Final Project

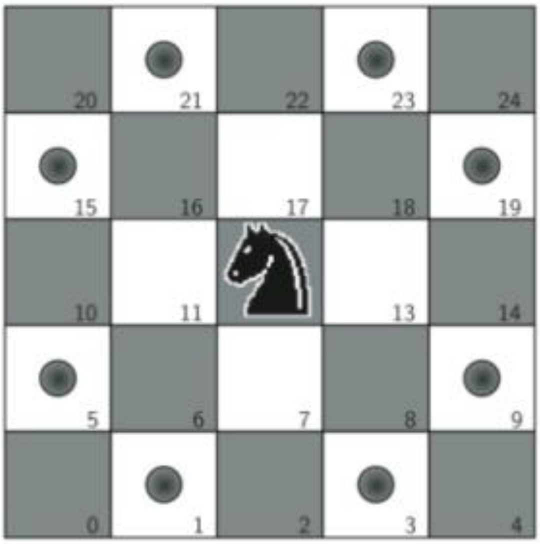
1. Obscure Binary Search Tree

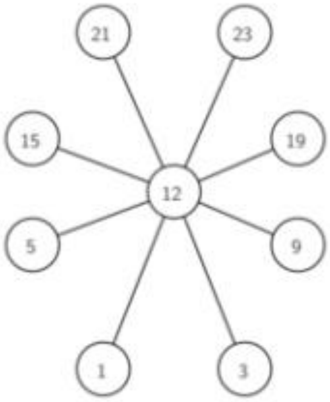
AVL Tree : Insert and Delete 0(log(n)), it's means AVL tree have constant time, and AVL tree is balanced.

Red Black Tree : Insert and Delete 0(log(n)), have more rotations during insertsion and deletion.

What's different? AVL Tree is use, if you need less insertsion and deletion, because it have less rotaion than Red Black Tree, if you need more insertsion and deletion, use Red Black Tree. Red Black Tree faster than AVL tree in several conditions.

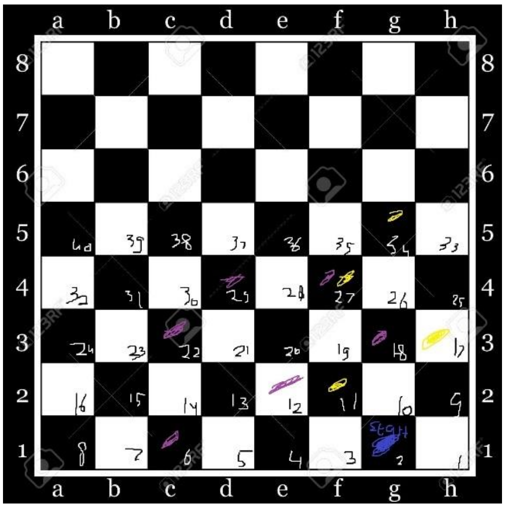
2. Knight's Travails

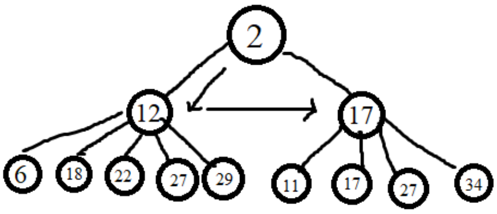




For this case we can see that the sortest path in unweighted graph, so we can use BFS to solve it.

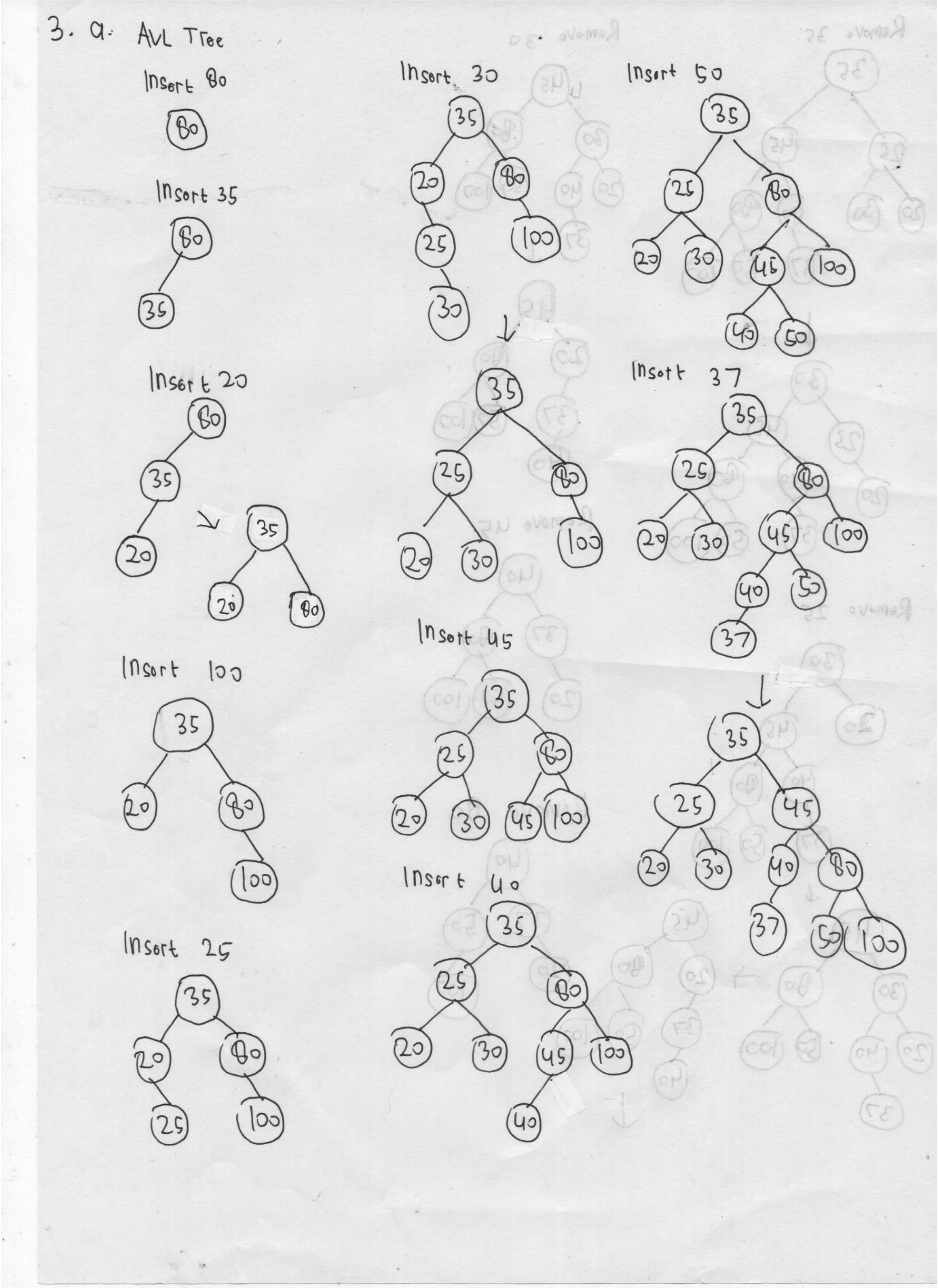
Start with knight's possible moves and select the move that had the least distance from the goal position. As the process goes by, it would be shown on the display that the knight moved all the way. And then continue to search for the next move until the whole process is complete.


a b c d e f g h
8 7 6 5 4 3 2 1
40 39 38 37 36 35 34 33
32 31 30 29 28 27 26 25
24 23 22 21 20 19 18 17
16 15 14 13 12 11 10 9
8 7 6 5 4 3 2 1
Start

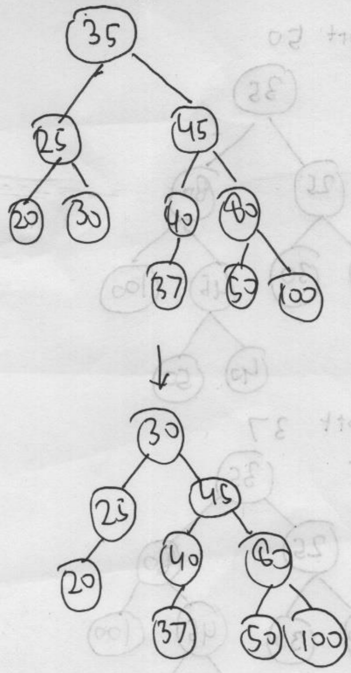

2
12
17
6
18
22
27
29
11
17
27
34

3. Tree Simulations

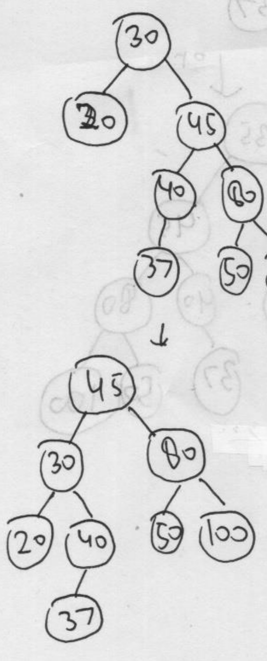
a. AVL Tree



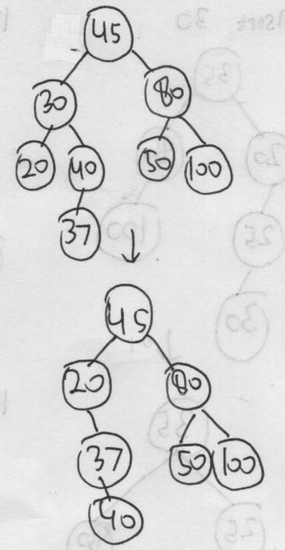
Remove 35



