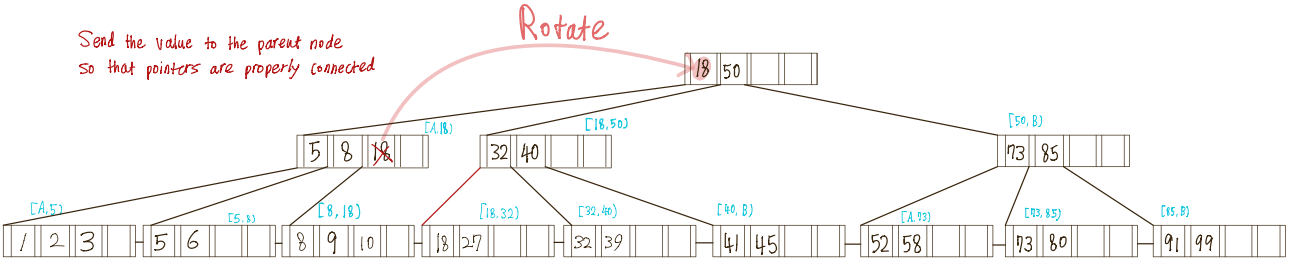
① Find the node
② Check if it satisfy "Minimum 50% Occupancy Rule"
③ If not, reorg. the node & update the parent if necessary



Q2. Insertion: a key "3"

