CS202 – HW3 Report

Elif Gülşah Kaşdoğan

Section 2

A)

Insert 9; 12; 10; 5; 1; 8; 20; 15; 13; 25 to AVLTree

Insert 9

Insert 12

Insert 10 → Insertion to left of right (inside), tree should be rebalanced: Double right left rotation on 9

Insert 5

Insert 1→Insertion to left of left (outside), tree should be rebalanced: Single right rotation on 9

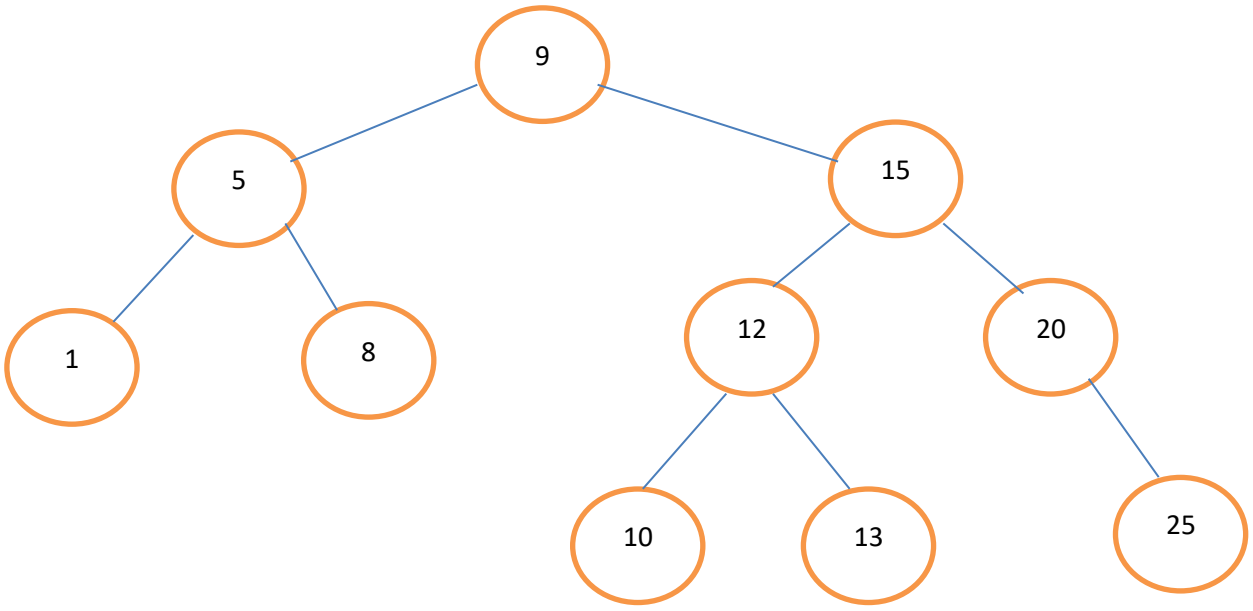
Insert 8→Inside insertion. Balance factor of 10(root) is 2, its left child 5's balance factor is -1. Left right rotation on 10.

Insert 20→ right of right(inside). 10 is right-high, single left rotation on 10.

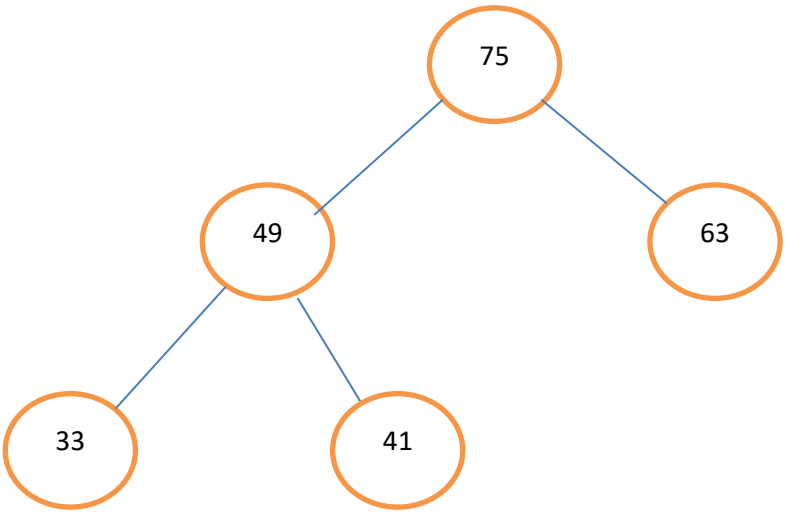
Insert 15

Insert 13→single right rotation on 20.

