Rules for DELETING a key:

Rule 1: If the key $k \in$ to the LEAF node x, then delete the key k from x

Rule 2: If the key $k \in$ to the internal node x:

a. if the child y that precedes k in a node x has at least t keys, then find the predecessor k' of k in the subtree rooted at y. Recursively delete k' and replace k by k' in x. Find and delete k' in a single downward pass.

b. if y has fewer than t keys, then, symmetrically, examine the child z that follows k in node x. If z has at least t keys, then find the successor k' of k in the subtree rooted at z. Recursively delete k' and replace k by k' in x. Find and delete k' in a single downward pass.

c. otherwise, if both y and z have only t-1 keys, merge k and all of z into y, so that x loses both k and the pointer to z, and y now contains $2t$-1 keys. Then free z and recursively delete k from y.

Rule 3: If the key $k \notin$ to the internal node x, take $x.c_i$ the root of the subtree that must contain k (if k is in the tree). If $x.c_i$ has only t - *1* keys, then use 3a or 3b to guarantee we descend to a node with $\geq t$ keys

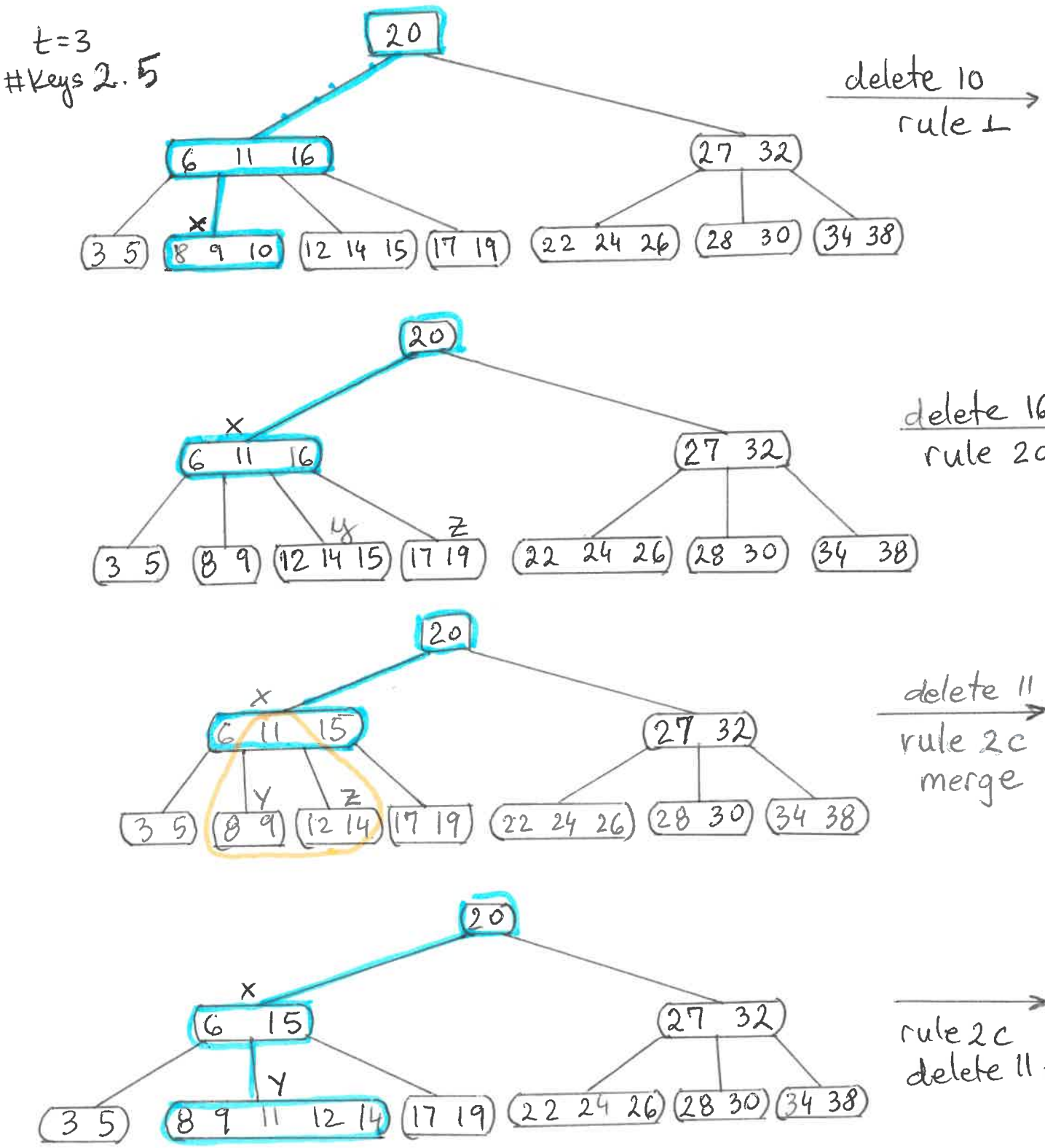
a. if $x.c_i$ has an immediate sibling with $\geq t$ keys, then give $x.c_i$ an extra key by:

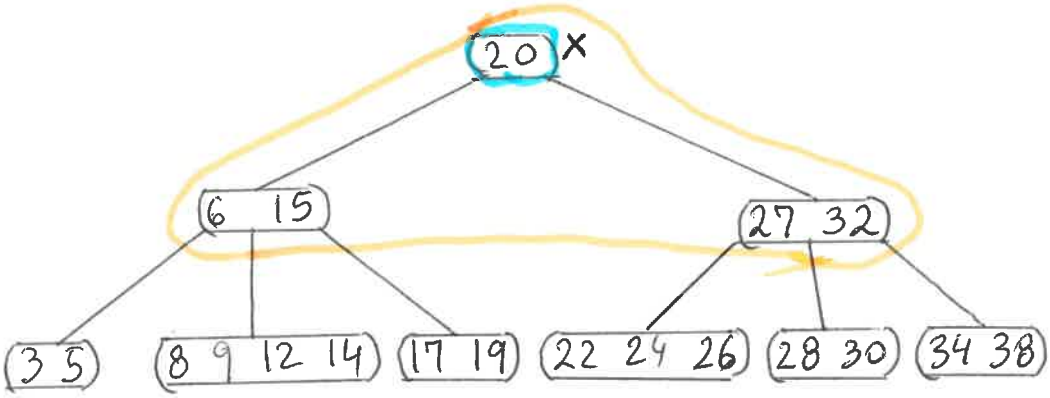
- moving a key from x to $x.c_i$,
- moving a key from $x.c_i$'s immediate left or right sibling up to x,
- moving the appropriate child pointer from the sibling into $x.c_i$

b. if both $x.c_i$'s immediate siblings have t-1 keys, merge $x.c_i$ with one sibling, which involves moving a key from x down into the new merged node to become the median for that node

B-tree delete operation

Example

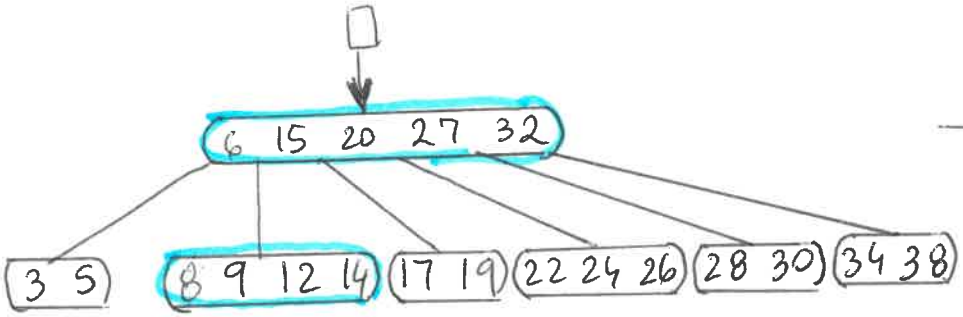


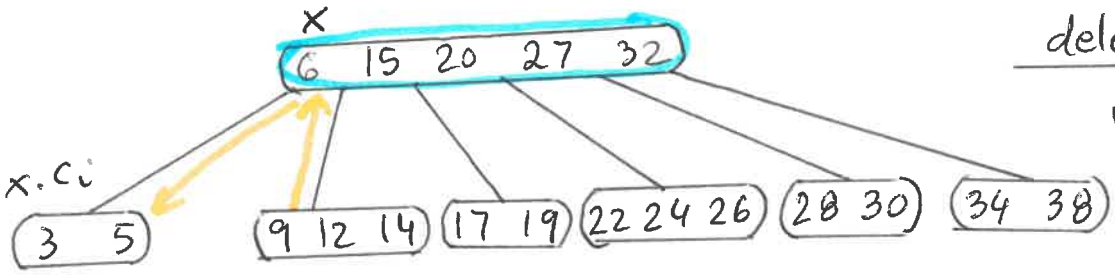


delete 8

rule 3b

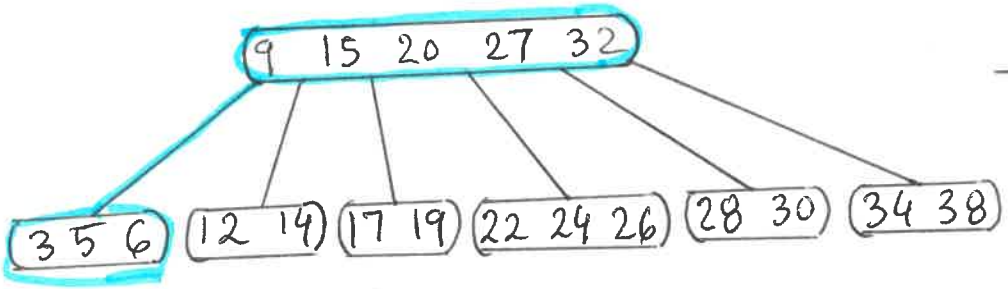
merge

