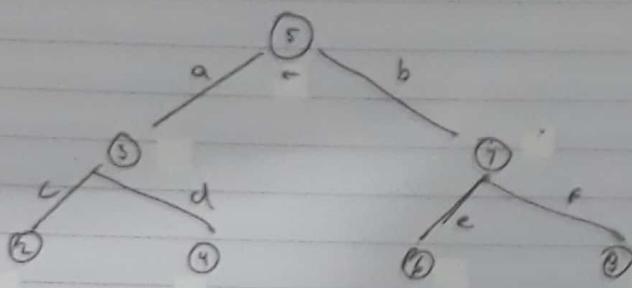


Nicholas (Night Stalker)

- Linear data structure = urutan mengisi dan mengakses elemen bisa langsung melihatnya
Pengaksesan hanya ke aktif elemen (Linear DS)
Non-linear \rightarrow Tidak perlu mengakses seluruh elemen untuk ke elemen terakhir (tree, B/B+)

2.



5 = Root node

key = atribut sedang membagi tree, dalam hal ini: value dari node

edge = garis penghubung (a,b,c,d,e,f)

Siblings = node dengan height sama dan parents sama. (2 Siblings 4) (6 Siblings 9)
(3 Siblings 7)

Parent = node yang memiliki keturunan. (3 Parent 2 & 4) (7 Parent 6 & 9) (5 Parent 3 & 7)

Child = node hasil perkembangan (2 & 4 child 3) (6 & 9 child 7) (3 & 7 child 5)

Leaf = node Paling bawah (2,4,6,9)

3. Full binary tree : masuk? Elemen Penya 0 atau 2 anak kecuali level Paling bawah

Complete binary tree = Semua level wajib terisi node kecuali level Paling bawah

Perfect binary tree = Semua node wajib terisi 2 node kecuali level Paling bawah

4. Balanced binary tree akan terbentuk apabila perbedaan tinggi level Paling bawah tidak ada yang lebih dari 1.

5. Maksimum element di level n = 2^n

Maksimum total element dengan tree k level = $2^{k+1} - 1$

Maksimum tinggi tree dengan n node = $\log_2(n)$

Maksimum tinggi dengan n node = $n-1$

6. Bentuk array. Root ada di indeks 0

arah kiri = $2i+1$

arah kanan = $2i+2$

Cari parents: $(P-i)/2$



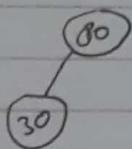
7. In Order Successor : Setelah diurutkan seluruh elemen dalam BST, Inorder Successor adalah elemen Paling besar terdekat

Inorder Predecessor : Setelah diurutkan seluruh elemen BST, Inorder Predecessor adalah elemen Paling kecil terdekat

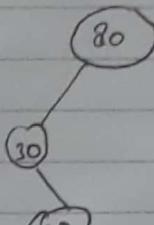
8



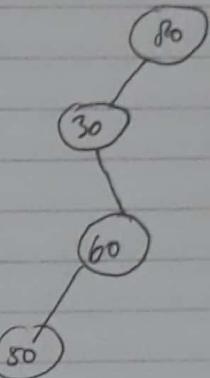
1



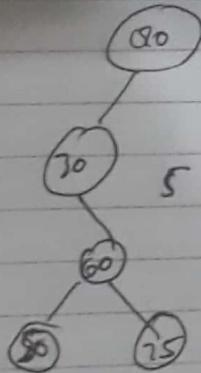
2



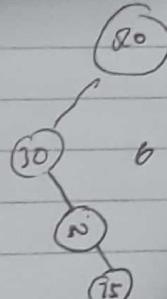
3



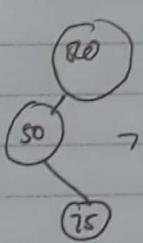
4



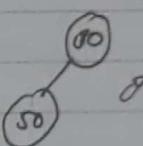
5



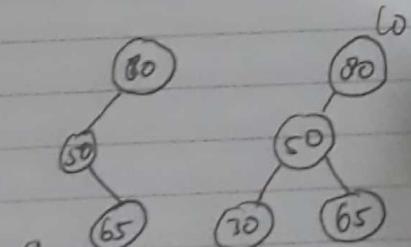
6



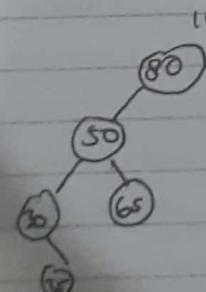
7



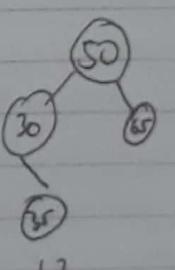
8



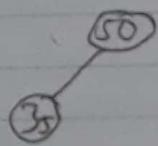
9



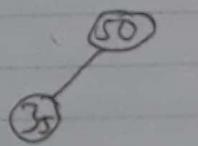
10



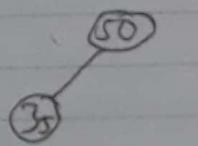
11



12



13



14

