CSE 2331 HOMEWORK 7

BRENDAN WHITAKER

1. Insert takes $clog_2(s)$, and ExtractMax takes $clog_2(s)$, where s is the number of elements in P.
Number of insertions: $n^{5/3}$.
Upper bound on time for single insertion: $clog_2(n^{5/3}) = \frac{5c}{3}log_2(n)$.
Upper bound on time for $n^{5/3}$ insertions: $(5/3)cn^{5/3}log_2(n) \in O(n^{5/3}log_2(n))$.
Lower bound on steps 1-6 (total insert time):

$$\begin{aligned} T_{insert}(n) &= \sum_{s=1}^{n^{5/3}} clog_2(s) \geq \sum_{s=n^{5/3}/2}^{n^{5/3}} clog_2(s) \geq \sum_{s=n^{5/3}/2}^{n^{5/3}} clog_2(\frac{n^{5/3}}{2}) \\ &\geq \frac{n^{5/3}}{2} clog_2(\frac{n^{5/3}}{2}) = \frac{n^{5/3}}{2} c(log_2(n^{5/3}) - log_2(2)) \\ &\geq \frac{n^{5/3}}{2} c\frac{5}{3}log_2(n) - \frac{n^{5/3}}{2} clog_2(2) \in \Omega(n^{5/3}log_2(n)). \end{aligned} \tag{1}$$

Thus $T_{insert}(n) \in \Theta(n^{5/3}log_2(n))$.

Steps 7-10 repeat until P is empty.
Thus the number of ExtractMax operations is the number of insertions.
Number of deletions: $n^{5/3}$.
Upper bound for single deletion: $clog_2(n^{5/3}) = \frac{5c}{3}log_2(n)$.
And we can see the running time will be identical to that of steps 1-6, so we have:
$T_{delete}(n) \in \Theta(n^{5/3}log_2(n))$.

2. (a) Insert takes cs time and ExtractMax takes constant time.
Number of insertions: $n^{5/3}$.
Upper bound on time for single insertion: $cn^{5/3}$.
Upper bound on time for $n^{5/3}$ insertions: $cn^{10/3} \in O(n^{10/3})$.
Lower bound on steps 1-6:

$$T_{insert}(n) = \sum_{s=1}^{n^{5/3}} cs \geq \sum_{s=n^{5/3}/2}^{n^{5/3}} cs \geq \sum_{s=n^{5/3}/2}^{n^{5/3}} cn^{5/3}/2 = cn^{10/3}/4 \in \Omega(n^{10/3}). \tag{2}$$

So $T_{insert}(n) \in \Theta(n^{10/3})$.
Number of ExtractMaxs: c.
Upper bound on time for single extract max: c.
Upper bound on n ExtractMaxs: cn.
Lower bound on steps 7-10:

$$cn. \tag{3}$$

(b) Time for single insertion: c.
Upper bound on time for $n^{5/3}$ insertions: $cn^{5/3}$.
Lower bound:

$$\sum_{i=1}^{n^{5/3}} c \geq \sum_{i=n^{5/3}/2}^{n^{5/3}} c = cn^{5/3}/2 \in \Omega(n^{5/3}). \tag{4}$$

Date: AU17.

SUPER SKETCHY (ITS CORRECT) JUST STUDY THIS THE RUNNING TIME OF THE DELETIONS IS PROPORTIONAL TO THE SIZE OF THE DATASTRUCTURE, AND THE DATASTRUCTURE'S SIZE IS DECREASING BY i FROM $n^{5/3}$.

Number of deletions: n.
Upper bound on time for single deletion: $c(max(s)) = cn^{5/3}$.
Upper bound on time for n deletions: $ncn^{5/3} = cn^{8/3}$.
Lower bound:

$$\sum_{i=1}^{n} c(n^{5/3} - i) \geq \sum_{i=1}^{n} c(n^{5/3} - n) = cn(n^{5/3} - n) \in \Omega(n^{8/3}). \tag{5}$$

3. Continued on hard copy.
4. Let $S_n = \{1, 2, ..., n\}$.
   (a) Just insert nodes from 1 to n in ascending order. You will get a tree that is just a line of right children with height n, hence height $\geq n/2$.
   (b) Start with $v = \lfloor n/2 \rfloor$. Then insert $\lfloor n/4 \rfloor$, $\lfloor 3n/4 \rfloor$, etc, dividing by 2 and taking floor of each subinterval created by the partition of S_n into sets of numbers with keys above/below that of the last added node. Then we We will get a complete (or nearly complete) binary search tree, and hence it will have height at most $2log_2(n)$, since it will actually have height $\lceil log_2(n) \rceil$, since n may not be an exact power of two.
   (c) Start with $\lfloor \sqrt{n} \rfloor + 1$. Then insert each node less than it until we hit 1, in descending order. So our tree has height at least $\sqrt{n}$. Now start again with the node $v = \lfloor \sqrt{n} \rfloor + 1 + \lfloor \frac{n-(\lfloor \sqrt{n} \rfloor + 1)}{2} \rfloor$. Then we do again the binary tree algorithm used in part by, dividing the remaining numbers in half and taking the floor, then adding that node, then subdividing each of the newly created intervals by that node into two, ad nauseum. Then the subtree rooted at v will have height at most $\lceil log_2(v) \rceil \leq log_2(n) \leq 2\sqrt{n}$ for all $n \in \mathbb{N}$.