

Insert 30



Jika root masih kosong buat node value 30, warna merah

Jika node itu tidak punya parent (base root) warna = hitam.



Insert 50

Cek tree, jika ada cek value,

Jika lebih besar traverse ke subtree kanan,

Jika sudah null, root = newroot (50)

Warnai merah



Insert 80

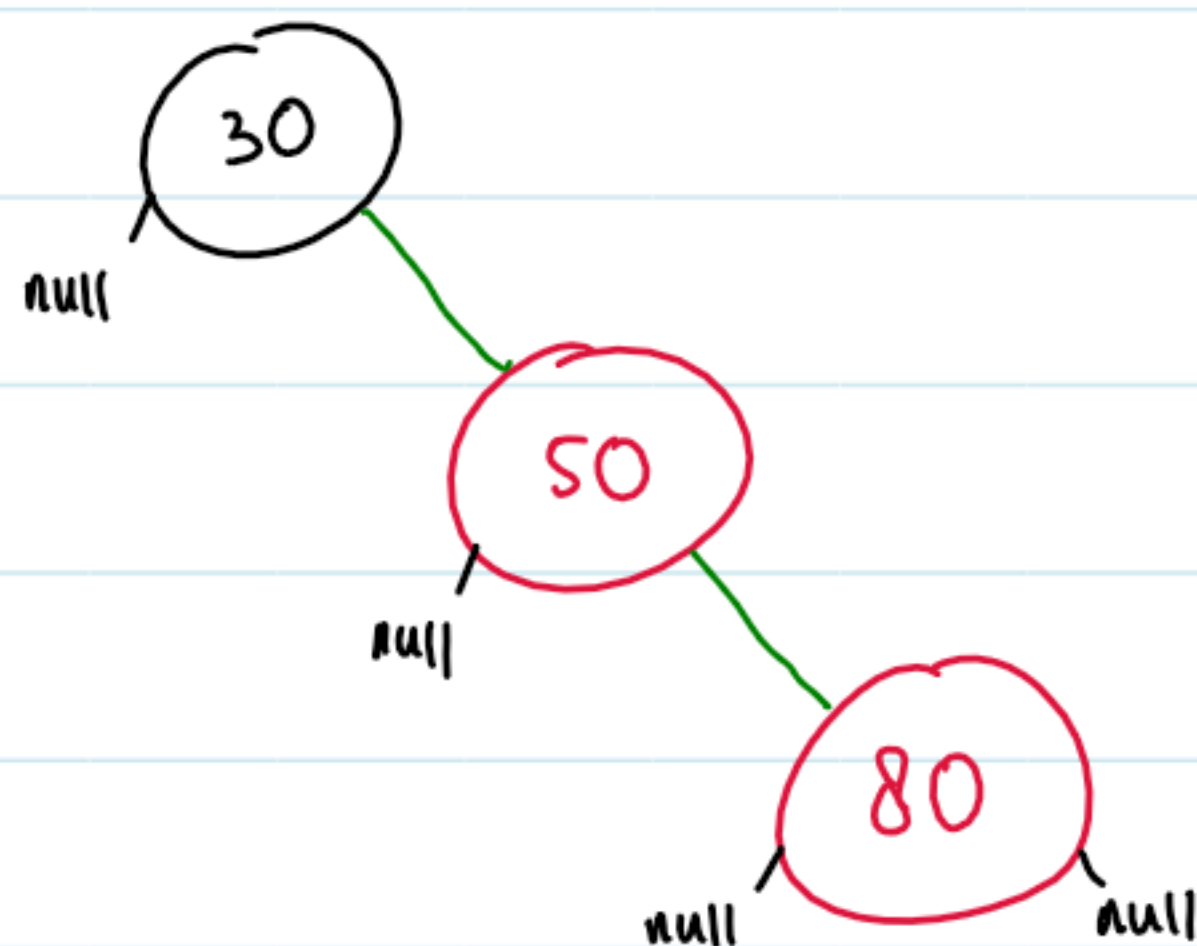
Cek tree, jika ada cek value,

Jika lebih besar traverse subtree kanan

else traverse subtree kiri.

Jika ketemu null, root = newroot (80)

Warna merah.



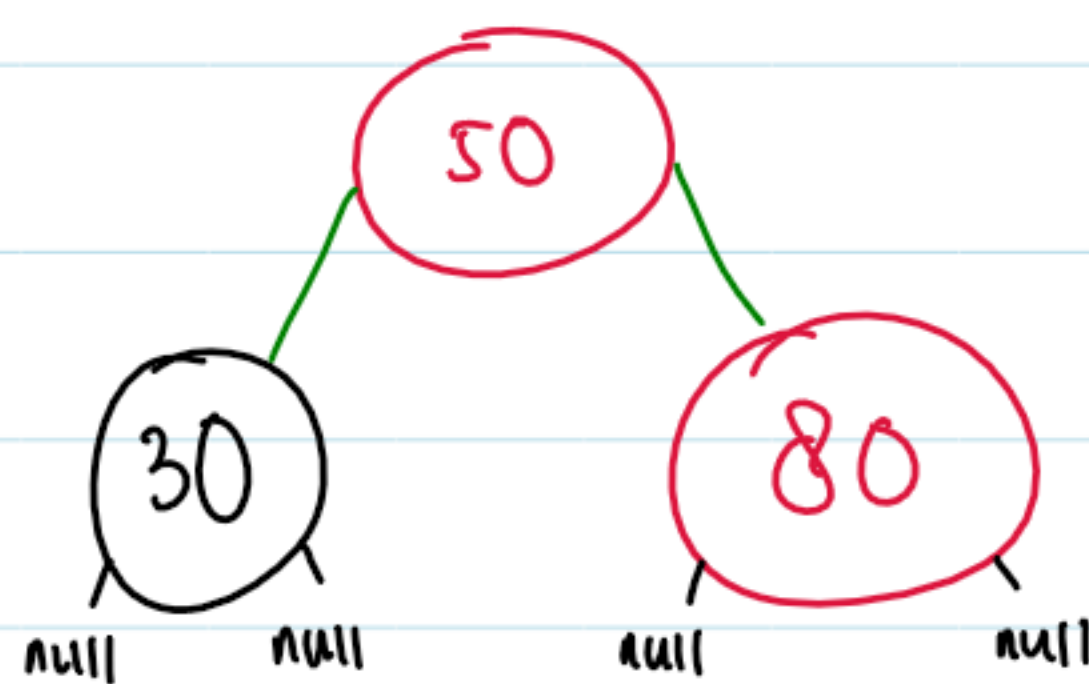
Jika beda tinggi sudah lebih dari 1,

rotate,

node adalah subtree kanan parent,

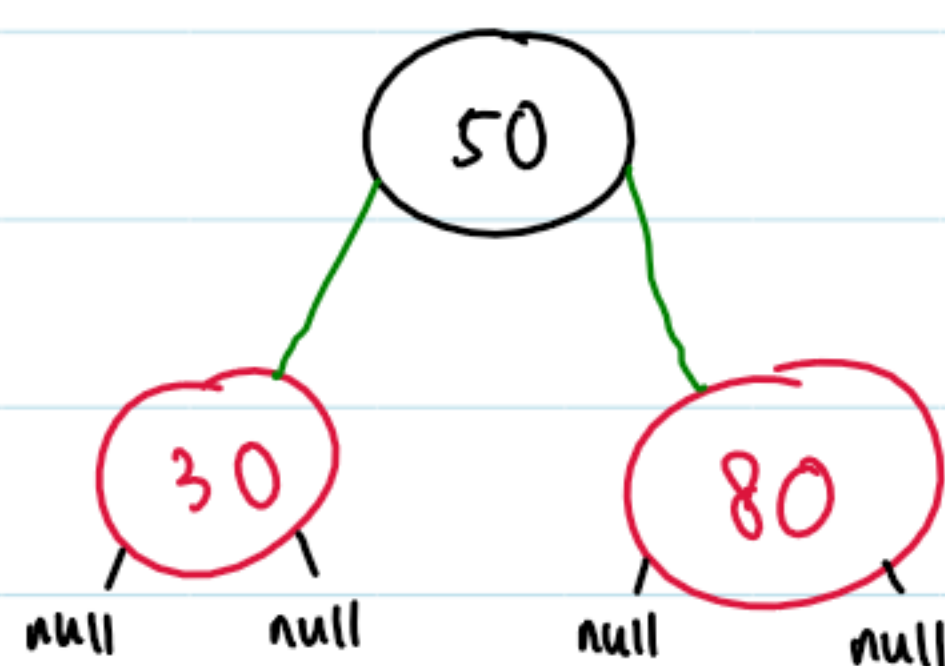
parent adalah subtree kanan grandparent,

maka single rotate left.



Cek warna baseroot. warnai hitam.

