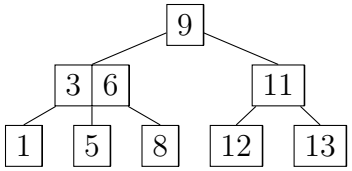
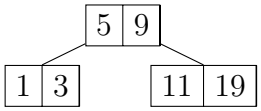
1 B-Tree

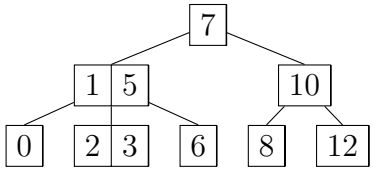
(a) Select all **VALID** 2-3 Tree structures.



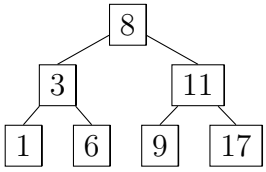
☐



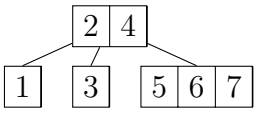
☐



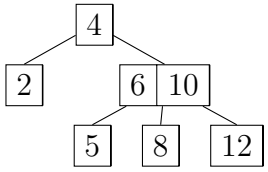
☐



☐



☐

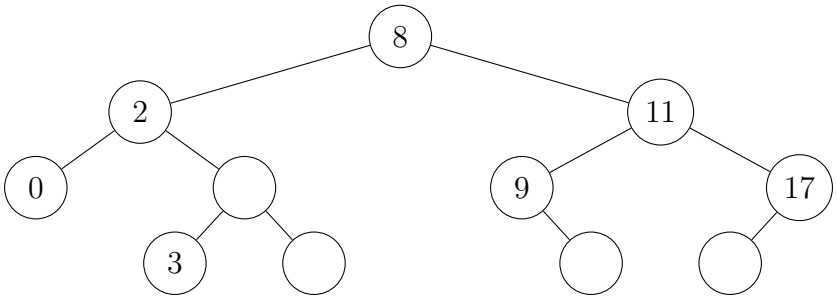


☐

(b) Draw the 2-3 tree that results from **inserting** these items in order: **9,2,6,4,5,11,12,3** (hint: try to keep the invariants.)

2 Binary Search Tree

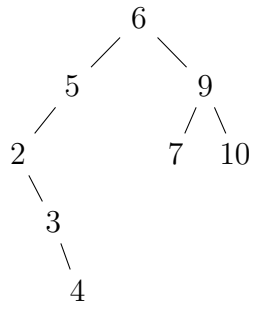
(a) Fill-in the blank nodes of the following binary search tree with valid **integer values**.



Will you want to pick up your worksheet later? Circle one: Yes / No
How confident do you feel with the material this week? Circle one: 1 / 2 / 3 / 4

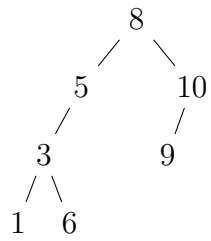
(b) Given the following binary trees, determine if each is a binary search tree, and whether the height of the tree is the same as the height of the optimal binary search tree containing the given elements.

i)



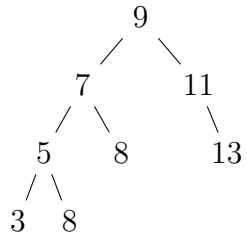
Binary Search Tree: ◯ Valid ◯ Invalid
Height: ◯ Balanced ◯ Unbalanced

ii)



Binary Search Tree: ◯ Valid ◯ Invalid
Height: ◯ Balanced ◯ Unbalanced

iii)



Binary Search Tree: ◯ Valid ◯ Invalid
Height: ◯ Balanced ◯ Unbalanced

(c) (Optional) Given any insertion order of distinct values $x_1, x_2, ..., x_n$ to insert into BST, we want to build a **balanced BST**. Assume that N is the size of the insertion list.

- `findMedian(list)` runs in $f(N)$ time,
- `insertIntoTree(x)` runs in $g(N)$ time and
- `partition(list, x)` which partitions `list` into two lists, one containing values less than `x` and another containing values more than `x`, runs in $h(N)$ time

Give a recurrence relation modeling the worst-case runtime to build this balanced BST.

$T(N) =$ _____ for $N > 1$

Will you want to pick up your worksheet later? Circle one: Yes / No
How confident do you feel with the material this week? Circle one: 1 / 2 / 3 / 4