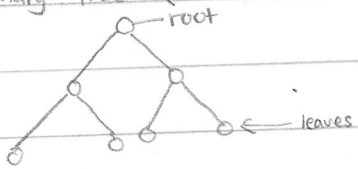
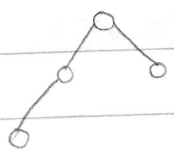
Review Exercise 2

Trees

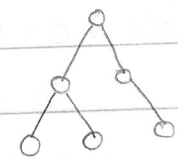
Full Binary Tree (same level and every non leaf has exactly two children)



Complete binary tree (next to last level is full)



or



1. a) How many nodes are in a full binary tree of height 1, 2, 3, 4, 5?

Height	Nodes	
1	← 1 node	$2^1 - 1 = 1$
2	← 3 nodes	$2^2 - 1 = 3$
3	← 7 nodes	$2^3 - 1 = 7$
4	← 15 nodes	$2^4 - 1 = 15$
5		$2^5 - 1 = 31$

b) What is the height of a complete tree that contains 14 nodes?

Height of 4.

c) If n is the number of the nodes in a full binary tree, what is the height of this full tree?

$2^h - 1 = n$

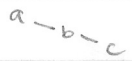
$2^h = n + 1$

$h = \log_2(n+1)$

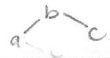
2. We can form several different binary search tree from the same set of data.

a) How many different binary search trees can you form from the strings a, b, c?

Show all of them.

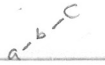
a b c → 

a c b → 

b a c → 

b c a → 

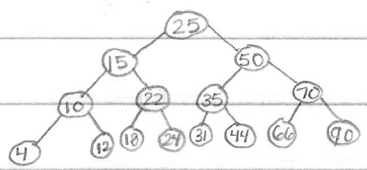
c a b → 

c b a → 

b) What are the heights of the shortest and tallest trees that you formed in question a?

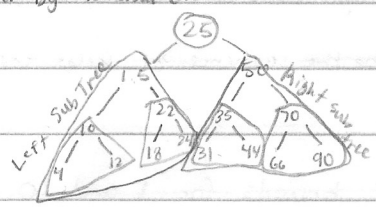
Shortest = 2 Tallest = 3

c) Construct a BST in the order given of the following keys. 25, 15, 50, 10, 22, 35, 70, 4, 12, 18, 24, 31, 44, 66, 90



d) Show node visit order by traversal order

In Order:



4 10 12 (15) 18 22 24 | (25) | 31 35 44 (50) 66 70 90

Pre-Order:

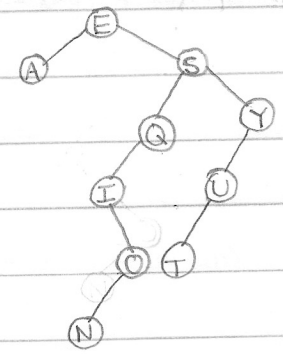
(25) | 15 10 4 12 22 18 24 | 50 35 31 44 70 66 90

Post-Order:

4 12 10 18 24 22 15 | 31 44 35 66 90 70 50 (25)

c. Textbook (p. 416) 3.2.1)

EASY QUESTION Binary Search Tree



In-Order:

A, E, I N O Q S T U Y,

Pre-Order:

E, A, S Q I O N Y U T,

Post-Order:

