Midterm 1 Revisions

Shashank Singh
sss1@andrew.cmu.edu
1
February 29, 2012

Problem 1

(a) $T(n) \in \Theta(n^2)$

(b) $T(n) \in \Theta(n)$

(c) The best possible algorithm will run in $\Theta(n)$ time.

(d) Let P_X, P_Y be polynomials with

$$P_X = \sum_{i=0}^{2M} a_i x^i, P_Y = \sum_{i=0}^{2M} b_i x^i,$$

where $\forall i \in \{0, 1, \ldots, 2M\}$, $a_i = 1$ if $i - M \in X$ and $a_i = 0$ otherwise, and $b_i = 1$ if $i - M \in Y$ and $b_i = 0$ otherwise. Compute $P = P_X P_Y$ in $O(M \log M)$ time using a Fast Fourier Transform. Then, $\forall i \in \{0, 1, \ldots, 4M^2\}$, let c_i be the coefficient of x^i in P, so that $c_i = 1$ if $i - 2M \in X + Y$, and $c_i = 0$ otherwise. Thus, the elements of $X + Y$ can be read in linear time from the coefficients of P, so that $X + Y$ is computed in $O(M \log M)$ time.

(e) See written test.

Problem 2

(a) See written test.

(b) Let $\Phi(n)$ be the sum of the token values in all of the arrays, i.e. $\Phi = \sum_{i=0}^{\log_2 n - 1} i * (\log n - i)$.

(c) The amortized cost of inserting the new unit size array is a constant actual time, plus $\log n$ time for the change in the potential function, giving $O(\log n)$ amortized time for inserting the new unit size array.

(d) The amortized cost of merging two arrays of size 2^i is 2^i time for actually merging the arrays, plus $2^{i+1}(\log n - (i + 1)) - 2 * 2^i(\log n - i) = 2^{i+1}$ time for the change in potential, giving $O(2^i)$ amortized time for merging two arrays of size 2^i.

(e) Since in n insertions, we add about $n * \log n$ stored tokens, and the amortized cost of an insert is, in the worst case $2^{\log_2 n} = n$, the average cost is at most $\frac{n \log n}{n} \in O(\log n)$.