Assignment 4

Given Date: 06-23-2022

Due Date: 06-30-2022 11:59 pm EDT

Question 1 (Heaps)

1. (25 points) Assume a priority queue implemented a heap. Show the heap after each of the operations performed one after the other:
   insert(5), insert(7), insert(3), insert(13), insert (2), insert(11), removeMin(), insert(1), removeMin(), removeMin()

Answer: Below, I have shown the heap after every operation.

Figure 1: insert(5)

Figure 2: insert(7)

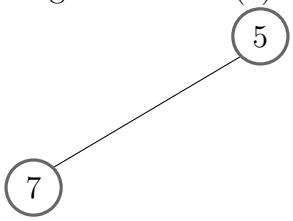


Figure 3: insert(3)

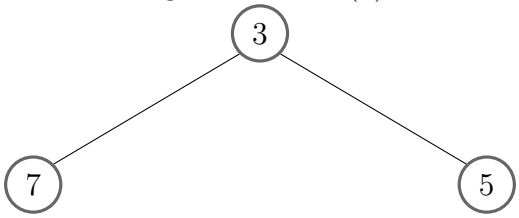


Figure 4: insert(13)

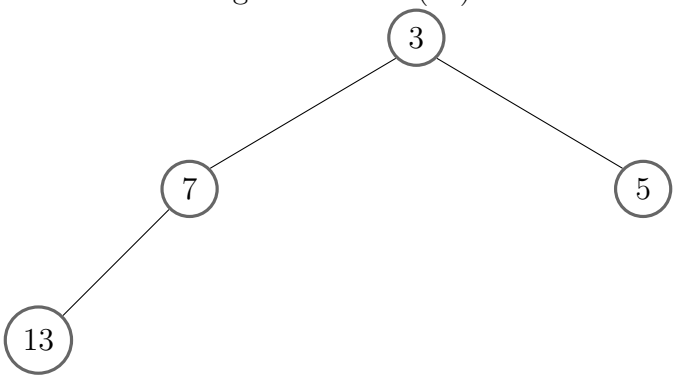


Figure 5: insert(2)

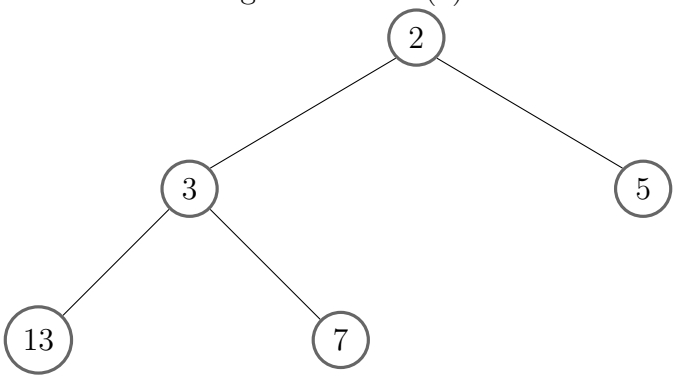


Figure 6: insert(11)

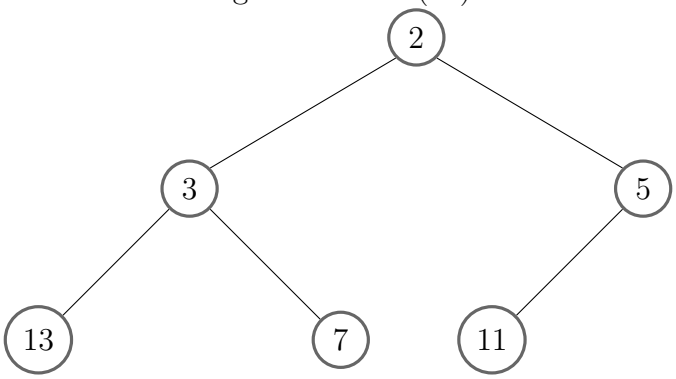


Figure 7: removeMin()

