Computer Science IV (January-April 2014)
Mid Term Test

Monday, Mar 10, 2014
Remark: In Question 1, part (a) is compulsory, but do part (b) OR (c)

1. Let $M(m,n)$ be the minimum number of comparisons needed to merge a sorted list of m distinct elements with another sorted list of n distinct numbers. Assume also that all the m elements are distinct from the n elements.

   The 'standard' merge algorithm takes at most $m+n-1$ comparisons proving that $M(m,n) \leq m+n-1$.

   (a) Show by a decision tree argument that

   $$\lceil \log \binom{m+n}{n} \rceil \leq M(m,n)$$

   (b) i. Show by an adversary argument that $M(n,n) \geq 2n-1$.
   ii. Does the above adversary argument extend to prove $M(m,n) \geq m+n-1$ (i.e. for the case when $m \neq n$)? If so, argue how, if not, give some range of m and n for which $M(m,n)$ can be less than $m+n-1$.

   OR

   (c) Given n distinct numbers arranged in increasing order in an array A, the problem is to test whether there is an element i in the array such that $A[i] = i$.
   i. Give an $O(\log n)$ algorithm for the problem.
   ii. Show, by an adversary argument, that if the elements are not necessarily distinct, then any algorithm to solve this problem would require $\Omega(n)$ comparisons.

2. Given two sorted lists of n elements each, give an efficient algorithm to find the median of the union of the two lists. Briefly argue the correctness and runtime of your algorithm.

3. Given a sequence of N objects $1, 2, \ldots N$, the goal is to cluster them into subsets of consecutive objects. There is a given cost $a_{i,j}$ associated with a cluster of the sequence $i+1, i+2, \ldots j$ and the goal is to group the given N objects into clusters minimizing the total cost. For example, if you decide on the two clusters $1, 2, \ldots 10$ and $11, 12, \ldots N$, then the cost of this clustering is $a_{0,10} + a_{10,N}$. Formulate this problem as a shortest path problem in graphs and give a dynamic programme solution to this. Prove briefly the correctness of your algorithm. What is the running time of your algorithm?

4. We will consider generalizations of the maxflow problem here.

   (a) In addition to the (positive integer) capacities on the edges of a given directed graph $G = (V, E)$, we are also given an integer *demand* d_v for each vertex v (note that d_v can be negative). When $d_v > 0$, the vertex can be considered a 'source', and when it is negative, the vertex can be considered as a 'sink'. The goal is to find a flow, a function on the edges, that apart from satisfying the capacity constraints, ensures that for every vertex v, $f^{in}(v) - f^{out}(v) = d_v$ where $f^{in}(v)$ is the amount of flow into the vertex v, and $f^{out}(v)$ is the amount of flow out of the vertex v. Let us call such a flow a circulation.
   - Show that, if there exists a feasible circulation, $\sum_{v \in V} d_v = 0$.
   - Explain how this problem can be solved using the 'standard' maxflow algorithms, by reducing the problem to the 'standard' maxflow problem.

(b) In addition to the upper bound for the capacity, suppose we have an integer lower bound for the flow across every edge. Assume that we also have an integer demand (as in the previous question) d_v for each vertex. Now the goal is to find a circulation if exists that meets the demand and capacity (both lower and upper) constraints.

Explain how this problem can be solved using the standard maxflow algorithm.