CURTIN UNIVERSITY (CRICOS number: 00301J)

Department of Computing, Faculty of Engineering and Science

Data Structures and Algorithms (COMP1002)

PRACTICAL 9 – ADVANCED TREES

AIMS

- To manually work through the algorithms for self-balancing trees:
  - 2-3-4 Trees
  - Red-Black Trees
  - B-Trees
- To compare the resulting trees created with each of the strategies (above).

BEFORE THE PRACTICAL:

- Read this practical sheet fully before starting.

Note:

- When converting 2-3-4 to Red-Black trees, there are two ways you can arrange a two-key node and still have a valid tree. This means you may not get the same tree as you did creating the Red-black tree directly. In the lecture notes the middle key was set up as the parent of the first key – use this approach.
- The visualisations of the B-trees split the node as it fills - not waiting for the 5th key to overflow it (as seen in the lecture notes). Use the lecture slide approach.

ACTIVITY 1: RED-BLACK TREES

Create three Red-Black Trees using the demo app, use the following lists of numbers:

- (10, 5, 50, 35, 40, 15, 95, 65, 20)
- (5, 10, 20, 30, 40, 50, 60, 70)
- (100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 5)

Manually create the same trees, drawing the tree after each insertion. Check it against the solution from the demo app… do they differ?

ACTIVITY 2: 2-3-4 TREES

Create three 2-3-4 Trees using the demo app, use the following lists of numbers:

- (10, 5, 50, 35, 40, 15, 95, 65, 20)
- (5, 10, 20, 30, 40, 50, 60, 70)
- (100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 5)

Manually create the same trees, drawing the tree after each insertion. Check it against the solution from the demo app… How is the algorithm in the lectures different to the visualisation? (max degree on demo = 4)

Now, manually convert the manually created 2-3-4 Trees into Red-Black Trees and check them against your solutions for Activity 1.

ACTIVITY 3: B-TREES

Create three B-Trees using the demo app, use the following lists of numbers:

- (10, 5, 50, 35, 40, 15, 95, 65, 20)
- (5, 10, 20, 30, 40, 50, 60, 70)
- (100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 5)

Manually create the same trees, using the algorithm from the lectures and drawing the tree after each insertion. Check it against the solution from the demo app. How is the algorithm in the lectures different to the visualisation?

ACTIVITY 4: REFLECTION

Note your answers to the following:

- Compare the heights of the resultant trees – how do they compare with a Binary Search Tree (BST) for the same input values?
- Compare the complexity of the algorithms, how much work would be required for the main operations: insert | find | delete? Compare this to BST.
- Compare the understandability of the algorithms, which would be easier to implement?
- Describe how an in-order traversal would work on each type of tree.

RESOURCES:

Useful Slides:

- Properties, violations, and recovery of Red Black Trees - https://www.youtube.com/watch?v=axa2g5oOzCE
- Rules of Red Black Tree's and each violation case and their respective solution - https://www.youtube.com/watch?v=PhY56LpCtP4

Demo apps and conversion:

(note they may not match the algorithms given in the lecture, you may find better ones!)

- **Red-Black Tree**: https://www.cs.usfca.edu/~galles/visualization/RedBlack.html
- 2-4 Tree: http://cs.armstrong.edu/liang/animation/web/24Tree.html
- 2-3-4 Tree Conversion - use rules from lecture slides to convert manually
- B Tree: https://www.cs.usfca.edu/~galles/visualization/BTree.html

SUBMISSION DELIVERABLE:

Your scanned solutions are due before the beginning of your next tutorial.

SUBMIT ELECTRONICALLY VIA BLACKBOARD, under the *Assessments* section.

MARKING GUIDE

Your submission will be marked as follows:

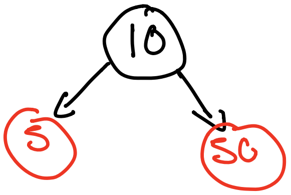
- [2]x3 Your trees for Activities 1-3 are correct.
- [4] One mark per reflection question.

ACTIVITY 1: RED-BLACK TREES

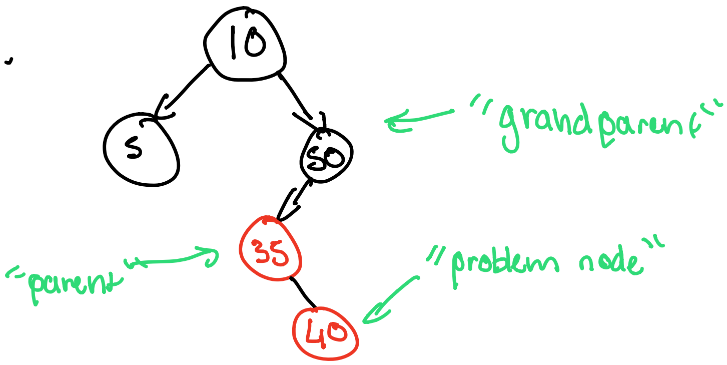
Create three Red-Black Trees using the demo app, use the following lists of numbers:

- (10, 5, 50, 35, 40, 15, 95, 65, 20)
- (5, 10, 20, 30, 40, 50, 60, 70)
- (100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 5)

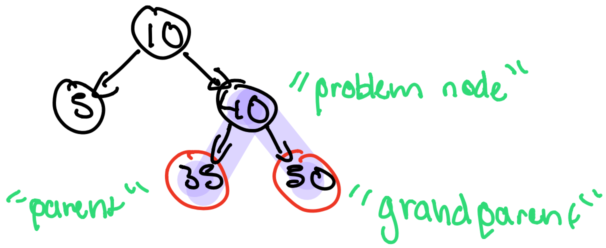
input
[10, 5, 50...



...35, 40...