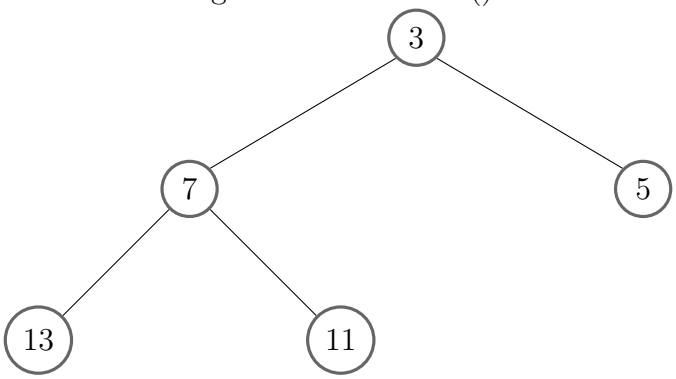


Figure 8: insert(1)

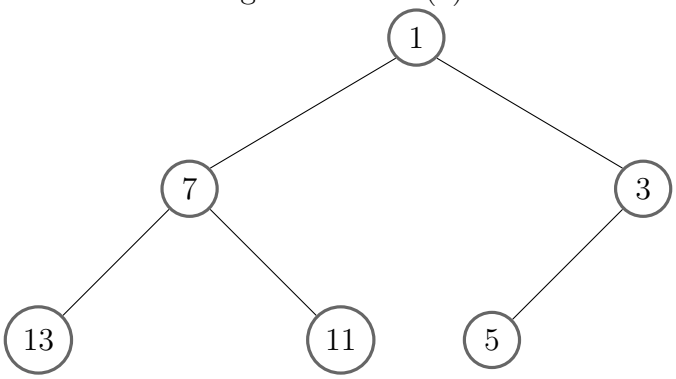


Figure 9: removeMin()

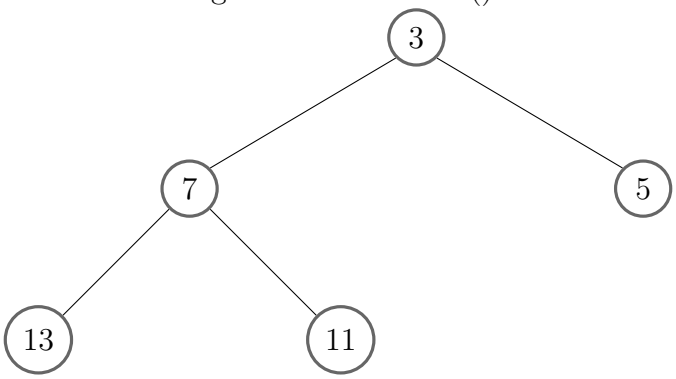
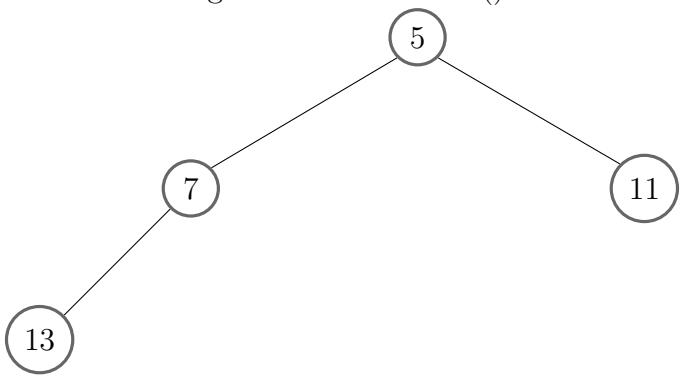


Figure 10: removeMin()



Question 2 (Search Trees)

1. (5 points) Show the (unbalanced) binary search tree after insertion of the following elements (in the order specified here):
   8 4 7 2 1 9 5 3 0

   Answer: In BST, when we are inserting a new element, we want to compare its value sorted in the current node (i.e., starting from root) and we decide either one of the following

   (1) Recurse down the left subtree if the value is less the current node value (2) Recurse down the right subtree if the value is greater the current node value (3) create a new node if a null node is found.

