Name: Georges Hatem

CS 590 Homework 4: Priority Queues and Heaps Reinforcement Exercises

Due Date: February 20, 2022

Problem 5.7.6:

A worst-case list with n elements for insertion sort would be a list in the reverse order, for example, a list as follows (supposing we have n elements in the list going from 1 to n):

n, n-1, n-2,……, 3, 2, 1

The reasoning behind this is as follows:

In an insertion sort, we at first start with an Array (let's suppose A) of n unordered elements. Let's suppose we have an integer x since we want to do compairason. We start by comparing the value of the 2nd element of Array A to the value of the first element of Array A. Then, based on that we go ahead and insert the lowest number between the 1st and 2nd element as the 1st element and the bigger one

as the 2nd element. Then, we go ahead and compare the value of the 3rd element to the value of the 2nd element. If the value of the 3rd element is bigger than the value of the 2nd element, then we will move on onto comparing the 4th element to the 3rd element. If the value of the 3rd element is lower than the value of the 2nd element, then we go ahead and compare the value of the 3rd element to the value of the 1st element and if the value of the 3rd element is bigger than the value of the 1st element, then in this case, the value of the 3rd element becomes the value of the 2nd element and the value of the 2nd element becomes the value of the 3rd element in Array A. We keep going this way until we reach the end of Array A, and we have Array A sorted in ascending order. The table below shows my explanation and my analysis provided above:

Let's consider the worst case insertion sort list below:

Worst Case Insertion Sort List Example					
15	**12**	**10**	**8**	**5**	**2**

The reason why this is an example of the worst case insertion sort list is because we are comparing each elements in the list to all the previous elements compared before it and placed in order in Array A. **This means that the running time of the Insertion Sort worst case Algorithm is O(n^2). This is explained below:**

At first, 12 is compared to 15, and 12 is less than 15. So, element number 1 of the Array above will become 12 and element number 2 of the Array above will become 15. This takes 1 compairason to be done. Then, 10 is compared to 15 and since 10 is less than 15, 10 will be placed as the 2nd element in the Array. Then, 10 is compared to 12, and since 10 is less than 12, then 10 becomes the first element in

the Array and 12 becomes the 2nd element and 15 becomes as indicated above, the 3rd element. We continue doing that for all values in the Array. Now, imagine why the list above would be insertion sort worth case scenario. It is because if the list is in reversed order as above, every element in the list will have to be compared with all previous elements in the list that have been ordered. This means that the element at n-1 will perform 1 comparaion, the element at n-2 will perform 2 compairasons. The element at n-3 will perform 3 compairasions. This means that suppose we have a list with n elements (from 1 to n), we will be doing that much comparaisons in thr worst case scenario of the insertion sort method:

$$\mathbf{1} + \mathbf{2} + \mathbf{3} + \cdots + (\boldsymbol{n} - \mathbf{3}) + (\boldsymbol{n} - \mathbf{2}) + (\boldsymbol{n} - \mathbf{1})$$

We know from Theorem 1.3.2 that:

$$\sum_{i=1}^{n}(i) = \frac{(n) * (n+1)}{2}$$

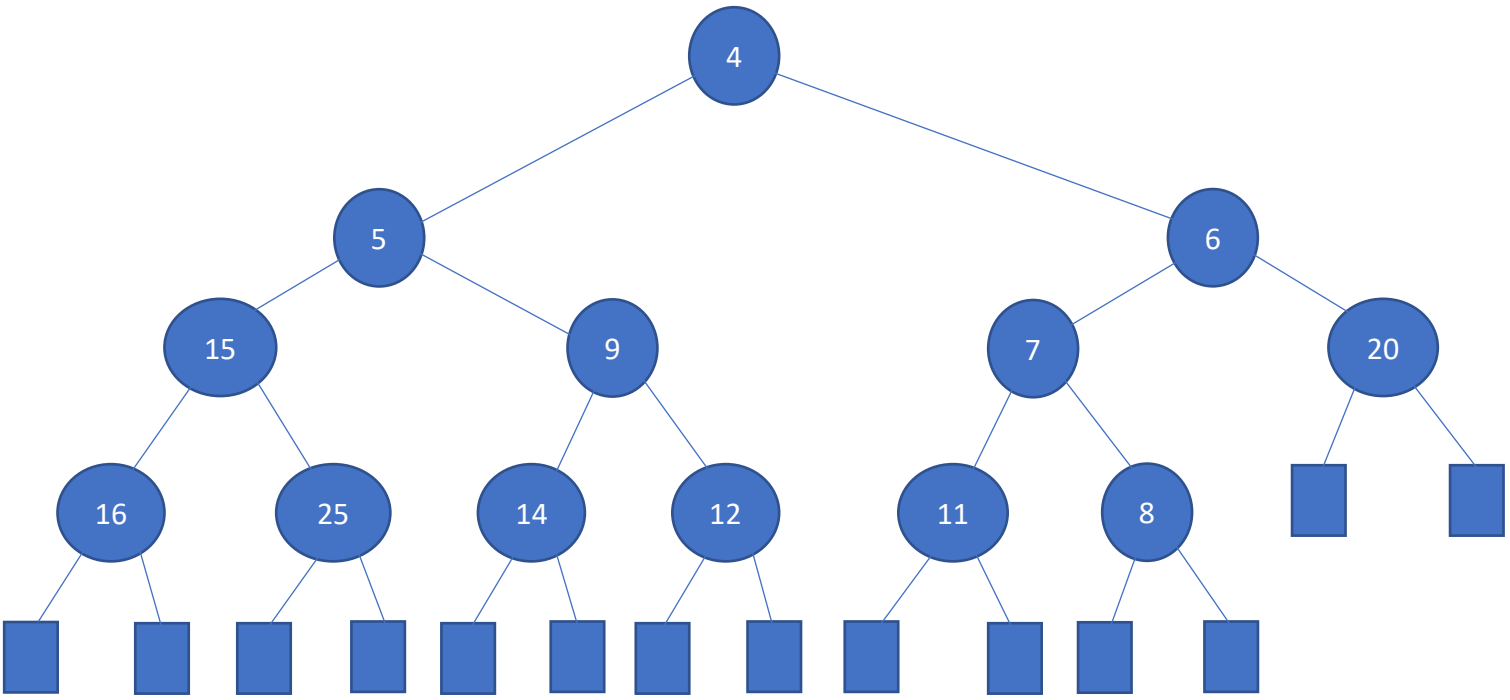
So, if we replace the n above by n-1 since we are adding from 1 to n-1, consecutively we get the following:

$$\sum_{i=1}^{n-1}(i) = \frac{(n-1) * (n-1+1)}{2} = \frac{(n) * (n-1)}{2}$$

This means that the running time for the worst case insertion sort list is Ω(n^2).

Problem 5.7.14:

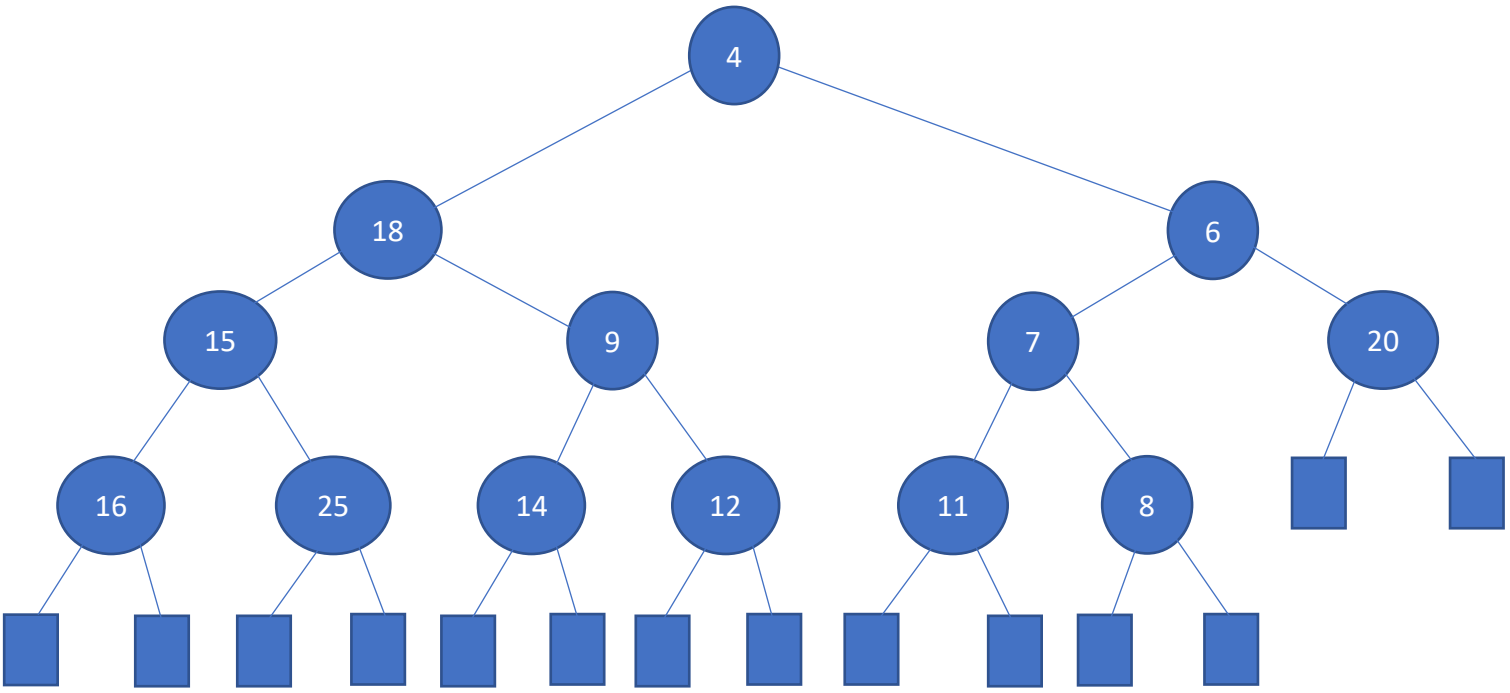
The given heap is:



The increase key on key 5 to value 18 is performed as follows:

Step 1:

Change the value of the key 5 to value 18.

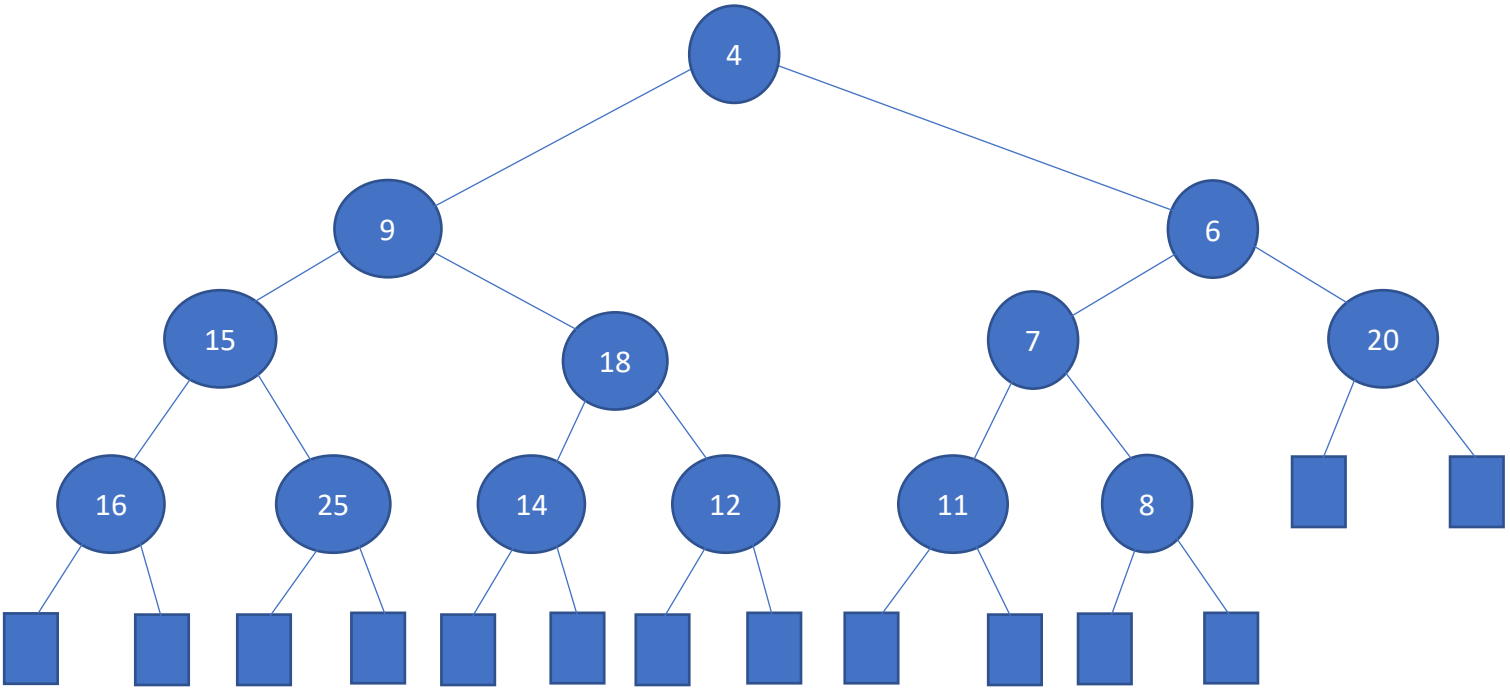


Step 2:

Min heapify the node whose value has been increased to 18.

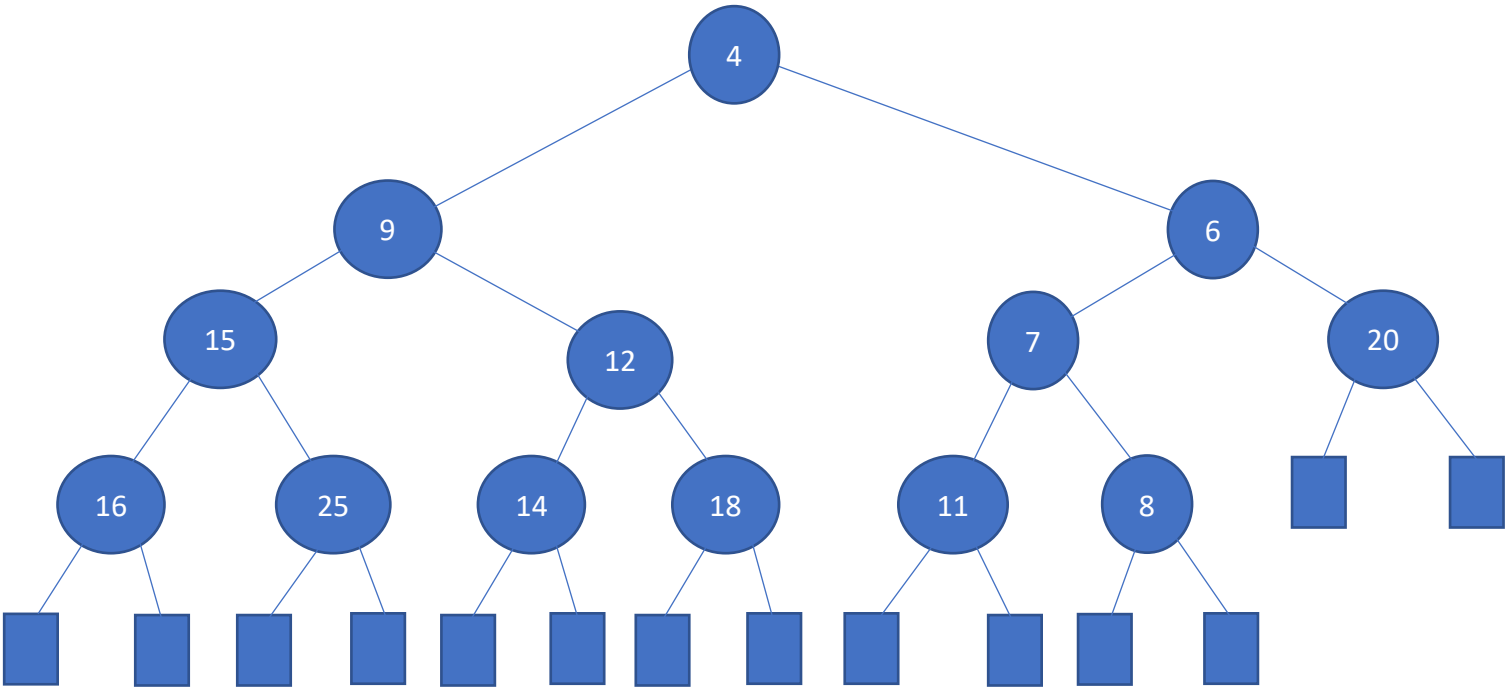
a.

Node with key 18 is greater than its right child 9. So, swap the nodes.



b.

Node with key 18 is greater than its right child 12. So, swap the nodes.



Since, min heap property is maintained,

the final heap after increasing the node with key 5 to value 18 is: