

RAM Algorithms 1

AVA Fall 2005, week 8

Mandatory assignment:

- **Sorting:** Given a priority queue data structure that supports *insert* in $f(m)$ time, and *extractmin* in $g(m)$ time, where m is the current size of the priority queue, how can you use it to sort a sequence of n numbers? What is the time taken by this sorting algorithm? How fast can you sort, given a priority queue with $f(m) = g(m) = \lg \lg m$?
- **Union-Split-Find on intervals:** Let $I = \{I_1, I_2, \dots, I_n\}$ be a set of n intervals that partition the universe $U = \{1, 2, \dots, m\}$, where $x \in I_j, y \in I_{j+1} \Rightarrow x < y$ (i.e., interval I_{j+1} immediately follows the interval I_j), for $1 \leq j \leq n - 1$. We want to maintain I under the following operations:
 - *find*(x): return the name of the interval containing x , for $x \in U$,
 - *union*(x): combine the interval containing x (say, I_j) with the immediately following interval (I_{j+1}), and
 - *split*(x): split the interval containing x into two intervals $I \cap [1, x]$ and $I \cap [x + 1, m]$.

Show that this problem is equivalent to the predecessor problem (i.e., show that given an efficient data structure for this problem, you can construct an efficient data structure for the predecessor problem, and vice versa).