| by Nez
Chounibyou Club

B - tree

Soal :

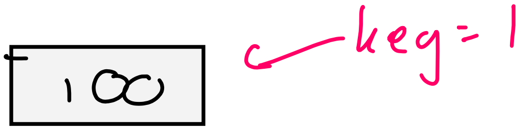
① Insert = 100, 20, 60, 50, 30
② Insert : 40, 90, 45, 25, 65
③ Delete : 90, 20, 60, 45, 50
④ Delete : 100, 40, 25, 65, 20

Soal ①

Isert Algorithm

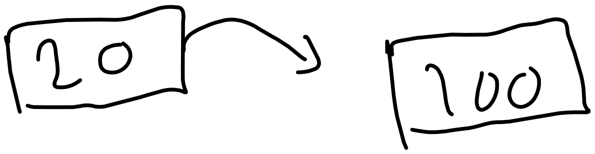
1. Menentukan M (max degree) = 3
 — maka 1 key max 2 ↳ m - 1 min 1 → m/2

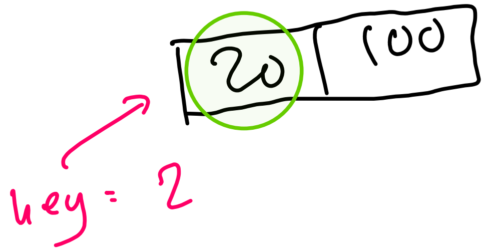
2. Insert 100



karena tree masih kosong jadi langsung saja isi 100

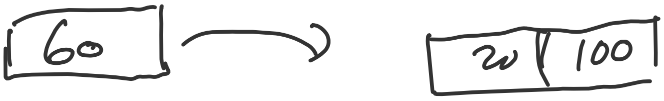
3. insert 20

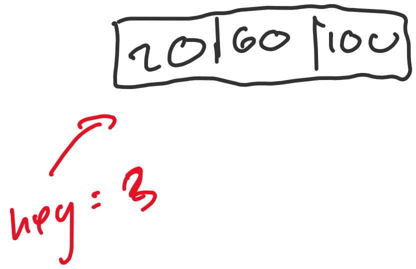




(karena 20 < 100 dan key masi 2 (maximemenuhi!))

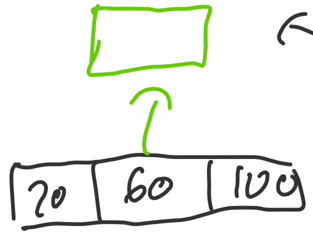
4. insert 60





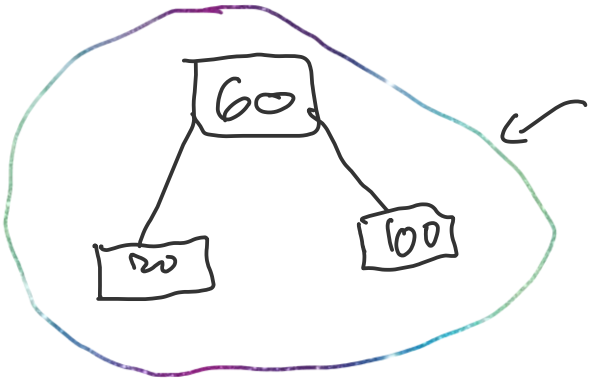
← ini sudah tidak memenuhi syarat (key min 1) dan m-1

∴



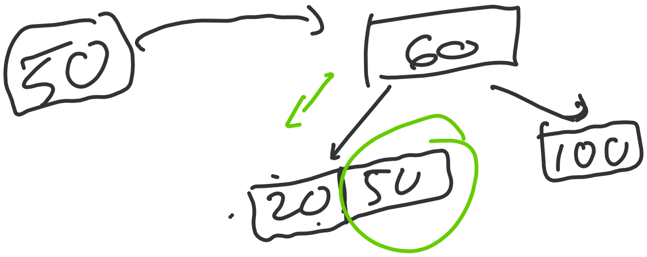
← kita ambil median dari key ini, lalu 60 naik ke atas

- Maka Tree nya akan berbentuk seperti ini



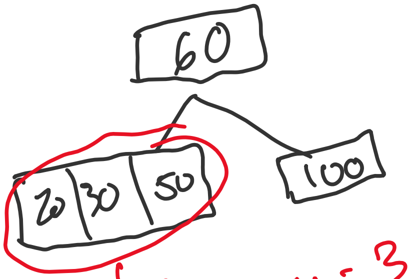
demikian adalah algoritma SPLIT

5. insert 50



kenapa 50 ada di kanan 20? karena 50 > 20 && 50<60

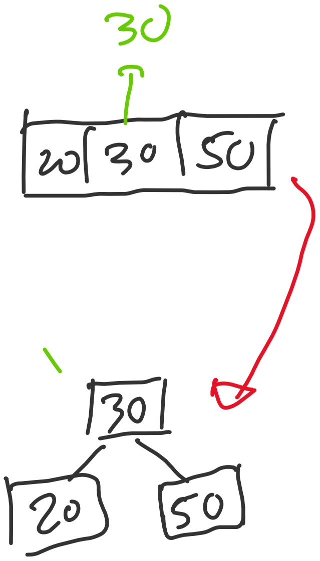
6. Insert 30



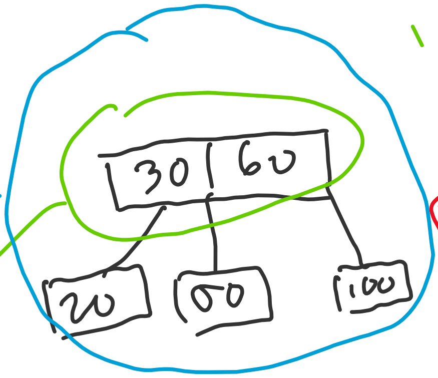
key : 3

melanggar aturan

Maka kita akan melakukan Split



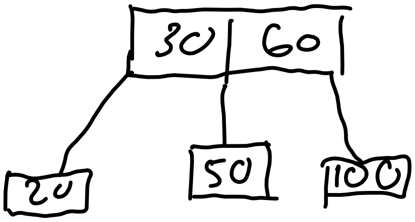
lalu kita hubungkan hasil split kita yaitu 30 dengan parent nya



lalu kita cek apakah parent nya memenuhi syarat sebuah key atau tidak dalam kasus ini Syaratnya adalah min key $m/2$ dan max $m-1$

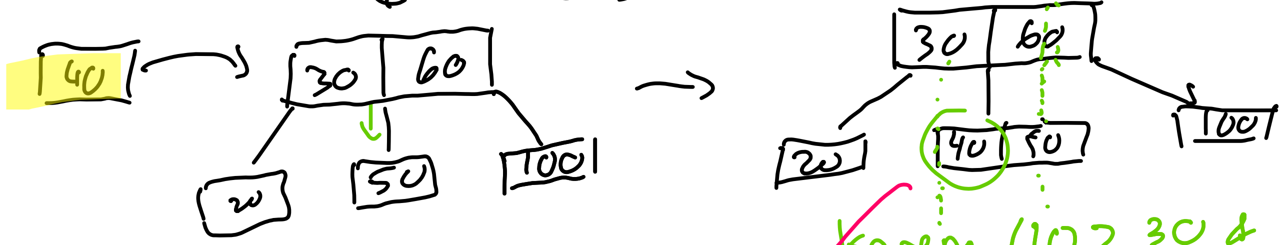
karena parent nya memenuhi maka tree ini benar

Soal 2} dari Tree →



1. insert 40

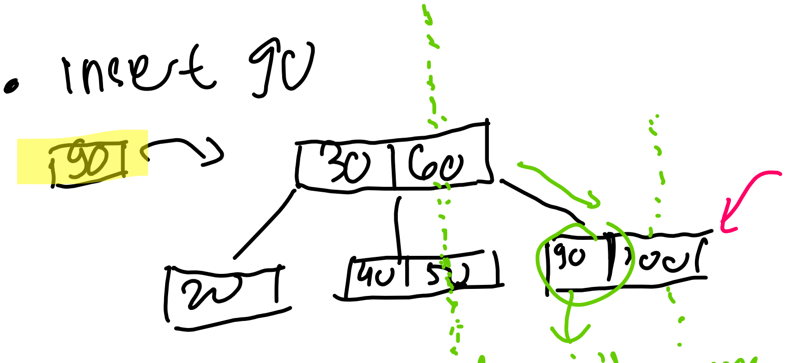
40 ada di antara 30 & 60



Karena 40 > 30 & 40 < 50

key : 2 ✓

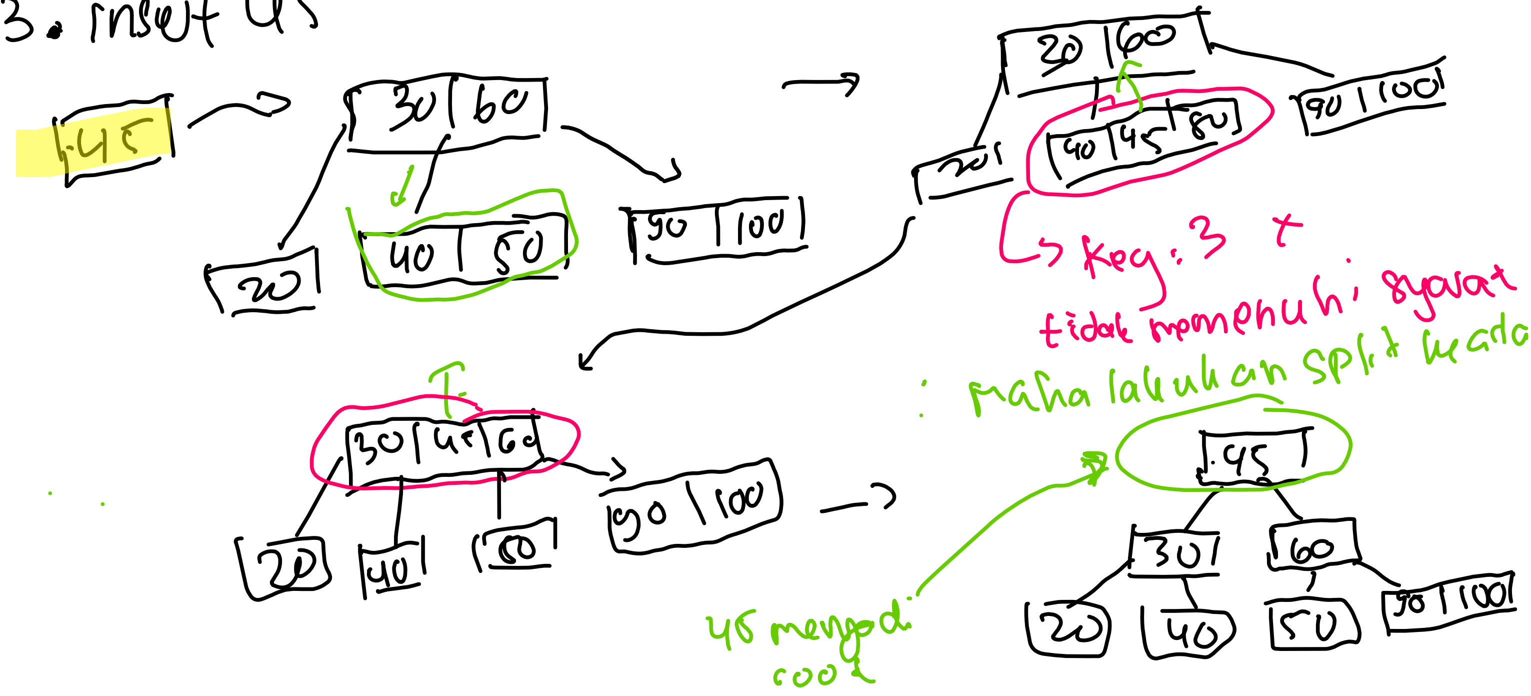
2. insert 90



key: 2 ∴ masi memenuhi syarat

90 disini karena 90 > 60 & 90 < 100

3. insert 45



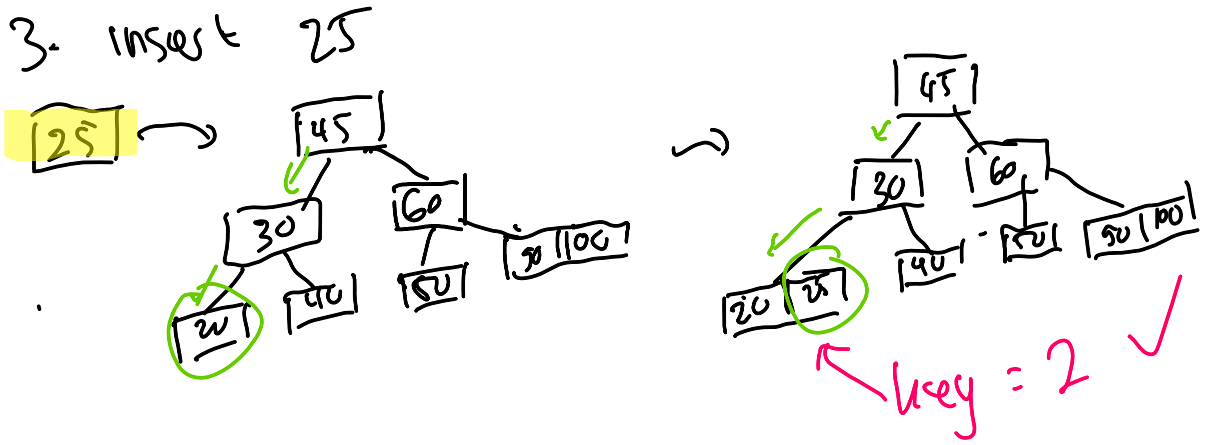
key: 3 ✗

tidak memenuhi syarat

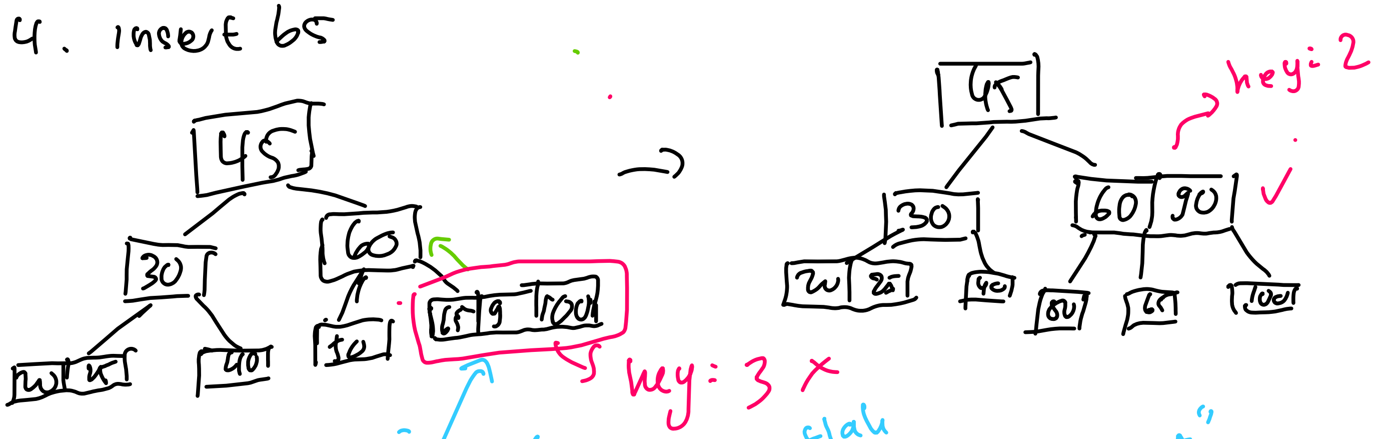
∴ maka lakukan split keatas