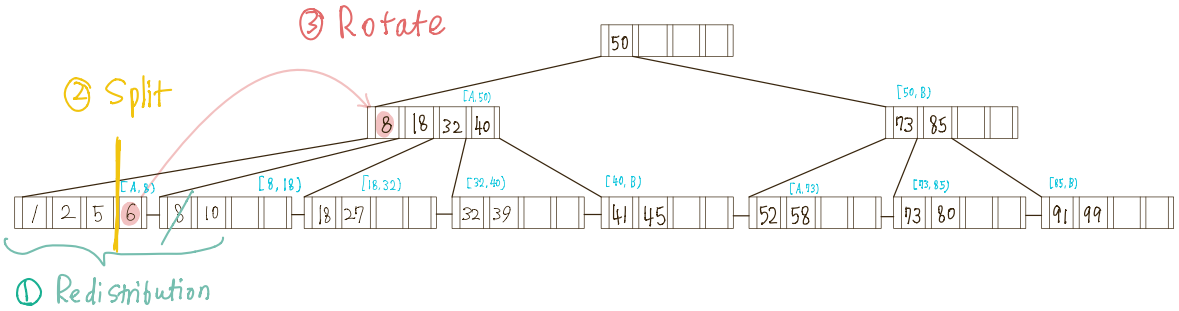


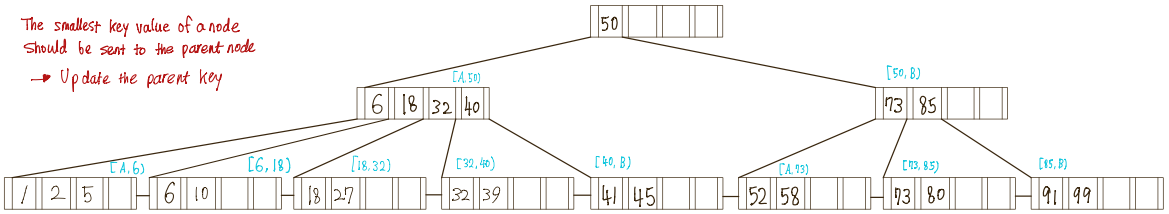




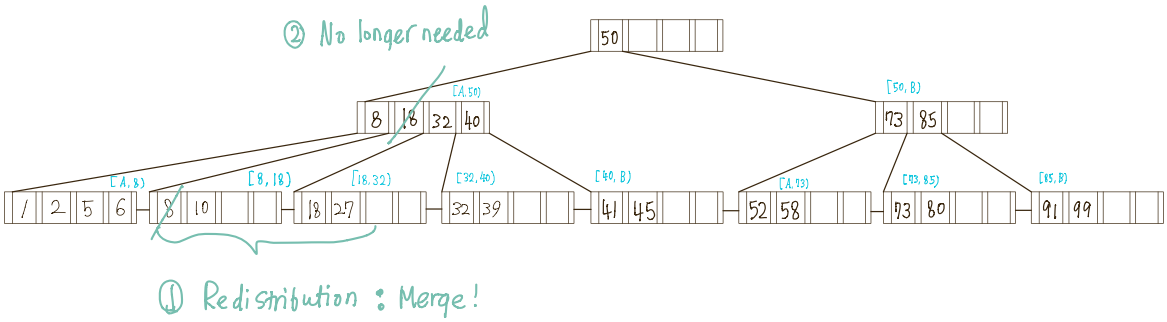
From the original B+ tree :

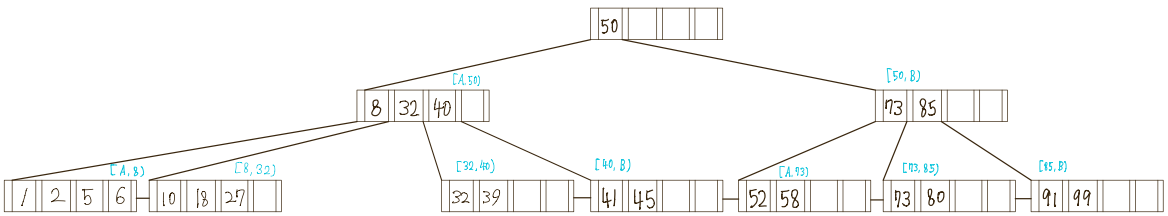
Q3. Deletion : a key "8" → Redistribution w/ the "left" sibling





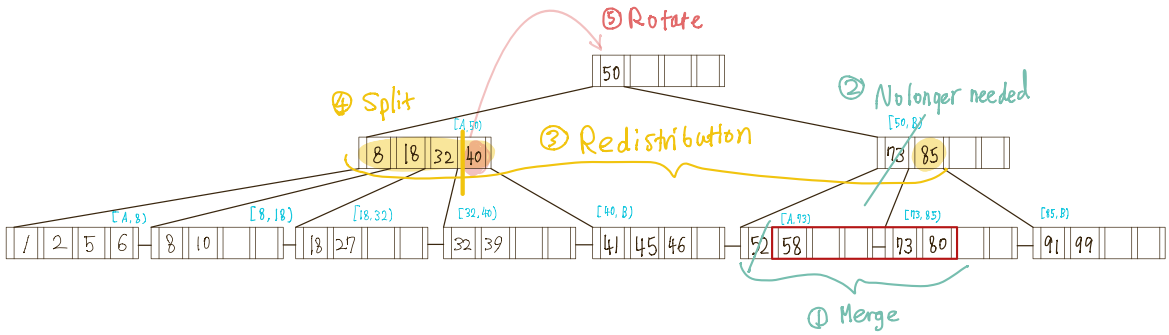
Q4. Deletion : a key "8" → Redistribution w/ the "right" sibling

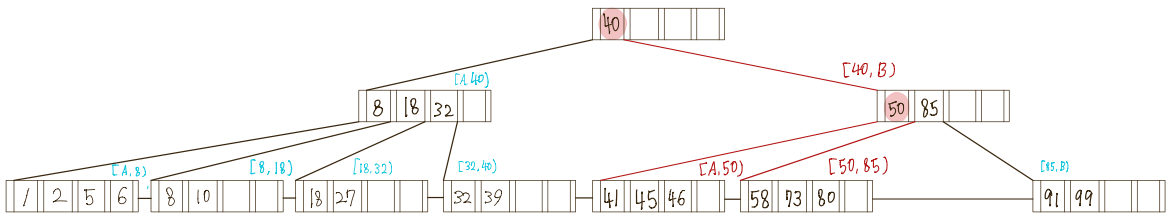




From the original B+ Tree :