Insert 25→single left rotation on 12.



B)



Heap Sort:

Start: 75, 49, 63, 33, 41

Swap: 41, 49, 63, 33, 75

Rebuild: 63, 41, 49, 33, 75

Swap: 33, 41, 49, 63, 75

Rebuild: 49, 33, 41, 63, 75

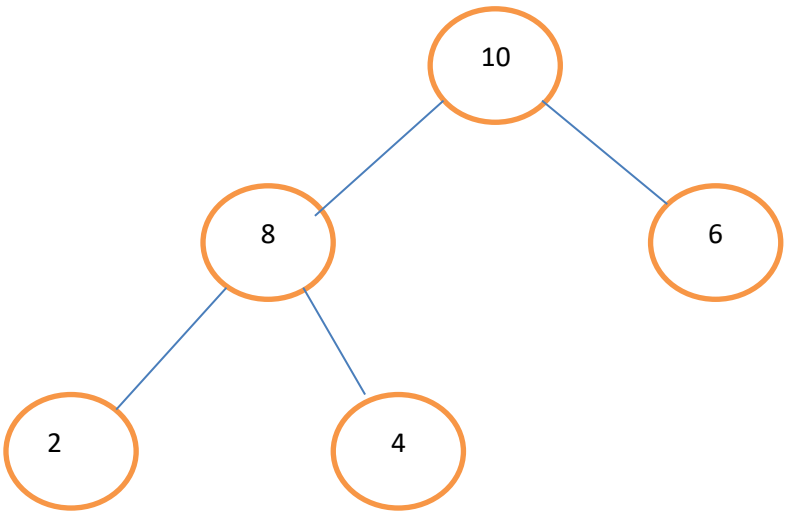
Swap: 41, 33, 49, 63, 75

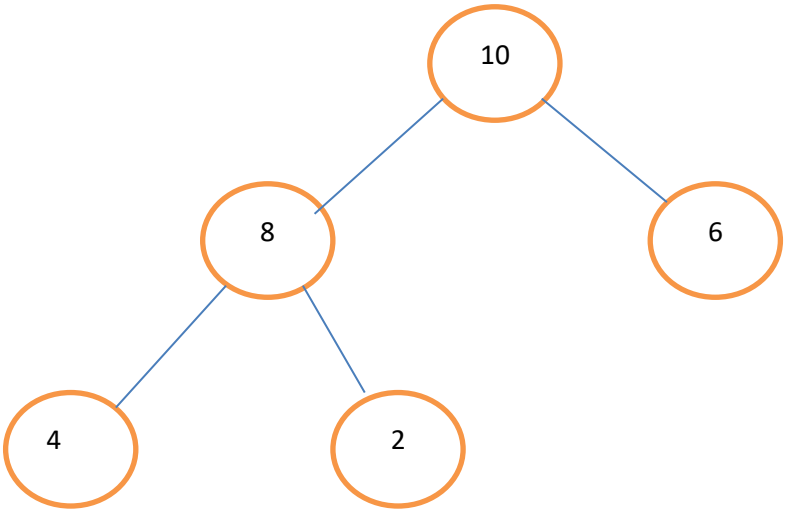
Rebuild: 41,33, 49, 63, 75

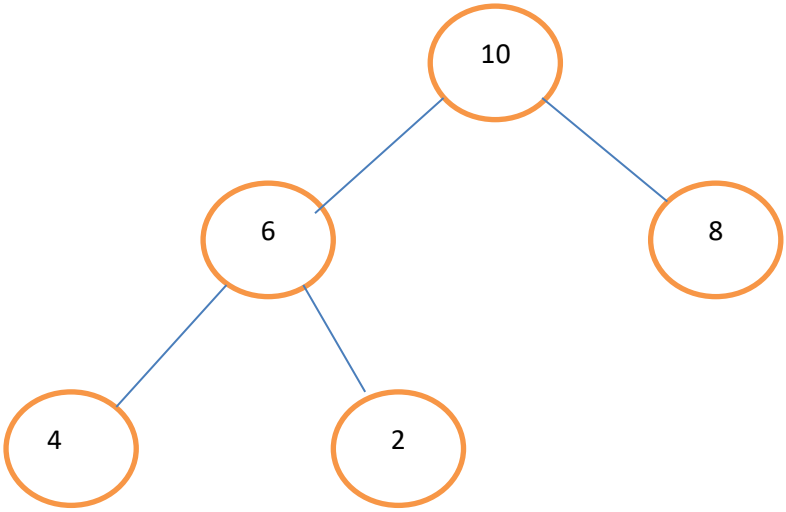
Swap: 33, 41, 49, 63, 75

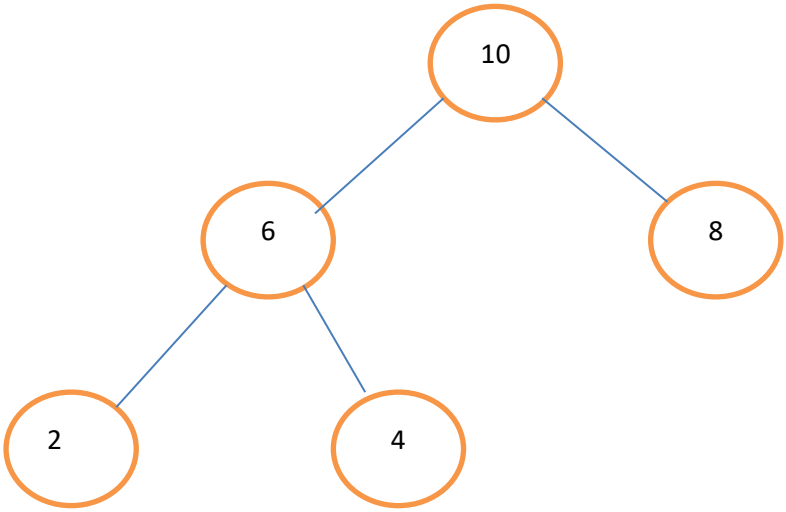
There is one element left in the heap →terminate

C)




10
