DQE Algorithms Fall 2007
Problem #1

1. An instance of the set-cover problem consists of a finite set X and a family F of subsets of X, such that each element of X belongs to at least one subset in F. A subset of F covers X if its union is equal to the set X. The set-cover problem is to find the cardinality of the smallest subset of F that covers X.

a. (10 pts) present a (reasonable) greedy approximation algorithm for the set-cover problem.
b. (20 pts) give an example instance of the set-cover problem for which the greedy algorithm is optimal, and another example for which it is not.
c. (20 pts) show that the size of the output of a natural greedy algorithm is at most a factor ln(n) greater than the optimal, where n = |X|.

Hint for part c: given any k-size greedy cover S1,S2,...,Sk we can assign costs to each element x as follows: when x is first covered, say by subset Si, we will assign to x the value of the reciprocal of the number of elements of Si not yet covered by the previous sets. First show that the size k of this greedy cover is precisely the sum of all the costs over X. Then show that for any subset Yof X, the sum of the greedy cover costs over Y is at most ln(n). Conclude that the size of the greedy cover can be at most ln(n) times the number of sets in the optimal cover.

DQE Algorithms Fall 2007
Problem #2

2. This problem involves the design of algorithms and data structures for an application involving web-based query processing. Biologists want to create a searchable database called the "Encyclopedia of Life" containing a webpage on all known species. They want the set of URLs for webpages to be indexed by the species Latin names. Also, they would like keyword queries to return the URL links to the top most likely species, even if the query is slightly misspelled. Your solution should consider the following problem scale: there are 2 Million web pages indexed, and each is titled with a unique Latin name averaging 25 characters each. However, the longest species name is about 200 characters. Query speed and index storage space are critical issues for the application.

a. (10pts) explain why the use of hashing is not appropriate for this problem.
b. (20 pts) describe a solution based on binary searching an array. Present an analysis of the time and space complexity.
c. (20 pts) Design your own solution for this problem, including description of appropriate data structures and algorithms. Compare your solution to the one based on a binary search in part c.