Ph.D. Qualifying Exam (Fall 2003)
Algorithms

Question 1.

Given a directed graph $G = (V, E)$, the transitive closure G^* is defined as a graph (V, E^*) such that $E^* = \{(i, j) \mid \text{ there is a path from vertex } i \text{ to vertex } j \text{ in } G\ \}$. Given G, we want to compute the transitive closure G^*. Define $t_{ij}^{(k)}$ to be 1 if there exists a path in graph G from vertex i to vertex j with all intermediate vertices in the set $\{1, 2, \ldots, k\}$, and 0 otherwise.

(a) (15%) Give a recursive definition of $t_{ij}^{(k)}$ for $k = 0, 1, 2, \ldots, n$ and $i, j \in V = \{1, 2, \ldots, n\}$. Your answer should make use of logical operators $\wedge$ and $\vee$.

(b) (20%) Give pseudo codes for a recursive process using memoization technique to compute all the $t_{ij}^{(k)}$ values for $k = 0, 1, 2, \ldots, n$ and $i, j \in V = \{1, 2, \ldots, n\}$. Analyse the worst case time and space needed. Express your answer in Θ notation.

(c) (15%) Explain how to compute the transitive closure efficiently in space $\Theta(n^2)$. Analyse the worst case running time in Θ notation.

Question 2.

Let k be a positive integer constant. We need to maintain a data structure to hold a set S of **real** numbers in the range $[0..k)$. Initially, S is empty. We need `Insert`(x) and `Delete`(x) operations for inserting and deleting element x. Note that $0 \leq x < k$. In addition, we want to support the operation `List`(j) where $j = 0, 1, 2, \ldots, k-1$ which returns a list of elements in S that fall within the range $[j, j+1)$ in increasing order.

(a) (10%) One implementation is to maintain a linear linked list of elements in sorted order. To support `Insert/Delete`, we need to traverse the linear list to find the appropriate location/element to insert/delete. To implement `List`(j), we traverse the sorted linear list, and prints only those elements that are at least j but less than $j + 1$. Analyse the worst case running time of `Insert`, `Delete` and `List` in terms of n, the current number of elements in the set S. Note that the size of S keeps changing as `Insert` and `Delete` are performed. Also, note that k is a constant.

(b) (40%) (Augmenting Data Structures) Design a data structure such that `Insert` and `Delete` both run in time $O(\lg n)$ where n is the current size of the set S, and `List`(j) should run in time independent of n, but linear in the size of the set $\{x \in S \mid j \leq x < j+1\}$. Thus, `List`$(j)$ runs in time linear in the output size.