45 menjadi root

3. insert 25



4. insert 65



Note : urutan key ini mutlak jadi yang naik 90 "nomatter what"

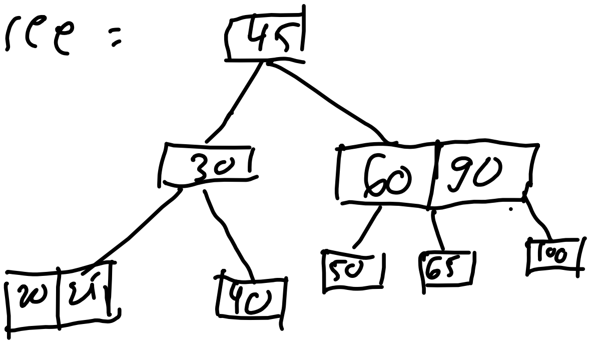
Soal 5

Note 2

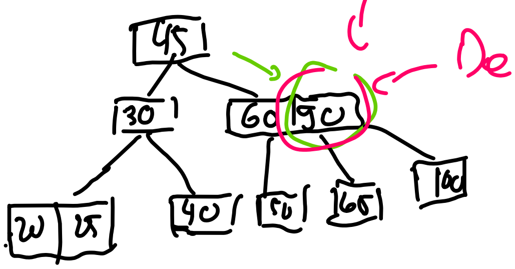
Delete algorithm terbagi dalam 2 kasus

① kasus dimana target key berada di Node leaf / node yang menunjuk NuLL

② kasus dimana target key menunjuk internal node / menunjuk parents tree :