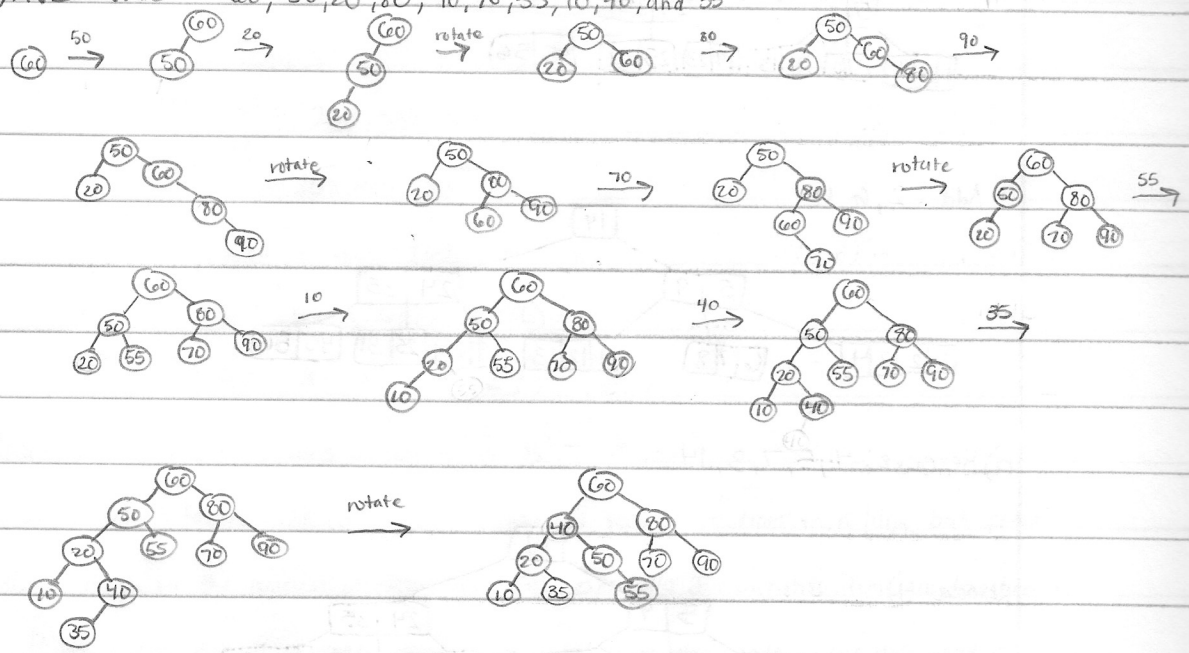
A, N O I Q T U Y S, E

3.2.4) BST has keys that are integers between 1 and 10, and we search for 5. Which sequence below cannot be the sequence of keys examined?

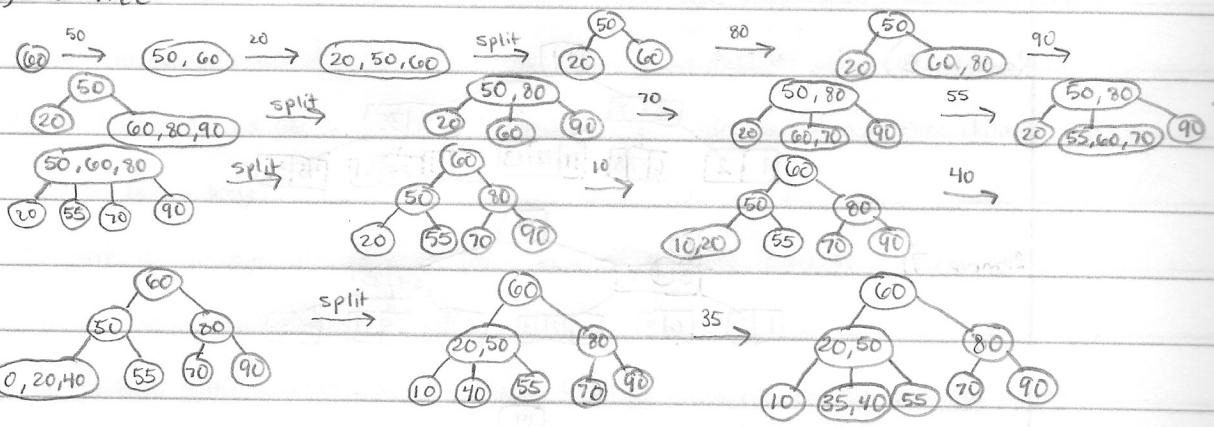
b. 4, 10, 8, 7, 53

Because 53 is out of bounds for 1 and 10 and the element 5 is not found in the sequence.