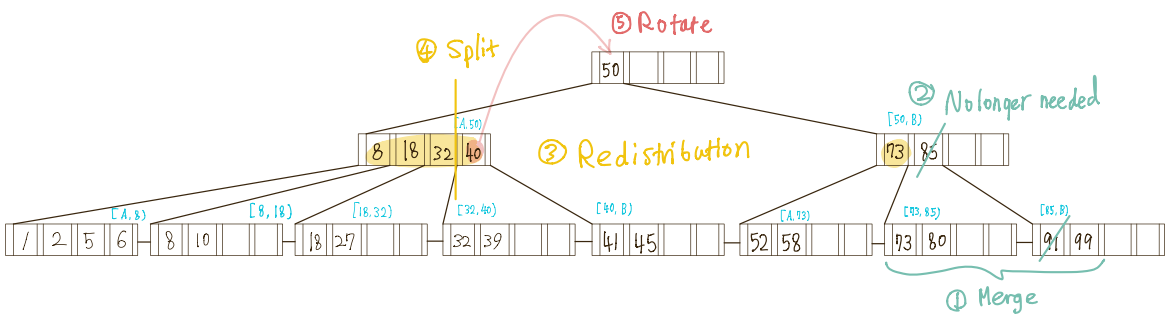
Q5. Insertion of key "46" & Deletion of key "52"

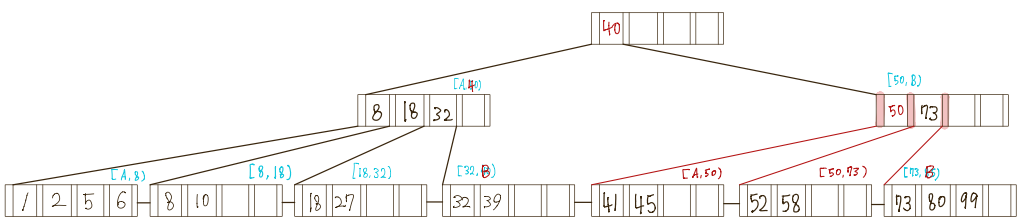




From the original B+ Tree :

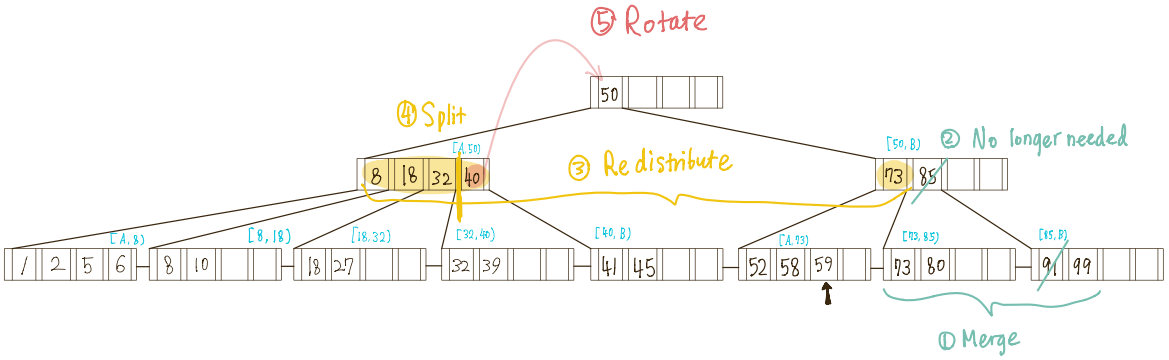
Q6. Deletion of a key "91"

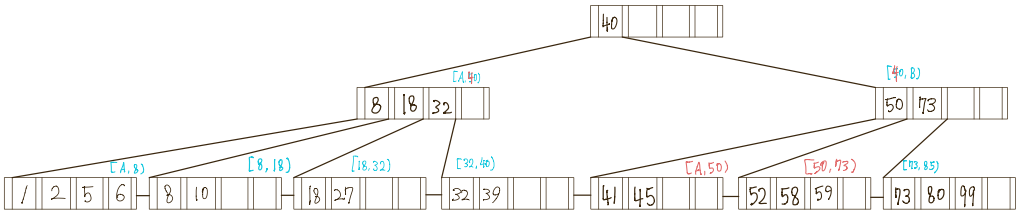




From the original B+ tree:

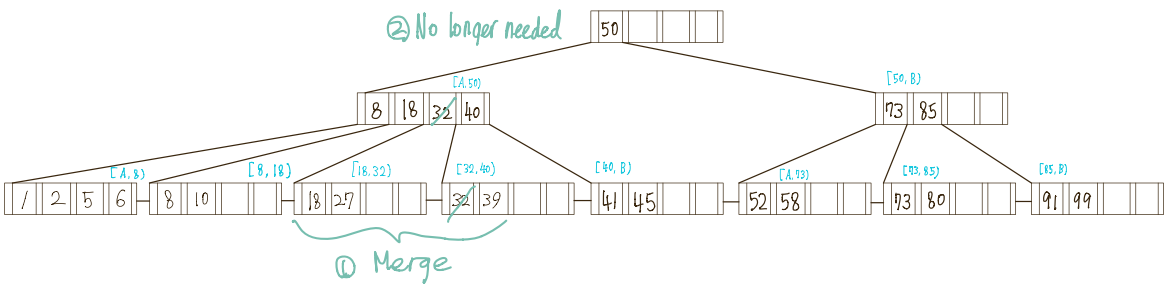
Q7. Insertion of key "59" and Deletion of key "91"

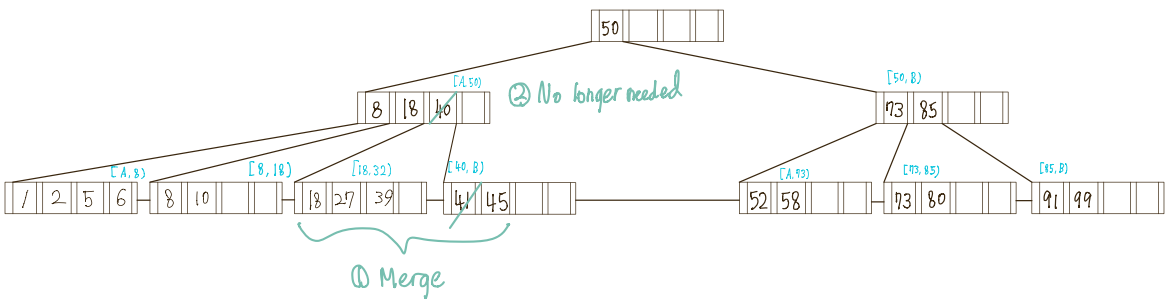


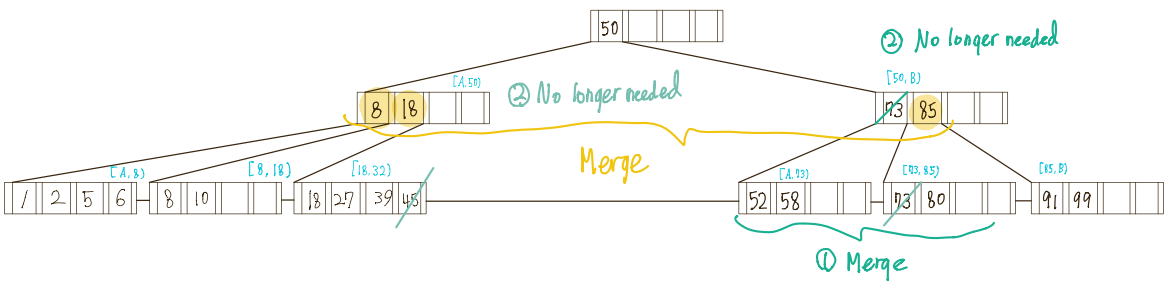


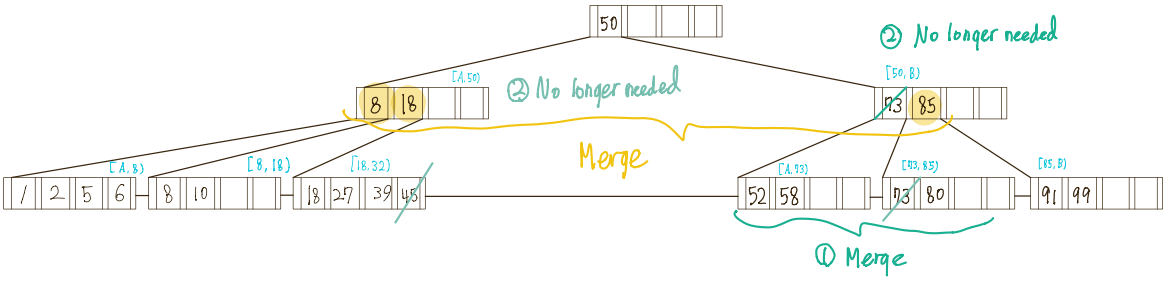
From the original B+ tree:

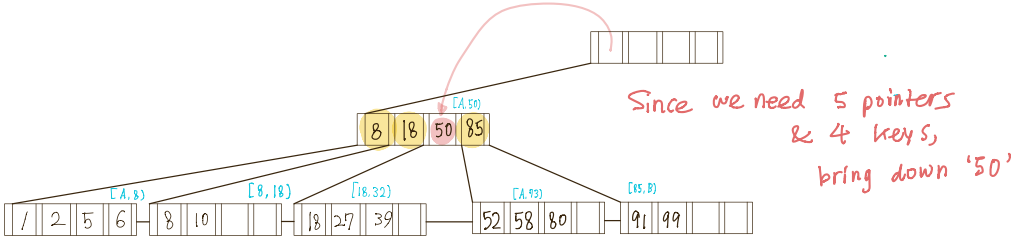
Q8. Deletion of keys 32, 39, 41, 45, and 73











Since we need 5 pointers & 4 keys, bring down '50'

B+ Tree Indexing (HW3)

Q1.

Root Node: $1 \sim 2d$ keys, $(2d+1)$ pointers
Internal Node: $d \sim 2d$ keys, $(2d+1)$ pointers

d : order of B+tree

$$2d \times (\text{key size}) + (2d+1) \times (\text{pointer size}) \leq \text{Block Size}$$

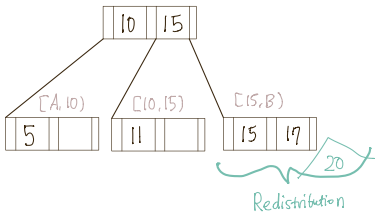
$$\begin{aligned} 2d \times 8 + (2d+1) \times 4 &\leq 30 \\ 16d + 8d + 4 &\leq 30 \\ 24d &\leq 26 \\ d &\leq 1.xx \\ 2d &\leq 2.xx \end{aligned}$$

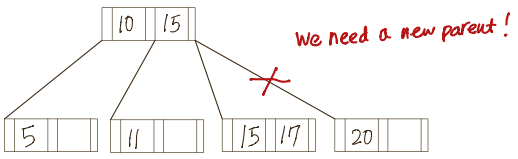
Find out the max. # of keys to fit in a node!

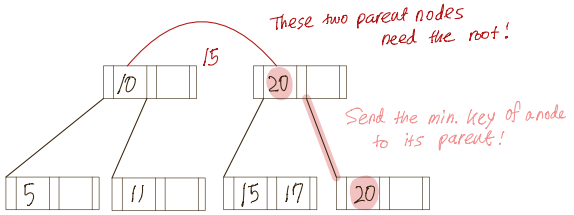
∴ The order of B+ tree : 1
Each node except for root can have $1 \sim 2$ keys, $1 \sim 3$ pointers

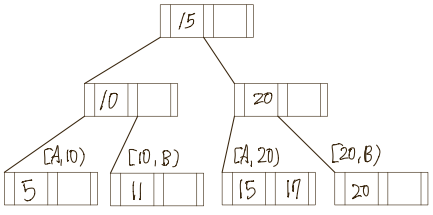
Height: 2→3 when "20" is inserted.
↳ height increases when # of pointers from a parent exceeds 3!

