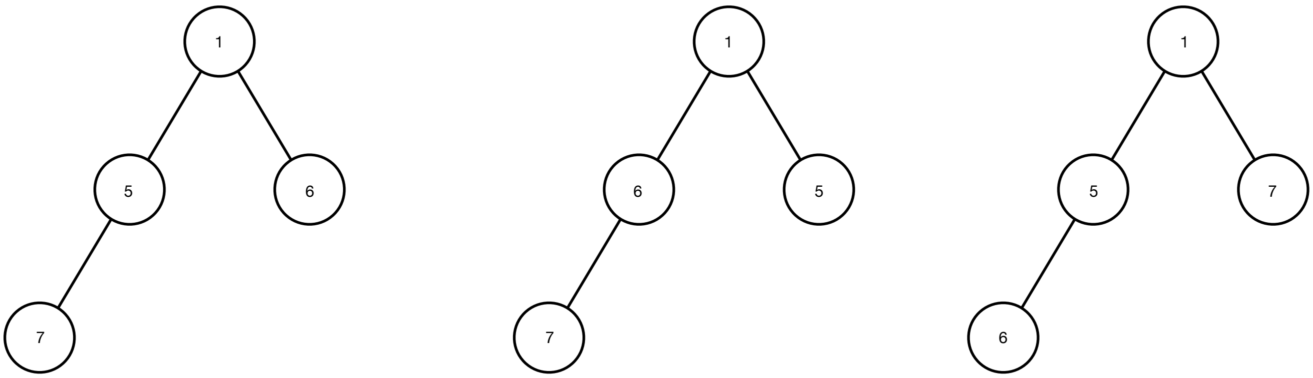
Title : Heaps and AVL Trees
Author : Bartu Atabek
ID : 21602229
Section : 2
Assignment : 3
Description : Contains answers to questions 1 & 2.

Question 1:

a) All valid min-heaps containing these 4 elements 5, 7, 6, 1.



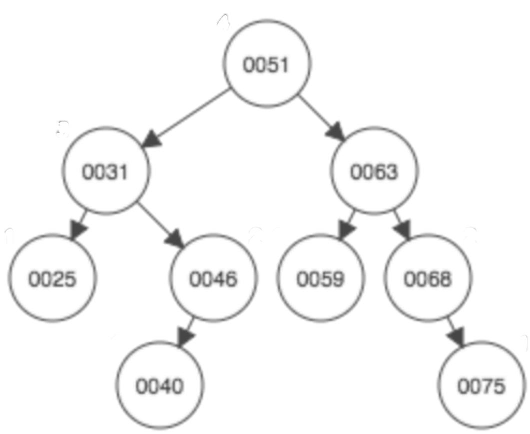
b) How many valid max-heaps containing 5 distinct elements can be built?

By using this Recurrence equation:
T(N) =(N-1)Ck* T(k) * T(N-k-1) where k = number of nodes on left subtree
T(1) = 1
T(2) = 1
T(3) = 2
T(4) = 3C2 * T(2) * T(1) = 3
T(5) = 4C3 * T(3) * T(1) = 8

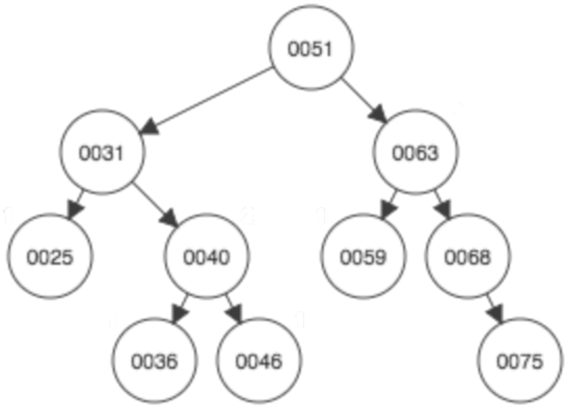
Total of 8 max-heaps containing 5 distinct elements can be built.

c)

