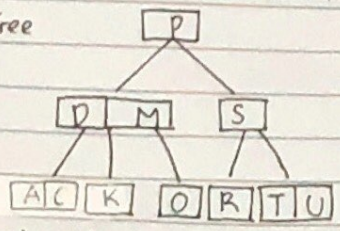
GSLC DS

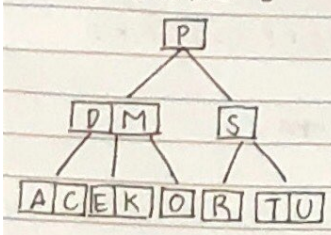
Nebrisca Valonnica 2501964371

1. B-Tree



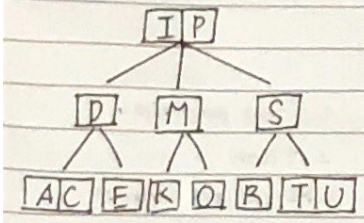
a. Proceed insertion (step by step simulation) on that tree above with these data : E, I, L, G, X

① memasukkan E



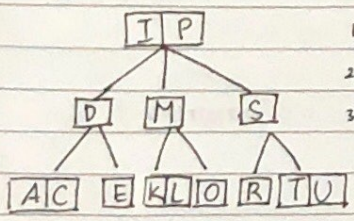
1. belok kiri
2. E ditaruh di anak tengah D dan M

② memasukkan I



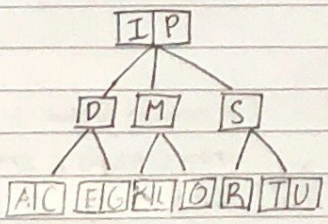
1. belok kiri
2. belok kanan
3. E dan K penuh
4. memisahkan E sama K
5. memisahkan O sama M
6. I naik sebelah P

③ memasukkan L



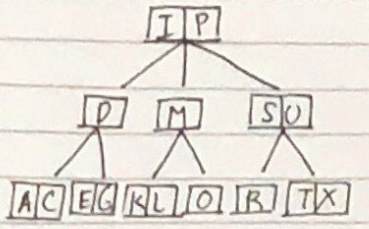
1. lurus
2. lebih kecil dari M
3. masuk sebelah K

④ memasukkan G



1. belok kiri
2. belok kanan
3. memasukkan di sebelah E

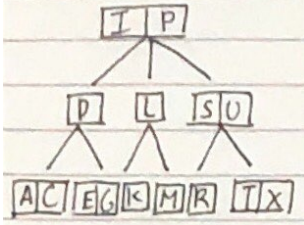
⑤ memasukkan X



1. belok kanan
2. belok kanan
3. split
4. U naik X turun
5. karena parent sudah memiliki 2 anggota masukkan T ke anak tengah

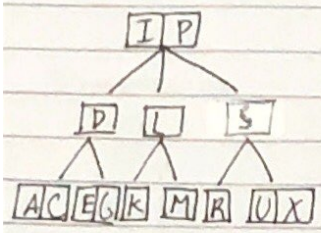
b. Proceed deletion (step by step simulation) after doing the insertion on point a with these order : O, T, D, P, X, S, I, C, M, K, E

① hapus O



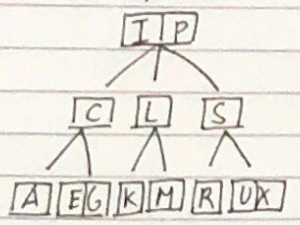
1. M menggantikan O
2. M digantikan L

② hapus T



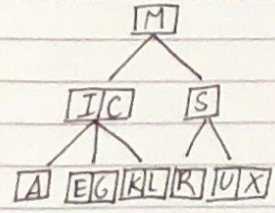
1. T menggantikan U

③ hapus D



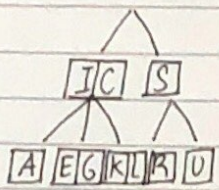
1. D menggantikan C

④ hapus P



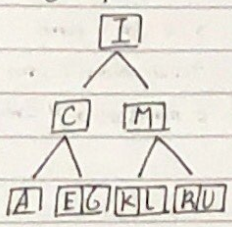
1. I turun ke kiri
2. KL disambung jadi anak kanan IC

⑤ hapus X



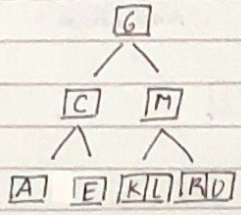
1. hapus X

⑥ hapus S



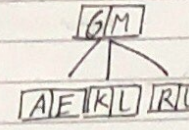
1. I naik menggantikan M
2. M turun
3. KU jadi anak kanan M
4. KL jadi anak kiri M

⑦ hapus I



1. G menggantikan I

⑧ hapus C

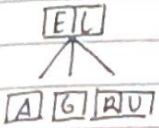


1. M naik ke tingkat G karena C di hapus A sama E disambungkan menjadi anak kiri nya G sama M

⑨ hapus M

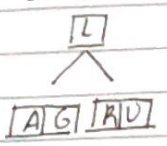
1. L menggantikan M

⑩ hapus K



1. G menggantikan K
2. E naik

⑪ hapus E



1. hapus E

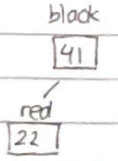
2. Red Black Tree

a. Create a Red black tree (step by step simulation) using the following sequence : 41, 22, 5, 51, 48, 29, 18, 21, 45, 3

① memasukkan 41

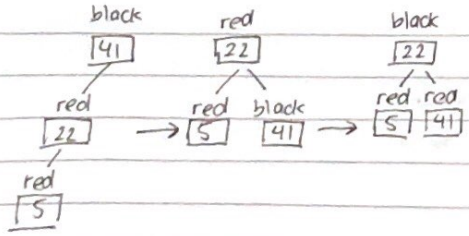
1. 41 masuk

② memasukkan 22



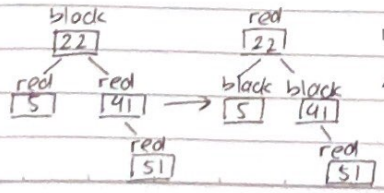
1. karena 22 lebih kecil dari 41 maka 22 menjadi anak kiri

③ memasukkan 5



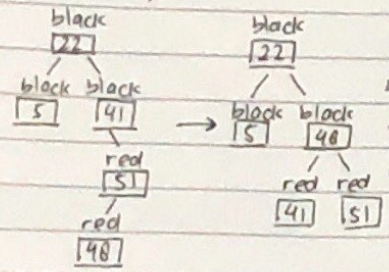
1. 5 lebih kecil dari 41 dan 22 maka lihat anak kiri
2. karena tidak seimbang maka di right rotate
3. pindahkan warna hitam ke parentnya

④ memasukkan 51



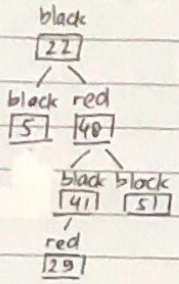
1. 51 lebih besar dari 41 dan 22 maka jadikan anak kanan
2. karena 51 red, turunkan warna hitam dari grandparent ke parent

⑤ memasukkan 48



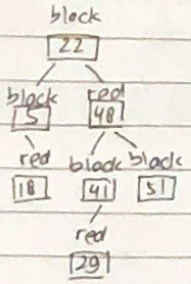
1. 48 lebih besar dari 22 dan 41 lihat anak kanan
2. 48 lebih kecil dari 51 jadikan anak kiri
3. lalu right rotate 51 dan 48, setelah itu left rotate 41, 48, 51
4. pindahkan warna hitam ke parent

⑥ memasukkan 29



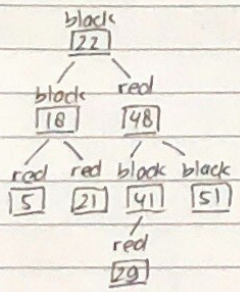
1. 29 lebih besar dari 22 maka lihat anak kanan
2. 29 lebih kecil dari 41 dan 48 maka lihat anak kiri, jadikan 29 anak kiri
3. turunkan warna hitam dari grand parent ke parent

⑦ memasukkan 18



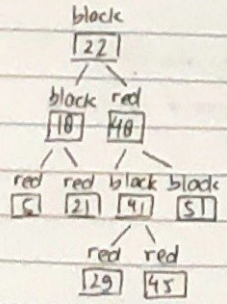
1. 18 lebih kecil dari 22 maka lihat anak kiri
2. 18 lebih besar dari 5 maka jadikan anak kanan

⑧ masukkan 21



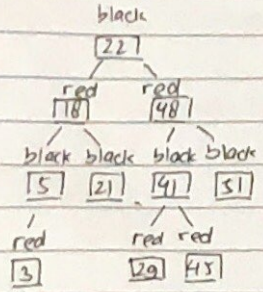
1. 21 lebih kecil dari 22, maka lihat anak kiri
2. 21 lebih besar dari 18 maka jadikan anak kanan
3. left rotate 5, 18, 21, naikkan black dari 5 ke 18

⑨ masukkan 45



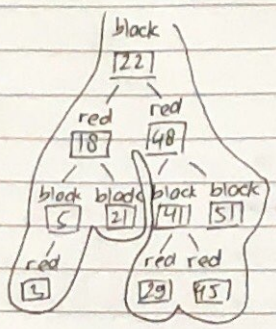
1. 45 lebih dari 22 maka lihat anak kanan
2. 45 kurang dari 48, maka lihat anak kiri
3. 45 lebih dari 41 jadikan anak kanan

⑩ masukkan 3



1. 3 lebih kecil dari 22, 18, 48, 5, 21, 41, 51, 29, 45 maka jadikan anak kiri
2. pindahkan warna hitam ke parent

b. In Order Traversal



3. AVL Tree

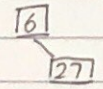
a. Into empty AVL Tree :

i. Insert (step by step simulation) the following values : 6, 27, 19, 11, 36, 14, 81, 63, 75

① masukkan 6

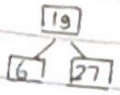
6

② masukkan 27



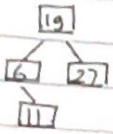
1. 27 lebih dari 6, maka buat 27 menjadi anak kanan 6

④ masukkan 19



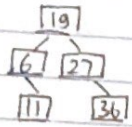
1. Karena 19 lebih besar dari 6, maka lihat ke anak kanan 6
2. Karena 19 lebih kecil dari 27 dan anak kiri 27 kosong ; masukkan 19 ke anak kiri 27
3. 6 menjadi tidak seimbang , maka right rotate dahulu 19 dengan 27 , barulah left rotate 6, 19, 27
4. 19 sekarang menjadi parent , 6 anak kirinya dan 27 anak kanan

⑨ masukkan 11



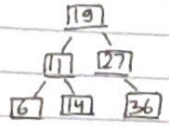
1. 11 lebih dari 6 , maka buat 11 menjadi anak kanan 6

⑤ masukkan 36



1. 36 lebih dari 27, maka buat 36 menjadi anak kanan 27

⑥ masukkan 14

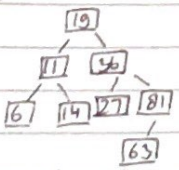


1. Karena 14 lebih kecil dari 19 maka lihat anak kiri 19
2. Karena 14 lebih besar dari 6 maka lihat anak kanan 6
3. Karena 14 lebih besar dari 11, maka lihat anak kanan 11
4. Karena anak kanan 11 kosong , masukkan 14 menjadi anak kanan 11
5. Karena 6 sekarang menjadi tidak seimbang , kita left rotate 6, 11, 14
6. 11 sekarang menjadi parent , dengan 6 anak kirinya dan 14 anak kanannya

⑦ masukkan 81

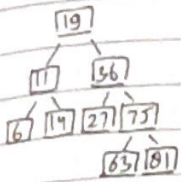
1. Karena 81 lebih besar dari 19 , lihat anak kanan 19
2. Karena 81 lebih besar dari 27 , lihat anak kanan 27
3. Karena 81 lebih besar dari 36, lihat anak kanan 36
4. Karena anak kanan 36 kosong , masukkan 81 menjadi anak kanan 36
5. Karena 27 sekarang menjadi tidak seimbang , kita left rotate 27, 36, 81
6. 36 sekarang menjadi parent dengan 27 anak kirinya dan 81 anak kanannya

⑧ masukkan 63



1. 63 kurang dari 81 , maka buat 63 menjadi anak kiri 81

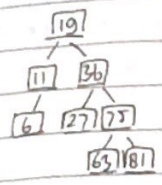
④ masukkan 75



1. karena 75 lebih besar dari 19, lihat anak kanan 19
2. karena 75 lebih besar dari 36, lihat anak kanan 36
3. karena 75 lebih kecil dari 81, lihat anak kiri 81
4. karena 75 lebih besar dari anak kanan maka lihat anak kanan 75
5. anak kanan 63 kosong, jadikan 75 anak kanan 63
6. karena 81 menjadi tidak seimbang, maka left rotate 75, 63 setelah itu lalu right rotate 81, 63, 75
7. 75 menjadi parent, dengan 63 anak kiri nya dan 81 anak kanannya

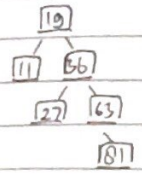
ii Delete (step by step simulation) the following values : 14, 75, 36, 19, 11

① hapus 14



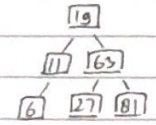
1. hapus 14

② hapus 75



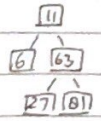
1. 63 naik menggantikan 81

③ hapus 36



1. karena 36 lebih besar dari 19
2. 36 anak kanannya 27
3. left rotate 27, 63, 81

④ hapus 19



1. 11 naik menggantikan 19

⑤ hapus 11

1. karena 6 sebagai root, maka rotasikan 6, 63, 27 ke kiri

4. Essay

a. What are the difference between binary tree and b-tree?

B-tree dan binary tree adalah jenis struktur non linear. Binary tree digunakan ketika catatan atau data disimpan dalam RAM karena kecepatan mengakses RAM jauh lebih tinggi dari pada disk. B-tree digunakan ketika data disimpan dalam disk yang mengurangi waktu akses dengan mengurangi ketinggian pohon dan meningkatkan cabang-cabang dalam node Perbedaan lainnya adalah B-tree dan binary tree harus memiliki simpul anaknya pada tingkat yang sama sedangkan binary tree tidak memiliki masalah tersebut. Binary tree memiliki maksimum 2 subtree atau node sedangkan di B-tree dapat memiliki M no dari subtree atau node dimana M adalah untuk B-tree

b. What are the difference between AVL Tree and Red black Tree? In what case would you want to use a red black tree over an AVL tree and vice versa?

Red black tree memiliki lebih sedikit pencarian karena tidak seperti nya seimbang sedangkan AVL Tree menyediakan pencarian yang lebih cepat dari pada red black tree karena mereka lebih seimbang. Red black tree, memiliki warna hitam atau merah sedangkan AVL Tree tidak memiliki warna. Red black tree menyediakan operasi penyisipan dan penindahan yang lebih cepat dari pada AVL Tree karena lebih sedikit rotasi yang dilakukan karena penyeimbangan yang relatif santai sedangkan AVL Tree menyediakan operasi penyisipan dan penghapusan yang kompleks karena lebih banyak rotasi yang dilakukan. Red black tree hanya membutuhkan 1 bit informasi per node sedangkan AVL Tree menyimpan faktor keseimbangan atau ketinggian dengan setiap node, sehingga memerlukan penyimpanan untuk integer per node Red black tree tidak memberikan pencarian yang efisien sedangkan AVL Tree menyediakan pencarian yang efisien