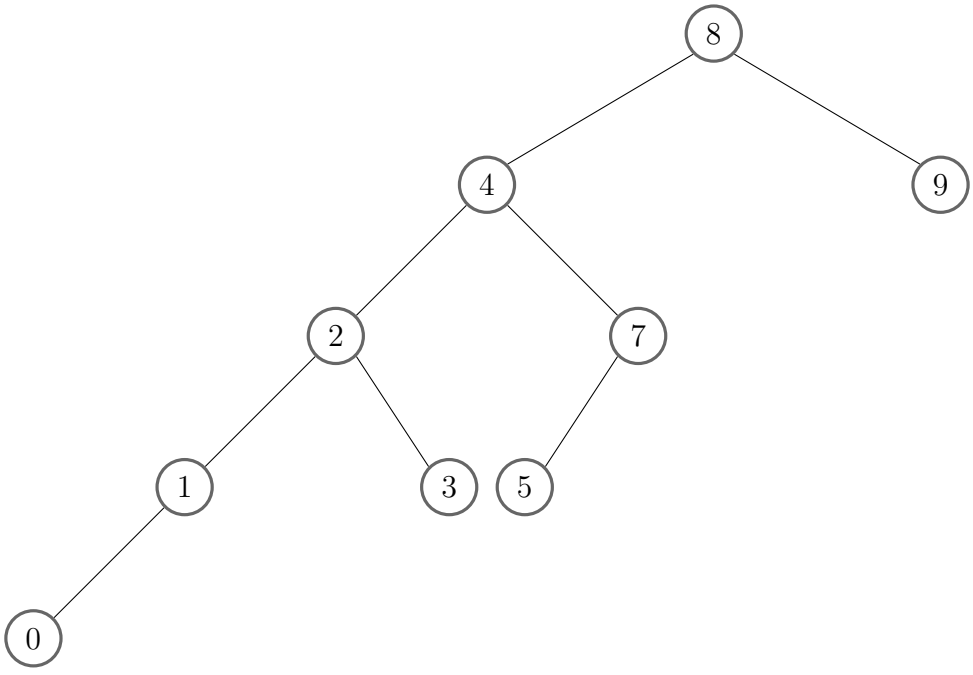


Figure 11: Unbalanced BST after insertion

2. (5 points) Show the search tree from Question 1 above after the following operations: Notice: (a) and (b) are independent, they should both be answered based on your BST in 1.

   (a) delete(8)

   (b) delete(4)

   Answer: First find the in-order successor of the node you are deleting. Then swap those two nodes. Then call delete on the swapped node.

   (a) delete(8): In-order successor of node 8 is node with value 9 which has no child. First swap node 8 and node 9 and then call delete on the new location of node 8. After that the graph will look like as follows