Warnai 30 merah



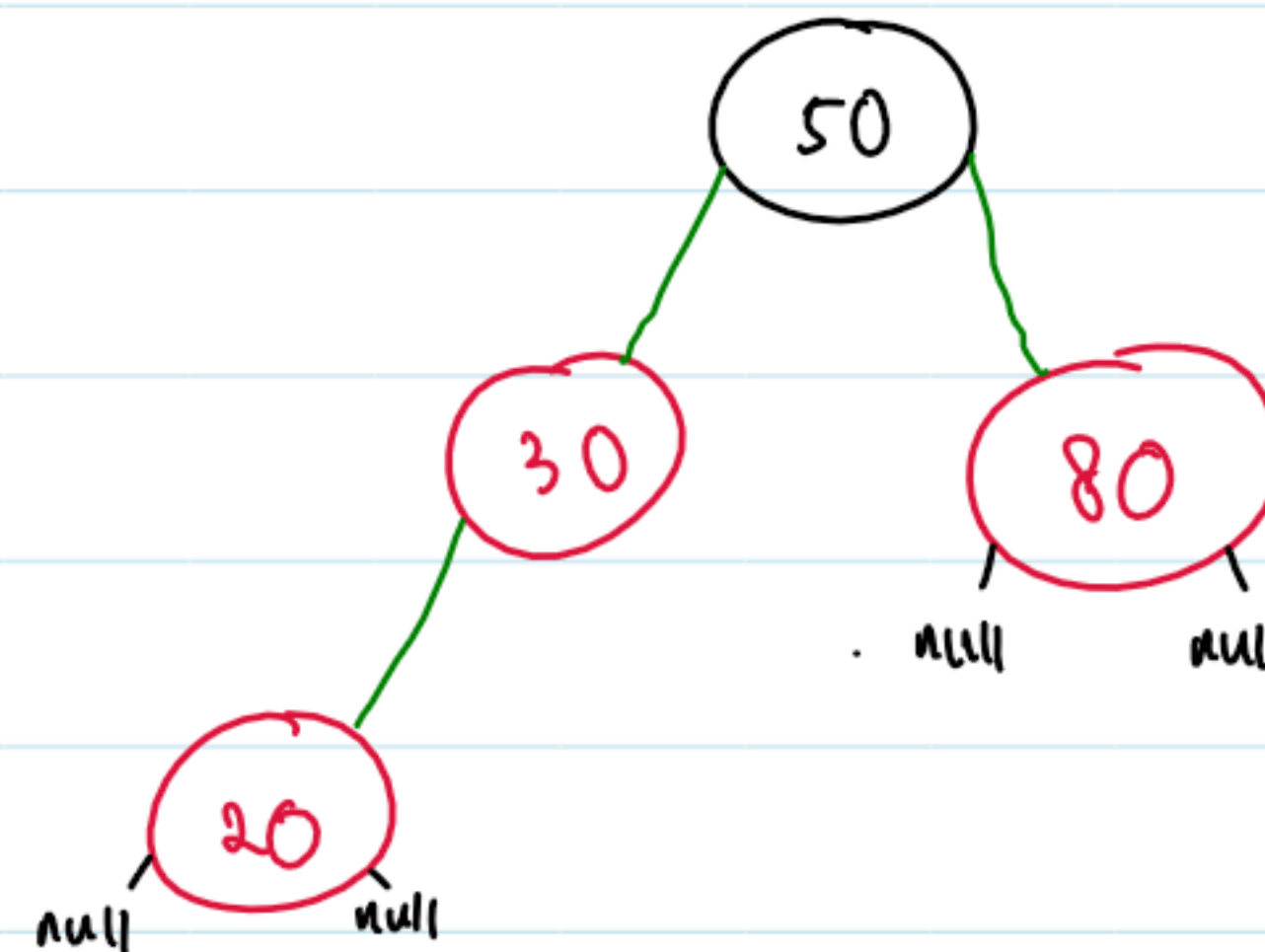
Insert 20

Cek root, jika ada cek value,

Jika lebih besar traverse ke subtree kanan,

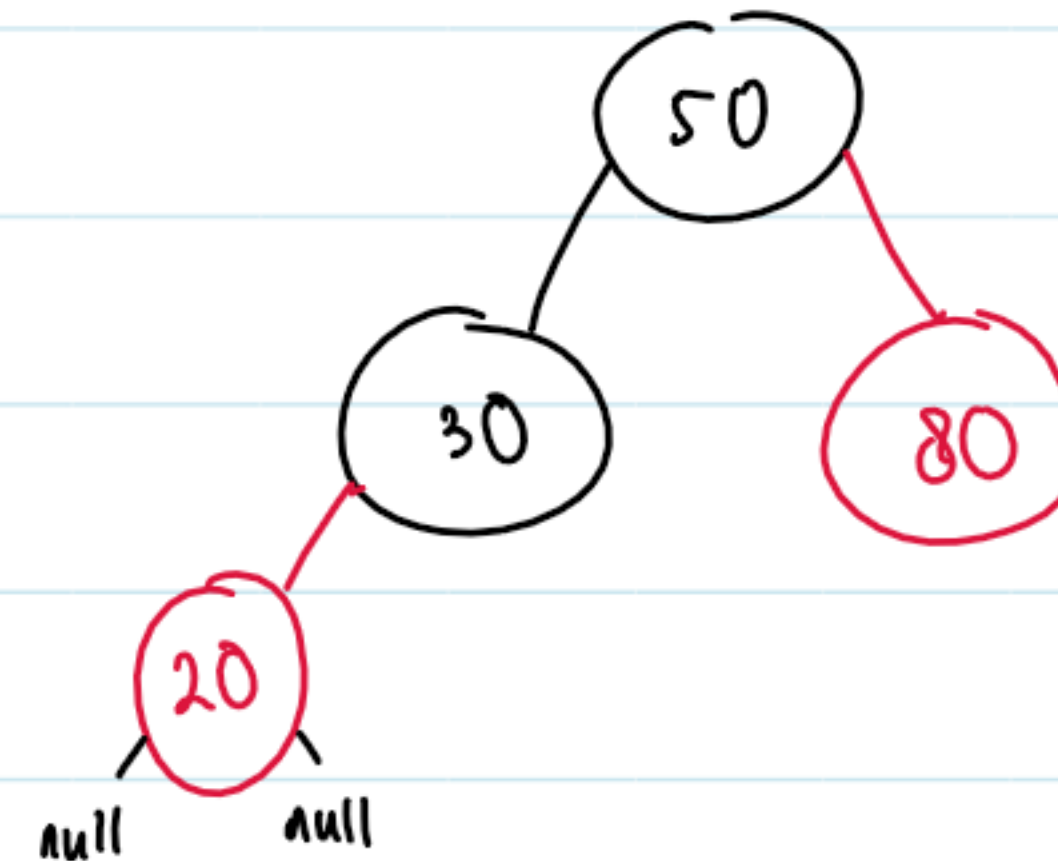
Jika sudah null, root = newroot (50)

Warnai merah



• 2 node warna merah berurutan melanggar aturan red-black.

• sebuah node merah parent dan child berwarna hitam.

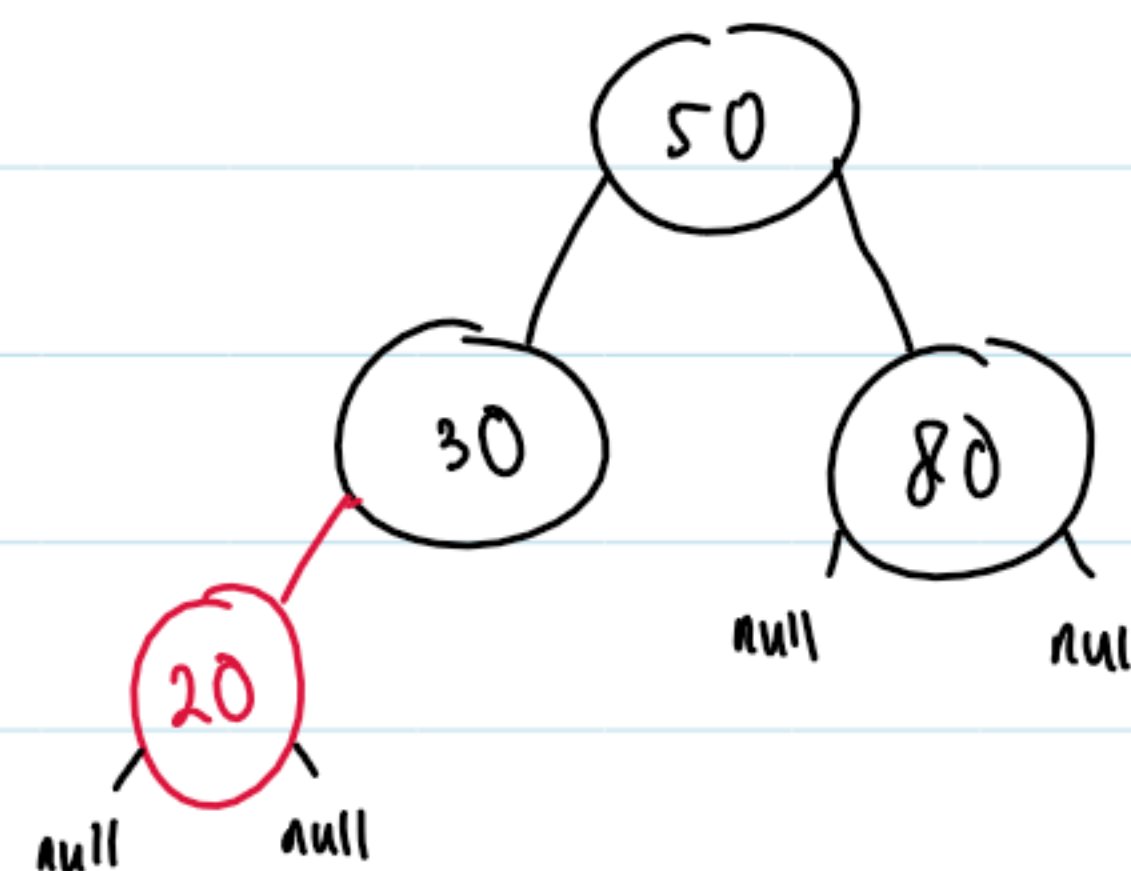


dari root ke semua null harus melewati jumlah node hitam yang sama.

root → left → left → left (2 node hitam)

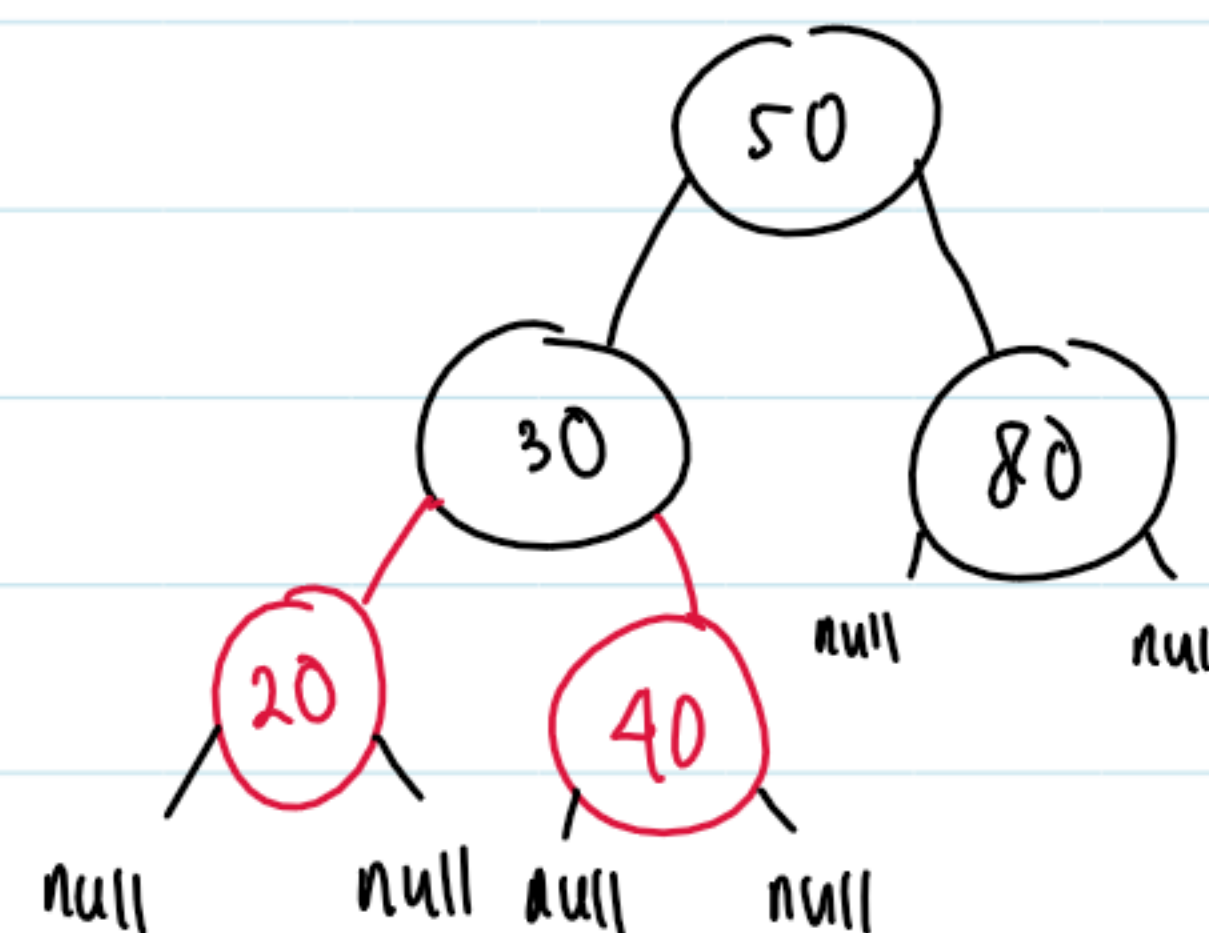
root → right → right (1 node hitam)

maka child kanan harus hitam juga.



Insert 40

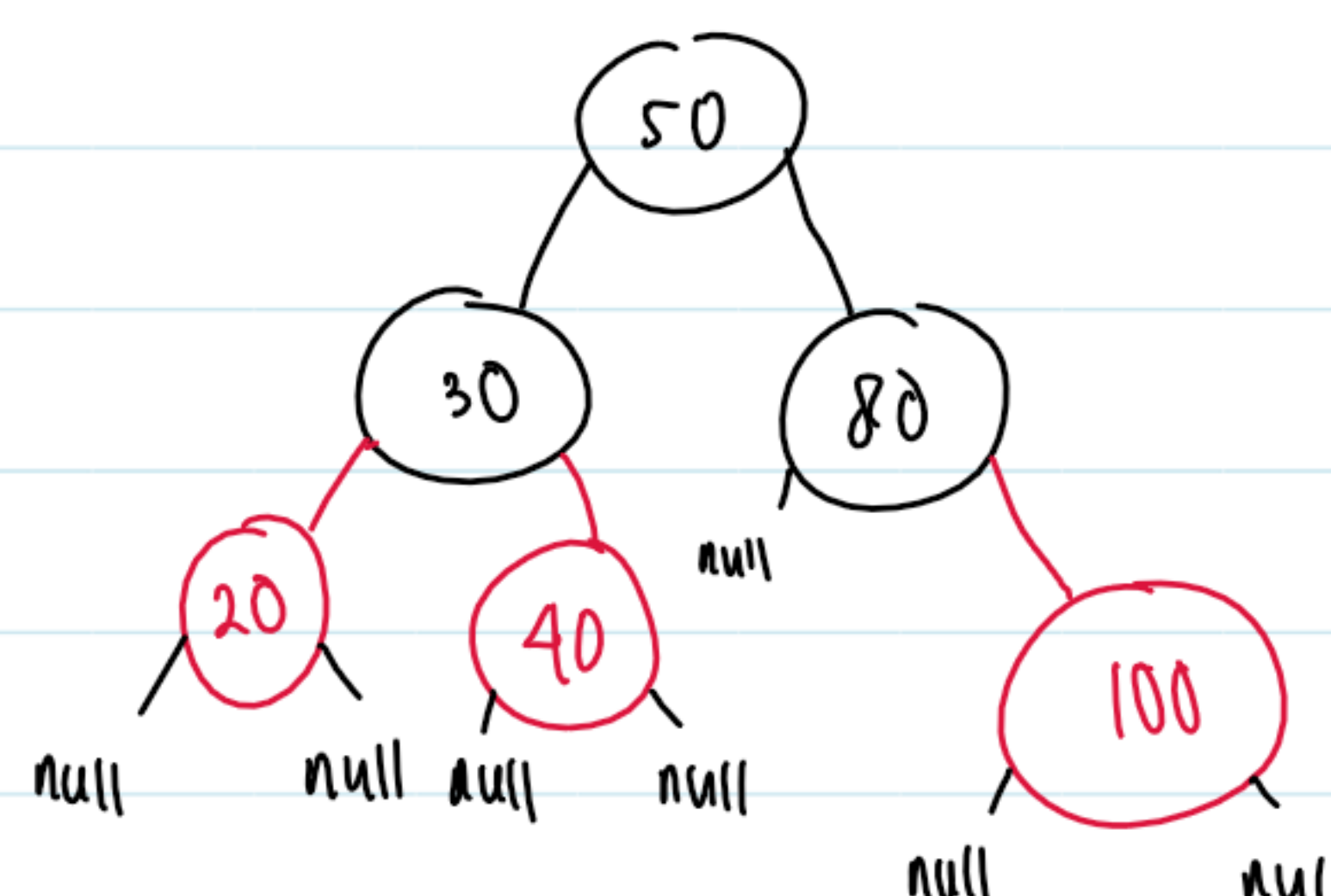
traverse ke kiri warnai merah.



pewarnaan masih belum ada masalah

Insert 100

traverse ke kanan warnai merah.

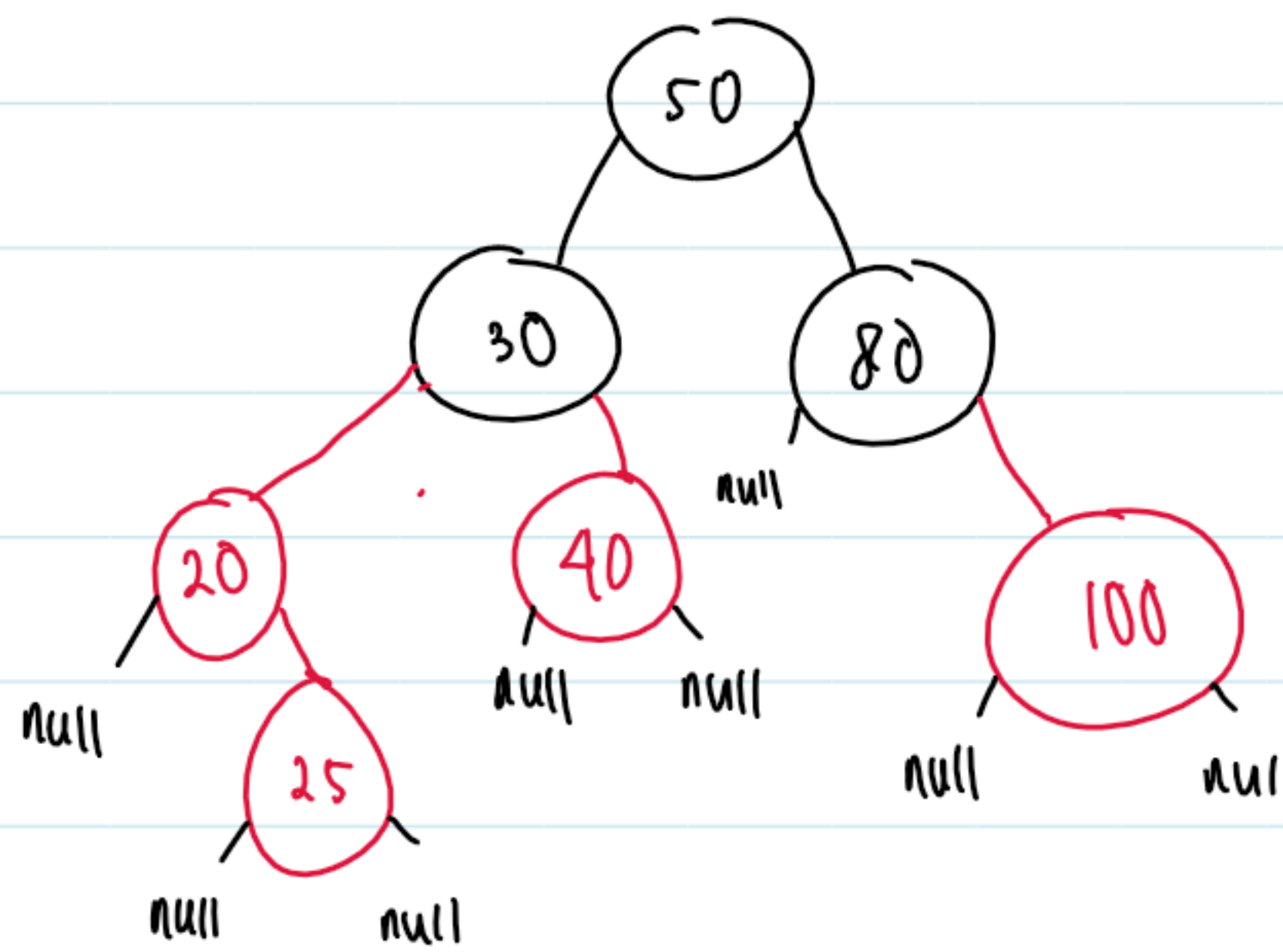


pewarnaan tidak ada masalah karena

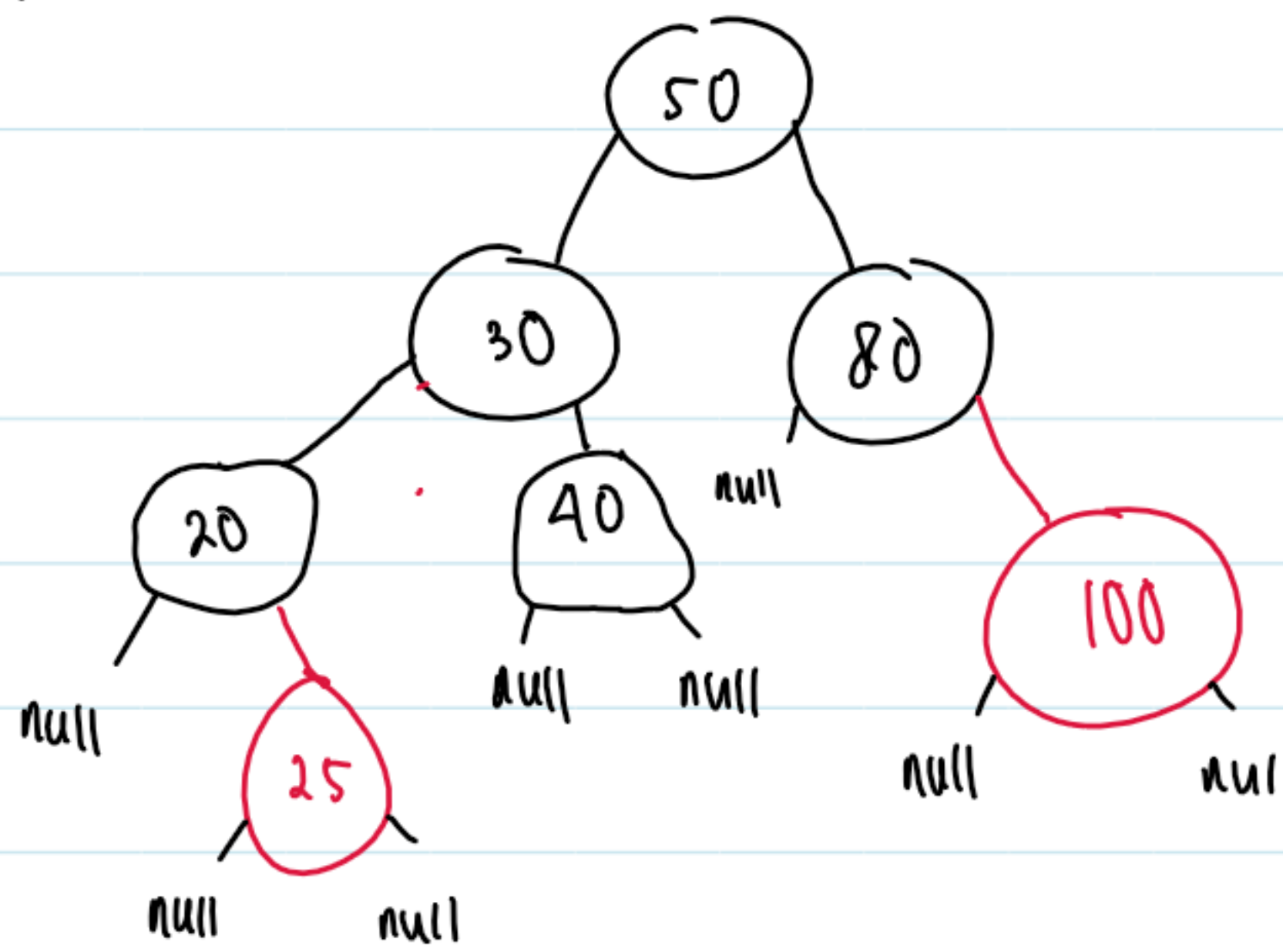
child dan parent dari node merah masih hitam.

