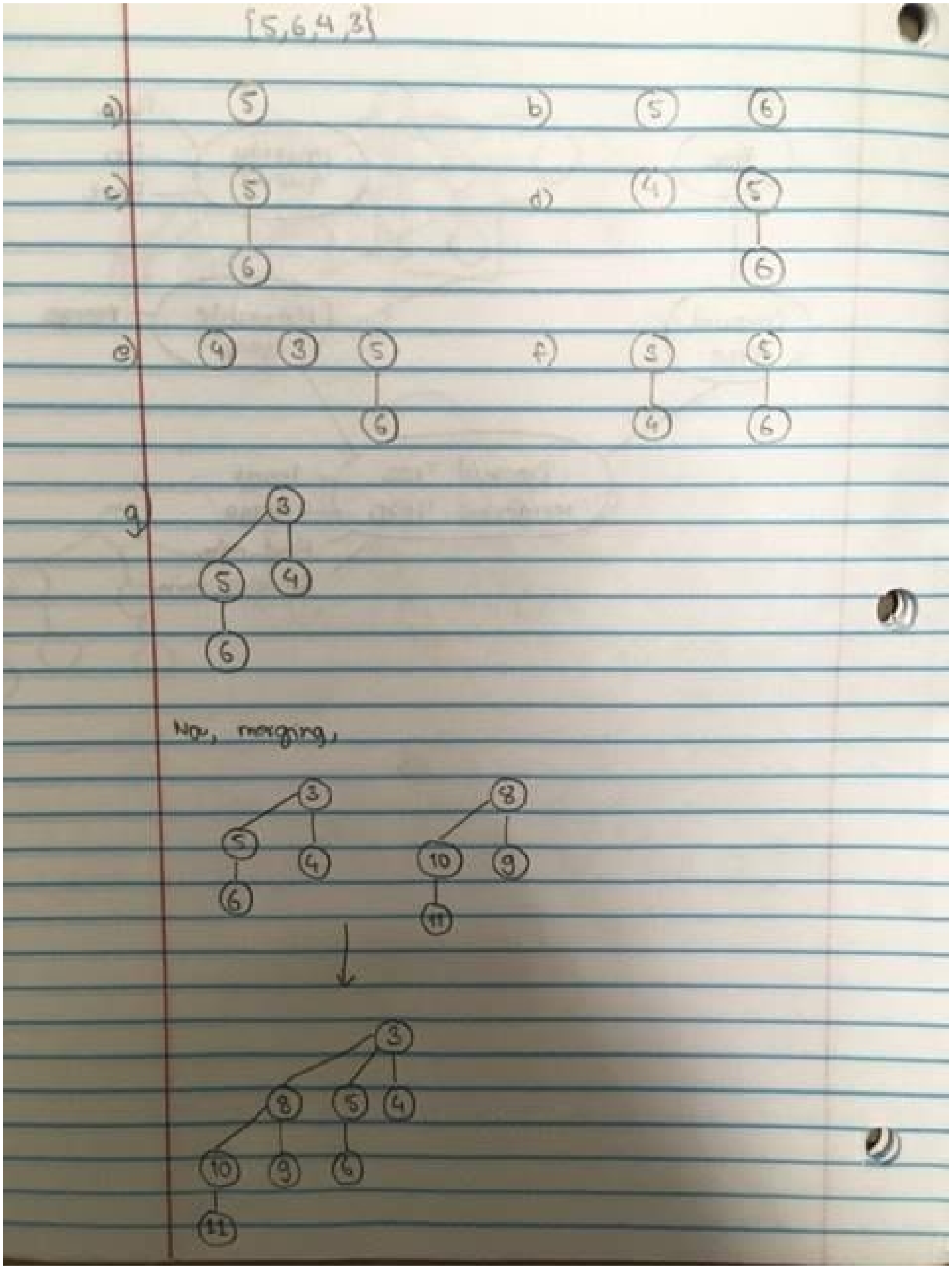
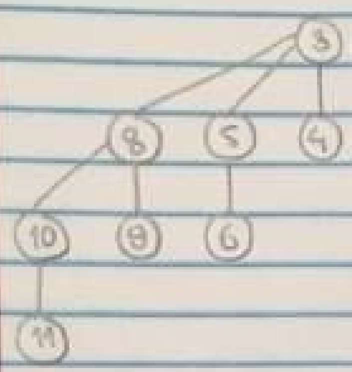
Lab 5: Merging Heaps - Tugii & Ash

Writeup A:

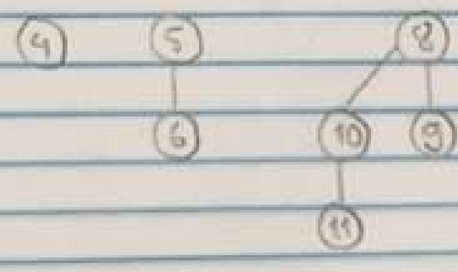


Now, removing 3 element

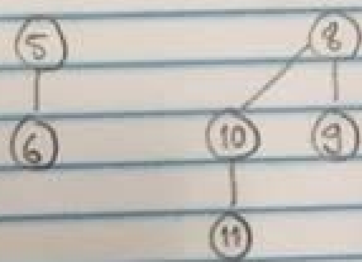
a)



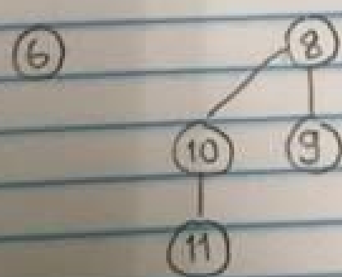
b)



c)



d)



Writeup B:

The greatest advantage of a binomial heap compared to a typical binary tree heap is that binomial trees allows the heaps to be merged in O(log n) time, whereas a typical binary tree heap takes O(n) time.

The greatest disadvantage of a binomial heap is that method to find the minimum takes O(log n) time in binomial heap compared to O(1) for binary tree heap. Implementation for a binomial heap can also be much more complex because we need to keep track of multiple children instead of just two, as in binary heap, and multiple trees.

Writeup C:

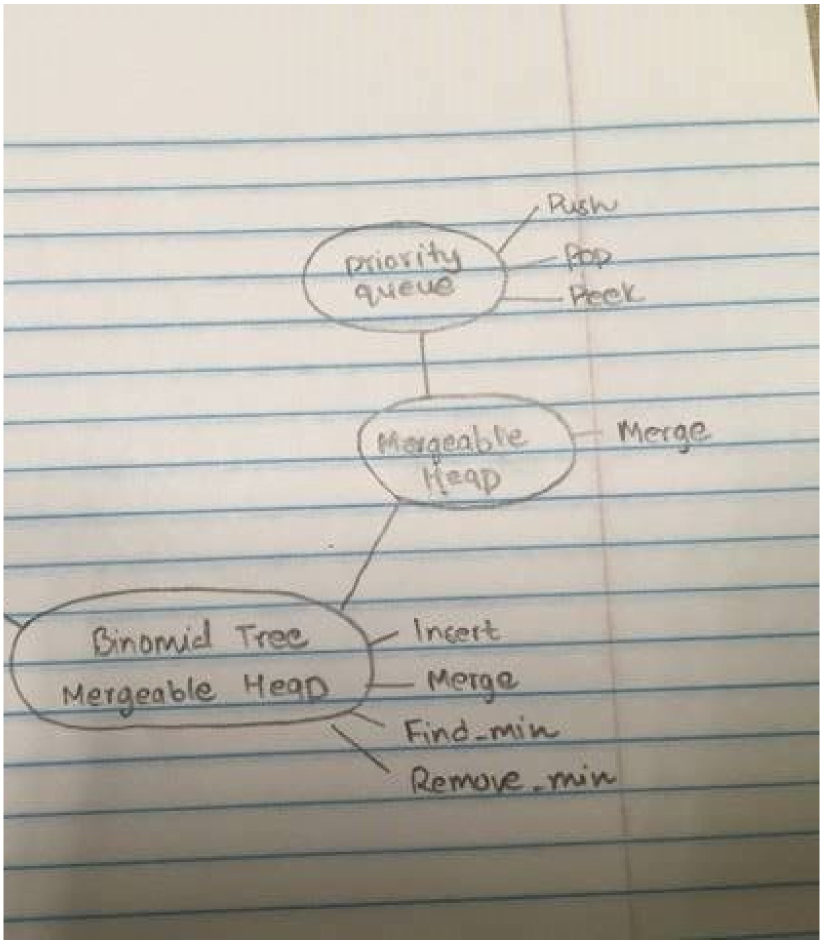
a) Insert - O(1) amortized time
To insert a new element into a heap, we can create a new heap that has only the new element and merge the two heaps. The merge will take O(log n), but the amortized time of insertion is O(1).

b) Merge - O(log n)
Merging two heaps with the same degree is simple since it only entails comparing the roots and adding one heap as a subtree to another. Since each tree has at most log n order, the merge operation will take O(log n).

c) Find_min - O(log n)
The finding minimum would take O(log n) as the number of trees in a binomial heap could be (log n) and the root nodes of all the trees need to be checked to find the minimum.

d) Remove_min O(log n)
Removing the minimum item would take O(log n) as the number of trees in a binomial heap could be (log n) and the root nodes of all the trees need to be checked to find and remove the minimum. After removing the minimum element, we reverse the children into a separate binomial heap and merge it with the original.

Design:



Implementation of the tests, the binomial heap as well as the extension is in the attached python script.