COMP90038 Algorithms and Complexity

Tutorial 8 AVL Tree and 2-3 Tree

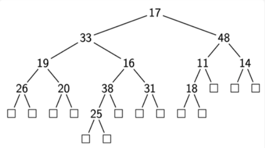
Tutor: Chuang(Frank) Wang

Slides were developed by Chuang Wang
Copyright @ The University of Melbourne (2020)

1. Review

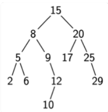
THE UNIVERSITY OF MELBOURNE

Trees



Binary Search Tree

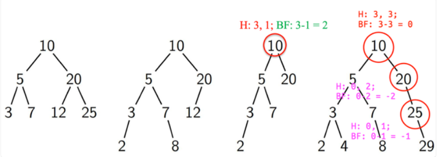
- 1. **Def**: a type of binary tree where **left < root < right (Think recursively)**
- 2. **Structure:** a node that holds a **single item** has **at most two children.**



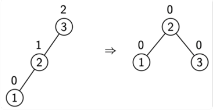
- 3. **The height of a tree**
  - **the number of edges** from the root to the deepest leaf.
  - the empty tree having height **-1.**
- 4. **balance factor:** **Height(Left subtree) - Height(right subtree) (Think recursively!)**

AVL Tree

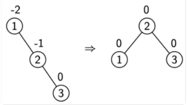
- 1. **Def:** a self-balancing BST in which every single node's balance factor is in **{-1, 0, 1}**



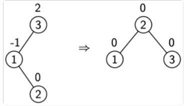
- 2. **Rotation Techniques (in case a node is imbalanced)**
  - Right Rotation



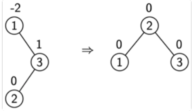
  - Left Rotation



  - Left-Right Rotation

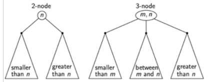


  - Right-Left Rotation

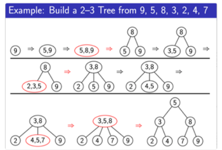


2–3 Trees

- 1. **Def:**
  - each node has either one value or two value
  - The node that holds 2 items has at most 3 children.
  - The node that holds 1 items has at most 2 children.
- 2. **Property:**



- 3. **Insertion:**



- **empty tree?**
  - Yes — create a node and put value into the node
  - No
    - *find the **leaf node** where the value belongs*
    - *put the value into the node*
    - *see #items in the node*
      - # = 2 — Continue
      - # = 3
        - split the node and promote the median of the three values to parent.
        - If the parent then has three values, continue to split and promote, forming a new root node if necessary

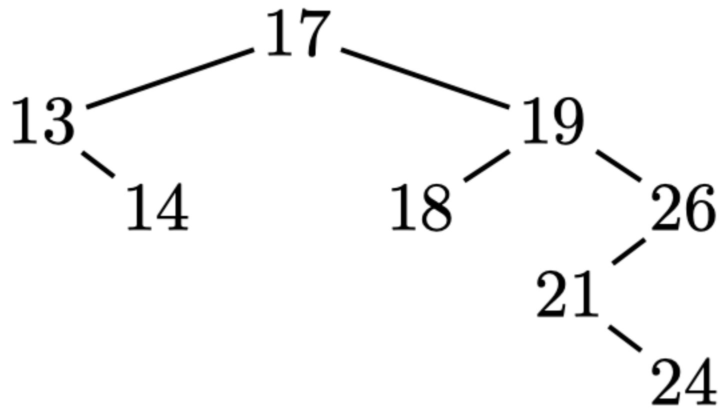
THE UNIVERSITY OF MELBOURNE

2. Tutorial Questions

Question 61

61. Construct a binary search tree (BST) by starting from an empty tree and inserting these keys, in the given order: 17, 19, 13, 26, 14, 18, 21, 24.

Answer:

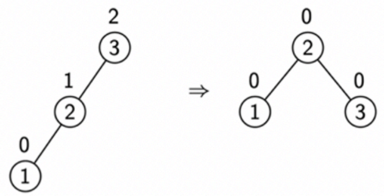


Question 52

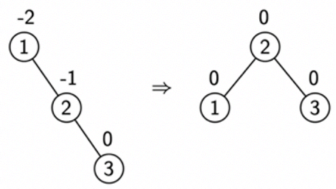
62. Construct an AVL tree from the empty tree by inserting the following keys in the given order: A, L, G, O, R, I, T, H, M.

1, 10, 7, 15, 18, 9, 20, 8, 13

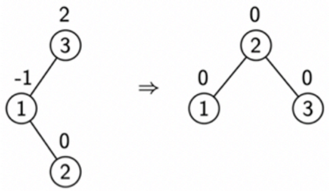
Right Rotation



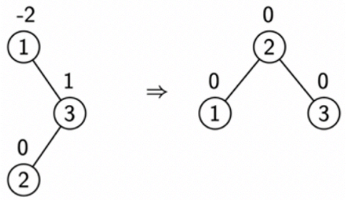
Left Rotation



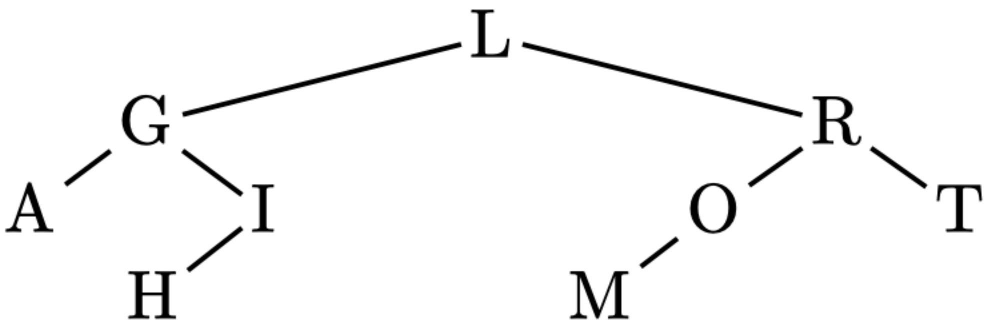
Left-Right Rotation



Right-Left Rotation

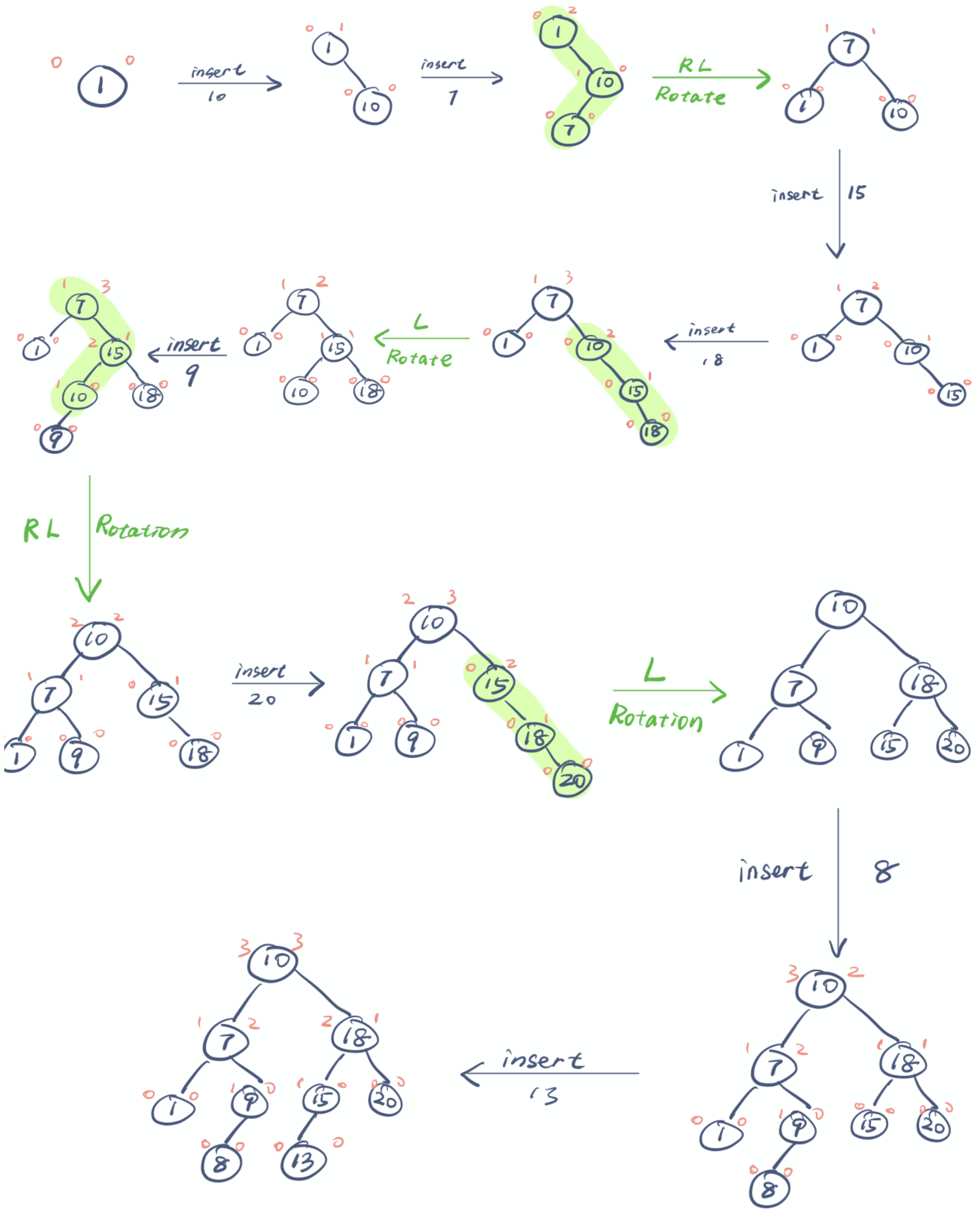


Answer:



Question 52

1, 10, 7, 15, 18, 9, 20, 8, 13



Question 63

63. Consider the set of five keys (let us say they are positive integers) $\{k_1, k_2, k_3, k_4, k_5\}$, satisfying $k_1 < k_2 < k_3 < k_4 < k_5$. There are 120 different permutations of these five keys. For exactly two of the 120 permutations, the following happens, when the keys are inserted one by one, in the order given by the permutation, into an initially empty AVL tree: First an LR-rotation takes place, then an RL-rotation takes place. Which two permutations generate that behaviour?

Answer: Without loss of generality, assume the set of keys is $\{1, 2, 3, 4, 5\}$. After the first three keys are inserted to make an AVL tree, that tree must be perfectly balanced. Hence the first three keys inserted must cause the $\langle$ zigzag path that requires an LR-rotation. After that, insertion of the two remaining keys must cause the $\rangle$ zigzag path that requires an RL-rotation. The only permutations that will achieve this are 3, 1, 2, 5, 4 and 5, 1, 4, 3, 2.

Question 64

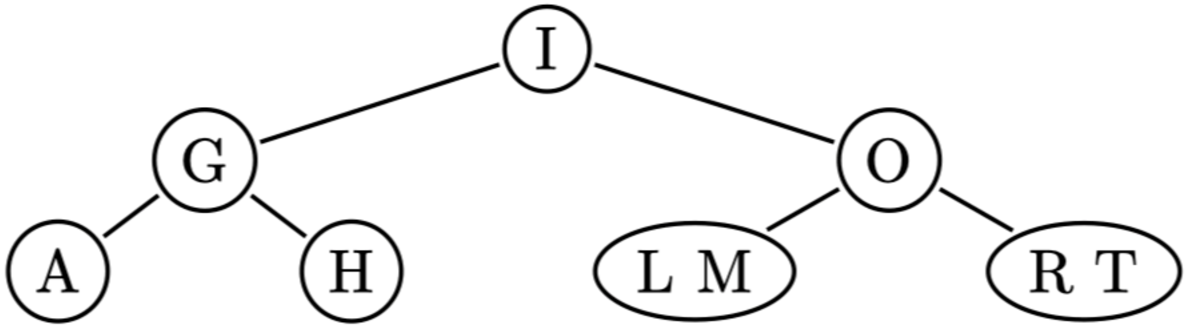
64. Construct a 2–3 tree from the empty tree by inserting the following keys in the given order: ~~A, L, G, O, R, I, T, H, M~~.

1, 12, 7, 15, 18, 9, 20, 8, 13

empty tree?

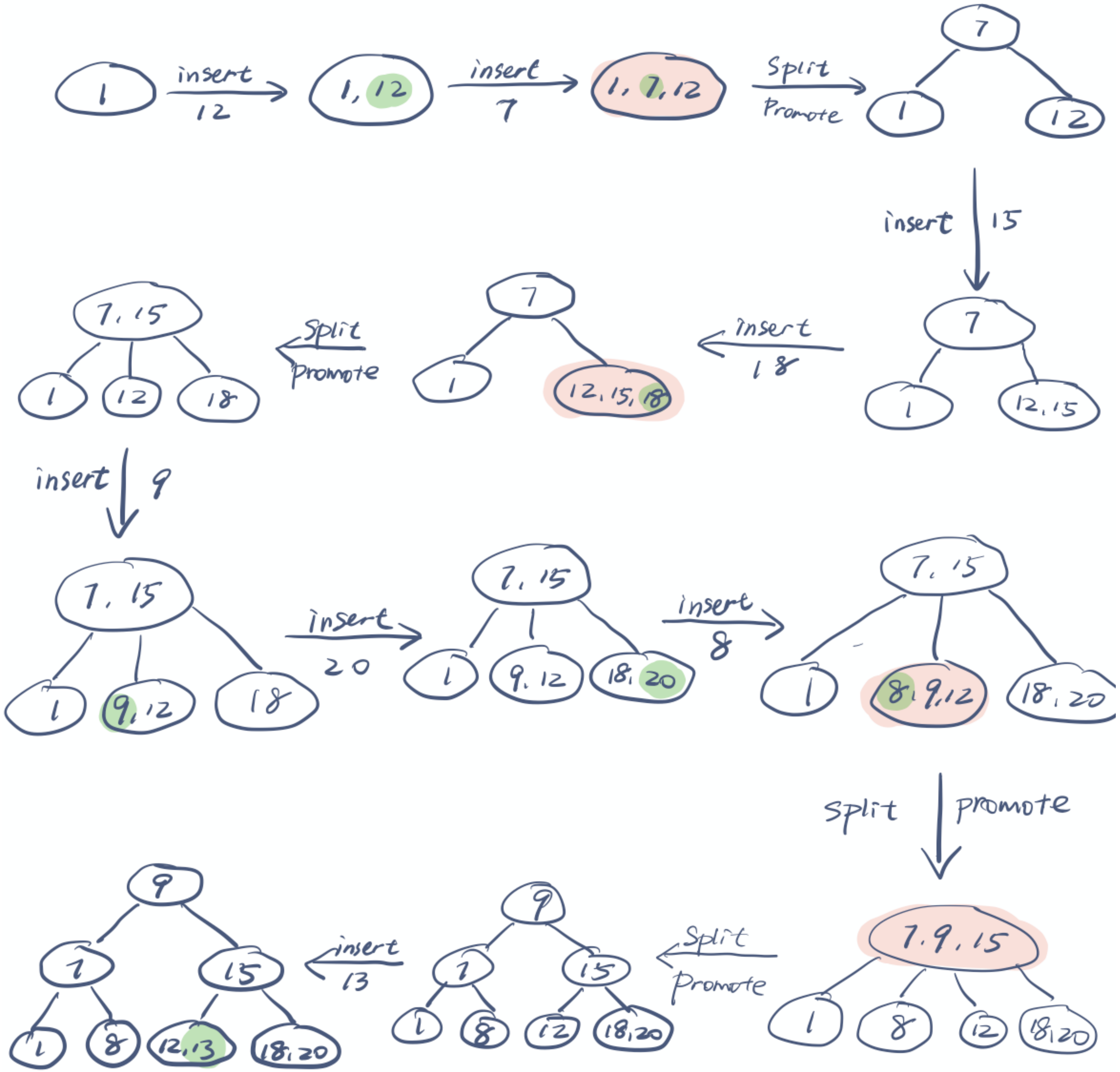
- Yes — create a node and put value into the node
- No
  - *find the **leaf node** where the value belongs*
  - *put the value into the node*
  - *see #items in the node*
    - # = 2 — Continue
    - # = 3
      - split the node and promote the median of the three values to parent.
      - If the parent then has three values, continue to split and promote, forming a new root node if necessary

Answer:



THE UNIVERSITY OF MELBOURNE

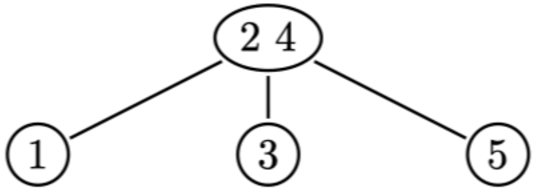
Question 64



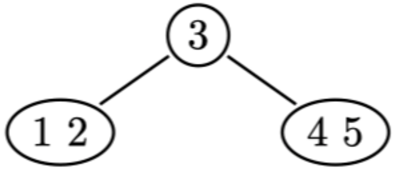
Question 65

65. Construct a 2–3 tree by inserting keys 1, 2, 3, 4, 5, in that order. Repeat the exercise, but this time insert the keys in this order: 2, 3, 4, 5, 1. Are the trees identical? Considering all possible orderings of the set $\{1, 2, 3, 4, 5\}$, how many different 2–3 trees can be produced?

Answer: Inserting 1, 2, 3, 4, 5, in that order, we produce this 2–3 tree:



Inserting 2, 3, 4, 5, 1, in that order, we produce this 2–3 tree:



These are the only possible 2–3 trees with the five given keys.

Question 60

60. Given a list of n distinct integers and a sequence of n boxes with fixed inequality signs between them, design an algorithm to place the numbers into the boxes while satisfying each inequality. For example, given 4, 6, 3, 1, 8 and the boxes $\boxed{} < \boxed{} > \boxed{} < \boxed{} < \boxed{}$, we can place the numbers as follows: $\boxed{1} < \boxed{8} > \boxed{3} < \boxed{4} < \boxed{6}$.

Answer: At first it is not clear that this is always possible. However, here is an algorithm. First sort the sequence. Now fill in the boxes from left to right, as follows. If there is a $<$ after the box, remove the smallest element from the list and place it in the box. If there is a $>$ after the box, remove the largest element from the list and place it in the box. Finally place the only remaining element in the last box. Note that some problem instances will have many solutions, but this algorithm is guaranteed to produce one solution.

To see why the algorithm works, think recursively. We want to be able to produce a solution for n boxes, assuming we already have a solution for the last $n-1$. We can always do that if we have reserved the first box for the appropriate extreme element (the smallest if there is a $<$ after the box, otherwise the largest).

Next Week:

- *Horspool's String Search Algorithm*
- *Hashing*

Thank you !

THE UNIVERSITY OF MELBOURNE

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

Warning

This material has been reproduced and communicated to you by or on behalf of the University of Melbourne pursuant to Part VB of the *Copyright Act 1968 (the Act)*.

The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice