CS466: Design and Analysis of Algorithms
Assignment 2
Spring 2016
Instructor: Mark Petrick

MingHao Lu
20465562

May 18, 2016

Question 1

Suppose the claim is not true, then the amortized cost must be of $o(\log(k))$. Consider the following sequence of operations. Given a list of n number, perform n insertions and then n deletemin. Since the average over this sequence(amortized cost of any operation) is $o(\log(n))$ and this sequence has 2*n operations, the total cost must be $2n * o(\log(n))$ or $o(n \log(n))$. This sequence of insertion and deletion gives us the n numbers in increasing order, effectively sorting them. Since the keys can only be accessed by pairwise comparison, this is a comparison sorting algorithm of $o(\log(n))$. However, we know comparison based sorting algorithms has a lower bound of $\Omega(\log(n))$. Contradiction. Therefore the claim must be true.

Question 2

i)

Linking(adding one tree as a leftmost child of another) is obviously constant time. Insertion is just linking the heap with a new heap of one node. Merging is a linking. Decrease key is removing a tree(same as removing a child from the parent node, which is constant time) and a linking. Since these operations only involve one or two constant time operations, they are O(1).

The only steps to take during deletemin is to delete the root node and consolidate all of its children into a single tree. Suppose the root has r children, then after removing the root you are left with r seperate trees. It'd take a constant time operation to decrease the number of trees left by one, so total cost is $\Theta(r)$. Since $r \leq n$, and if the tree has only two layers then $r = n - 1$, we conclude deletemin cost $\Theta(n)$ worst case.

ii)

cost(i) is the cost for severing the root ($O(1)$) and joining the children of the root together. Let k be the number of children the root has. Each joining operation removes one tree and takes $O(1)$. Since there are k trees and we apply join until only one tree is left, this takes $O(k-1)$. Together with the sever, this is $O(k)$. Since $k \leq n$, cost(i) is $O(n)$.

Let Φ_i be the number of nodes in the tree, Φ_0 is zero and Φ_n is non-negative. Then we have $\Phi_n > \Phi_0$. Change in potential from deletemin is -1. $charge(i) = cost(i) + \Phi_i - \Phi_{i-1} = O(n) - 1 = O(n)$