Solution 1. Left-weighted Leaf node

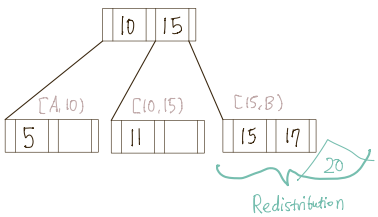


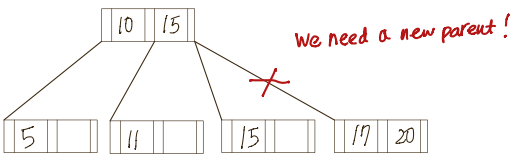


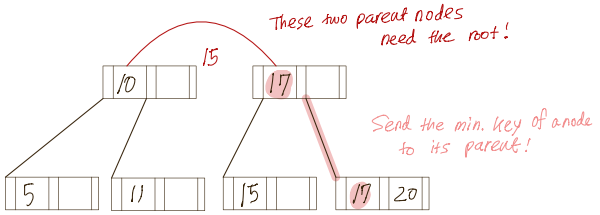


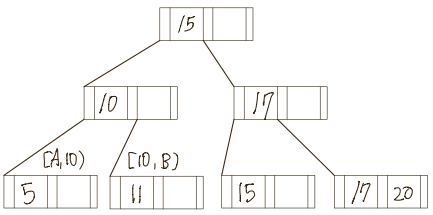


Solution 2. Right-weighted Leaf Node









Q2.

Root Node: $1 \sim 2d$ keys, $(2d+1)$ pointers
Internal Node: $d \sim 2d$ keys, $(2d+1)$ pointers

d : order of B+ tree

$$2d \times (\text{key size}) + (2d+1) \times (\text{pointer size}) \leq \text{Block Size}$$

$2d \times 4 + (2d+1) \times 4 \leq 30$
$8d + 8d + 4 \leq 30$
$16d \leq 26$
$2d \leq 3.xx$

Find the maximum # of keys in a node ($2d$)
Then find an integer w/ min 50% occup.

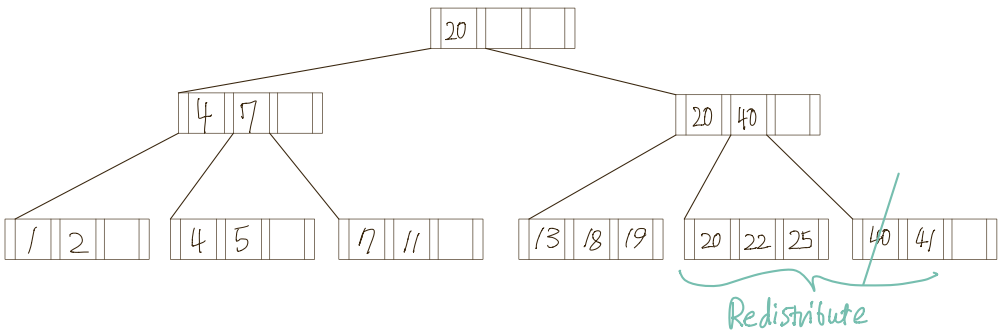
∴ The order of B+ tree : $d = 2$
(= min # of keys in a node)

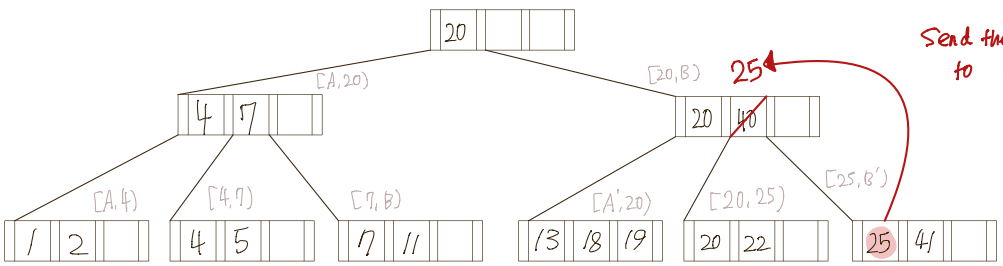
$2d = 3$ max # of keys

$\lceil \frac{2d}{2} \rceil$ by 50% occupancy rule

Each node except for root can have 2~3 keys,
3~4 pointers

Redistribution after deleting "40"
↳ Deletion drops the # of keys below 2





Send the minimum key of a node to its parent node

* Choice of Index

Given the query below,

```
Select *
from   Emp
where  age = 20  and  salary > 20000
```

Clustered B+ tree index on the composite index (age, salary) can be applied to improve the running time.