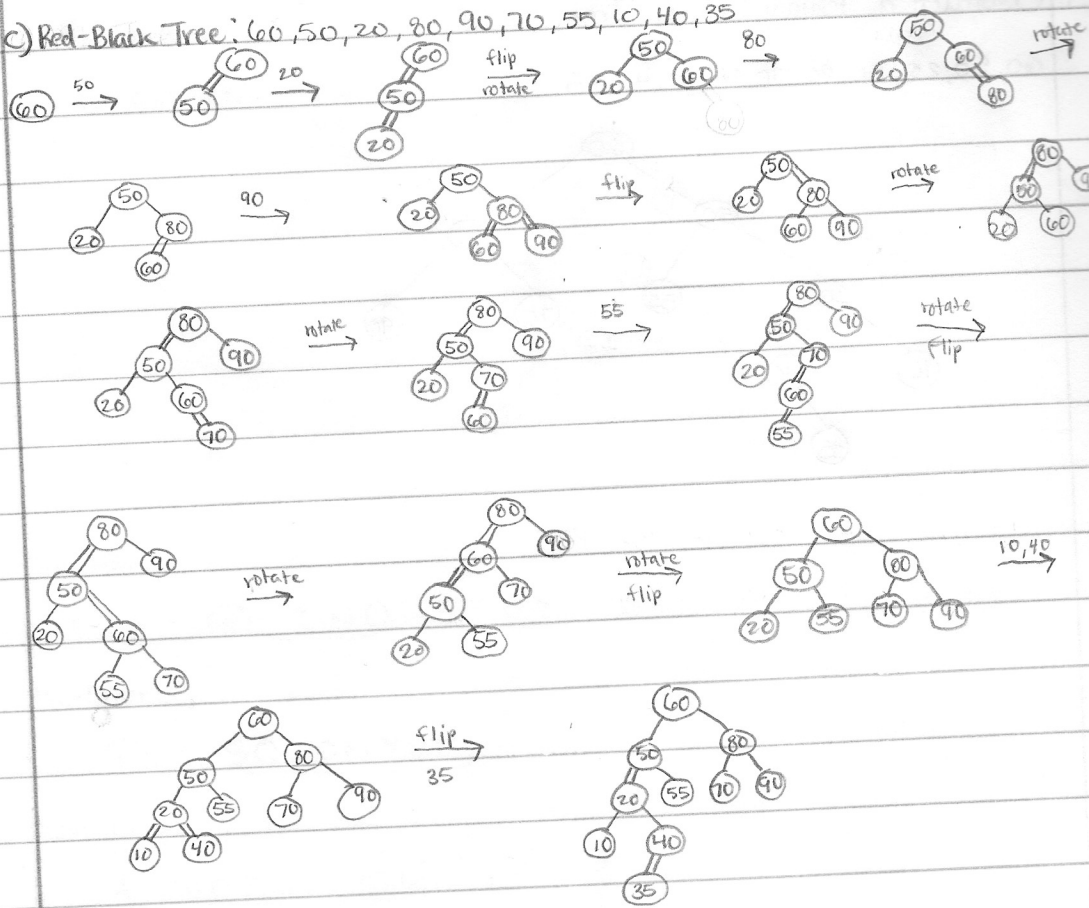
3. a) AVL Tree 60, 50, 20, 80, 90, 70, 55, 10, 40, and 35



b) 2-3 Tree

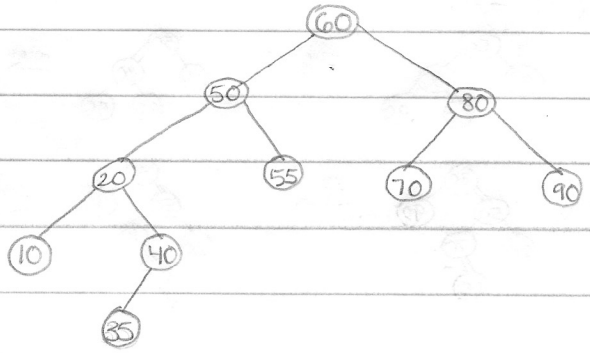


3. c) Red-Black Tree: 60, 50, 20, 80, 90, 70, 55, 10, 40, 35

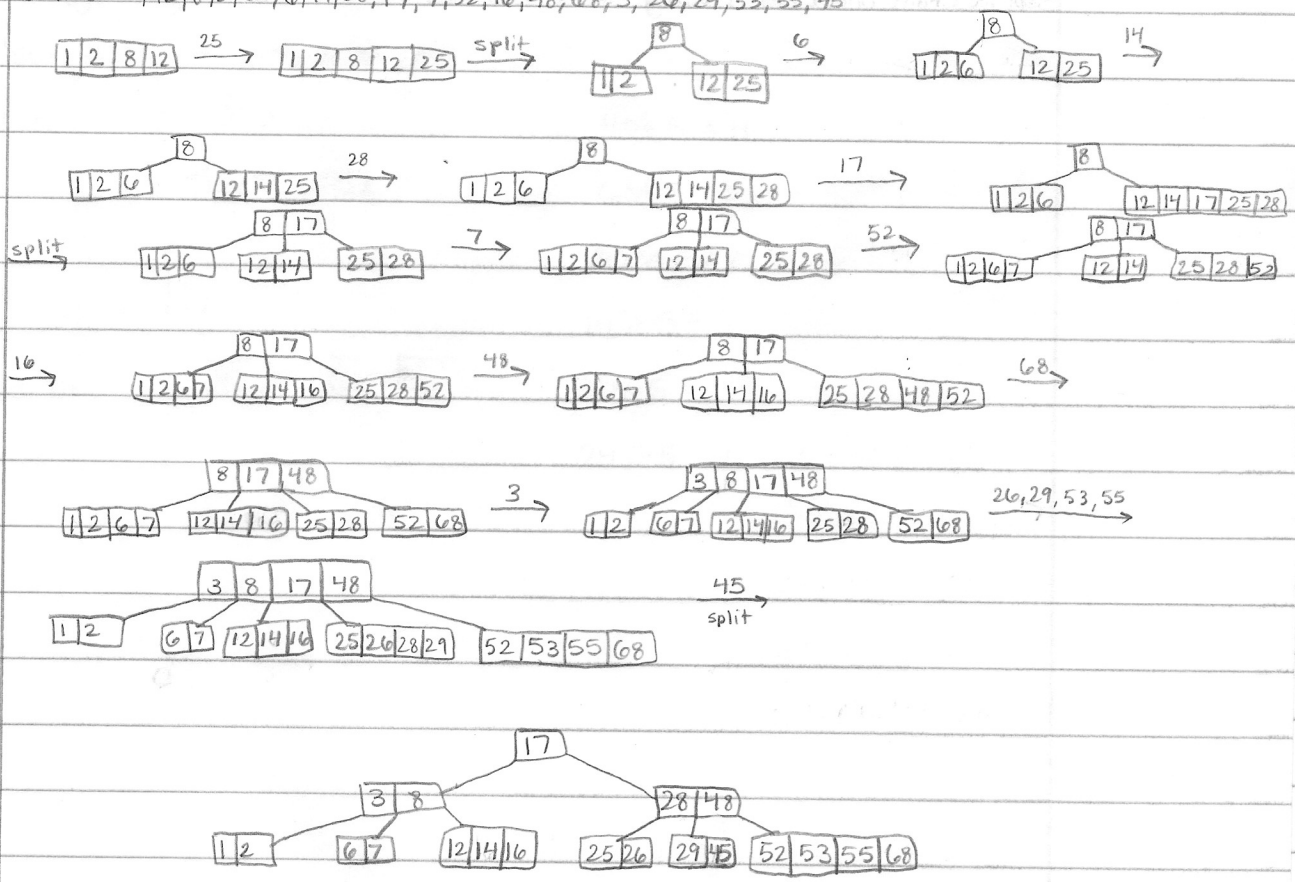


4. Construct a Binary Search Tree

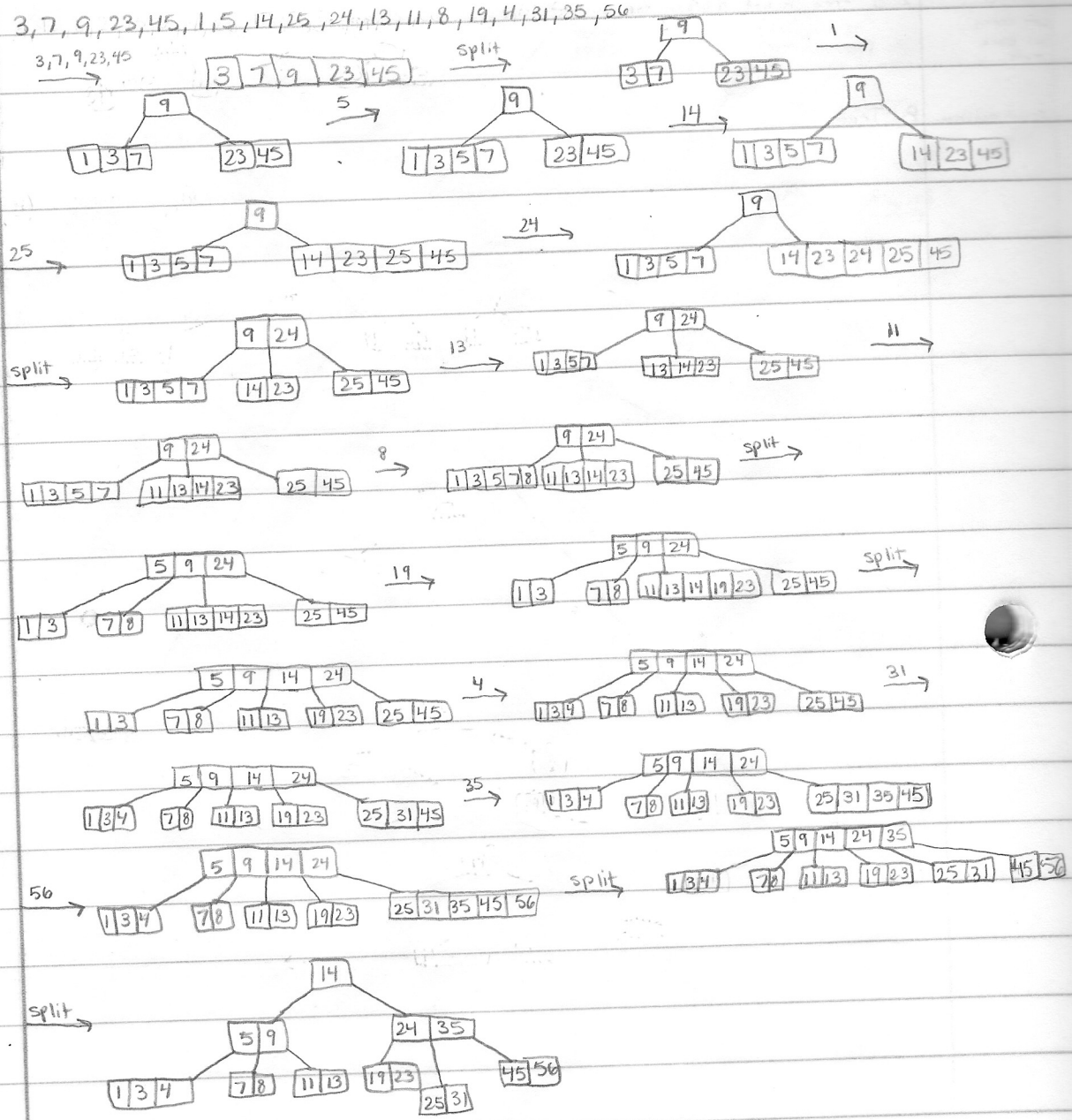
60, 50, 20, 80, 90, 70, 55, 10, 40, 35



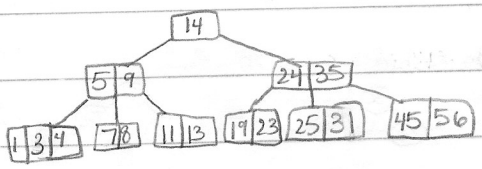
5. B-tree : 1, 12, 8, 2, 25, 6, 14, 28, 17, 7, 52, 16, 48, 68, 3, 26, 29, 53, 55, 45



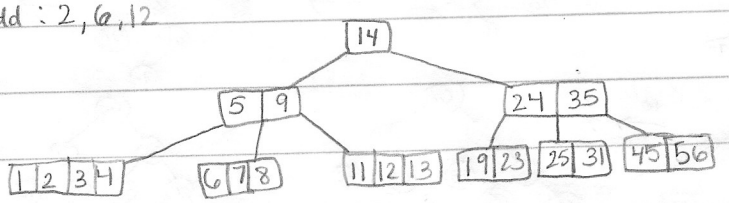
6. 3, 7, 9, 23, 45, 1, 5, 14, 25, 24, 13, 11, 8, 19, 4, 31, 35, 56



7.

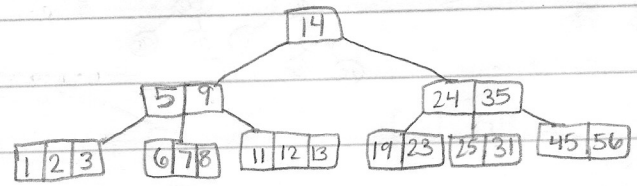


a) Add : 2, 6, 12

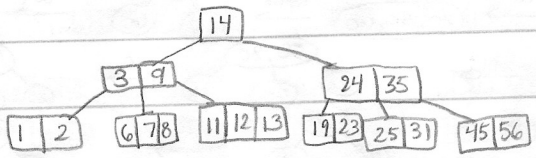


b) Remove: 4, 5, 7, 3, 14

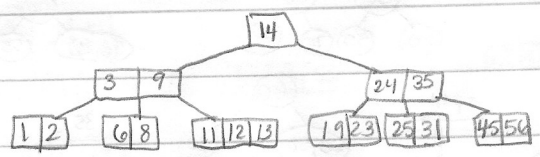
Remove 4)



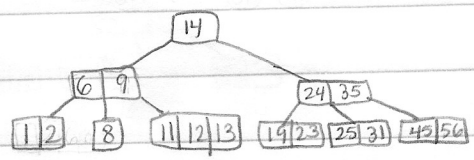
Remove 5)

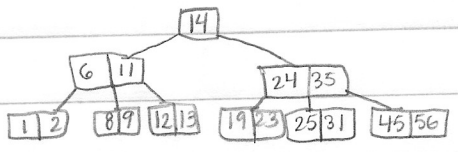


Remove 7)

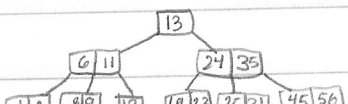


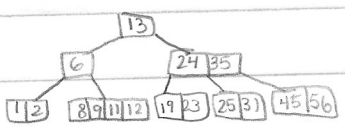
Remove 3)

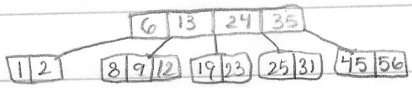




Remove 14)

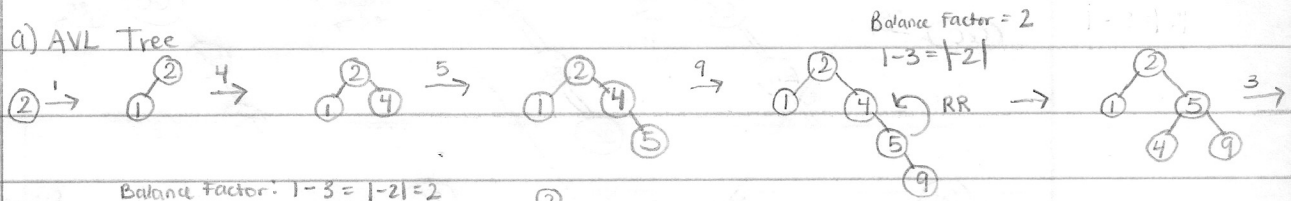


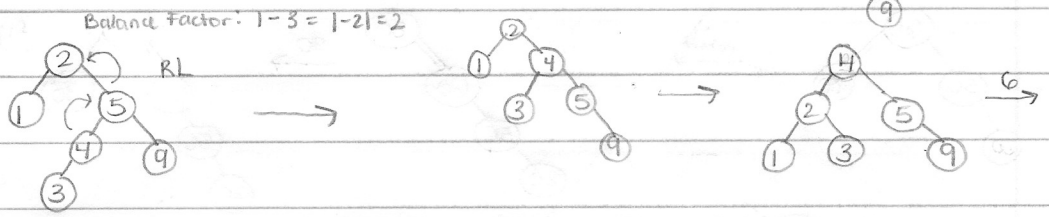


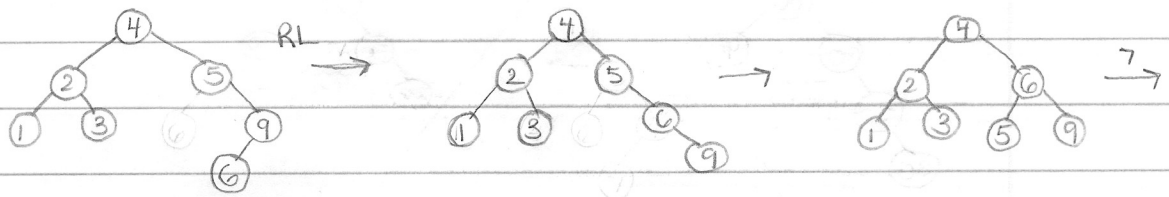


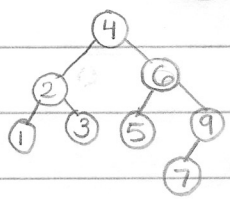
8. For data 2,1,4,5,9,3,6,7

a) AVL Tree

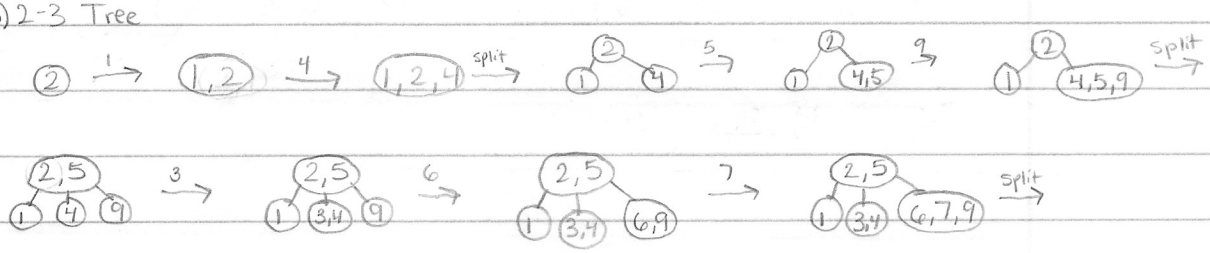


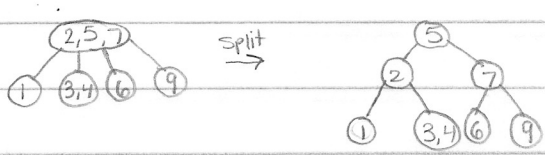






b) 2-3 Tree





9. The maximum number of items in a B-tree of order m and height h:

root $m-1$

level 1 $m(m-1)$

level 2 $m^2(m-1)$

⋮

level h $m^h(m-1)$

So, the total number of items is:

$$(1+m+m^2+m^3+\dots+m^h)(m-1) =$$

$$[(m^{h+1}-1)/(m-1)](m-1) = m^{h+1}-1$$

10. Base case: $h=0$

$$2^{0+1}-1 = 2-1 = 1$$

Induction:

$$2(2^{h+1}-1)+1 = 2^{h+2}-1$$

$$2^{h+1}-1+2^{h+1} = 2^{h+2}-1$$

11. order (m) = 128 $128^{3+1} - 1 = \boxed{268,435,455}$

h = 3

$$2^{h+1} - 1 = 268,435,455$$
$$+1 \quad\quad +1$$
$$2^{h+1} = 268,435,456$$

$$h+1 = \log_2(268,435,456)$$

$$h = \log_2(268,435,456) - 1$$

$$h = 28 - 1 = \boxed{27}$$

12. What are the main reasons of using a B-Tree?

- When searching tables held on disc, the cost of each disc transfer is high but doesn't depend much on the amount of data transferred, especially if consecutive items are transferred.
  - If we use a B-tree of order 101, say, we can transfer each node in one disc read operation.
  - A B-tree of order 101 and height 3 can hold $101^4 - 1$ items (approximately 100 million) and any item can be accessed with 3 disc reads (assuming we hold the root in memory)
- If we take m = 3, we get a 2-3 tree, in which non-leaf nodes have two or three children (i.e., one or two keys)
  - B-trees are always balanced (since leaves are all at the same level), so 2-3 trees make good type of balanced tree.

13. Linear probing is the simplest collision resolution strategy in an open addressing. Linear probing resolves a collision during hashing by examining consecutive locations in the hash table - beginning at the original hash index - to find the next available one.

14.

Linear	Quadratic
i	i
i+1	$i+1^2$
i+2	$i+2^2$
i+3	$i+3^2$
i+4	$i+4^2$
⋮	⋮

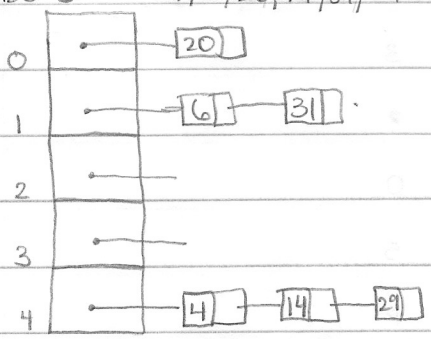
15. $h_1(16) = 16\%13 = 3$ $\quad h_2(16) = 7 - 16\%7 = 5$

Probe sequence	⟹	Table location for hash table size 13
1) $h_1(16) = 3$		3
2) $h_1(16) + h_2(16) = 8$		8
3) $h_1(16) + 2h_2(16) = 13$		0
4) $h_1(16) + 3h_2(16) = 18$		5
5) $h_1(16) + 4h_2(16) = 23$		10
6) $h_1(16) + 5h_2(16) = 28$		2
7) $h_1(16) + 6h_2(16) = 33$		7
8) $h_1(16) + 7h_2(16) = 38$		12
9) $h_1(16) + 8h_2(16) = 43$		4
10) $h_1(16) + 9h_2(16) = 48$		9

16. The size 13 hash table works well because everything fits.

17. h(key) = key % 5

size 5 4, 6, 20, 14, 31, 29



4 % 5 = 4

6 % 5 = 1

20 % 5 = 0

14 % 5 = 4

31 % 5 = 1

29 % 5 = 4