Insert 25

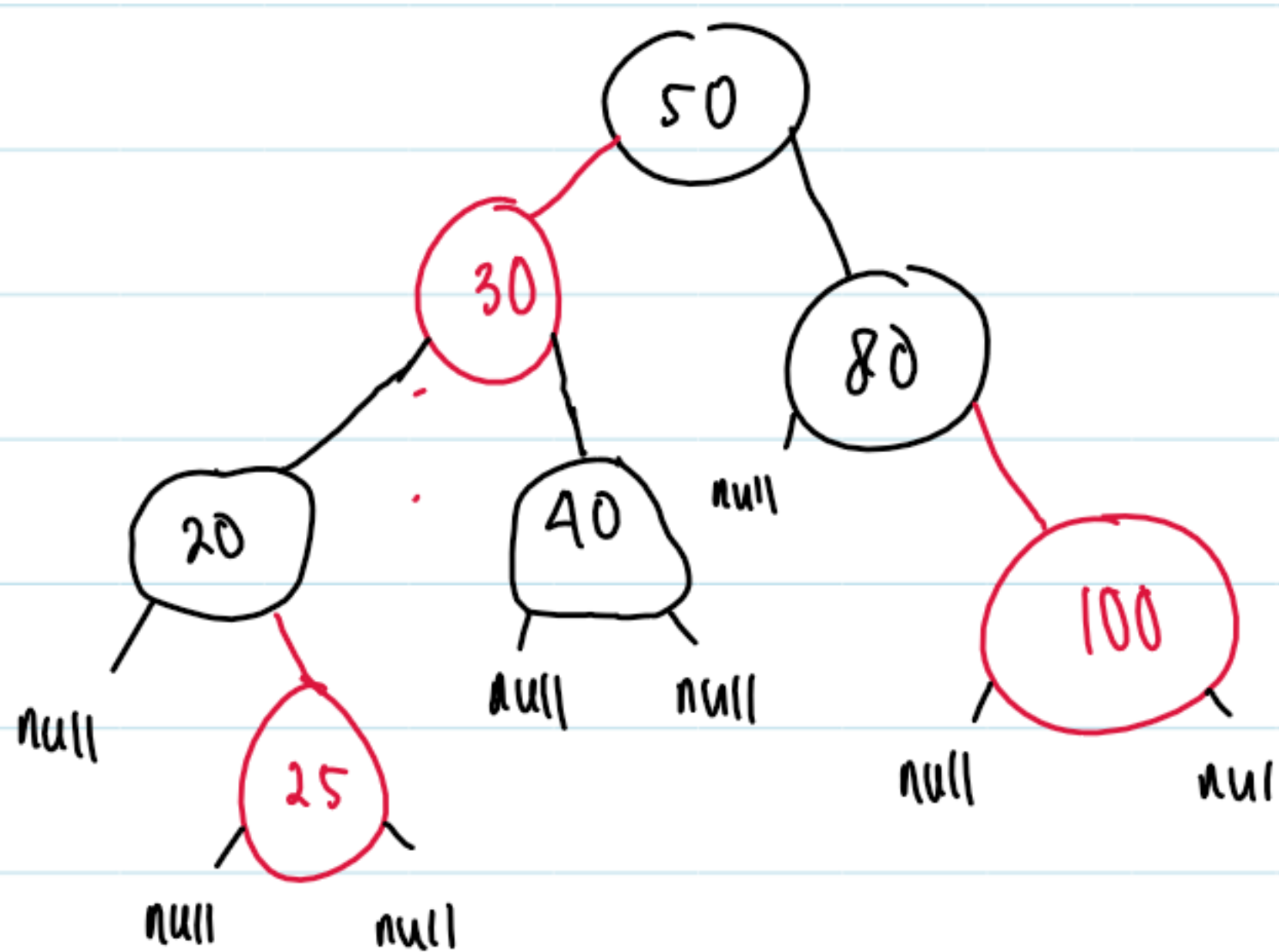
traverse left if value is smaller else traverse right.



parent and uncle dari node merah harus hitam.

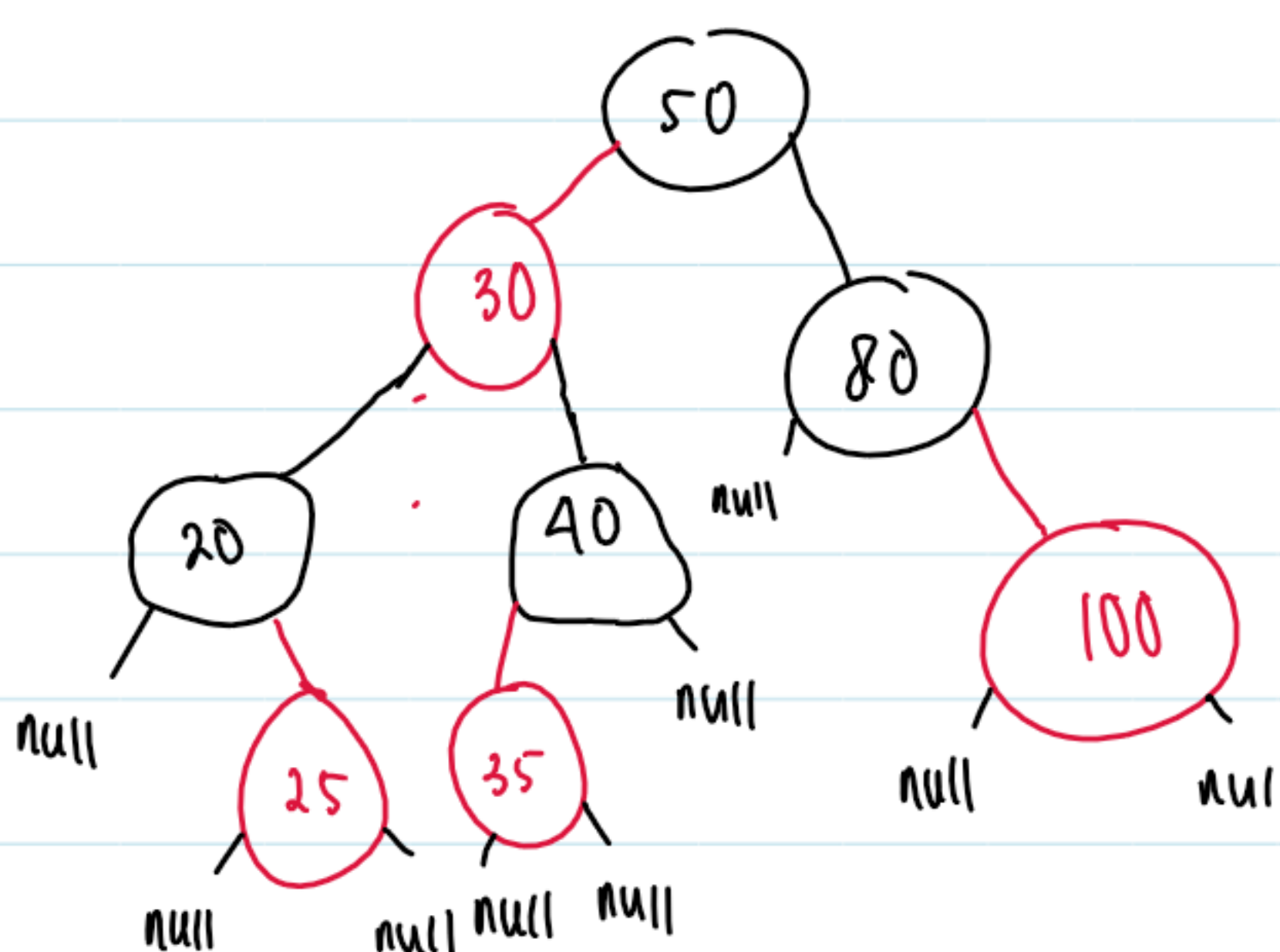


agar root ke semua null menemui jumlah node hitam yang sama,
node 30 jika warna merah



Insert 35

traverse ke kiri jika value lebih kecil else traverse kanan.



Search

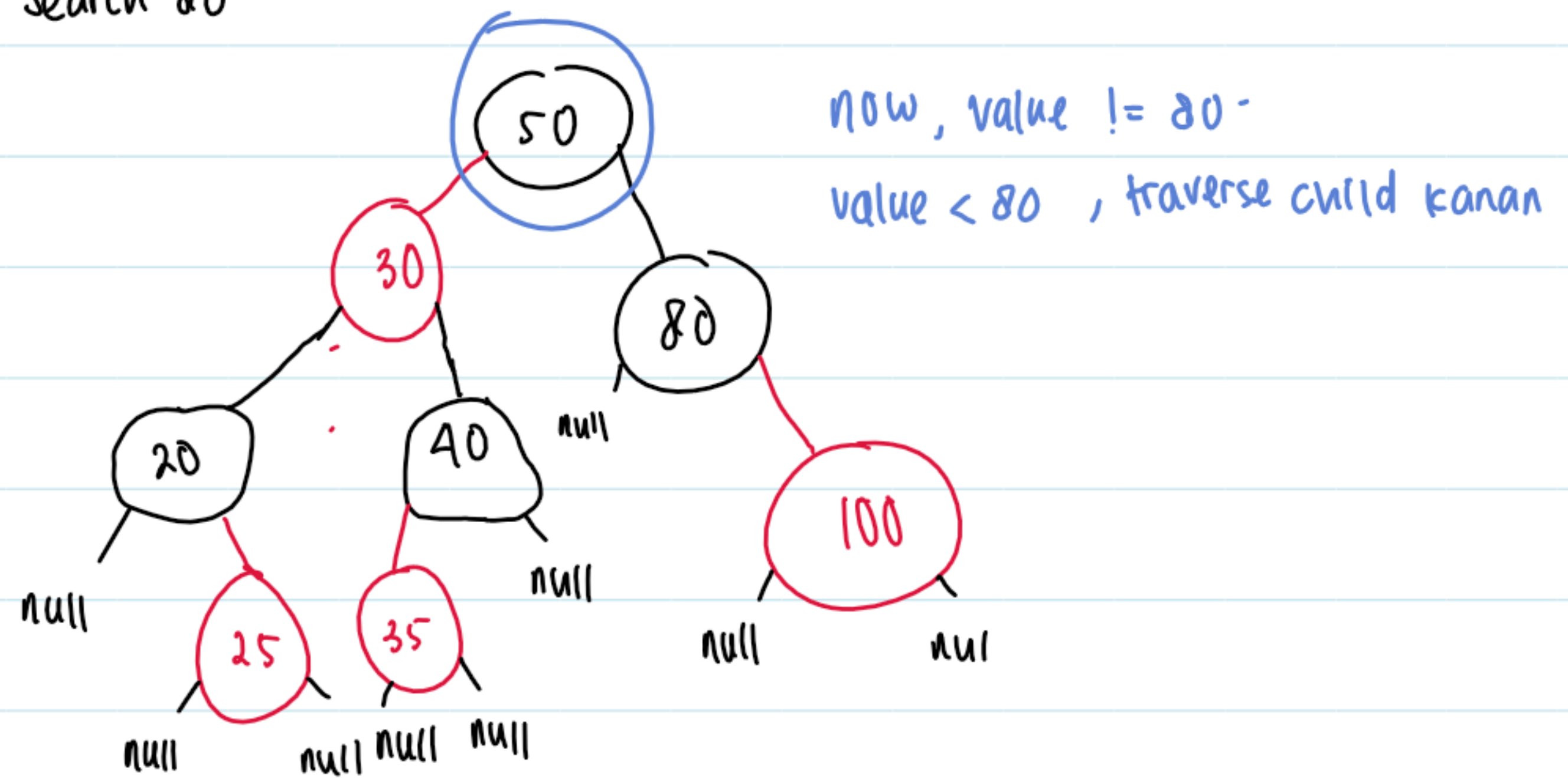
jika value root itu 80 return true.