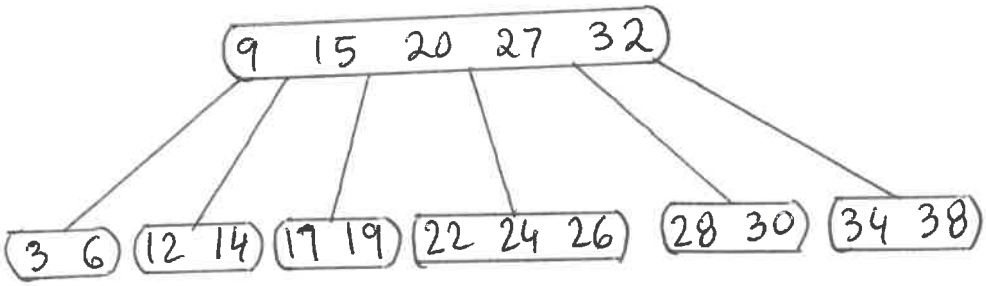




delete 5

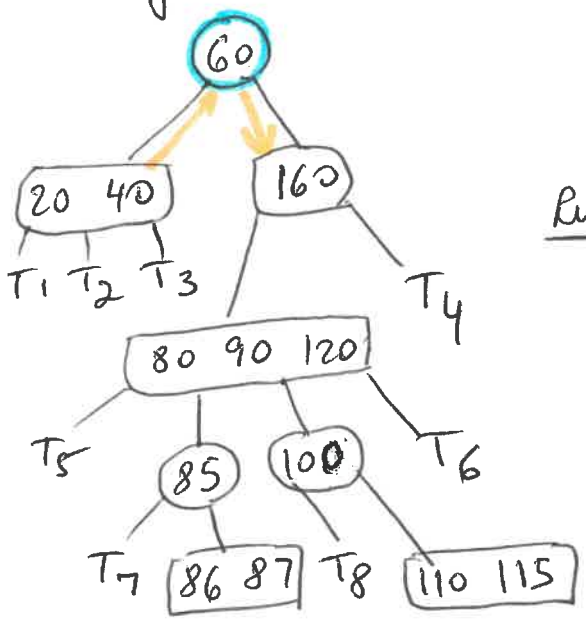
rule 3a



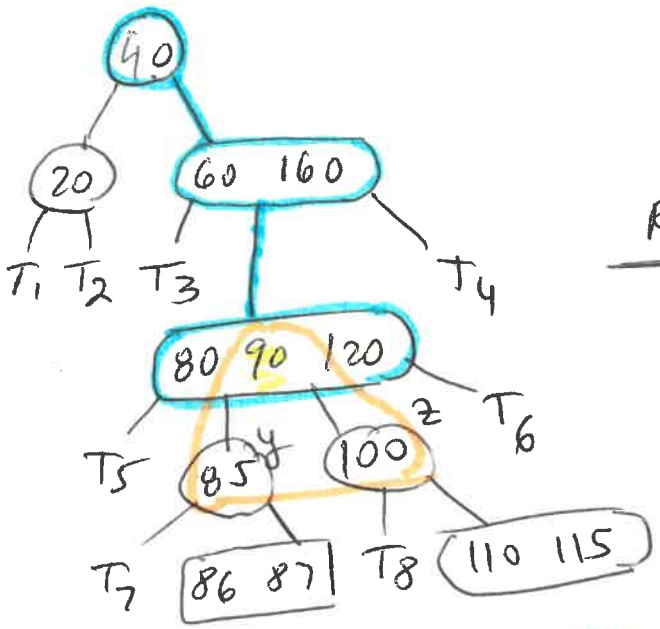


- Let $t = 2$. Delete the key 90 from the following B-tree:

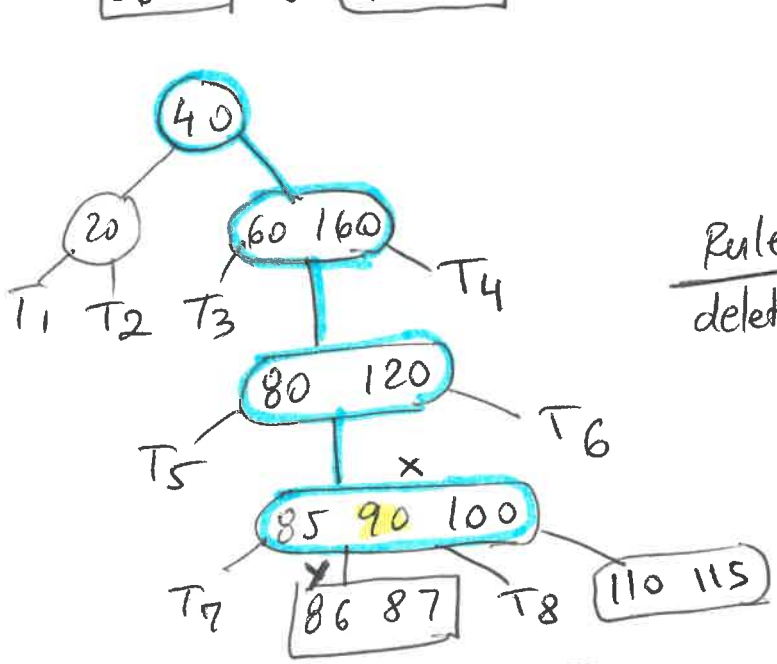
#Keys: 1..3



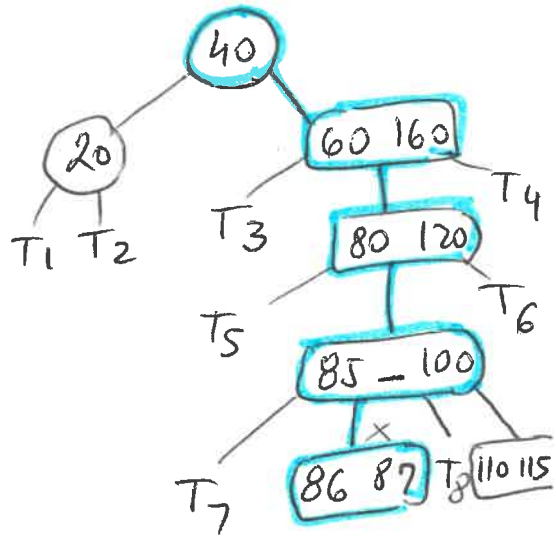
Rule 3a →



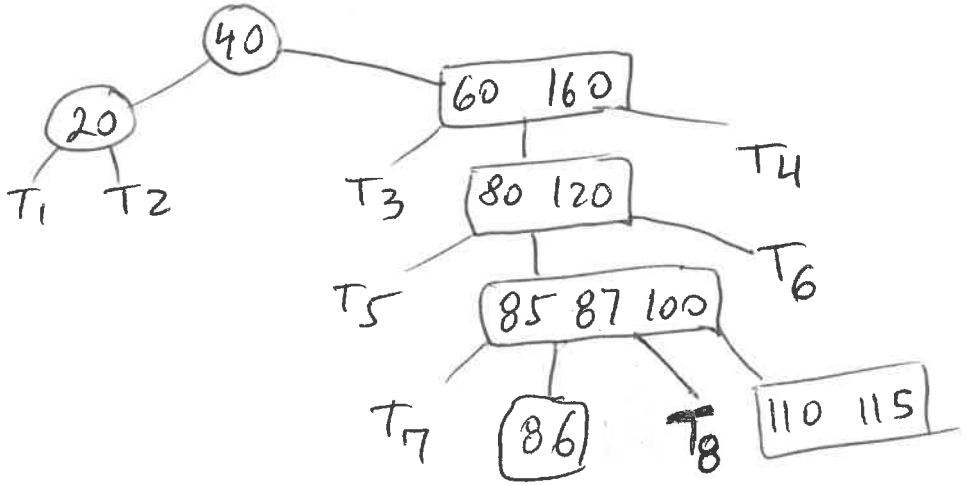
Rule 2c merge →



Rule 2a delete-max(y) →

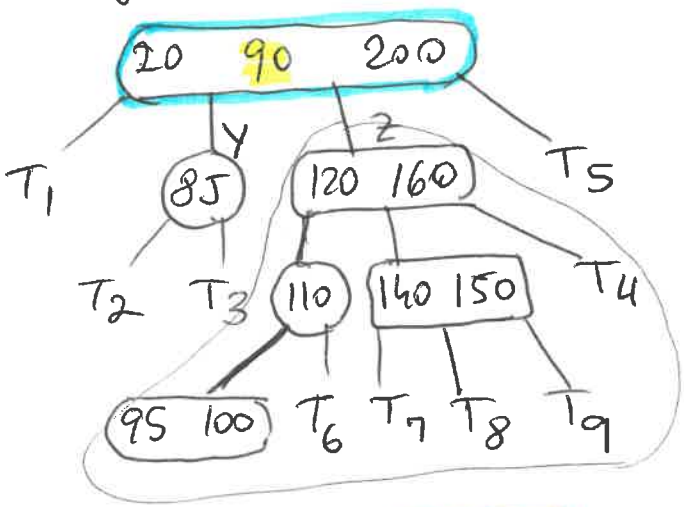


Rule 1 delete-max() →

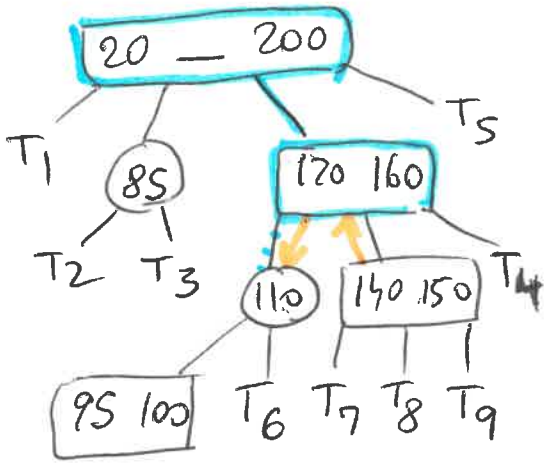


- let $t=2$. Delete the key 90 from the B-tree below:

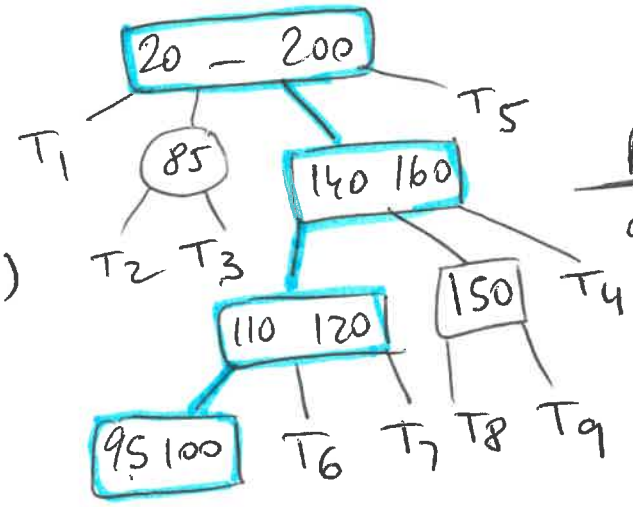
#Keys : 1 .. 3



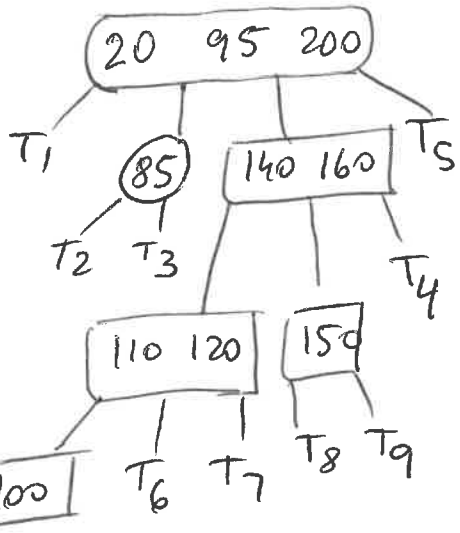
Rule 2b
delete-min(z)



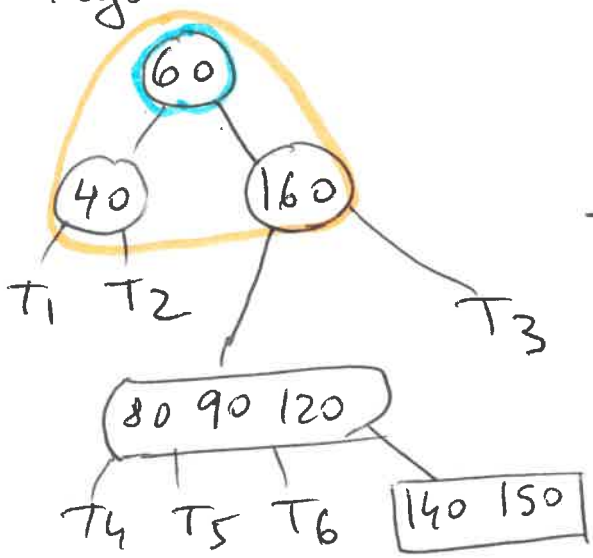
Rule 3a
delete-min()



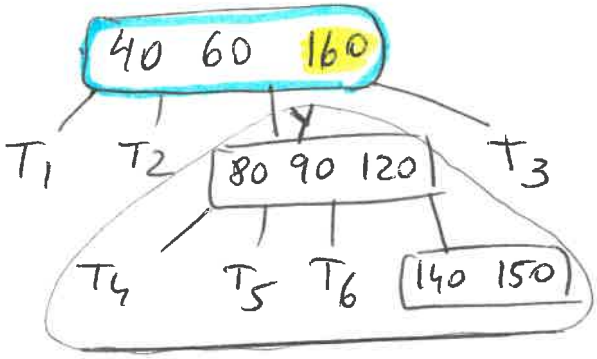
Rule 1
delete-min()



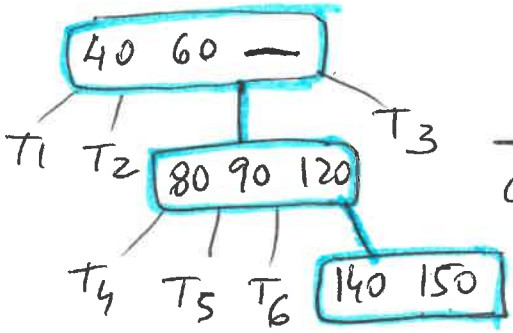
• Let $t=2$. Delete the key 160 from the B-tree below:
#keys: 1..3



Rule 3b
merge



Rule 2a
delete-max(y)



Rule 1
delete-max()