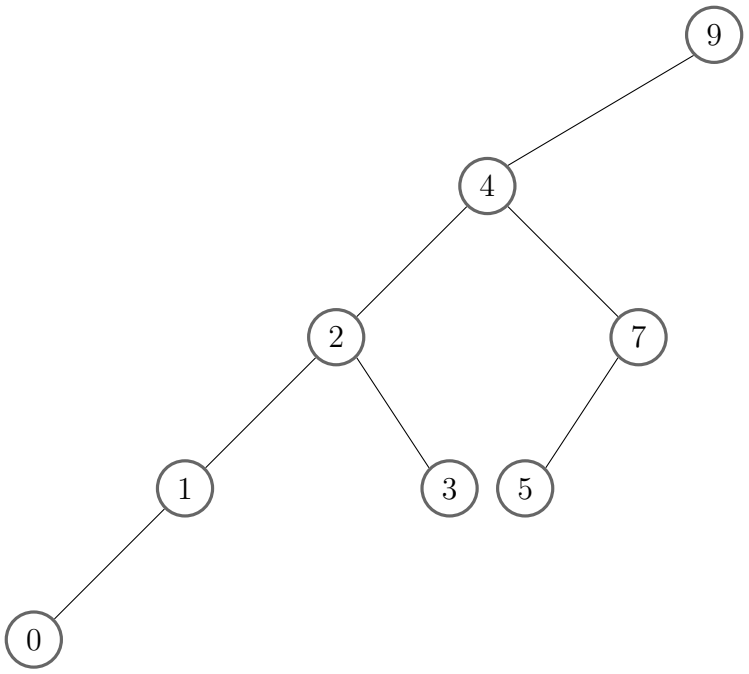


Figure 12: BST after delete(8)

(b) delete(4): In-order successor of node 4 is node with value 5 which has no child. First swap node 4 and node 5 and then call delete on the new location of node 4. After that the graph will look like as follows

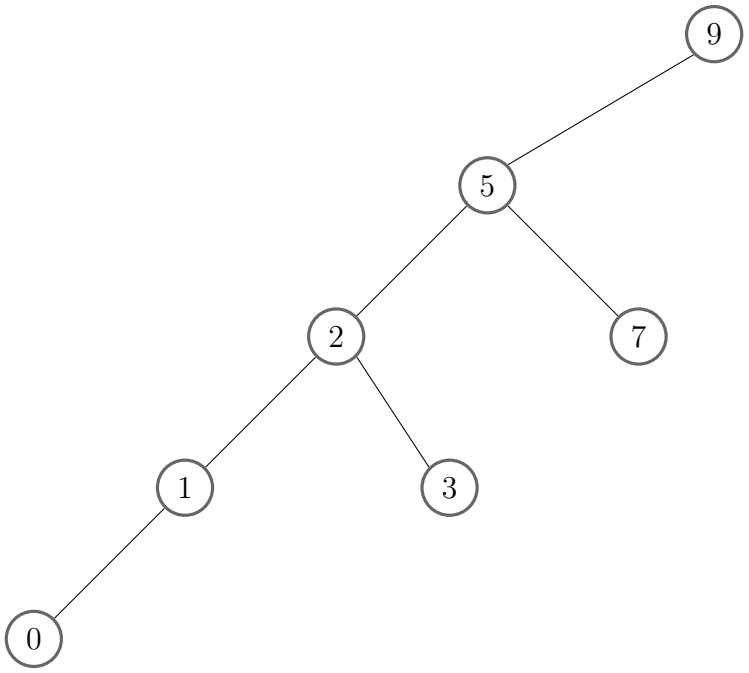


Figure 13: BST after delete(4)

3. (5 points) Show the (unbalanced) binary search tree after insertion of the following elements (in the order specified here):
   8 7 6 5 4 3 2 1 0

   Answer: In BST, when we are inserting a new element, we want to compare its value sorted in the current node (i.e., starting from root) and we decide either one of the following

   (1) Recurse down the left subtree if the value is less the current node value (2) Recurse down the right subtree if the value is greater the current node value (3) create a new node if a null node is found.

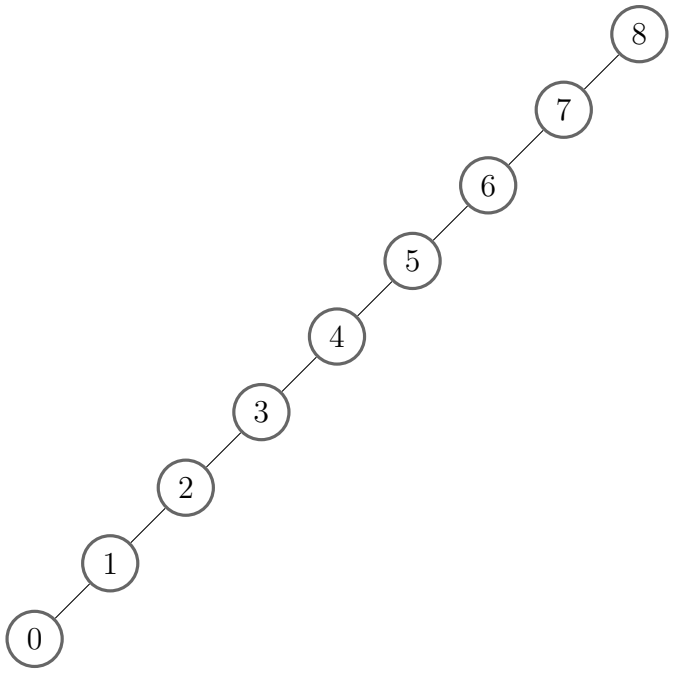


Figure 14: Unbalanced BST after insertion

4. (10 pts) Consider the AVL tree in Figure 15. Show the AVL tree after each of the following operations performed one after the other: insert(42), insert(90), delete(62), insert(92), delete(50)
   Notice: Replace the deleted item with its successor for this question

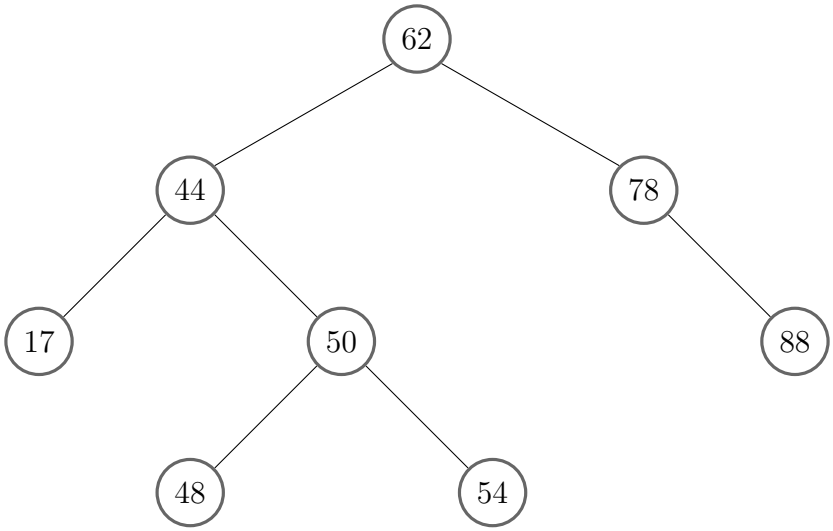


Figure 15: AVL tree of Question 2

Answer: For AVL tree, rebalancing needs to be performed only for an internal node, the height of its children differ more than 1. For no case, in this question, we are not hampering the balance of the AVL tree thus we do not need to perform the rebalancing.

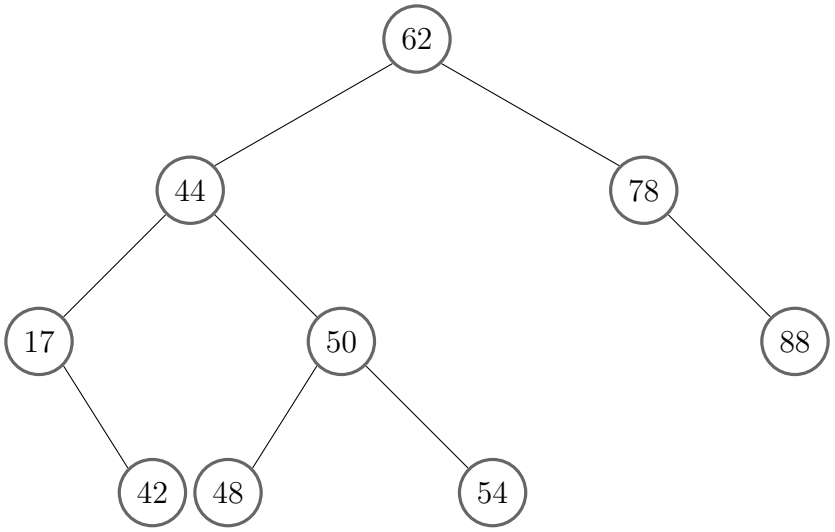


Figure 16: insert(42)

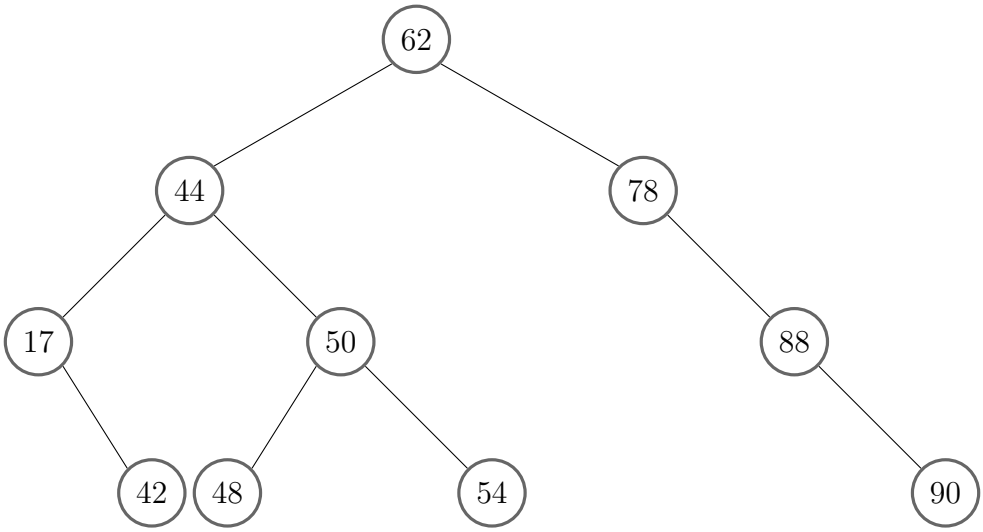


Figure 17: insert(90)

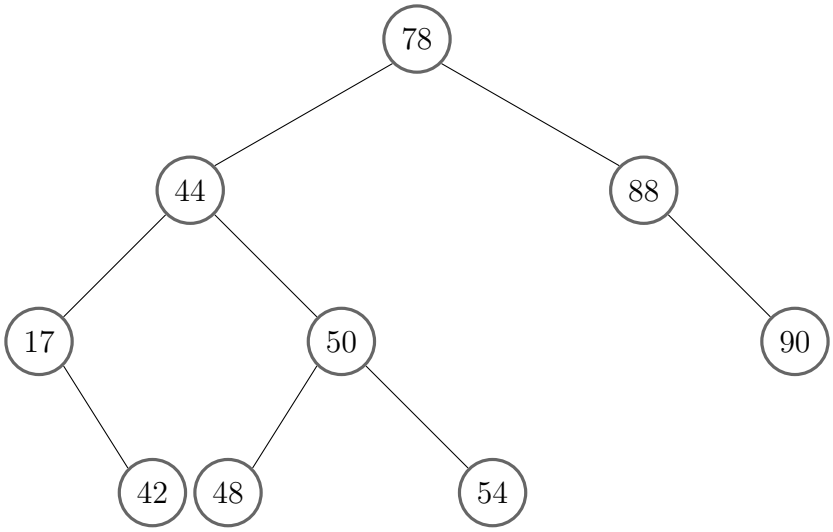


Figure 18: delete(62)

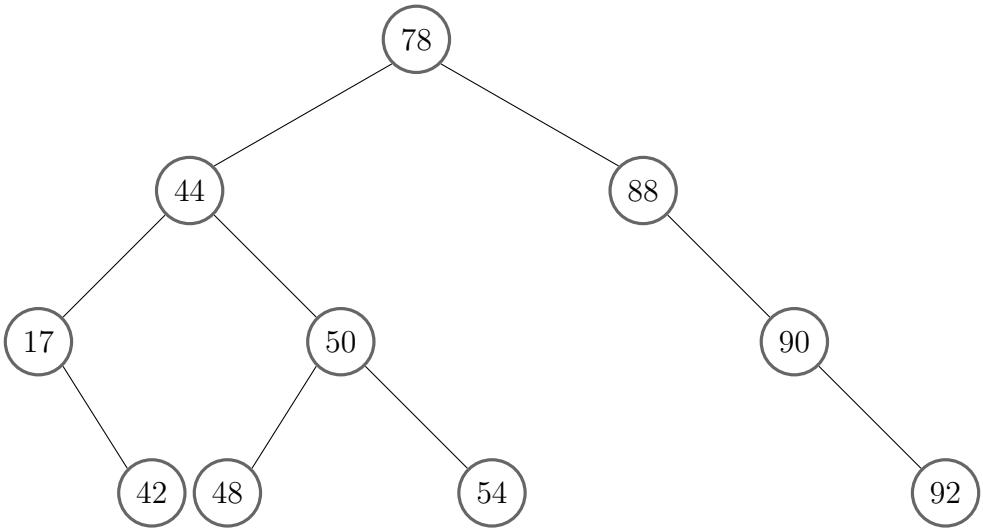


Figure 19: insert(92)

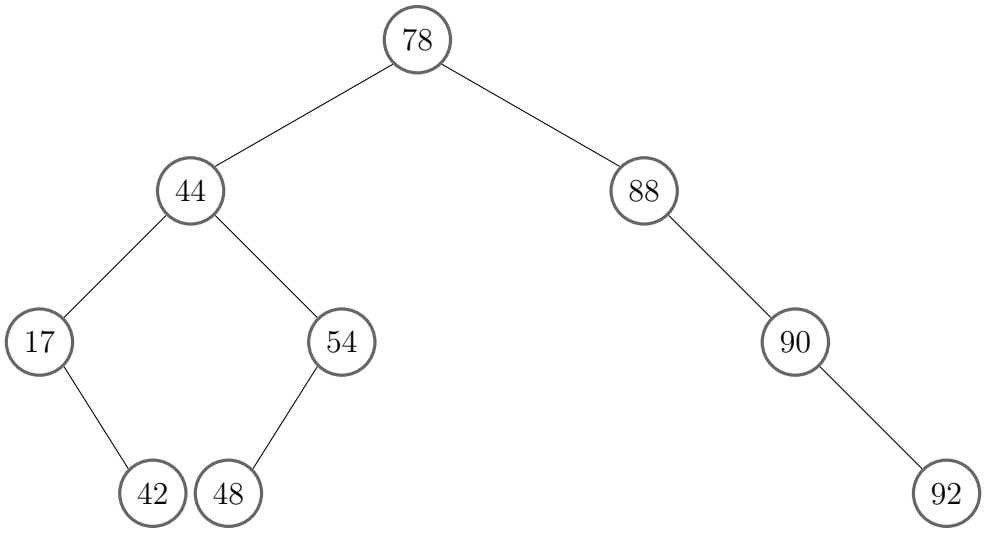


Figure 20: delete(50)

Question 3 (Hashing)

(25 points) Assume a hash table implemented as an array of 13 entries (indexed 0 through 12). Assume that you have a hash function that hashes entry x to $h(x) = x \mod 13$. Assume also that in the event of a collision, an entry is stored in the next available entry in the array. Show the hash table after insertion of the following entries:
4, 19, 30, 13, 20, 5

Answer:

Initially the hash table is empty with 13 entries

h(4) = 4 mod 13 = 4. Thus the element 4 will be inserted at index 4.

h(19) = 19 mod 13 = 6. Thus the element 19 will be inserted at index 6.

h(30) = 30 mod 13 = 4. There is a collision. The next available index in the array is 5. Thus the element 30 will be inserted at index 5.

h(13) = 13 mod 13 = 0. Thus the element 13 will be inserted at index 0.

h(20) = 20 mod 13 = 7. Thus the element 20 will be inserted at index 7.

h(5) = 5 mod 13 = 5. There is a collision. The next available index in the array is 8. Thus the element 5 will be inserted at index 8.

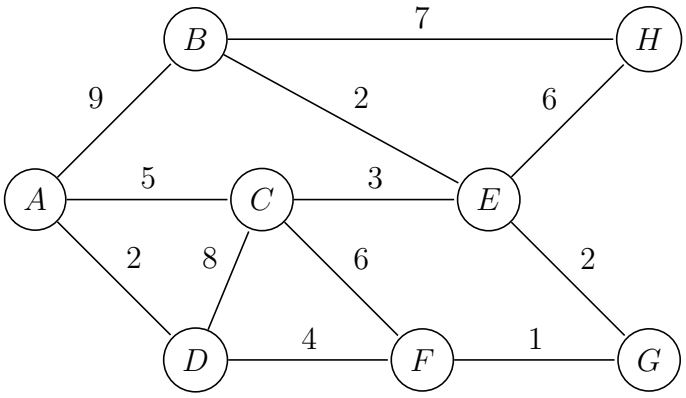
After inserting the entries the hash table will look like as follows –

Table 1: Hash Table

13				4	30	19	20	5				

Question 4 (Minimum Spanning Tree)

(25 points) Consider the following weighted undirected graph. Show the order of the edges selected by Prim's algorithm to form a minimum spanning tree.
Assume Prim's algorithm starts at node A



Answer: PRIM'S ALGORITHM is used to find the minimum spanning tree. Spanning tree is the subset of a graph G, such that it will have same number of vertices as G but number of edges will be one less than total number of vertices in G (with no cycle) so, V' = V, E' = $\|V\| - 1$.

Here the number of nodes is 8 thus the allowable number of vertices in MST is 7.

Always pick the vertex that has the smallest weight from the already visited node to keep the tree structure.

(1) Starting from A, out of the connected vertices [(A,B), (A,C), (A,D)] the least cost vertex is **(A,D)** (with cost 2)

(2) out of the connected vertices [(A,B), (A,C), (D,C), (D,F)], the next least cost vertex is **(D,F)** (with cost 4)

(3) out of the connected vertices [(A,B), (A,C), (D,C), (F,G), (F,C)], the next least cost vertex is **(F,G)** (with cost 1)

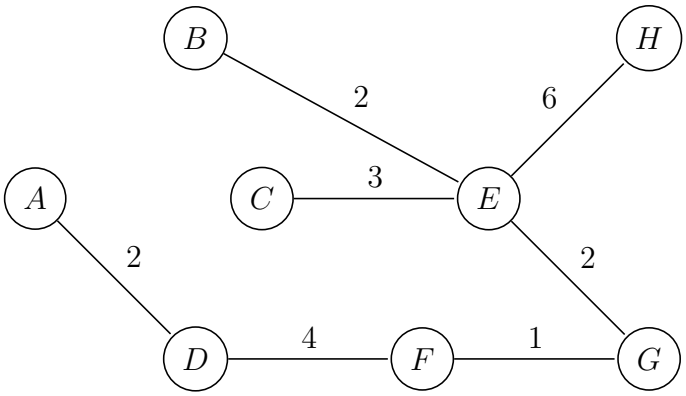
(4) out of the connected vertices [(A,B), (A,C), (D,C), (F,C), (G,E)], the next least cost vertex is **(G,E)** (with cost 2)

(5) out of the connected vertices [(A,B), (A,C), (D,C), (F,C), (E,H), (E,B), (E,C)], the next least cost vertex is **(E,B)** (with cost 2)

(6) out of the connected vertices [(A,B), (A,C), (D,C), (F,C), (E,H), (E,C)], the next least cost vertex is **(E,C)** (with cost 3)

(7) out of the connected vertices [(A,B), (A,C), (D,C), (F,C), (E,H)], the next least cost vertex is (A,C) but since C is already visited it will create a cycle the next least cost vertex to a non-visited node is **(E,H)** (with cost 6)

The MST will look like as follows,


B
H
2
6
A
C
3
E
2
2
D
4
F
1
G