traverse child kiri jika 80 lebih kecil dari value

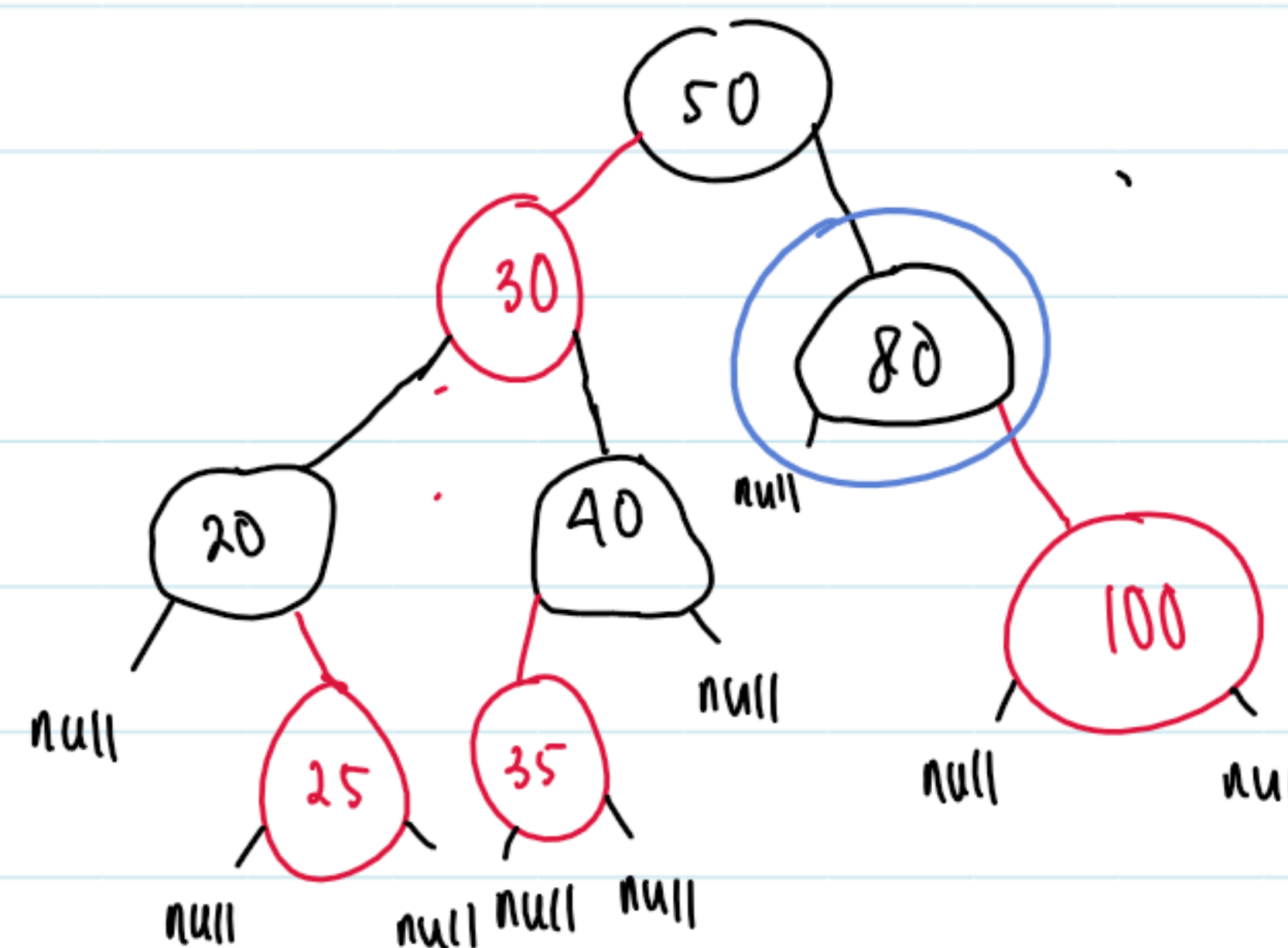
else traverse child kanan.

jika sampai null return false // ga ketemu.

Search 80

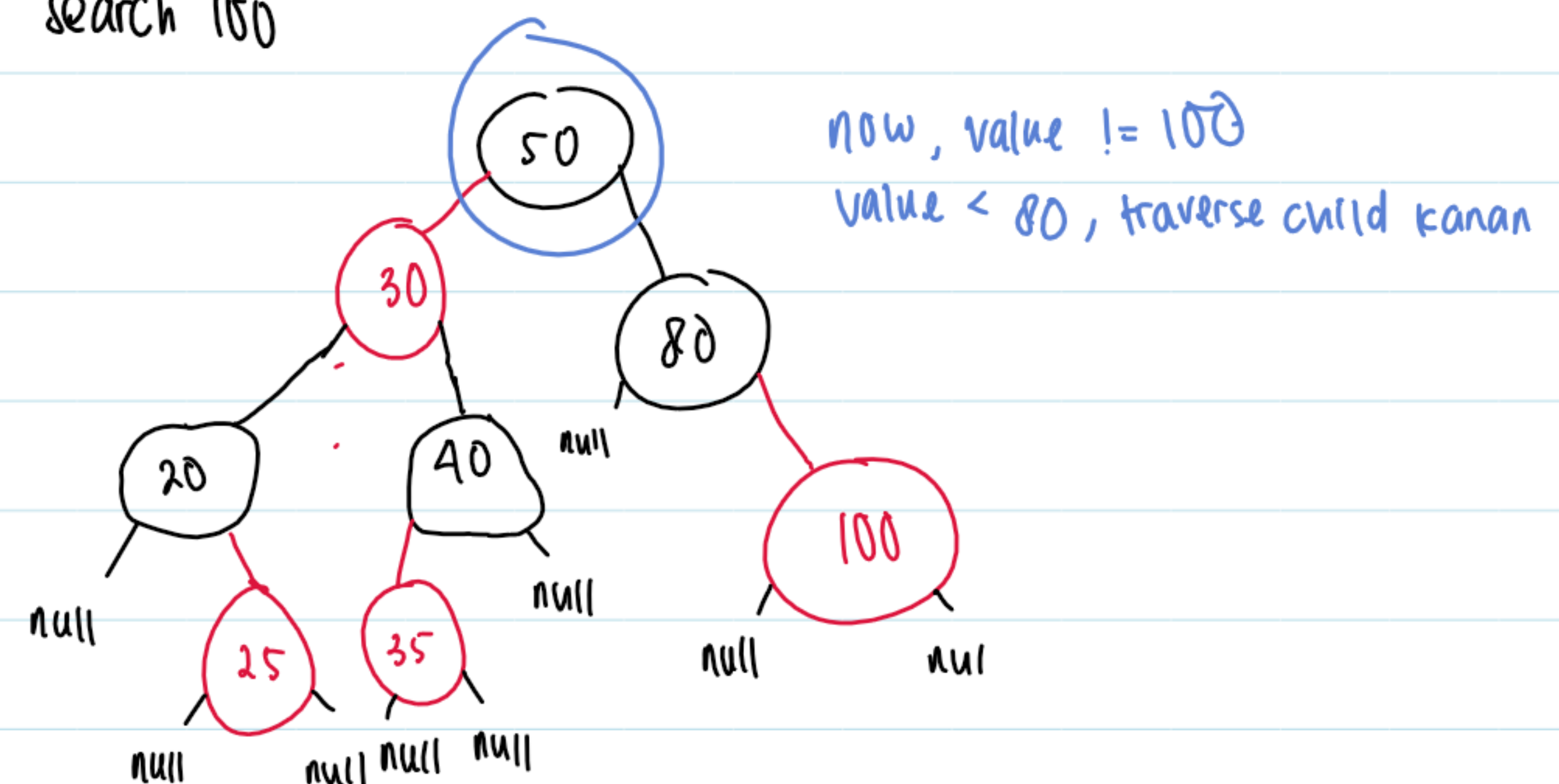


now, value != 80
value < 80, traverse child kanan

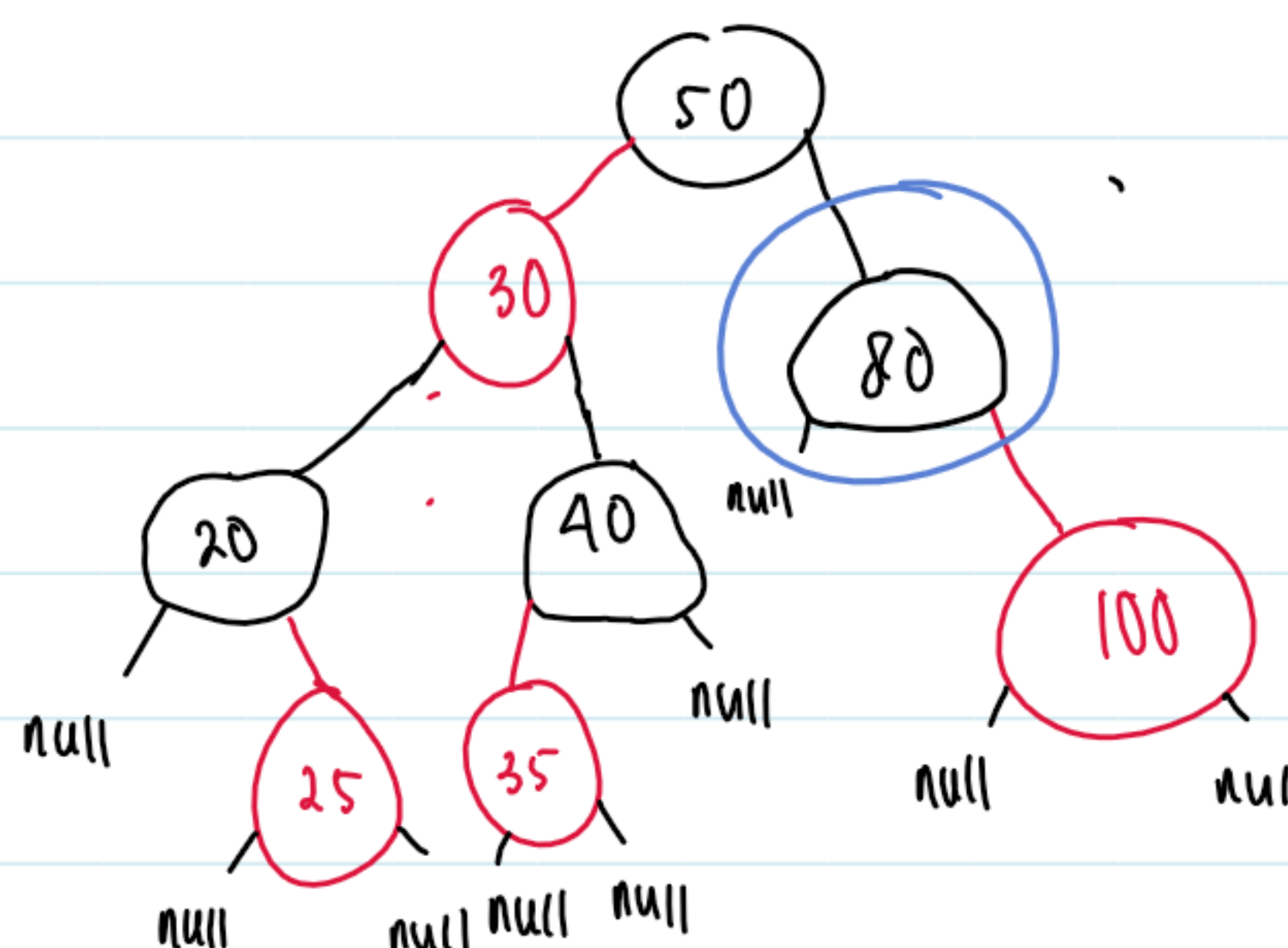


now, value = 80
return true.