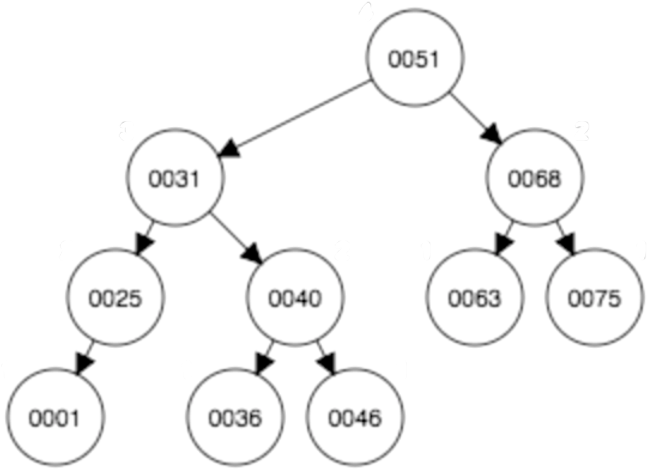
i) Insert 46, 59, 51, 31, 68, 63, 25, 75, 40 to an empty AVL tree.



ii) Insert 36 to cause a single right rotation.

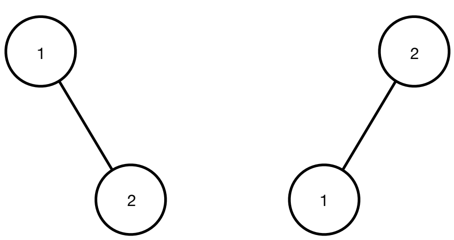


iii) Insert 1; to the AVL tree. Now, delete 59 such that it causes a single left rotation in the tree



d) “Insertion order of the same set of elements into an AVL tree affects the resulting tree structure”.

The above statement is true because an AVL tree is a sorted tree (BST) so that the insertion order of the elements could cause the nodes to align differently therefore changing the resulting tree structure.



As it can be seen from the example above inserting {1, 2} in different orders will affect the resulting tree structure.

Question 2:

Array Size	Random	Ascending	Descending
1000	730	990	990
2000	1366	1989	1989
3000	2113	2988	2988
4000	2635	3988	3988
5000	3408	4987	4987
6000	4044	5987	5987
7000	4782	6987	6987
8000	5350	7987	7987
9000	6036	8986	8986
10000	6727	9986	9986

Table 1: Experimental results obtained indicating the number of rotations for the number of elements inserted

Theoretically in an AVL tree operations such as insertion deletion and search takes O(log n) time in the worst case. In the context of this question when inserting elements into the AVL tree in three different orders (random, ascending, descending) the experimental results obtained agree with the theoretical results. For inserting ascending or descending ordered elements into the AVL tree which are the worst cases since it will create an unbalanced tree and require rotations to balance it and inserting an element to the AVL tree could cause at most two rotations in the worst case and zero rotations in the base case. Therefore, an insertion of an element will cost between [0,2] rotations and in average case we can approximate it into 1 rotation. With this logic since the insertion of one element into the AVL tree will cost one rotation n insertions will cost n number of rotations so it’s O(n). When we look at table 1 it can be seen that upon inserting 1000 elements into the AVL tree the number of rotations for ascending and descending numbers is 990 which is approximately O(n) and the difference is caused because inserting an empty tree will not cost any rotations and when a rotation is made the tree is temporarily balanced so that the next insertion may not require any rotations at all. The number of rotations for random numbers is way less than the theoretical values because assuming we are inserting uniformly random items the tree will not be clustered at one side and balanced equally therefore requiring less number of rotations.

Different patterns of insertion affect the number of rotations in the AVL tree. As discussed in part d of the first question different patterns of insertion affect the number of rotations in the AVL trees. If we insert sorted number such as ascending or descending order the tree will become clustered on one side continuously and thus will require more rotations in order to keep the height balanced between both sides. On the other hand inserting uniformly random numbers could keep the balance property intact and less number of rotations will be required. The experimental values achieved in Table 1 supports this idea so that inserting with different patterns such as random, ascending and descending will in fact affect the number of rotations and inserting elements in a sorted manner will cause more rotations as explained above.