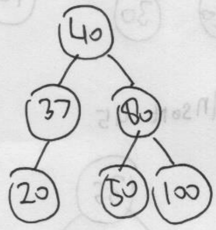
Remove 25



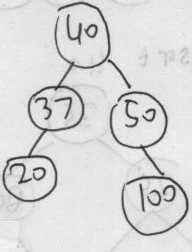
Remove 30



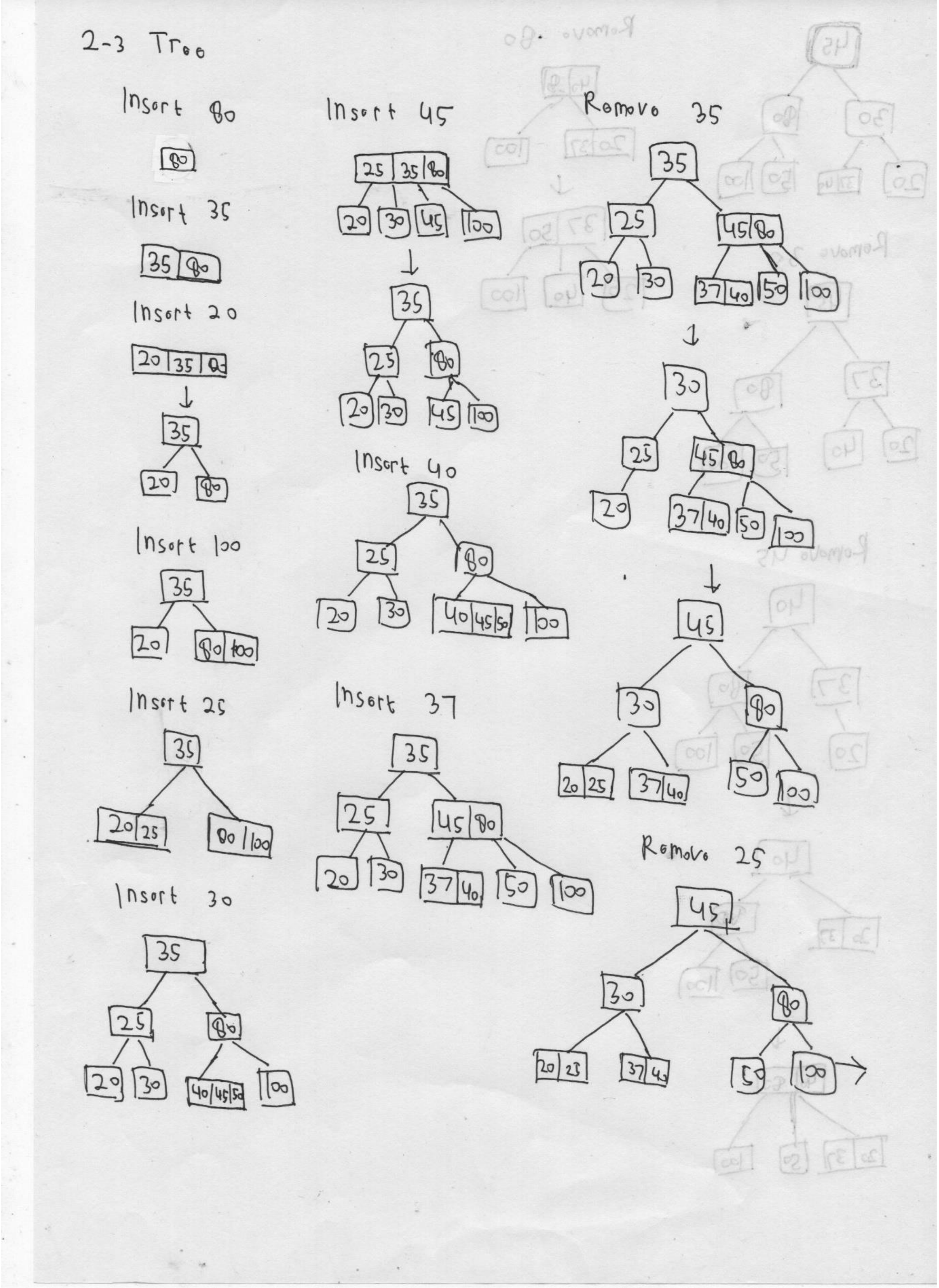
Remove 45

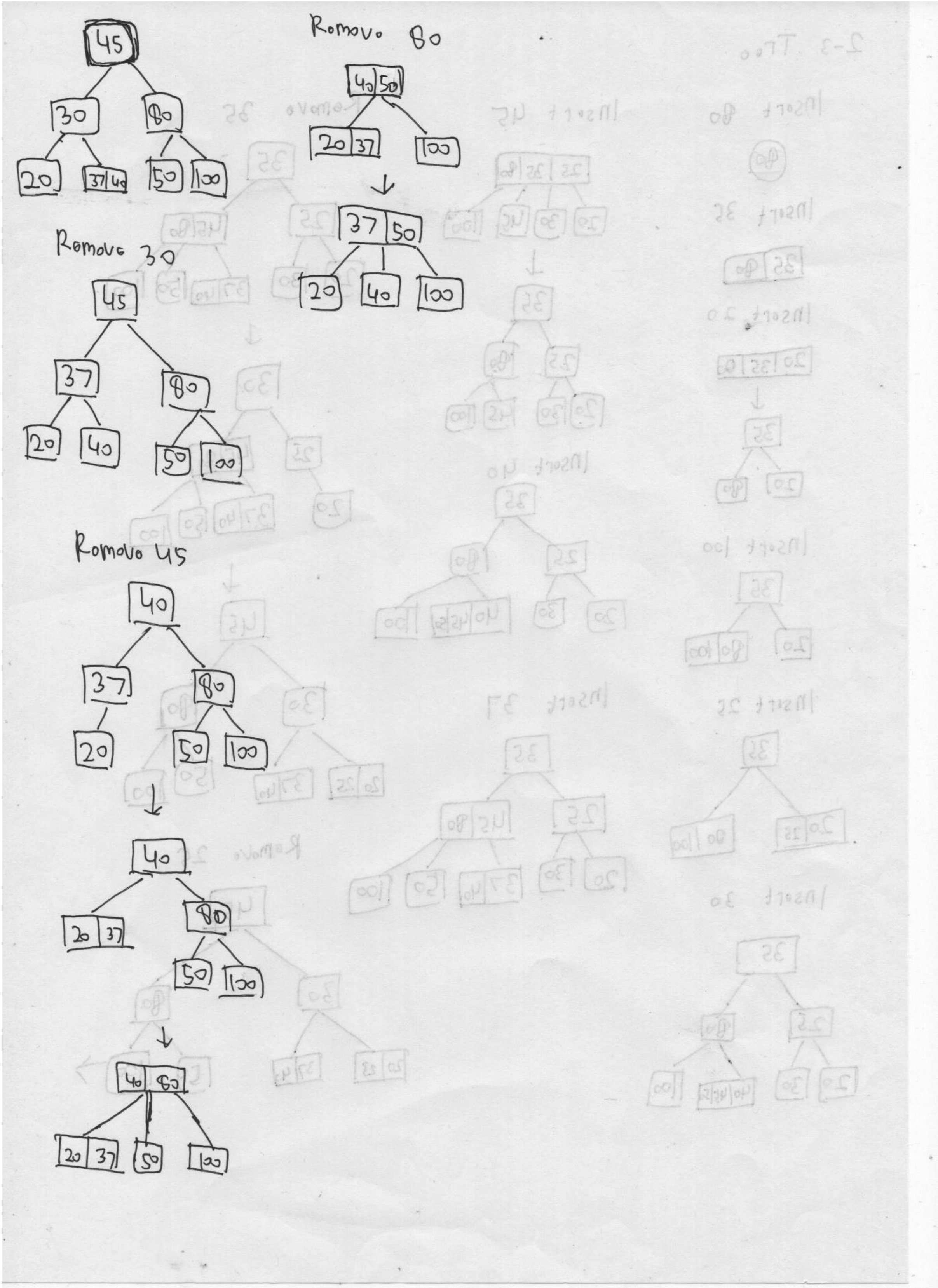


Remove 80

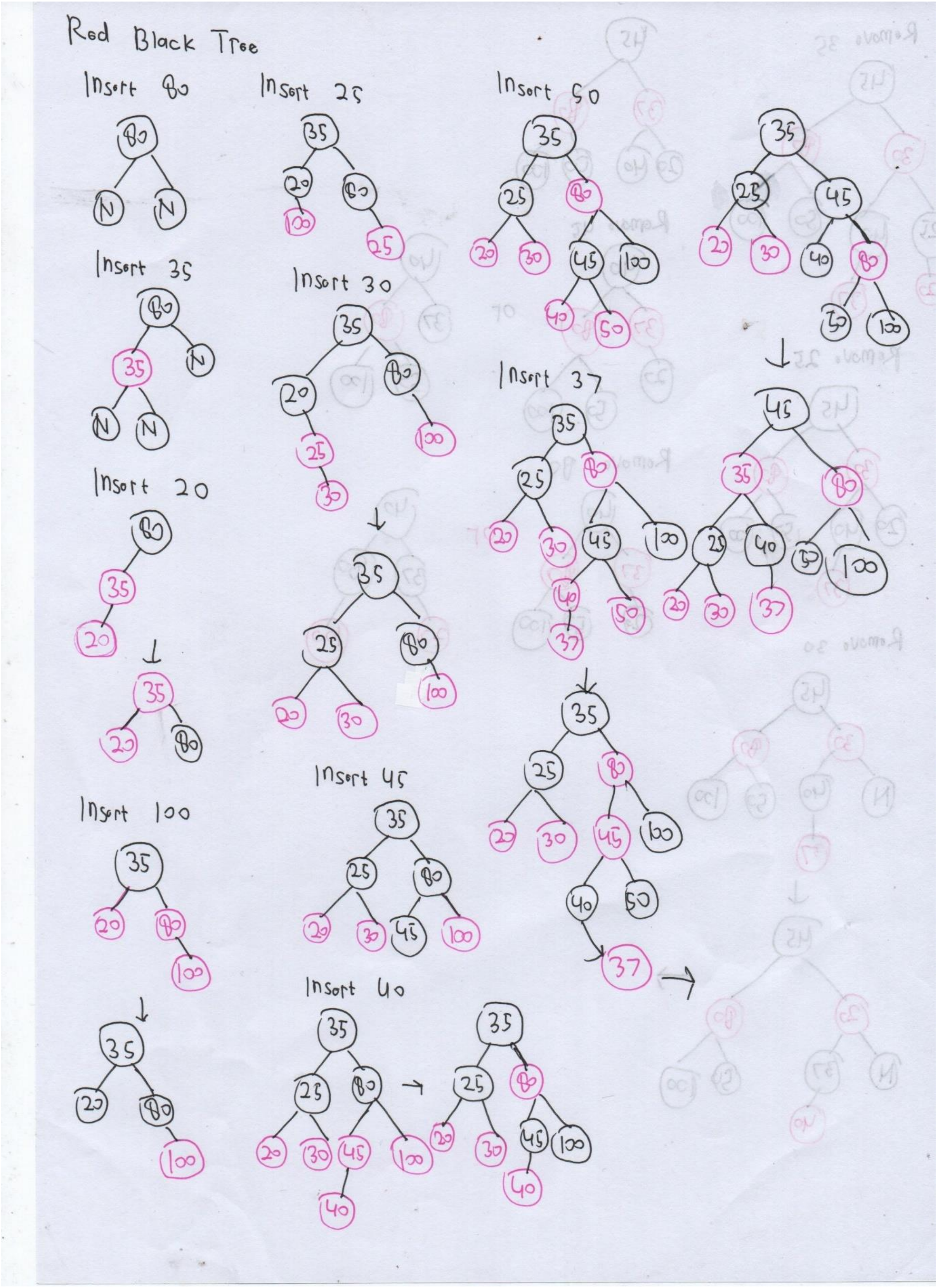


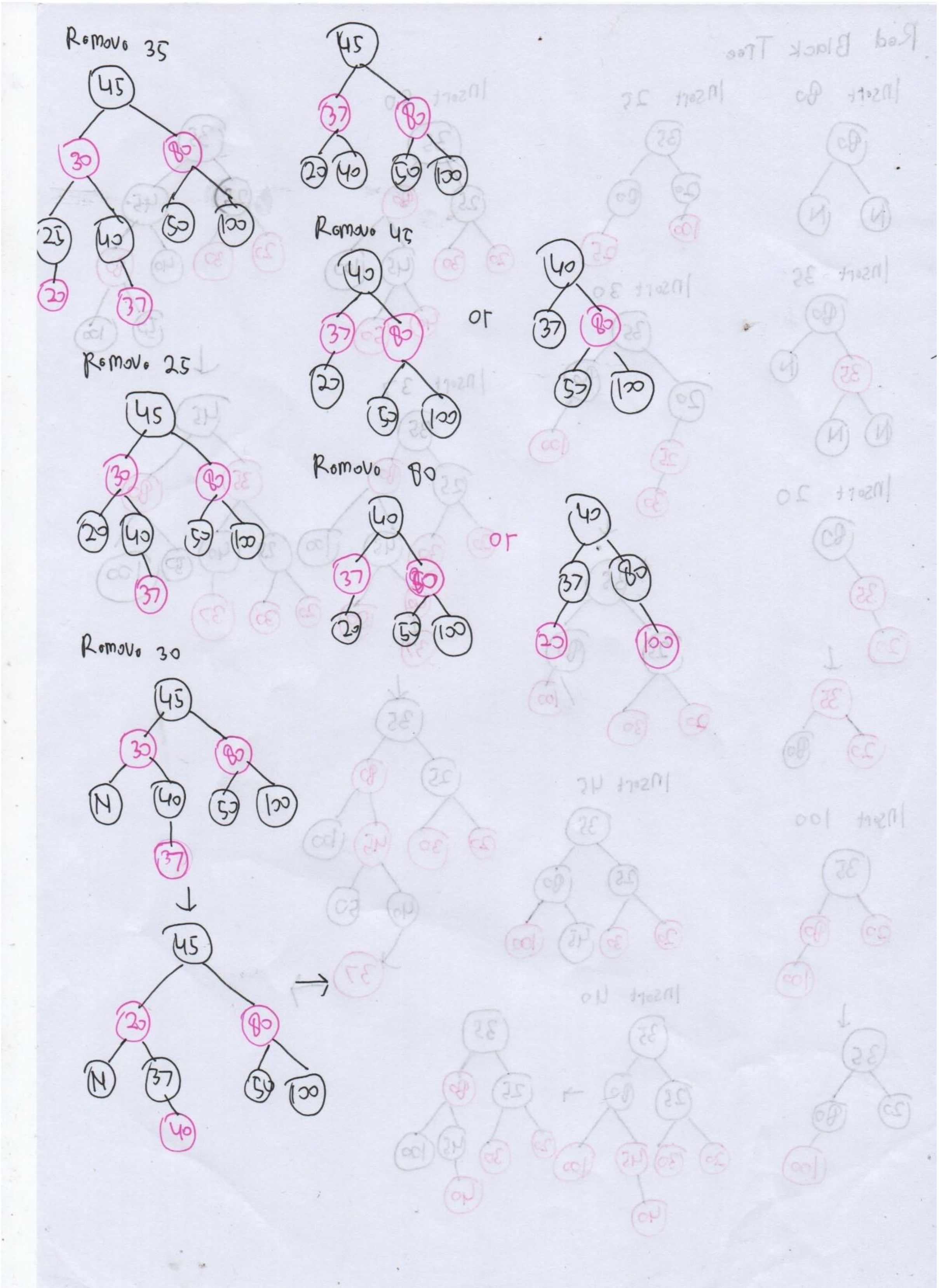
2-3 Tree


2-3 Tree
Insert 80
80
Insert 35
35 | 80
Insert 20
20 | 35 | 80
35
20
80
Insert 100
35
20
80 | 100
Insert 25
35
20 | 25
80 | 100
Insert 30
35
25
80
20
30
40/45/50
100
Insert 45
25 | 35 | 80
20
30
45
100
35
25
80
20
30
45
100
Insert 40
35
25
80
20
30
40 | 45 | 50
100
Insert 37
35
25
45 | 80
20
30
37 | 40
50
100
Remove 35
35
25
45 | 80
20
30
37 | 40
50
100
30
25
45 | 80
20
37 | 40
50
100
45
30
80
20 | 25
37 | 40
50
100
Remove 25
45
30
80
20 | 25
37 | 40
50
100