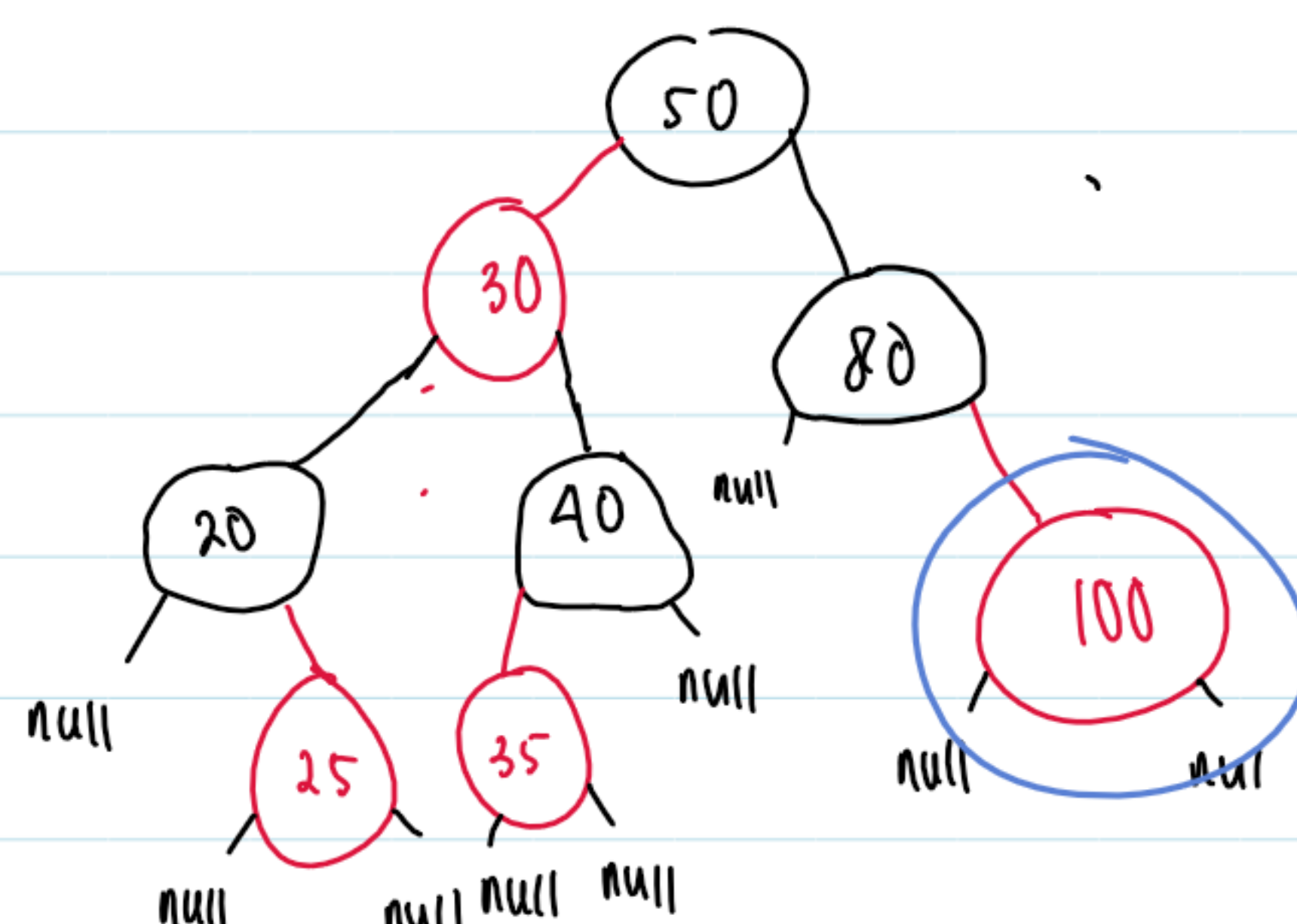
search 100



now, value != 100
value < 80, traverse child kanan

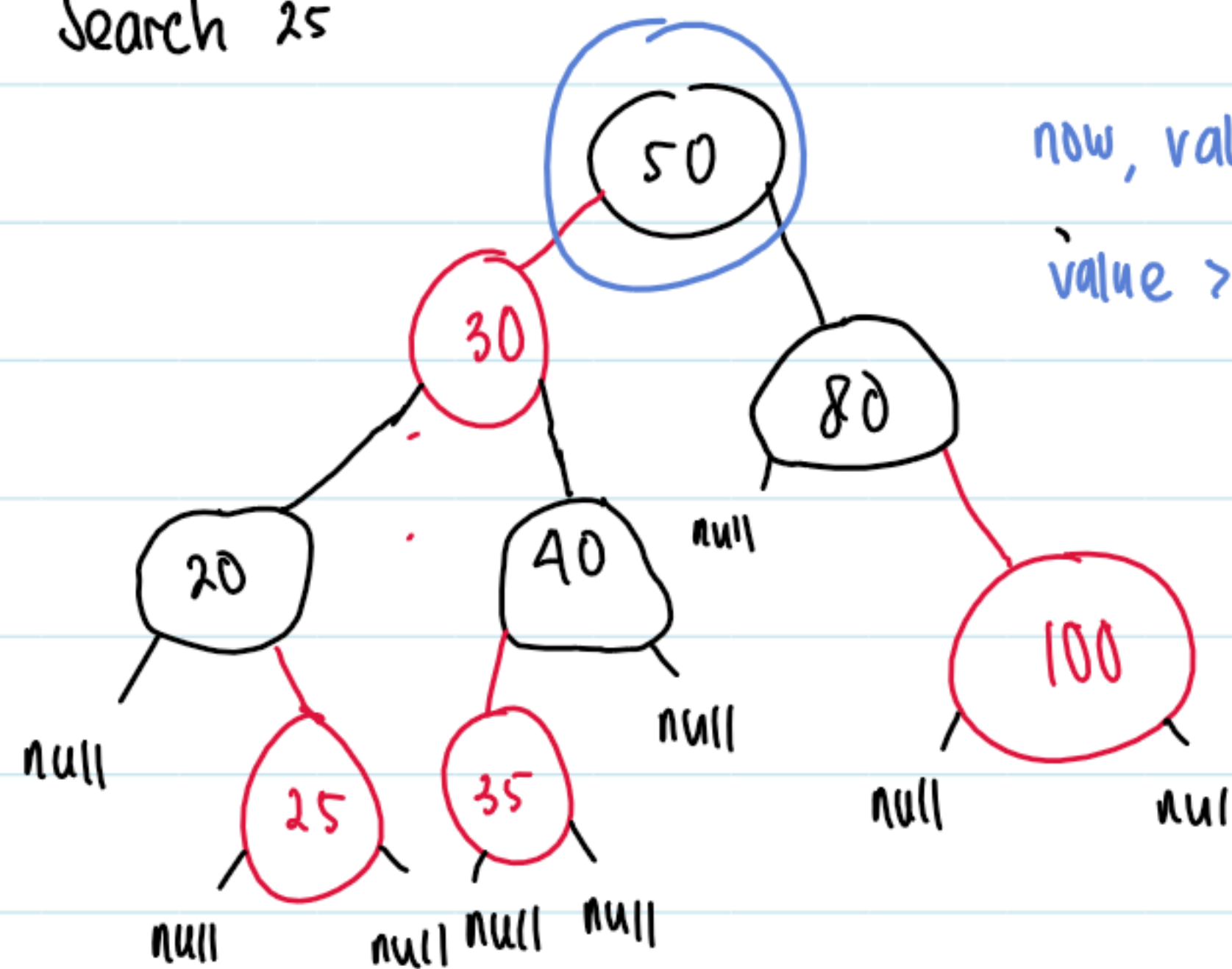


now, value != 100
value < 100, traverse child kanan.



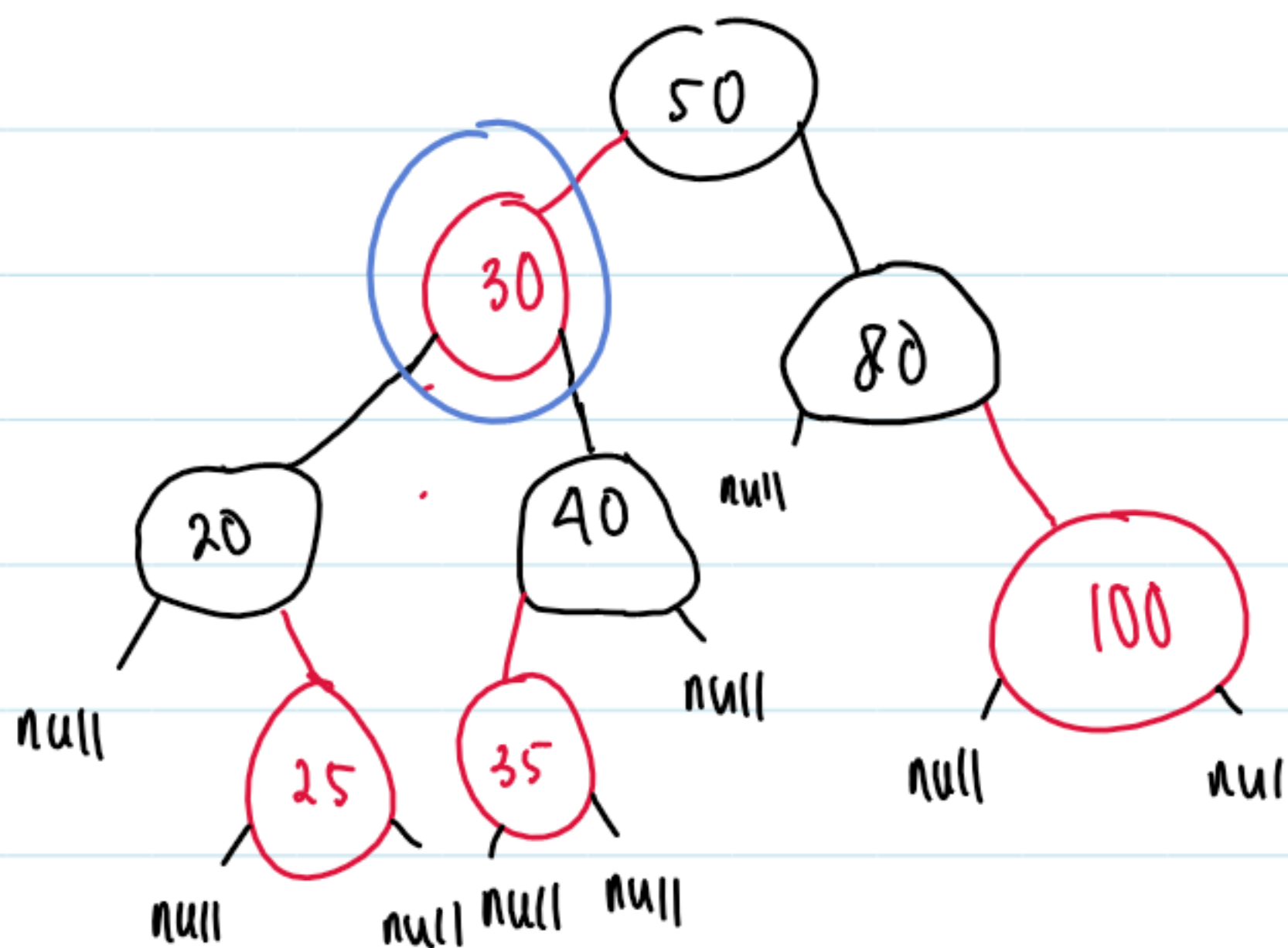
now, value = 100
return true.

Search 25



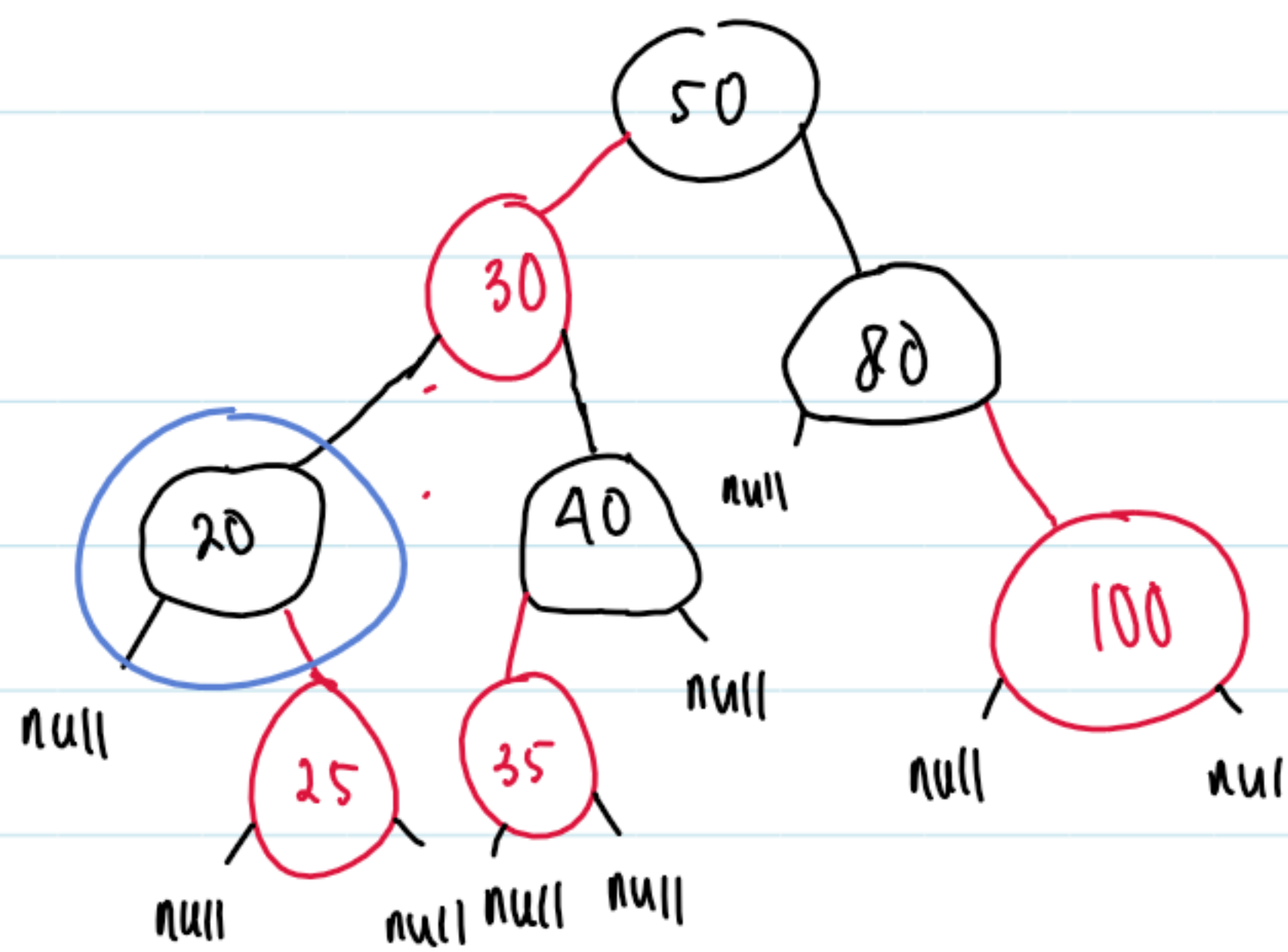
now, value != 25

value > 25, traverse left child.



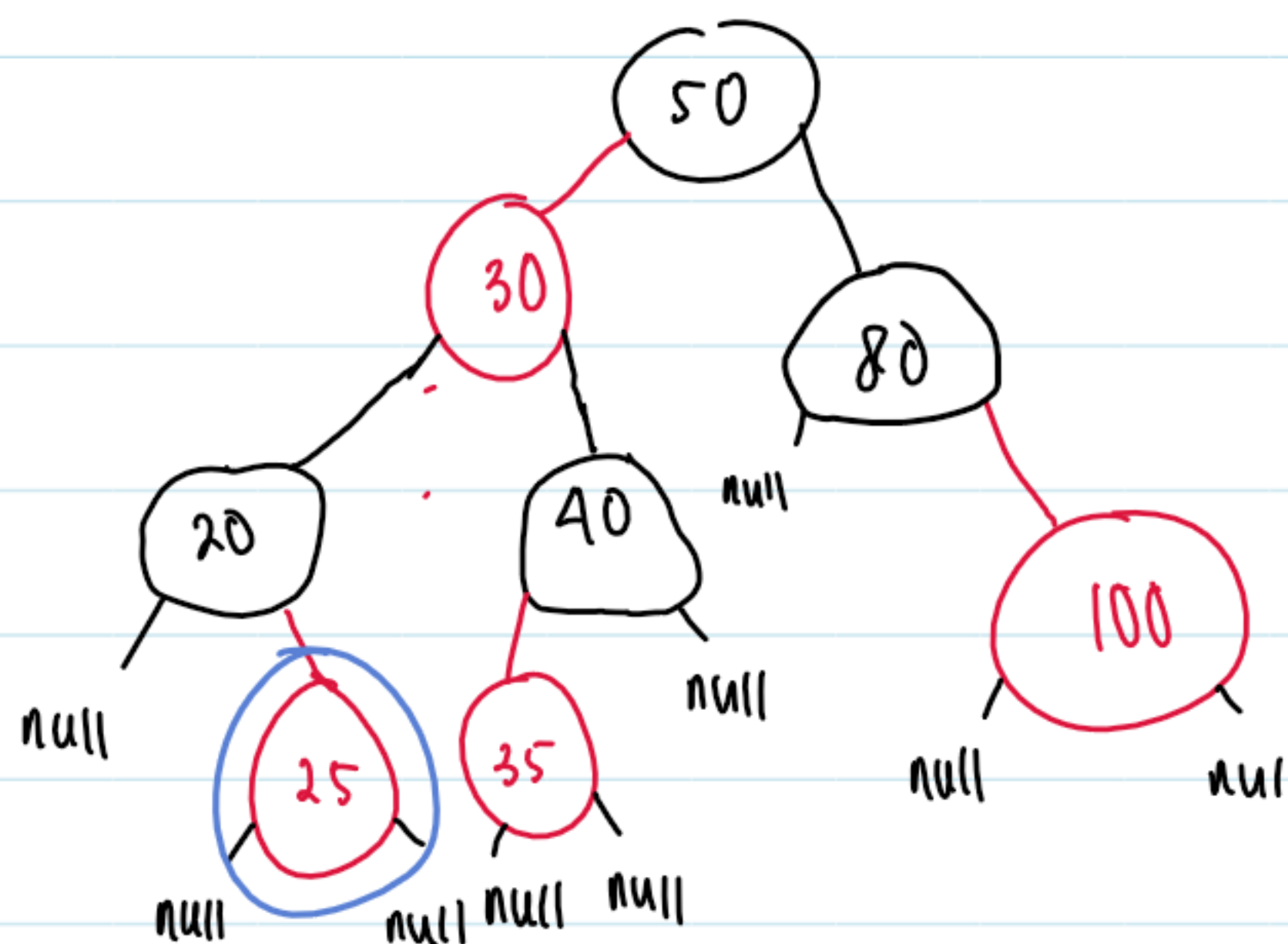
now, value != 25

value > 25, traverse left child.



now, value != 25

value < 25, traverse right child.



now, value = 25

return true

Delete

first step : find x

if root != x :

if x < root.value : traverse child kiri

else if x > root.value : traverse child kanan

} algo search

else :

Jika ga punya child : delete node x

Jika punya child :

cari node value terbesar di child kiri :

dari child kiri traverse kekanan sampai ketemu null

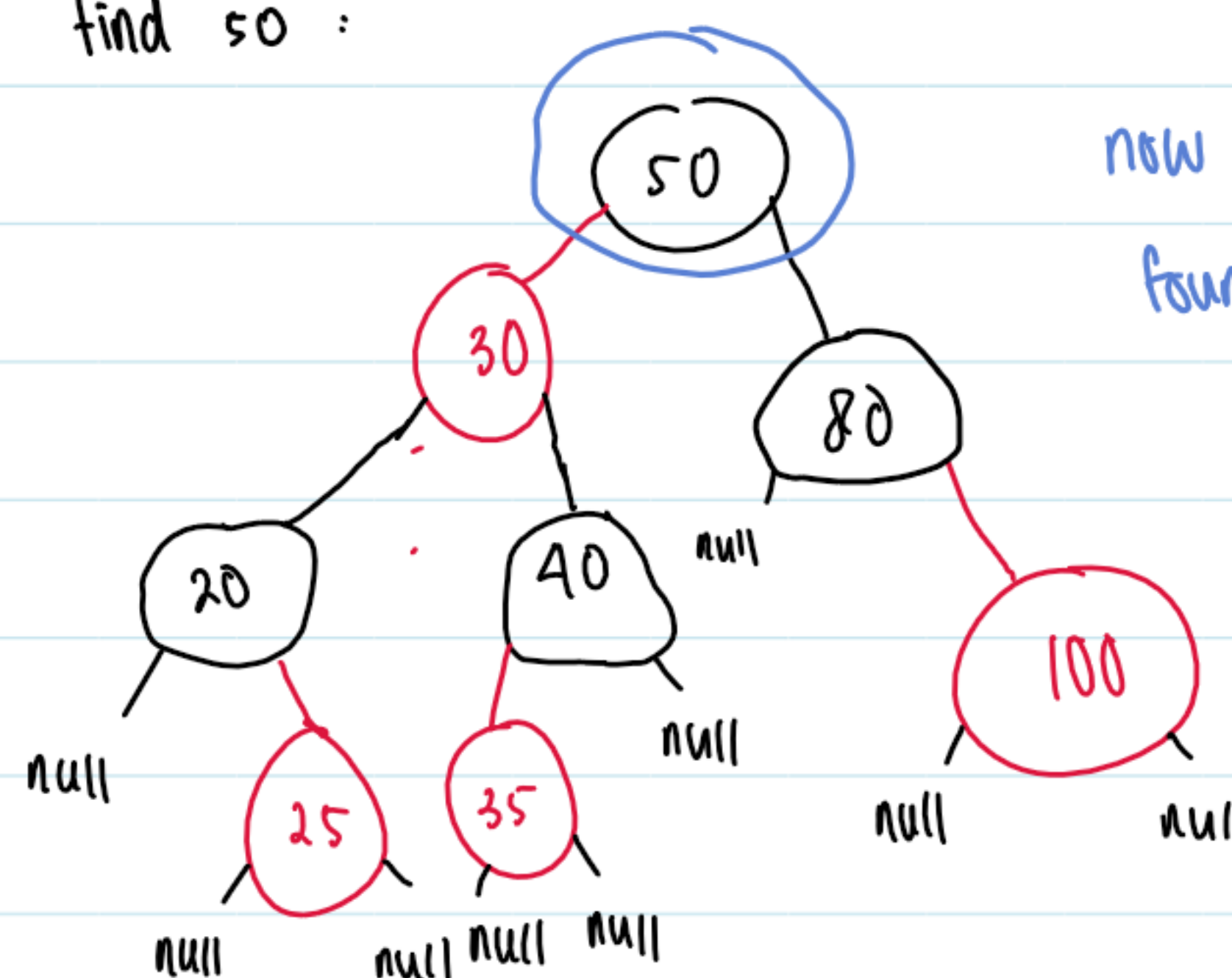
ganti value root dengan value terbesar di child kiri

Delete node value terbesar di child kiri

fix tree agar tidak melanggar aturan red black tree

Delete 50

find 50 :

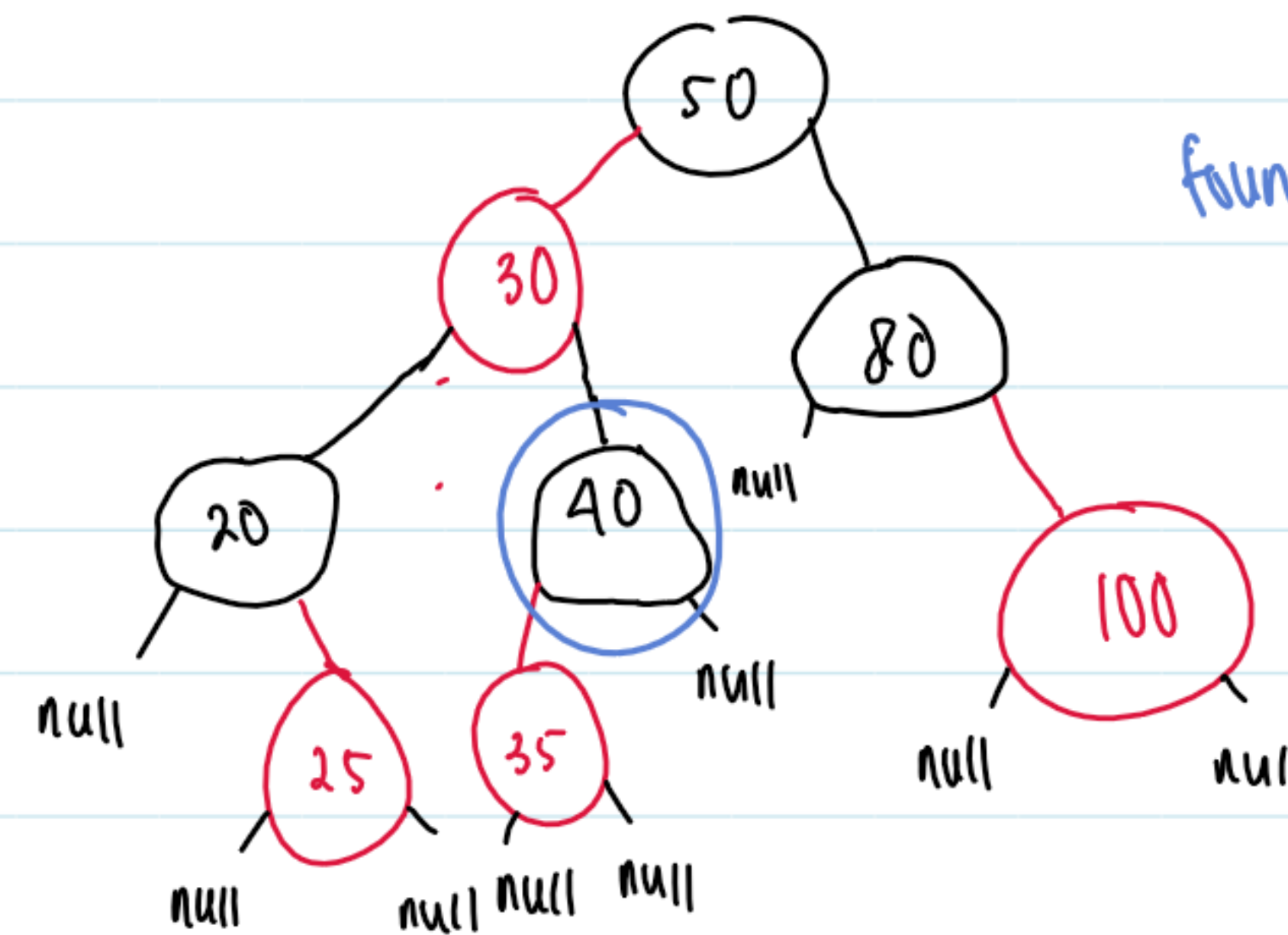


now, value = 50

found 50

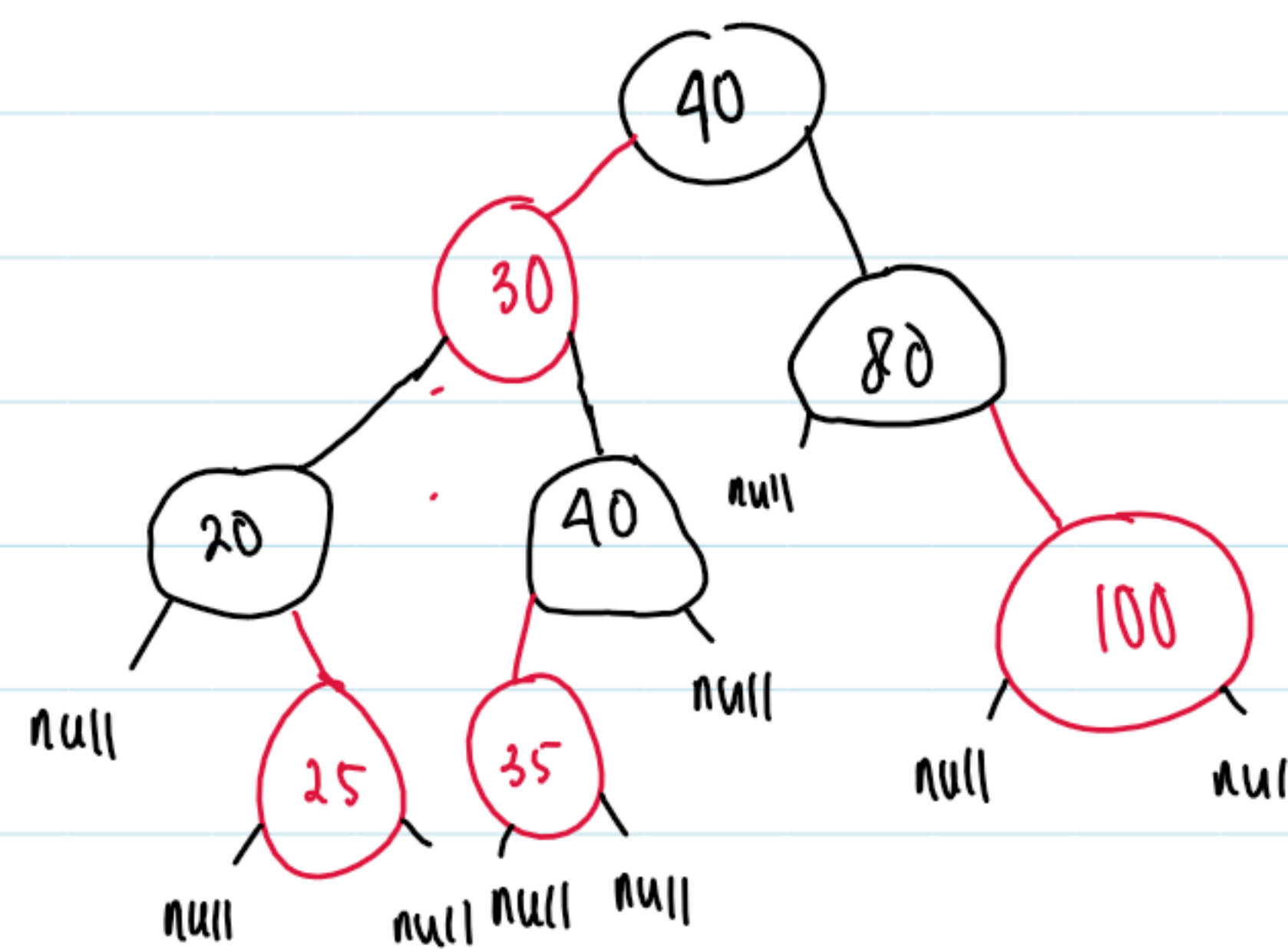
Cari node value terbesar di child kiri 50

dari child kiri 50 traverse kekanan sampai null

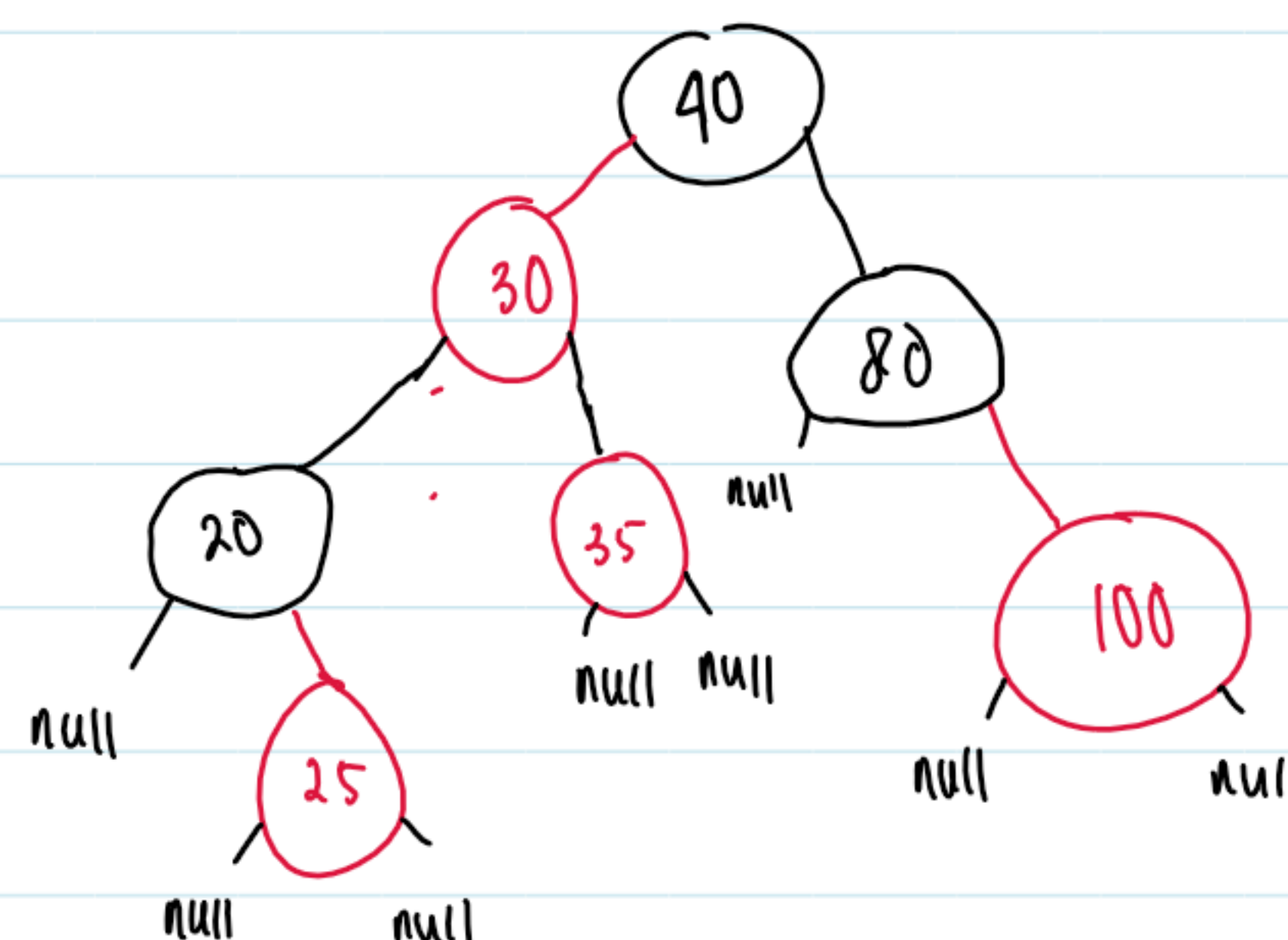


found largest element

ganti nilai root dengan 40



Delete node 40



fix tree, recolor node

child node 30 harus hitam