8
6
4
2


10
6
8
4
2


10
6
8
2
4

Analysis Report:

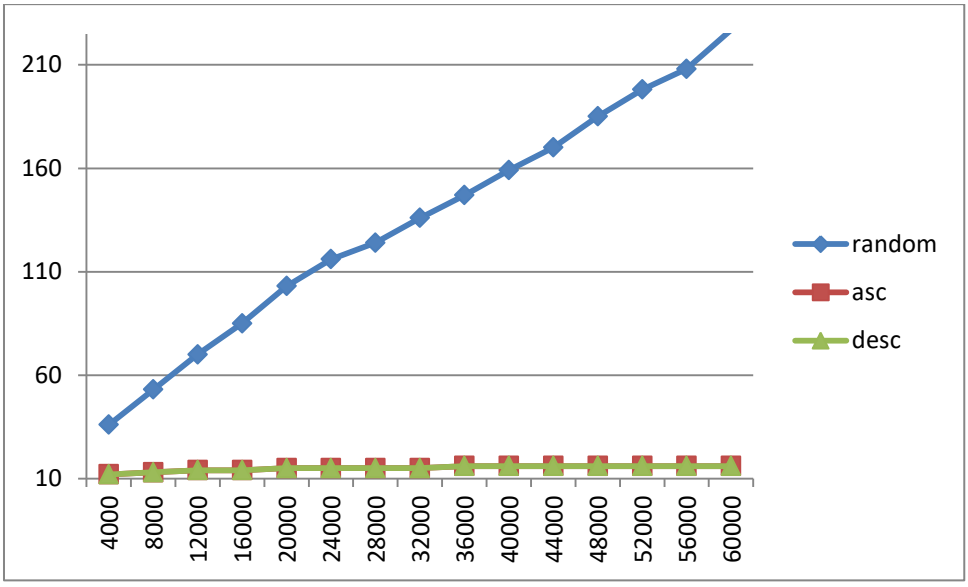
Ascending order insertion: Tree will grow to the right; since it must be kept balanced, left rotation will be performed at least once for every 3 insertions. As tree grows, more than one left rotation will be performed for different problematic nodes. Height of tree for ascending order insertion will be minimum height which an AVL tree can achieve. Height is order $log_2 n$ where n is number of items.

