New CS 473 Spring 2015 | Arash Khatibi (khatibi2)
Homework 6 Problem 2 | Liang Tao (ltao3)
Chen Zhang (czhang49)

1 Part A

Define the following variables:
T: A binary tree T.
P(Q): The length of a random root-to-leaf path in the binary tree T.
When we meld two arbitrary heap-ordered binary tree Q_1 and Q_2. The worst case is that the algorithm will trace down both Q_1 and Q_2. The best case is that the algorithm will only trace down one tree (when all the elements in Q_1 has smaller or larger priority than Q_2). Each time an operation is carried out, we will trace down one step on Q_1 or Q_2, either to the left or to the right. Thus we know that the maximum step we trace down is the summation of the length of the two trees, which is $P(Q_1) + P(Q_2)$. Now we will prove that

$$E[P(Q_1) + P(Q_2)] = O(\log n).$$

It suffice to show that for a tree T with n nodes, $E[P(T)] = O(\log n)$, since if we remove the root of T, it will readily become two trees with a total number of nodes equal to n-1.
Now we claim that $E[P(n)] \leq \log(n+1)$. If $n = 0$, this claim is readily satisfied. Otherwise, suppose that the left and right subtrees of T contains l and r nodes, respectively. Notice that $l < n$ and $r < n$. With strong induction hypothesis, we have $E[P(l)] \leq \log(l+1)$ and $E[P(r)] \leq \log(r+1)$. Thus we can have the following equations:

$$\begin{aligned}
\frac{1}{2}(E[P(l)] + E[P(r)]) &\leq \frac{1}{2}(\log(l+1) + \log(r+1)) \\
&= \log(2\sqrt{(l+1)(r+1)}) \\
&\leq \log(l+1) + (r+1) \\
&\leq \log(n+1)
\end{aligned}$$

Thus the strong induction proof holds true and we have that $E[P(Q_1) + P(Q_2)] = O(\log n)$.

2 Part B

Define the following variables:
$|\gamma|$: The length of path γ.
h_Q: The length of a path from root to leaf.
Γ: The set of all paths from the root to leaf with length exceeding $(c+1)\log n$.

Note that to reach for any leaf from the root via any fixed path γ, the probability is $2^{-|\gamma|}$. Suppose that Γ is the set of all paths from the root to leaf with length exceeding $(c+1)\log n$, we have the following relation:

$$Pr[h_Q > (c+1)\log n] = \sum_{\gamma\in\Gamma} 2^{-|\gamma|} < \sum_{\gamma\in\Gamma} 2^{-(c+1)\log n} = |\Gamma| n^{-(c+1)} \leq n^{-c}$$

Thus we see that we have a exponentially decaying probability of length of path exceeding $(c+1)\log n$ with the number of node n. Note the running time of Meld(Q_1,Q_2) is $P(Q_1)+P(Q_2)$. This proves that the probability that we have running time of Meld(Q_1,Q_2) having $O(\log n)$ is large.

3 Part C

MakeQueue: Create an empty set. This is O(1) time.
FindMin(Q): Return the $Key(Q)$, which is the root of the tree, this takes O(1) time.
DeleteMin(Q): Return Meld(left(Q),right(Q)).
Insert(Q,x): Make the new element itself a new tree which takes O(1) time and return Meld(Q,x).
Delete(Q,x): Let Q_x be the tree with root at x, replace Q_x with DeleteMin(Q_x) which calls Meld() once and then return Q.
DecreasePriority(Q,x,y): Let Q_x be the tree with root at x, detach it from the parent, change the value of $Key(x)$ to y and return Meld(Q,Q_x)