

1. AVL Trees (20 pts)

- Insert the following values in the given order [34, 74, 98, 2, 64, 41, 86, 59, 40, 58, 50, 56, 45, 16, 44] into an empty AVL tree. Show the tree after all insertions.
- Add an element to the AVL tree such that it will cause a **right rotation** of the tree. State the element and show the tree after. Use the tree from part A.
- Add an element to the AVL tree such that it will cause a **left rotation** of the tree. State the element and show the tree after. Use the tree from part B.
- Delete an element from the AVL tree such that it will cause a **double rotation** to the tree. State the element and show the **final tree** after deletion. Use the tree from part C.

a) Empty AVL tree

Insert 34

34

Insert 74

34
↙
74

Insert 98

74
↙ ↘
34 98

Insert 2

74
↙ ↘
34 98
↙
2

Insert 64

74
↙ ↘
34 98
↙ ↘
2 64

Insert 41

64
↙ ↘
34 74
↙ ↘
2 41
↘
98

Insert 86

64
↙ ↘
34 86
↙ ↘
2 4 74 98

Insert 59

64
↙ ↘
34 86
↙ ↘
2 41 74 98
↘
59

Insert 40

64
↙ ↘
34 86
↙ ↘
2 41 74 98
↘
40 59

Insert 58

64
↙ ↘
41 86
↙ ↘
34 59 74 98
↙ ↘
2 40 58

Insert 50

64
↙ ↘
41 86
↙ ↘
34 58 74 98
↙ ↘
2 40 50 59

Insert 56

58
↙ ↘
41 64
↙ ↘
34 59 86
↙ ↘
2 40 56 74 98

Insert 45

58
↙ ↘
41 64
↙ ↘
34 50 59 86
↙ ↘
2 40 45 56 74 98

Insert 16

58
↙ ↘
41 64
↙ ↘
34 50 59 86
↙ ↘
2 40 45 56 74 98
↙
16

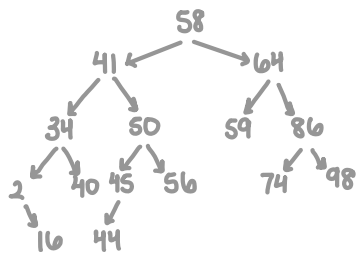
Insert 44

58
↙ ↘
41 64
↙ ↘
34 50 59 86
↙ ↘
2 40 45 56 74 98
↙ ↘
16 44

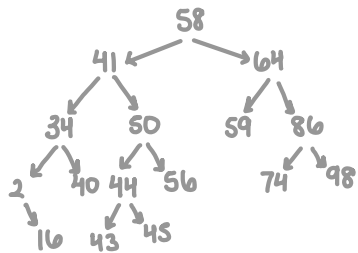
Final Tree

58
↙ ↘
41 64
↙ ↘
34 50 59 86
↙ ↘
2 40 45 56 74 98
↙ ↘
16 44

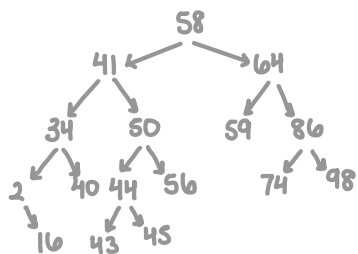
b) Right Rotation of tree



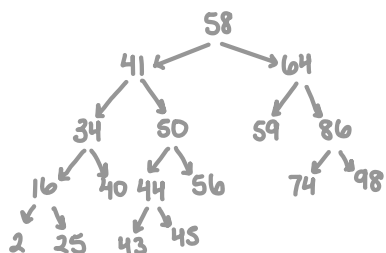
Insert 43



c) Left Rotation of tree



Insert 25



2. Heap (10 pts)

- Create a binary max on top heap with the keys [11, 16, 33, 36, 26, 41, 17, 84, 8, 57, 30, 71, 53, 23, 87, 95, 62]. Show the heap after all insertions.
- Show the heap when removing the max value 3 times. Show the steps.

