CIS 575. Introduction to Algorithm Analysis
Remarks on Assignment #3, Spring 2014

Question 1

1. Observe that each `while` loop is iterated n times. Thus Q.RemoveFirst is called n times, with each call taking constant time; hence the total time used for such calls is in $\mathbf{O(n)}$.

2. For P.Insert, observe that in the ith iteration, the size of the priority queue will grow from $i-1$ to i. Our assumptions thus entail that there exists $f \in O(\lg(i))$ such that the i'th iteration takes time $f(i)$. There are now at least two ways to proceed:

   - We may apply *Howell*'s Theorem 3.28, which is applicable since $\lg(i)$ is a smooth function (as $\lg(2i) = \lg(i) + 1 \leq 2\lg(i)$ for $i \geq 2$), to infer that the total running time for P.Insert is in $\mathbf{O(n \cdot \lg(n))}$.
   - We may approximate the total running time for P.Insert as (with c such that $f(i) \leq c\lg(i)$)

     $$\sum_{i=1}^{n} f(i) \leq \sum_{i=1}^{n} c\lg(i) = c\sum_{i=1}^{n} \lg(i) = c\lg(\prod_{i=1}^{n} i) = c\lg(n!) \leq c\lg(n^n) = cn\lg(n) \in \mathbf{O(n\lg(n))}$$

     (one may remark that the bound is tight as one can show that $n\lg(n) \in O(\lg(n!))$).

3. The case for P.RemoveMin is similar to the previous case, except that it is now the *first* iterations that are the most expensive (as the priority queue gets smaller and smaller); there exists $g \in O(\lg(i))$ such that the ith *last* iteration takes time $g(i)$. But clearly, reversing the order doesn't affect the total time. We therefore infer that also P.RemoveMin has a total running time in $\mathbf{O(n \cdot \lg(n))}$.

4. The total running time is composed of:

   (a) the time spent in Q.RemoveFirst, which we saw is in $\mathbf{O(n)}$;

   (b) the time spent in P.Insert, which we saw is in $\mathbf{O(n\lg(n))}$;

   (c) the time spent in P.RemoveMin, which we saw is in $\mathbf{O(n\lg(n))}$;

   (d) the time spent in Q.InsertLast, which is easily seen to be in $\mathbf{O(n)}$;

   (e) the time spent on running the `while` loops, excluding the calls in the bodies but including loop test evaluation, which is easily seen to be in $\mathbf{O(n)}$.

   Using a general property of big-O, that if $f_1 \in O(g_1)$ and $f_2 \in O(g_2)$ then $f_1 + f_2 \in O(\max(g_1, g_2))$, we infer that the total running time of Sort is in $\mathbf{O(n\lg(n))}$.

Question 2 For (a), the outer loop iterates $5n$ times with the ith iteration taking time in $\Theta(i)$. Since i is a smooth function, we can infer that the total running time is in $\Theta((5n) \cdot (5n)) = \mathbf{\Theta(n^2)}$.

Alternatively, one can give a tight estimate of $\sum_{i}^{5n} \frac{i}{2}$.

For (b), the outer loop iterates n^2 times with the ith iteration taking time in $\Theta(\log i)$. Since $\log i$ is a smooth function, we can infer that the total running time is in $\Theta(n^2 \log n^2) = \mathbf{\Theta(n^2 \log n)}$.

For (c), the outer loop iterates approximately $\lg n$ times: for the first iteration, the inner loop iterates n times; for the second iteration, the inner loop iterates $n/2$ times, etc. Since $n + n/2 + n/4 + n/8 \ldots \leq 2n$, we infer that the total running time is in $\mathbf{\Theta(n)}$.