Individual Assignment : Richard William

1. Explain the differences between linear and non-linear data structures!
2. Describe the following terminology in a tree: base root, key, edge, siblings, parent, child, and leaf!
3. Explain the following types of binary trees: full, complete, and perfect!
4. What makes a tree balanced?
5. Explain the four properties of a binary tree!
6. Explain the intuition of implementing a binary tree using an array!
7. Explain the differences between inorder successor and inorder predecessor!
8. Draw the following binary search tree step by step (14 pictures):

- Insert 80, 30, 60, 50, 75
- Delete 60, 30, 75
- Insert 65, 30, 35
- Delete 80, 65, 35

Jawaban:

1. Linear data struct dapat di traverse dalam 1 kali jalan saja, sedangkan non-linear harus berulang kali. Linear data struct terhubung dengan prev dan next, sedangkan non linear dengan tingkatannya.

2. Base root: Node paling atas dalam sebuah tree.

   Edge: garis yang menghubungkan antar node.

   Siblings: Node yang memiliki parent yang sama dan berada dalam tingkatan yang sama.

   Parent: Node yang memiliki anak (child)

   Child: Node yang terhubung dengan parent.

   Leaf: Node yang tidak mempunyai child.

3. Full Binary Tree: Binary tree yang setiap nodenya mempunyai 2 atau 0 child kecuali pada tingkat paling akhirnya.

Complete Binary Tree: Binary tree yang setiap levelnya mempunyai 2 child kecuali yang paling bawah.

Perfect Binary Tree: Binary tree yang mempunyai 2 child di setiap nodenya dan berakhir di level yang sama kecuali level paling bawah

4. Yang membuat tree balance adalah ketika selisih antara tinggi subtree kiri dan kanan tidak lebih dari 1.

5. Maximum number of nodes : 2^k , k = level node

   Maximum number of nodes on binary tree: $2^{k+1} - 1$, k= level node

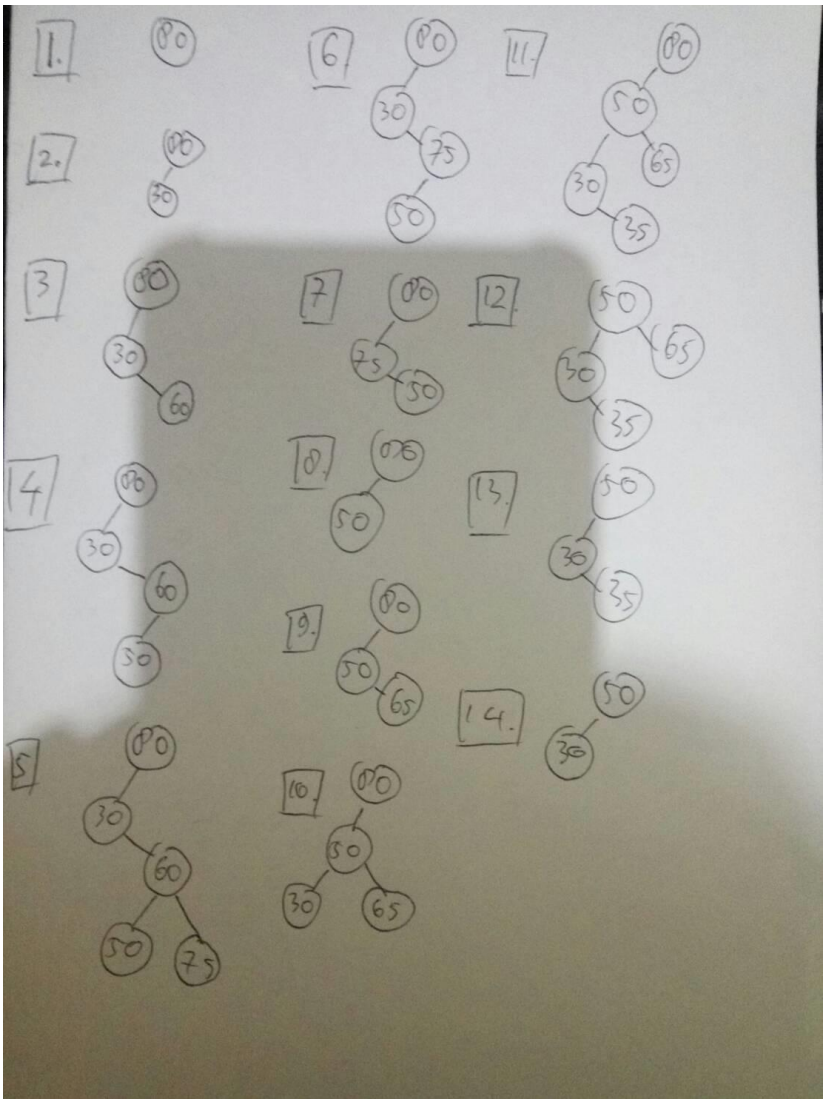
   Tinggi minimum dari binary tree : $^2\log(n)$, n = jumlah node

   Tinggi maximum dari binary tree: n-1, n = jumlah node

6. Implementasi binary tree dalam array dibaca dari sebelah kiri ke kanan dengan root dari tree memiliki index 0 yang kemudian dilanjutkan dengan komponen childrennya jika children berada di posisi kiri, gunakan rumus 2p+1 dengan p bernilai index dari parentnya, dan jika children berada di posisi kanan, gunakan rumus 2p+2 dengan p bernilai index dari parentnya.

7. Inorder predecessor adalah node dengan nilai dibawah/sebelum dari nilai yang akan dihapus nilai ini merupakan subtree sebelah kiri dari tree yang akan dihapus. Kemudian inorder Successor merupakan nilai lebih besar dari value yang akan diganti dan merupakan subtree sebelah kanan dari tree yang akan dihapus.

8. →→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→


1.
80
2.
80
30
3
80
30
60
4
80
30
60
50
5
80
30
60
50
75
6
80
30
75
50
7
80
75
50
8.
80
50
9.
80
50
65
10.
80
50
30
65
11.
80
50
65
30
35
12.
50
65
30
35
13.
50
30
35
14.
50
30