Derek Lu

2/11/18

Textbook: (R) 9.5, 9.6, 9.18, 9.19, 9.24.

9.5. A simple modification to the UnsortedPriorityQueue class to execute in O(1) time would be to add a global variable that keeps track of the minimum element.

Necessary modifications:

① Add declaration of global variable

② Assign global variable in the insert method if it is indeed the min value

③ Update the global variable whenever the removeMin() method is called.

④ The min() method would return the global variable directly instead of traversing up to O(n) times to find it

9.6 Based off the existing removeMin() method that simply removes the minimum element from the unsorted list, using a global variable holding the minimum element would adjust the Big Oh to O(1). The element could just be found by its key and removed from the list without traversing to the O(n) times.

9.18 $\sum_{i=1}^{n} \log i = \log 1 + \log 2 + \log 3 + \ldots \log n = F(i) = \int_1^n \log(i)\, di$

where n is the total number of entries on the heap.

Let $F(i) = \int \log i\, di$. Using integration by parts ($\int u\,dv = uv - \int v\,du$)

$\int_1^n \log i\, di \qquad u = \log i \quad du = \frac{1}{i}\, di$

$\int dv = \int di \quad v = i$

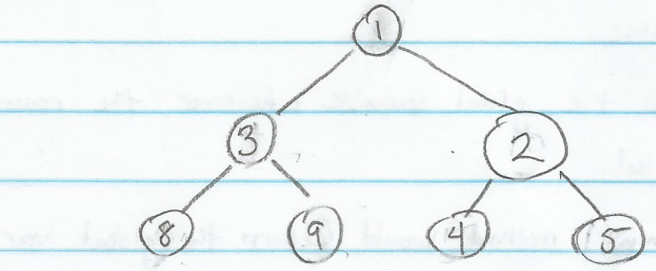
$\int_1^n \log i\, di = \log i(i) - \int i \cdot \frac{1}{i}\, di = i\log(i) - i$

$= (n\log(n) - n) - \underbrace{(\log(1) - 1)}$

$= +c$ (constant, not significant for Ω notation)

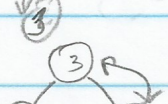
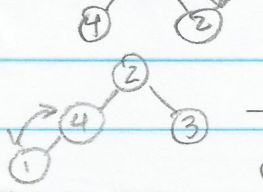
Therefore $\sum_{i=1}^{n} \log(i) = \Omega(n\log(n))$ for heap-sort

9.19



Preorder traversal → (1, 3, 8, 9, 2, 4, 5)

9.24 It is proven that the height of a heap tree is $O(\log n)$. Thus when inserting a single value at the left most position at the bottom layer of the heap, if the new value is less than the root, it must make its way up one level at a time until it becomes the new root

COMMAND	Tree
Ex: heap.insert(4);	
heap.insert(3);	
heap.insert(2);	
heap.insert(1);	

$(\Omega(n\log n))$