Descending order insertion: Tree will grow to the left as items inserted. To keep the tree balanced right rotation will be performed for different problematic nodes. Height of tree for descending order insertion will be in order $log_2 n$ where n is number of items in tree. This will have the minimum height like ascending order insertion.

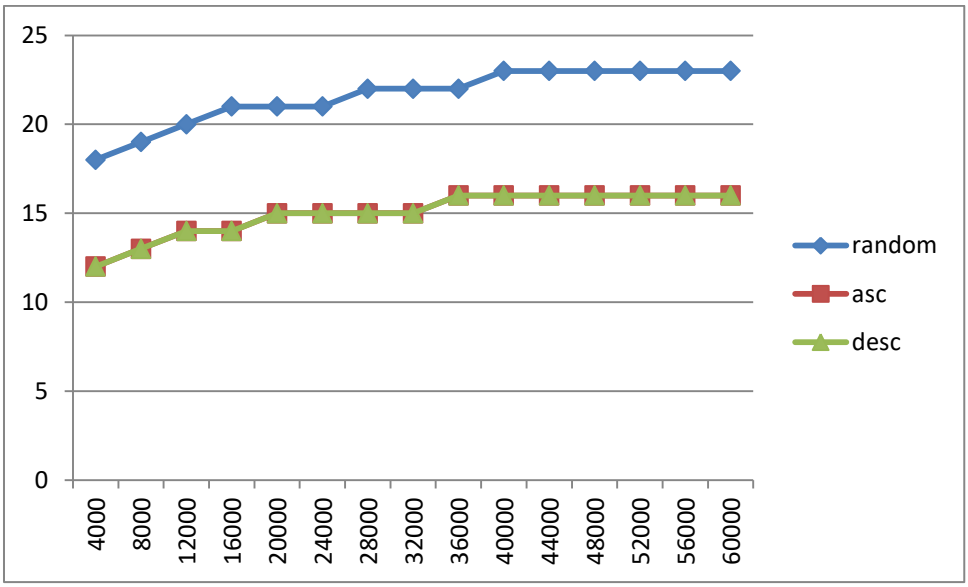
I would expect ascending and descending order insertion to have height in logarithmic order, random integers would leave gaps in between siblings and as trees grow rapidly height of tree would increase faster than ascending/descending insertion. My empirical results is as I expected in terms of order of heights however I did not expect that much difference between random and ordered insertions.

AVL Tree will work O(logn) in worst case for searching, deletion and insertion. In worst case scenario of AVL Tree insertion and deletion all leaves are balanced and other nodes are left or right high. Theoretically maximum height should be 1.44*logn, my random values were much higher. I tested my code for different trees but I could not find my mistake. Even though my results for random is not accurate, for an AVL tree with 4000 elements we should have 18 as max height, ascending and descending order insertion should yield logn height which is approximately 12. I prepared two tables one for empirical and one for theoretical results.

Empirical Result:



Expected result:



C:\Users\gulsa\Desktop\HW3\AVL\bin\Debug\AVL.exe

```
Part b - Height analysis of AVL trees
---------------------------------------------------------
Array Size      Random       Ascending       Descending
---------------------------------------------------------
4000            36             12               12
8000            53             13               13
12000            70             14               14
16000            85             14               14
20000            103             15               15
24000            116             15               15
28000            124             15               15
32000            136             15               15
36000            147             16               16
40000            159             16               16
44000            170             16               16
48000            185             16               16
52000            198             16               16
56000            215             16               16
60000            227             16               16
64000            235             16               16
68000
```