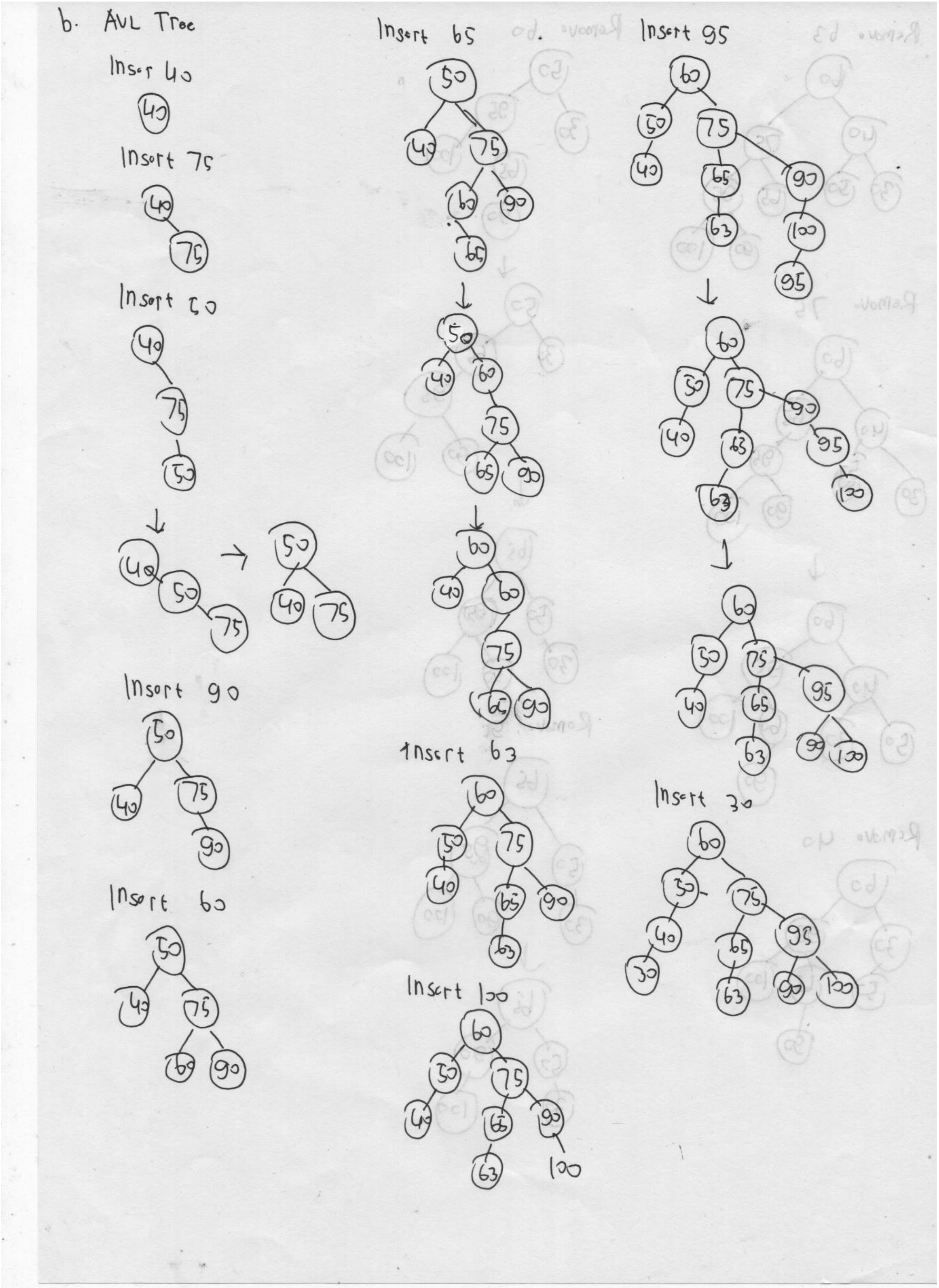

Romove 80
Romove 30
Romove 45
45
30
80
20
37 | 45
50
100
40 | 50
20 | 37
37 | 50
40
37
40
20 | 37
40 | 80

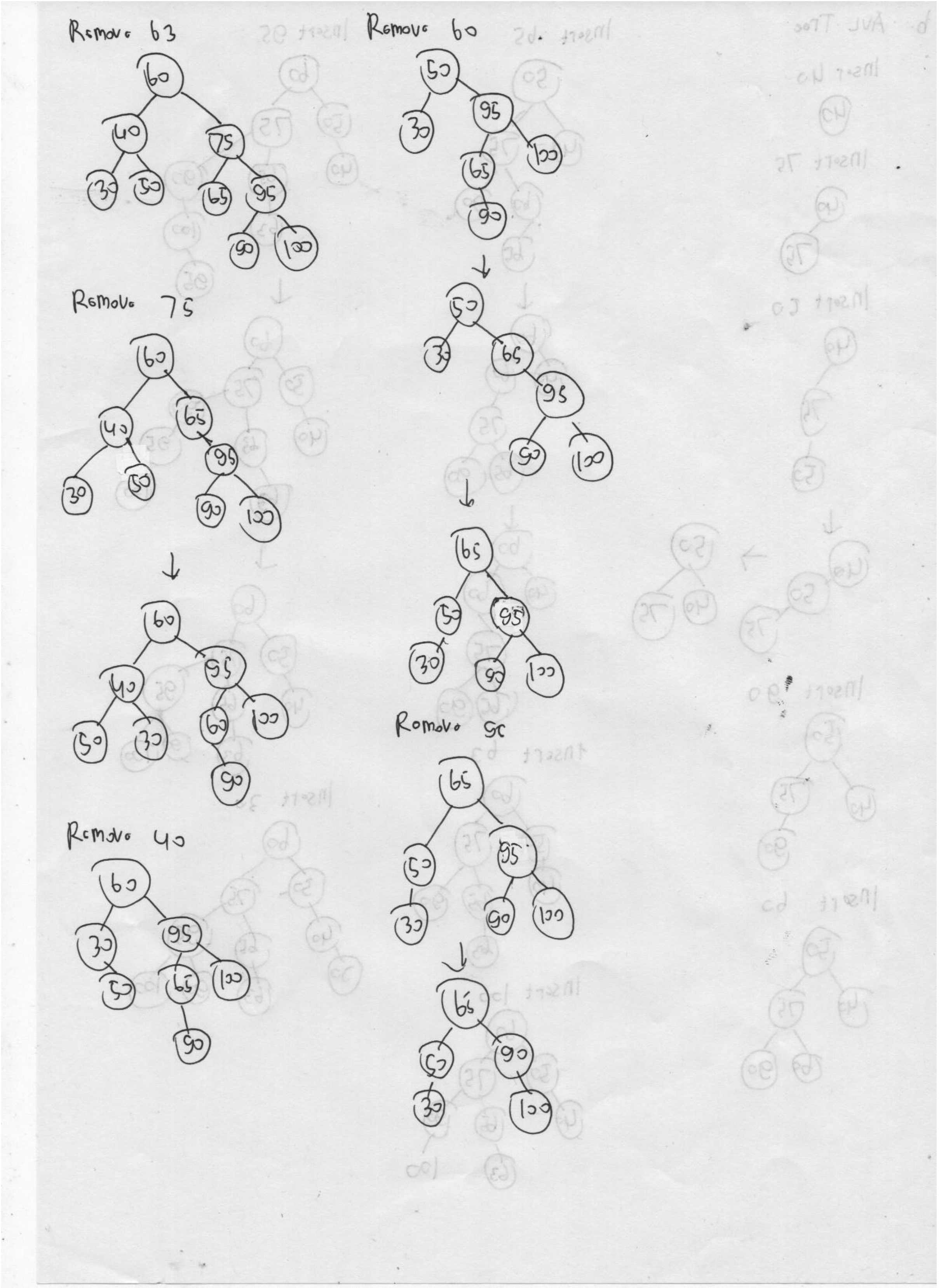
Black Red Tree



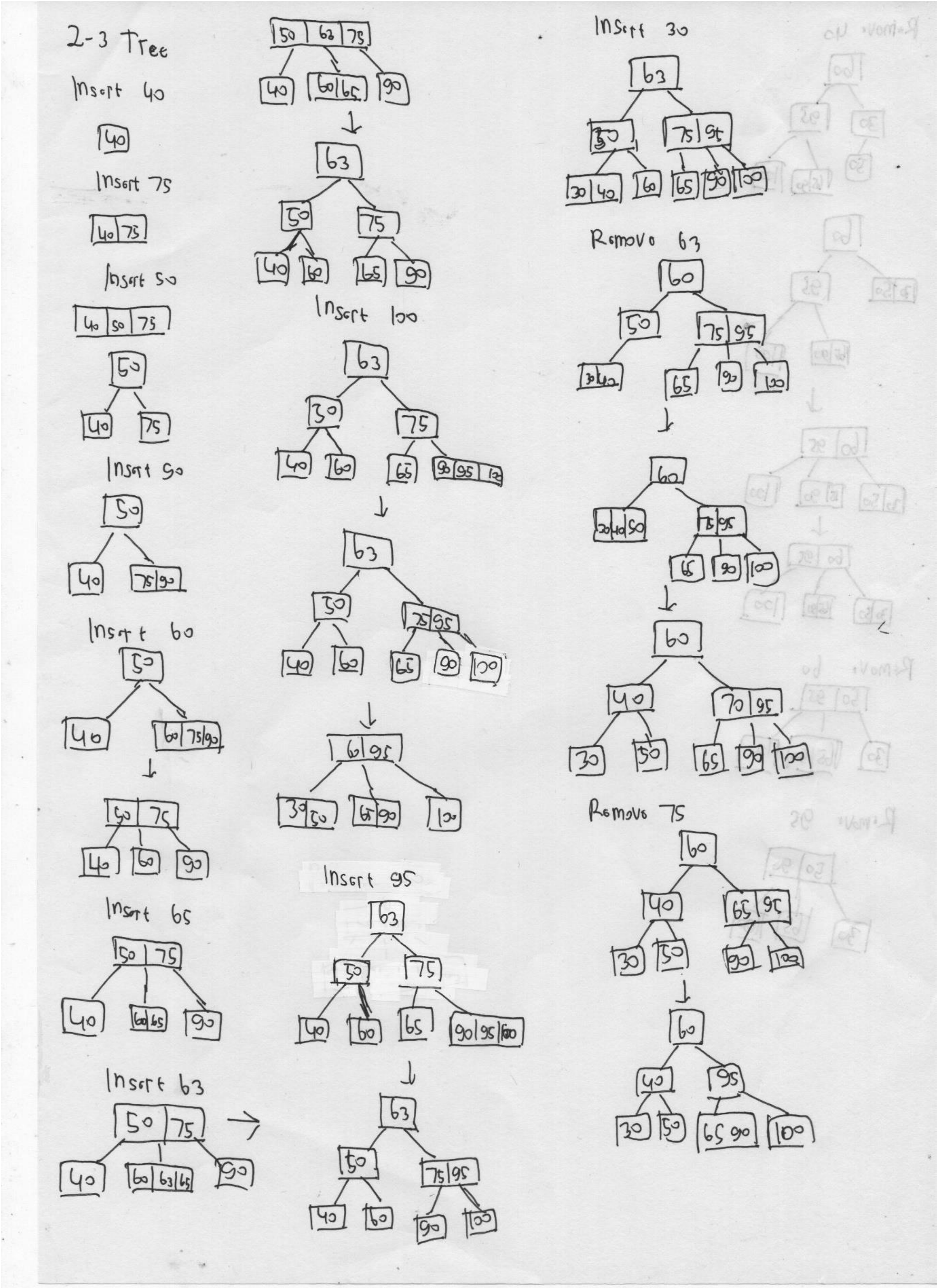

Remove 35
45
30
80
25
40
50
100
20
37
45
37
80
20
40
50
100
Remove 45
40
37
80
20
50
100
or
40
37
80
50
100
Remove 25
45
30
80
20
40
50
100
37
Remove 80
40
37
80
20
50
100
or
40
37
80
20
100
Remove 30
45
30
80
N
40
50
100
37
45
20
80
N
37
50
100
40

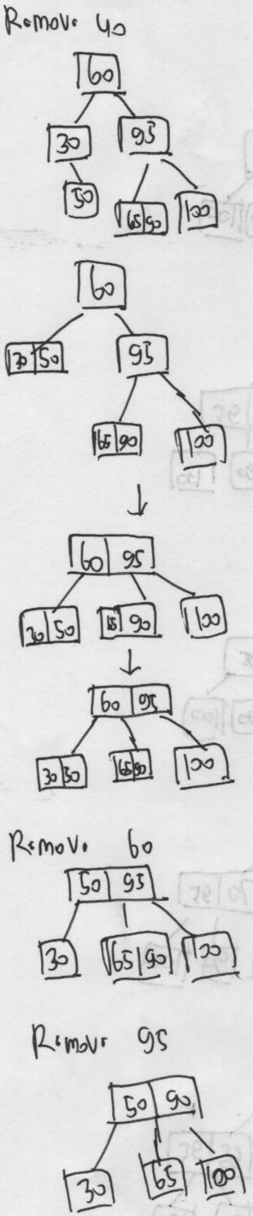
b. AVL Tree


b. AVL Tree
Insert 40
Insert 75
Insert 50
Insert 90
Insert 60
Insert 65
Insert 63
Insert 100
Insert 95
Insert 30

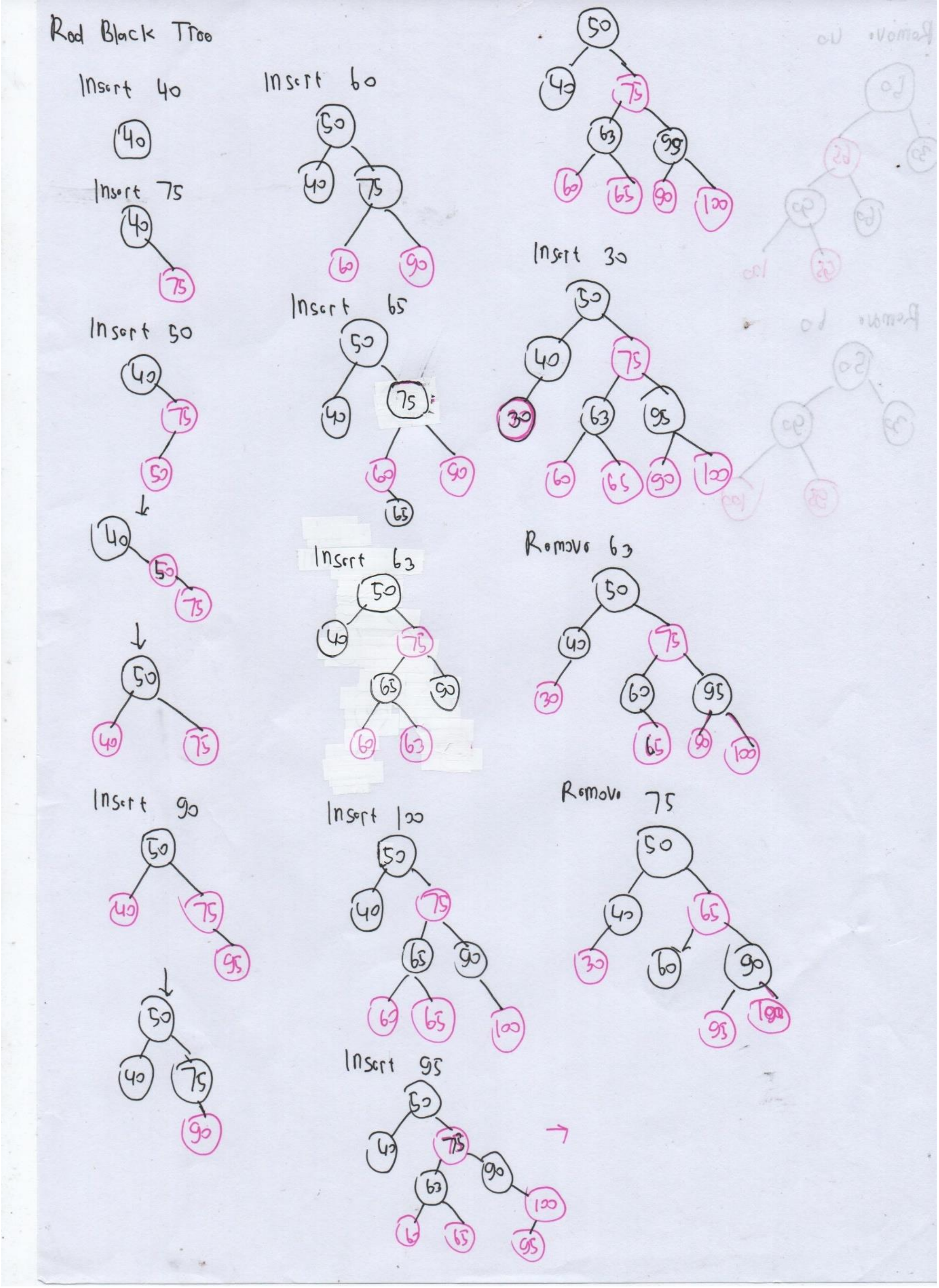

Remove 63
Remove 75
Remove 40
Remove 60
Remove 95

2 3 Tree

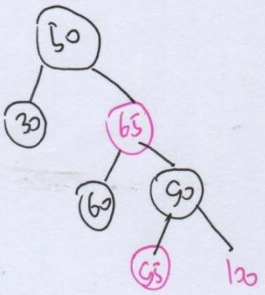

2-3 Tree
Insert 40
Insert 75
Insert 50
Insert 90
Insert 60
Insert 65
Insert 63
Insert 100
Insert 95
Insert 30
Remove 63
Remove 75


Remove 40
60
30
95
50
65|90
100
60
30|50
95
65|90
100
60|95
30|50
65|90
100
60|95
30|50
65|90
100
Remove 60
50|95
30
65|90
100
Remove 95
50|90
30
65
100