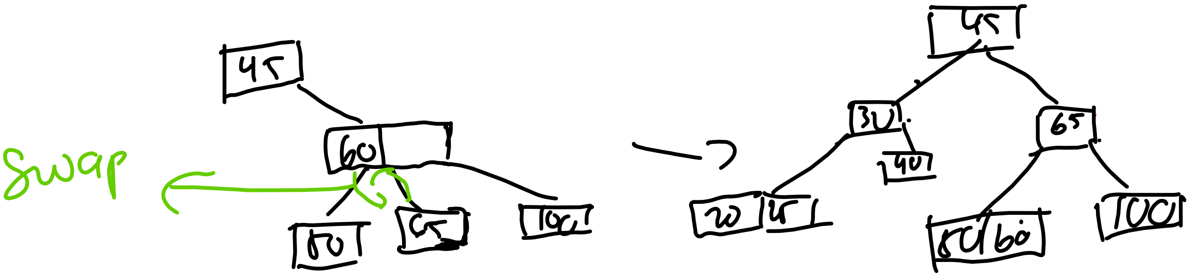
1. Delete 90



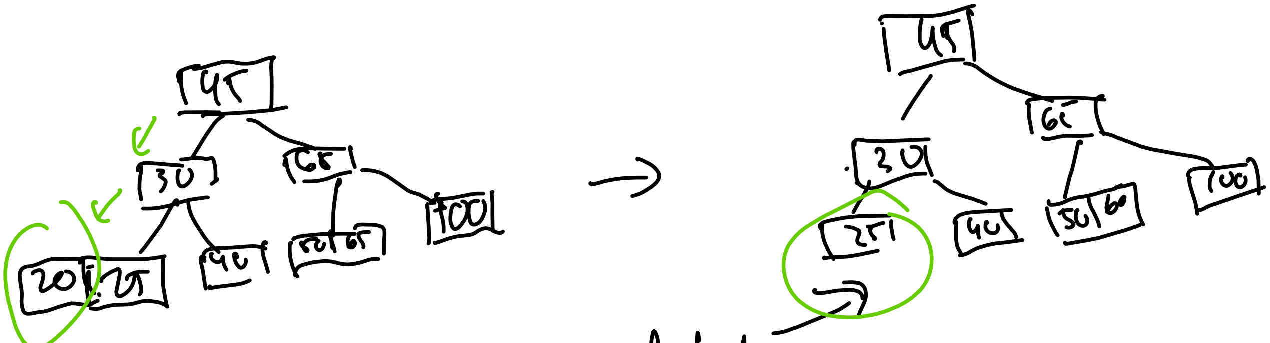
maka di node ini harus 1 key karena tadi key nya 2 trus di delete /

Delete

lalu kita merge



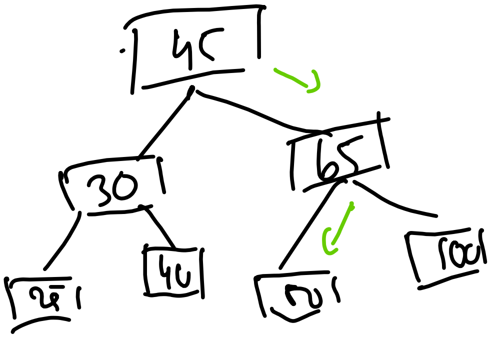
2. Delete 20
find



Langsung delete
karena 20 di leaf dan masi mempunyai key lain dalam 1 node

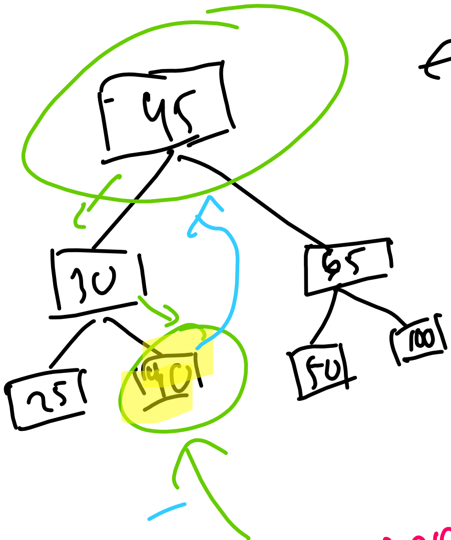
3. Delete 60

Sama seperti (2.) langsung delete



4. Delete 45

find 45 =

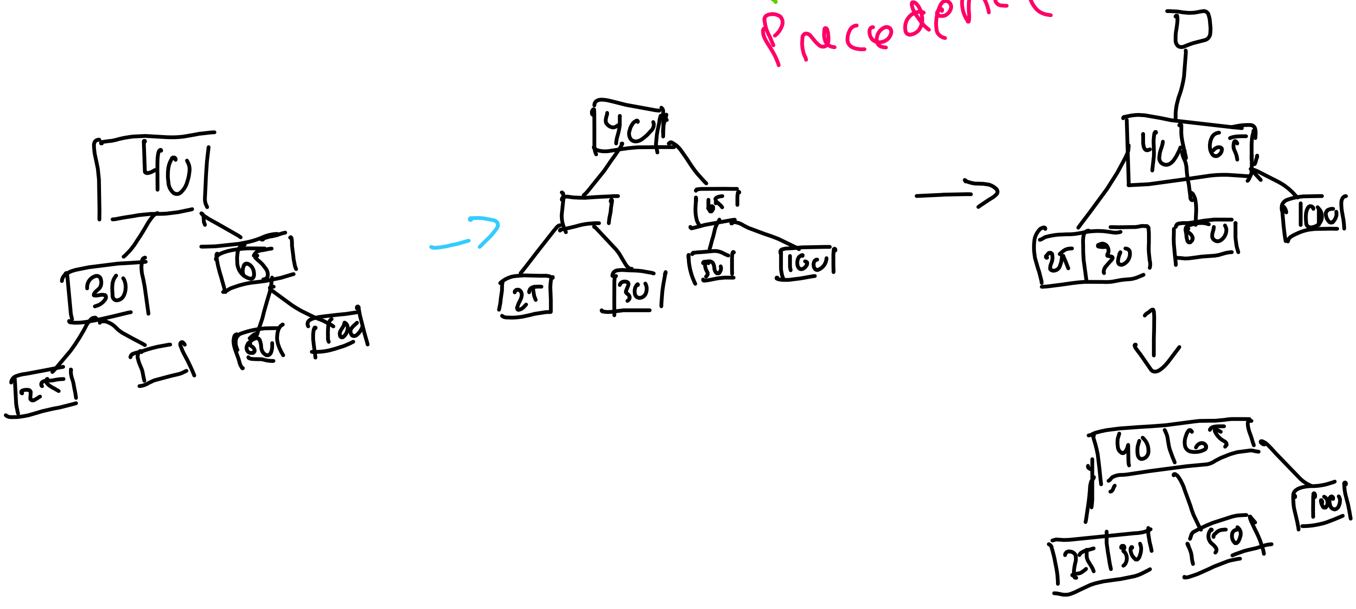


kita akan pakai precedence

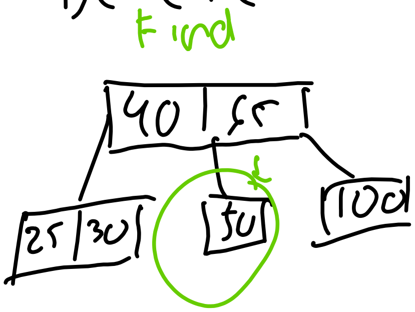
jadi

ke kiri 1 x lalu ke kanan terus

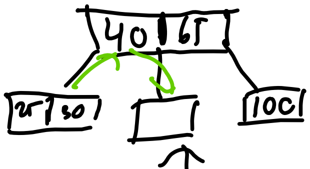
Precedence



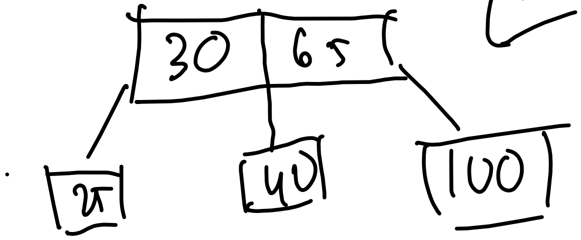
5. Delete 50



→

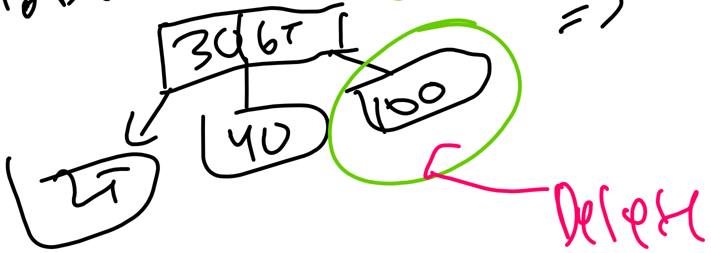


karena di dalam Node ini key nya terlalu sedikit maka Pinjam sebelah

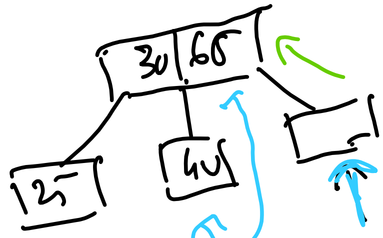


Soal 4)

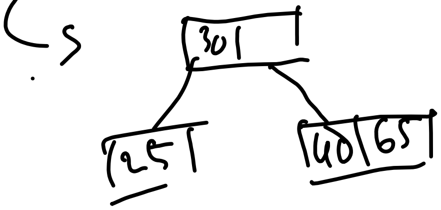
1. Delete 100



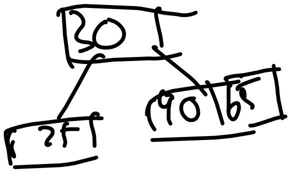
⇒



kurang dari m/2 maka harus minjam sebelah

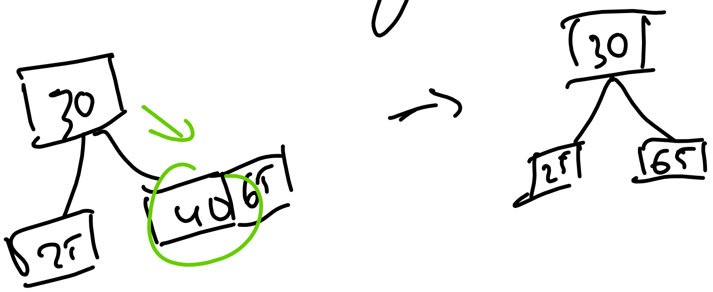


→

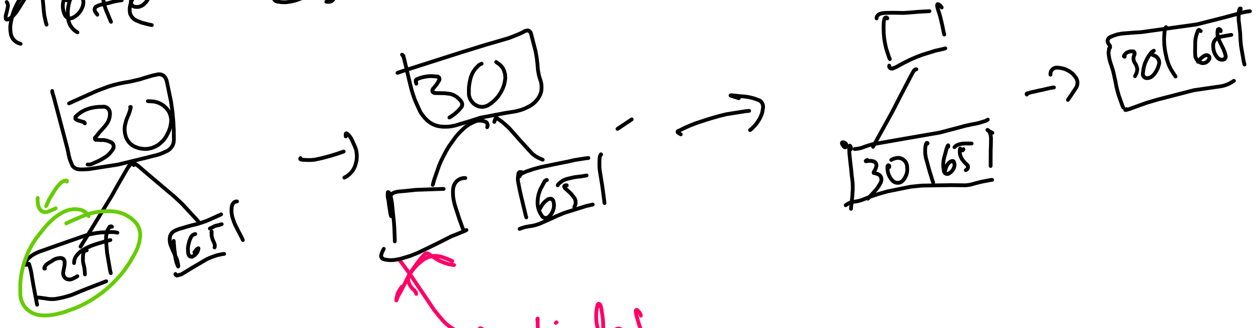


2. Delete 40

langsung Delete saja



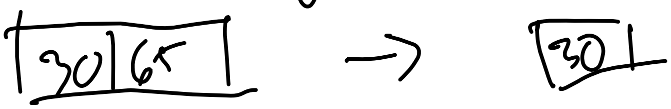
3. Delete 25



tidak memenuhi syarat (m/2) / min key

4. Delete 65

lansung delete saja



5. Delete 30

lansung delete saja

30 → Null

TAPI BOONG!!

ini tidak bisa terjadi karena saat kita delete 30 node nya tidak memenuhi syarat (m/2) jadi tidak bisa di delete

tapi untuk kasus ini kita bisa buat validasinya