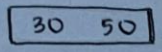
1. a) Insert : 30, 50, 80, 20, 40, 100, 25, 35

B-tree (max.degree = 3) ← max children = 3
max key sehap node = 2
min key sehap node = 1

① Insert 30

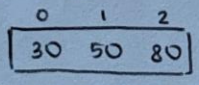
Karena tree kosong, key bisa langsung di-insert di root node.

② Insert 50



Karena max key dalam sehap node = 2, 50 masih bisa di-insert dalam node tersebut (tidak melanggar properti)

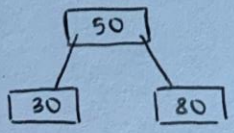
③ Insert 80



Melanggar properti (node sudah penuh), maka harus di-split.

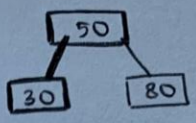
Karena node penuh, key yang berada di tengah (median) akan dilempar ke atas.

Setelah di-split :



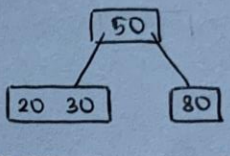
Semua key yang berada di kiri median akan menjadi anak kiri median, dan semua key di sebelah kanan menjadi anak kanan median.

④ Insert 20



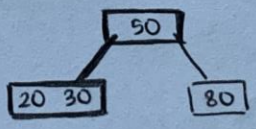
Karena 20 < 50, maka masuk ke sebelah kiri (sama seperti BST)

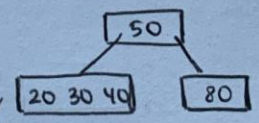
Setelah bertemu node leaf, insert key.



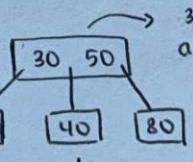
Node memiliki 2 key (tidak melanggar properti), jadi tidak perlu split.

⑤ Insert 40





Ternyata node melebihi max key, jadi perlu split (sama seperti poin 3)

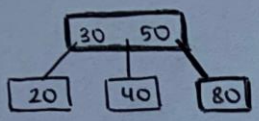


30 dilempar ke atas. ternyata node tidak penuh jadi tidak ada masalah.

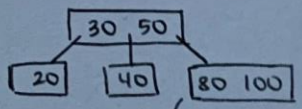
Elemen (key) di sebelah kiri 30 jadi anak kiri 30

Key di sebelah kanan 30 jadi anak kanan 30

⑥ Insert 100

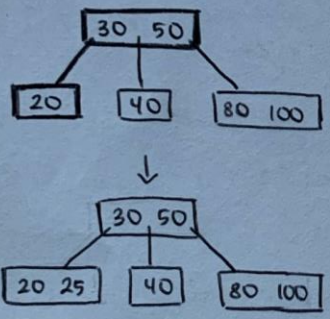


Setelah ketemu tempatnya, kita insert key-nya.



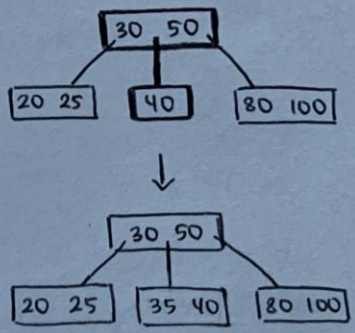
↳ jumlah key dalam node tidak melebihi max key, jadi tidak perlu split.

⑦ Insert 25



Penjelasan kurang lebih sama dengan poin ⑥.

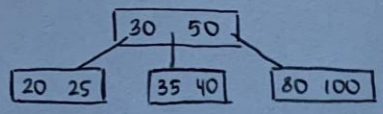
⑧ Insert 35



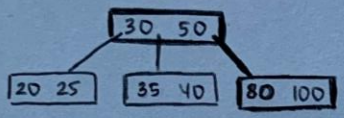
Mirip dengan poin ⑥ dan ⑦.

1. b) Search : 80, 100, 25

① Search 80

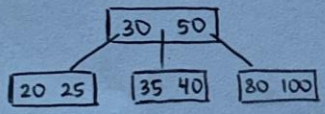


Pertama, bandingkan nilai yang ingin di-search dengan key yang ada pada node root. 80 > 50, maka akan masuk ke subtree sebelah kanan dari 50.

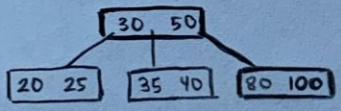


Element 80 found

② Search 100

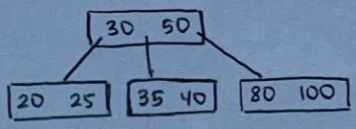


100 > 50, maka masuk ke subtree kanan.



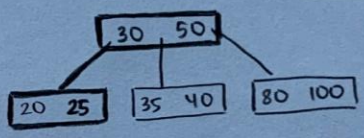
Element 100 found

③ Search 25



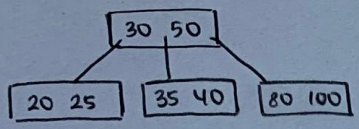
-> Bandingkan 25 dengan key pada node root.

-> Karena 25 < 30, maka masuk subtree kiri dari 30

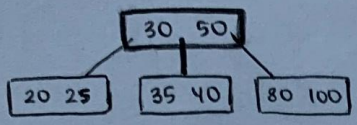


Element 25 found

1. c) Delete : 50, 20, 100

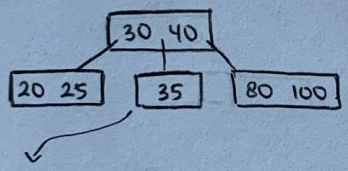


① Delete 50



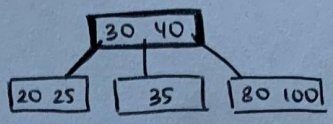
Key yang ingin dihapus ternyata berada pada internal node. Maka, yang harus dilakukan adalah mencari predecessor (key maksimal yang terletak di kiri key yang ingin kita hapus)

Predecessornya adalah 40, maka key yang ingin kita hapus (50) akan kita replace dengan predecessor yaitu 40.

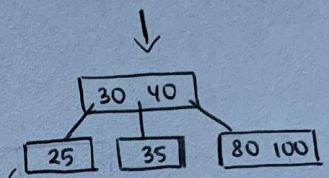


Setelah 50 di-replace dengan 40, ternyata node asal dari 40 masih memenuhi properti (min key = 1). Jadi tidak ada masalah / langkah lain yang harus dikerjakan.

② Delete 20

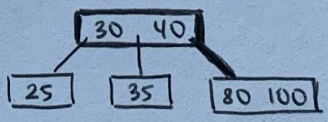


Key yang ingin dihapus berada pada leaf node.

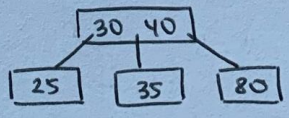


↳ setelah key dihapus, ternyata node masih memenuhi properti yaitu min key = 1. Jadi, sudah tidak ada langkah lain yang harus dikerjakan. Proses deletion berhenti sampai situ saja.

③ Delete 100



Key yang ingin dihapus berada pada leaf node.



Penjelasan sama dengan poin ②.