a) Empty heap

Insert 11

11

Insert 16

16
11

Insert 33

33
11 16

Insert 36

36
33 16
11

Insert 26

26
36 16
33 11

Insert 41

41
33 36
11 16

Insert 17

17
41
33 36
11 16

Insert 84

84
41 36
33 16
11 17

Insert 8

84
41 36
33 16
11 17
8

Insert 57

84
57 36
33 16
11 17
8 26

Insert 30

84
57 36
33 16
11 17
8 26 30

Insert 71

84
57 71
33 36
11 16
8 26 30

Insert 53

84
57 71
33 53
11 16
8 26 30

Insert 23

84
57 71
33 53
11 16
8 26 30 17 23

Insert 87

87
57 84
33 53
11 16
8 26 30 17 23

Insert 95

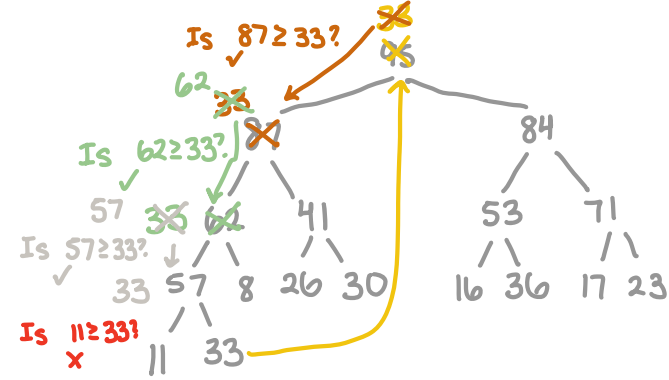
95
87 84
57 53
11 16
33 8 26 30 17 23

Insert 62

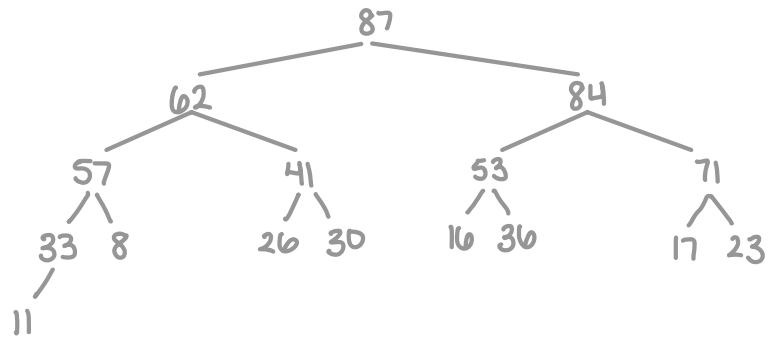
95
87 84
62 53
11 16
57 8 26 30 17 23

b) Remove max 3 times

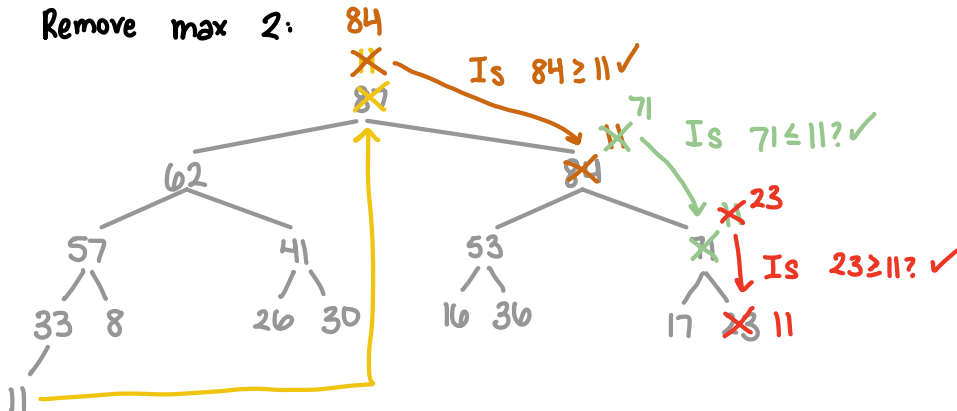
Remove max 1: 87



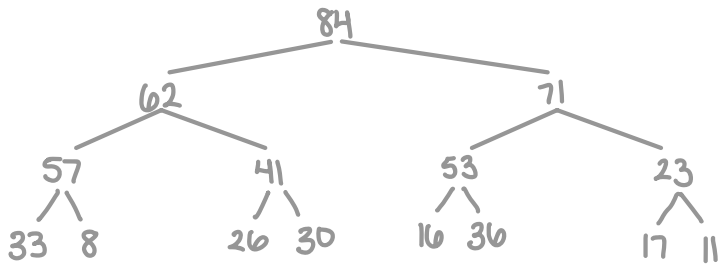
Tree after extracting:



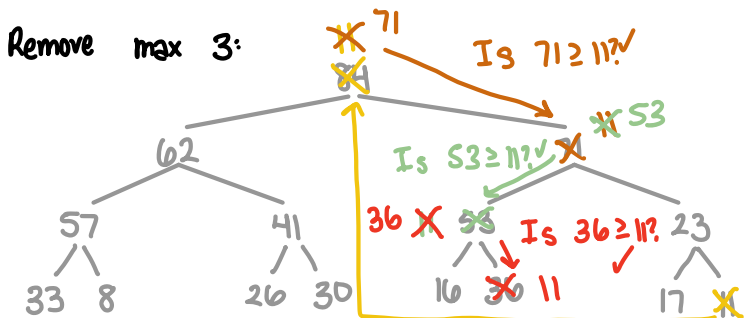
Remove max 2: 84



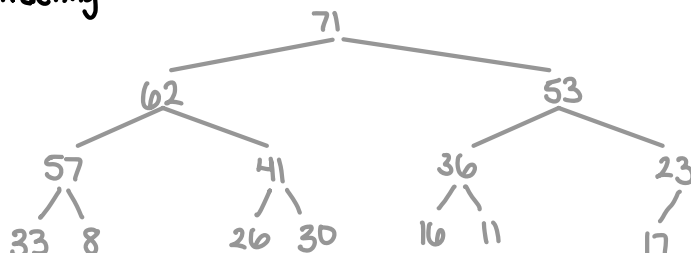
Tree after extracting:



Remove max 3: 71

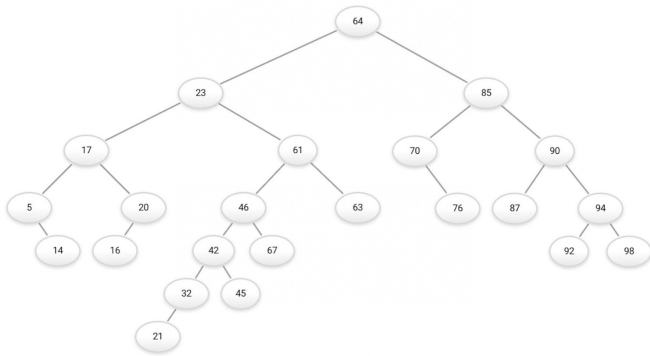


Tree after extracting:



3. Binary Trees (20 pts)

Given the following tree:



- Give post-order, pre-order of the tree.
- Give level-order and in-order of the tree
- When looking for the value 67, give the path traced by breadth first search and depth first search algorithm
- Given a tree with the preorder: [E, A, B, F, H, I, D, J, C, G, K] and the inorder: [B, A, F, H, I, E, D, J, C, G, K], draw the binary tree.

a) Post order:

14, 5, 16, 20, 17, 21, 32, 45, 42, 67, 46, 63, 61, 23, 76, 70, 87, 92, 98, 94, 90, 85, 64

Preorder:

64, 23, 17, 5, 14, 20, 16, 61, 46, 42, 32, 21, 45, 67, 63, 85, 70, 76, 90, 87, 94, 92, 98

b) Level-order:

64, 23, 85, 17, 61, 70, 90, 5, 20, 46, 63, 76, 87, 94, 14, 16, 42, 67, 92, 98, 32, 45, 21

In-order:

14, 5, 17, 16, 20, 23, 21, 32, 42, 45, 46, 67, 61, 63, 64, 76, 70, 85, 87, 90, 92, 94, 98

c) BFS:

64 → 23 → 85 → 17 → 61 → 70 → 90 → 5 → 20 → 46 → 63 → 76 → 87 → 94 → 14 → 16 → 42 → 67

DFS:

64 → 23 → 17 → 5 → 14 → 20 → 16 → 61 → 46 → 42 → 32 → 21 → 45 → 67

d)