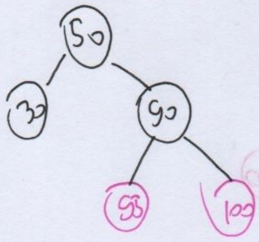
Black Red Tree



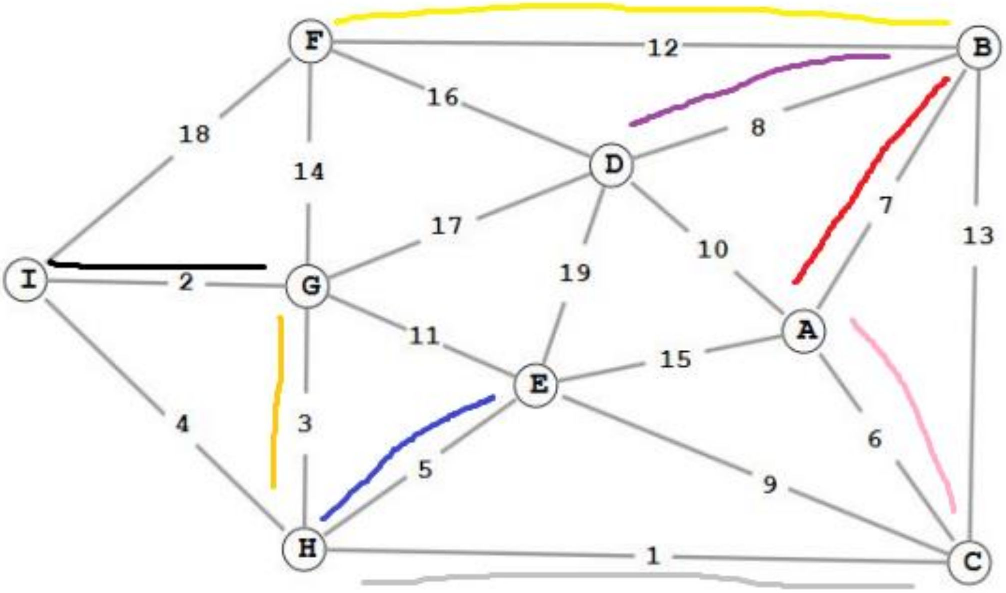
Remove 40



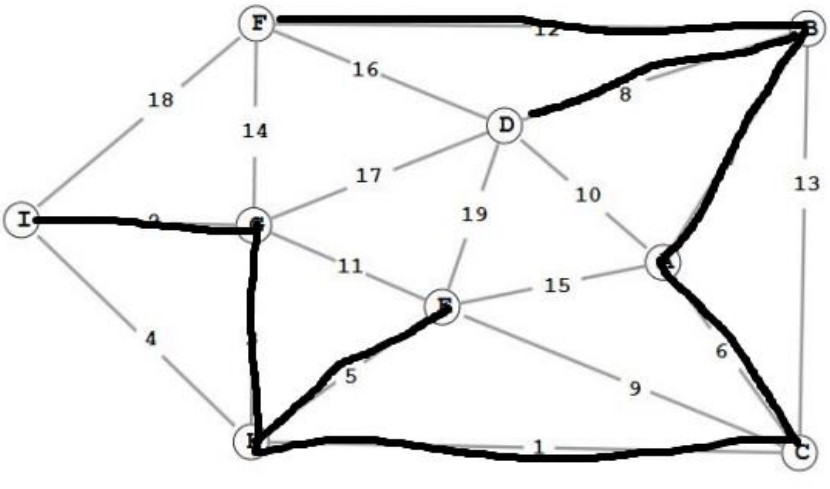
Remove 60



4. Disjoint Set & Graphs
Using Prim



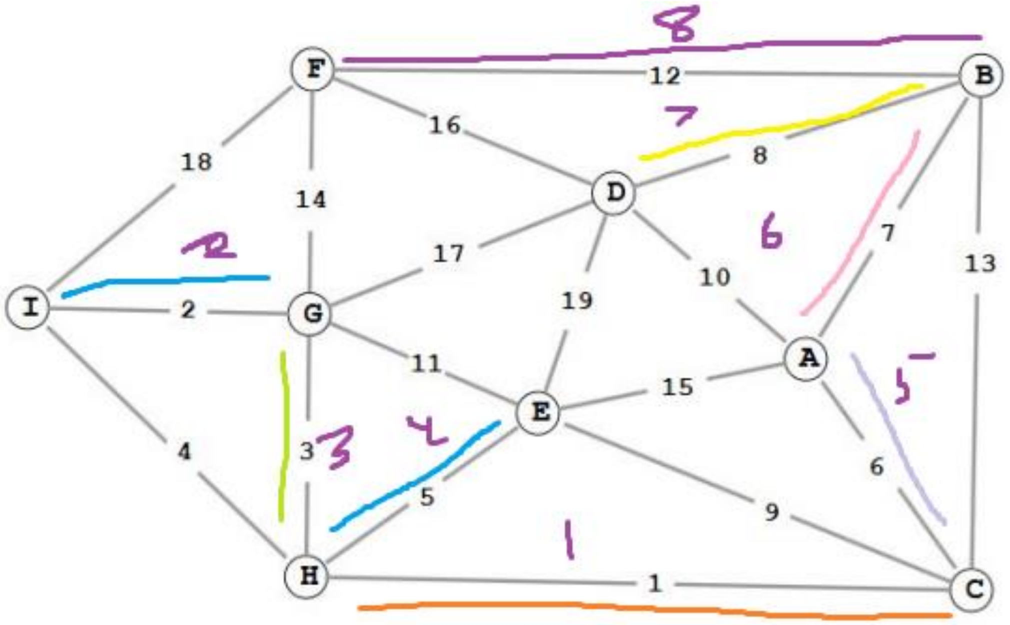
Keterangan :
Shortest Path
I to G = 2
G to H = 3
H to C = 1
C to A = 6
A to B = 7
B to D = 8
B to F = 12
H to E = 5



So minimum cost is = 2+ 3+ 1+6+7+8+12+5 = **44**

Using Kruskal

Pilih yang edge nya paling kecil



Smallest Edge

H to C = 1

I to G = 2

G to H = 3

H to E| = 5

C to A = 6

A to B = 7

B to D = 8

B to F = 12

Mengapa 9,10,11 tidak termasuk? Karena edge tersebut bagian dari siklus

So minimum cost is 1+2+3+5+6+7+8+12 = **44**