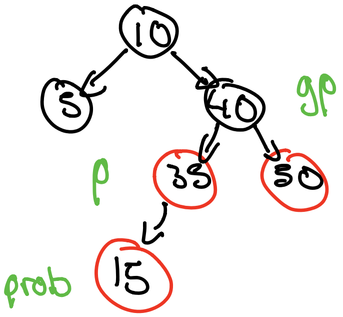


switch

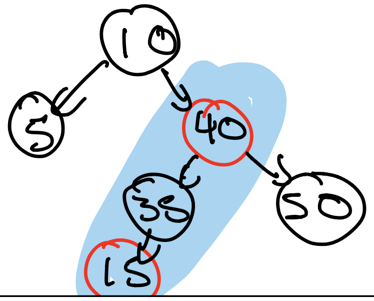


post order traversal = {35, 40, 50}

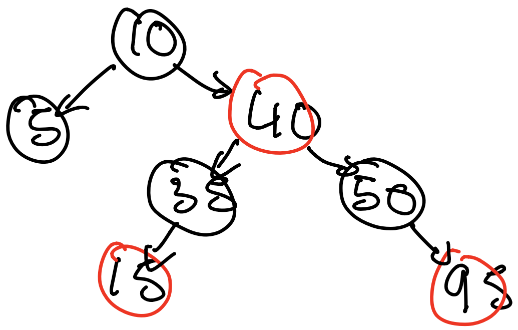
... 15



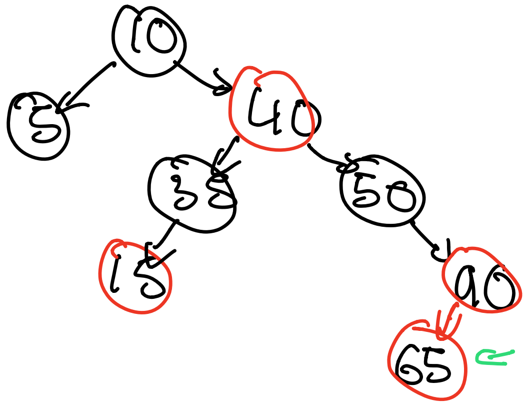
swap colors



...95



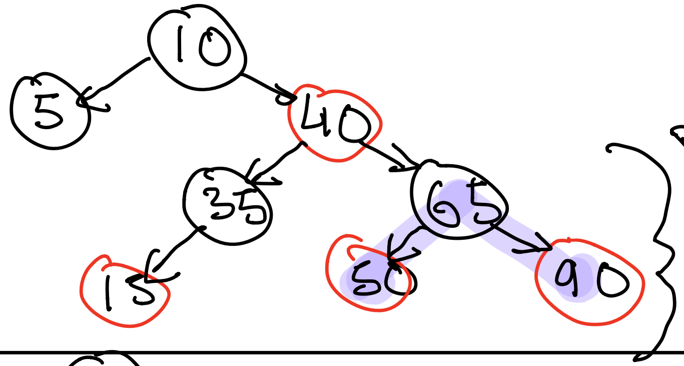
65



* can't change colours > switch!

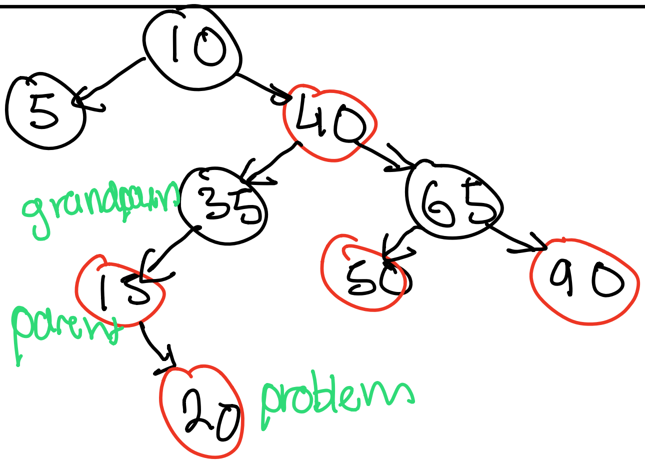
<= "problem node"

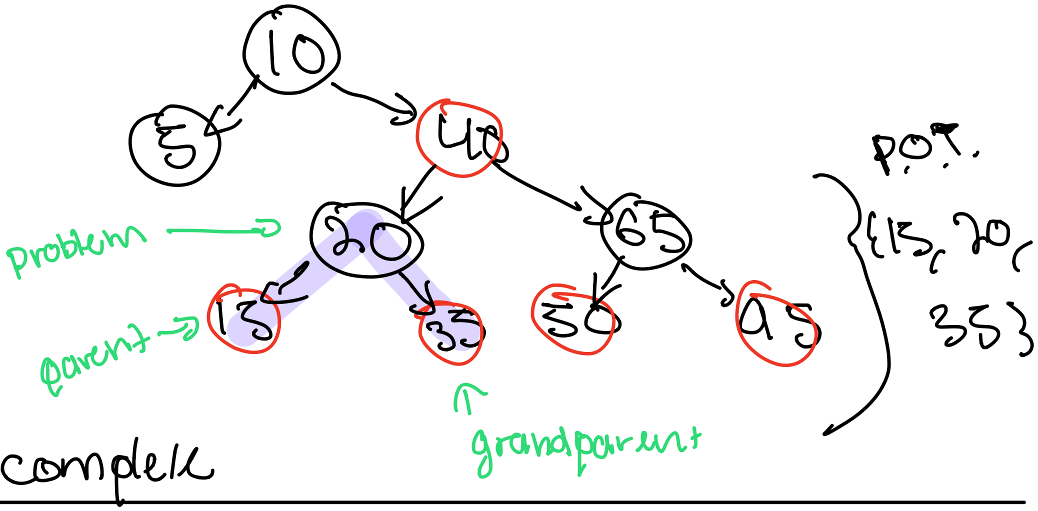
switch



Post Order Traversal
:(50, 65, 90)

...20]



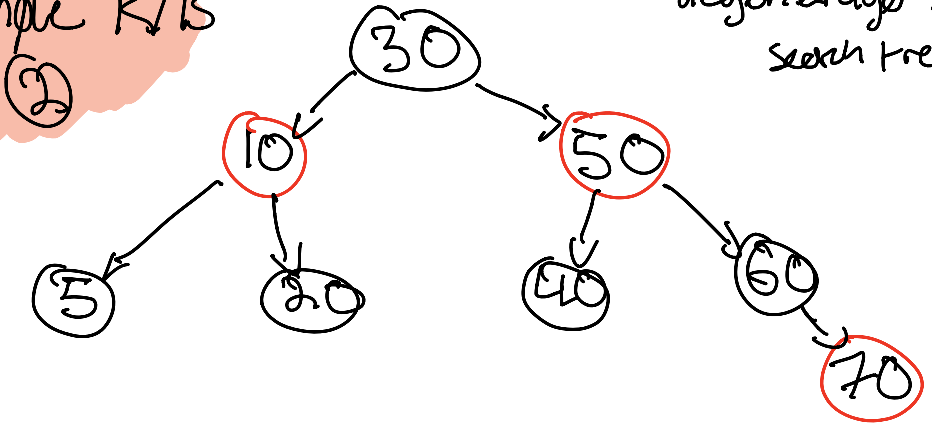


1.1 complete

Input = [5, 10, 20, 30, 40, 50, 60, 70]

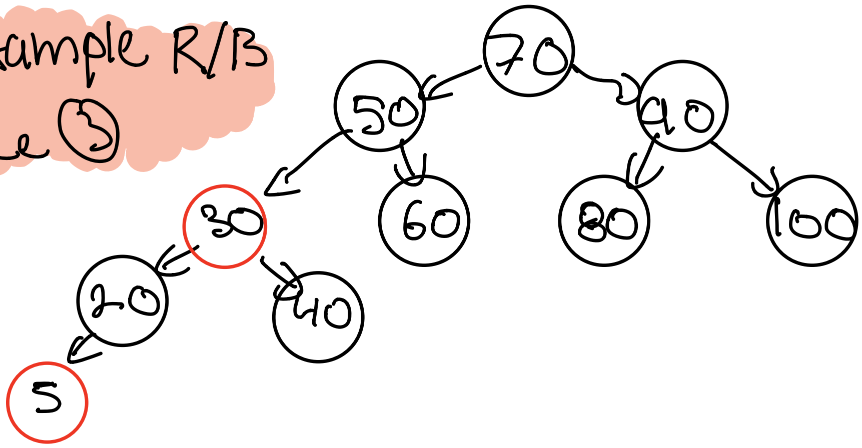
degenerage binary search tree *

example R/B Tree ②



Input = [100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 5]

example R/B Tree ③



ACTIVITY 2: 2-3-4 TREES

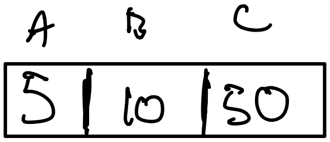
Create three 2-3-4 Trees using the demo app, use the following lists of numbers:

- (10, 5, 50, 35, 40, 15, 95, 65, 20)
- (5, 10, 20, 30, 40, 50, 60, 70)
- (100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 5)

2-3-4 Trees

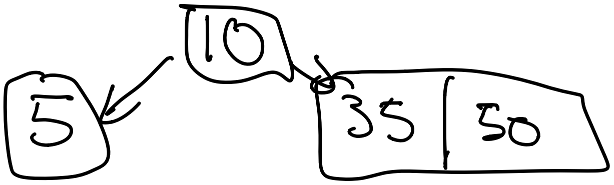
Input = [10, 5, 50, 35, 40, 15, 95, 65, 20]

[10, 5, 50..

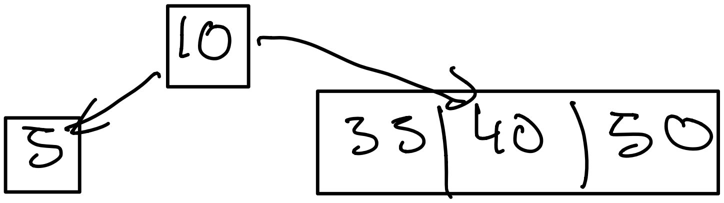


(top-down insert. If node = full → split.)

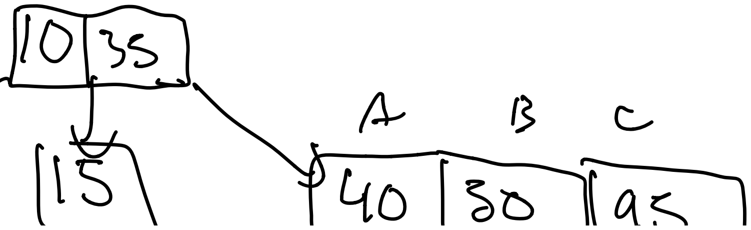
..35



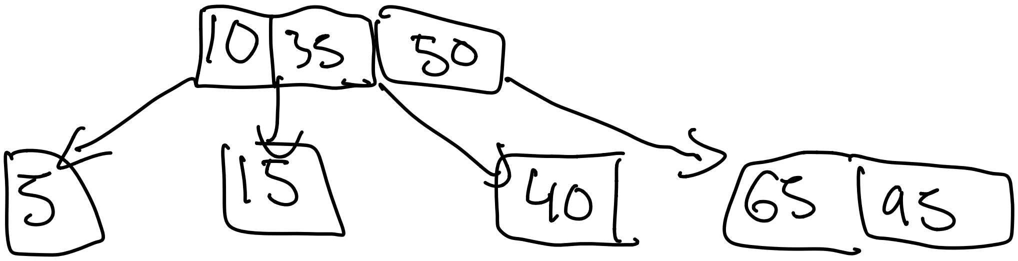
..40



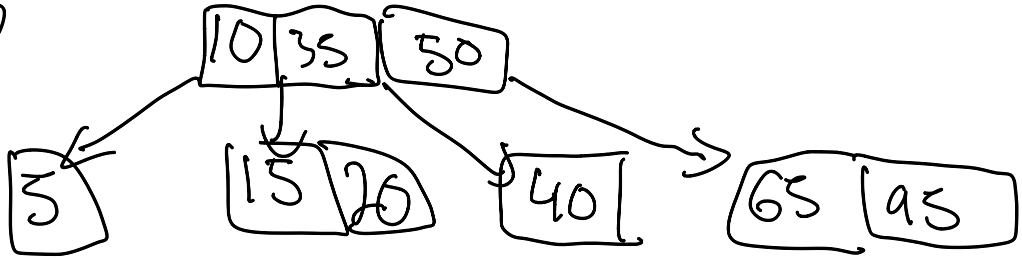
..15, 95...



...65

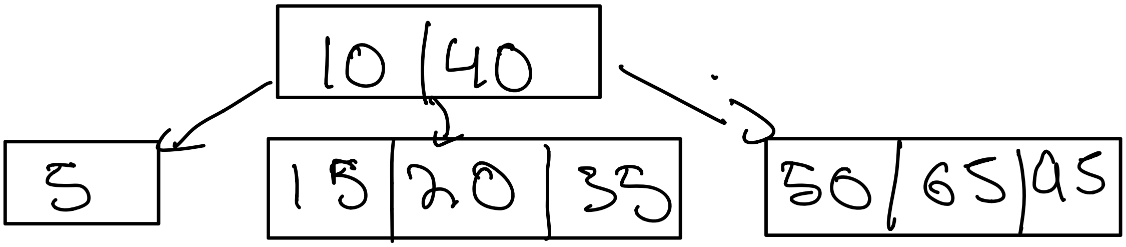


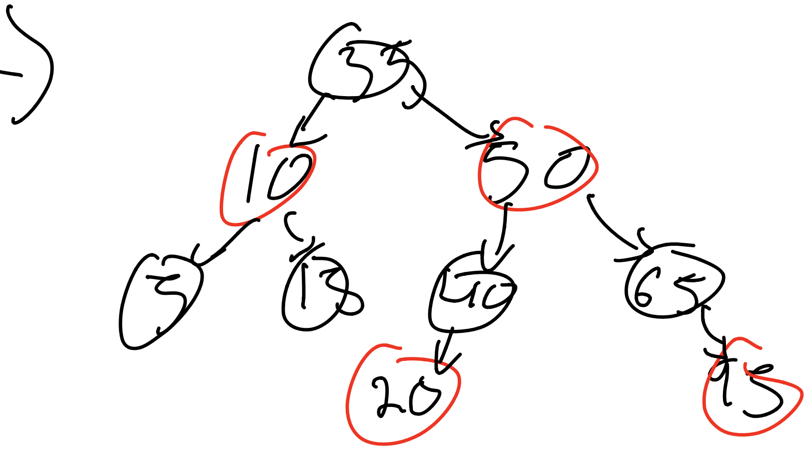
...20



vs

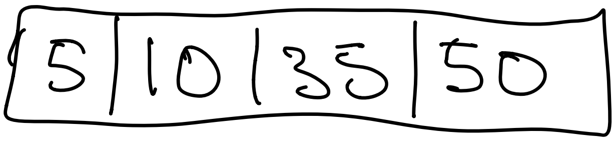
demo result



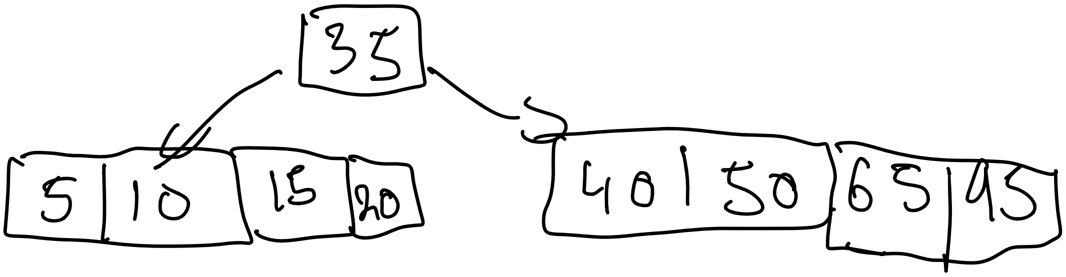


B Trees

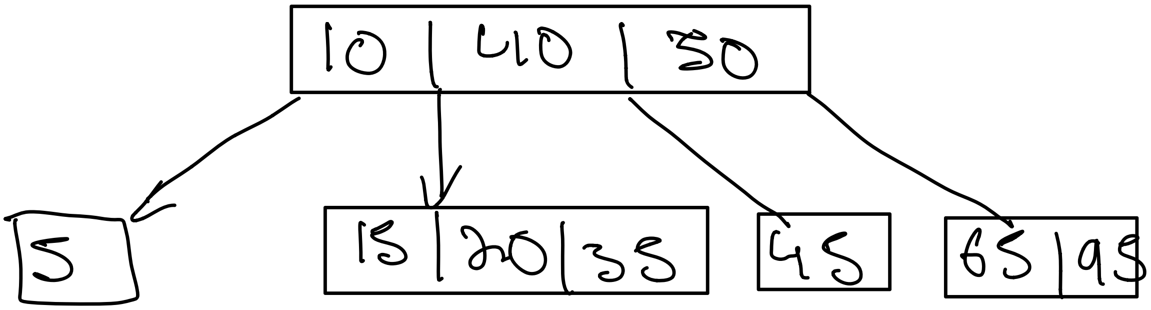
Input = [10, 5, 50, 35, 40, 15, 95, 65, 20]



40, 5



vs demo



10
40
50
5
15
20
35
45
65
95