Winter Term 2001/02
November 19, 2001

Prof. Dr. Susanne Albers
Institute of Computer Science

Algorithms Theory

Problem Set 3

To be submitted on December 3, 2001 (in the lecture)

Problem 1: (5 + 5 points)

In the lecture we studied two UNION FIND data structures.

a) In the first solution sets are represented by linear lists. Show that there is a sequence of $n-1$ UNIONs that costs $\Omega(n \log n)$.

b) In the second solution sets are represented by trees. In a UNION operation, the smaller set becomes part of the larger set. During a FIND operation, path compression is performed. Consider a sequence of $n-1$ UNIONs and f FINDs. Show that if all the UNIONs occur before the FINDs, the total running time is $O(n+f)$.

Problem 2: (10 points)

Prove that if the edge costs of a connected, undirected graph are distinct, then there is a unique minimum spanning tree.

Problem 3: (4 + 2 + 4 points)

a) Suppose that all edge costs in a connected, undirected graph are integers in the range from 1 to $|V|$. How fast can you make Kruskal's algorithm run?

b) Analyze the running time of Prim's algorithm if the priority queue is implemented as a 2-ary heap.

c) Describe an efficient algorithm that, given a connected, undirected graph G, determines a spanning tree of G whose largest edge cost is minimum over all spanning trees of G. Show that your solution is correct.

Problem 4: (10 points)

In this problem we study a simple hashing scheme where collisions are resolved by chaining. As usual, we wish to support the operations *Search*(x, S), *Insert*(x, S) and *Delete*(x, S), for a set $S \subseteq U$ and an element $x \in U$. The size m of the hash table should be $\Theta(|S|)$. If m is much smaller we cannot avoid long lists in our hash table; if m is much larger, the hash table uses a lot of space compared to $|S|$.

Suppose that, for any table size m, we know a hash function $h_m : U \longrightarrow \{0, \ldots, m-1\}$ that executes *Search*, *Insert* and *Delete* opertions in $O(1)$ time. Develop a dynamic hashing scheme that always maintains a load factor β with $\frac{1}{4} \leq \beta \leq 1$ and supports all operations in $O(1)$ amortized time. The load factor, at any time, is the size of the set S currently stored, divided by current hash table size, i.e. $\beta = |S|/m$.