Computer Science 409, Spring 2001: Final Review

1. (a) Give one advantage of Prim's algorithm over Kruskal's algorithm.

   (b) Give one advantage of Dijkstra's algorithm for single-source shortest-path problem over the Bellman-Ford algorithm, and one advantage of Bellman-Ford over Dijkstra.

2. Can you always topologically sort a directed graph? If not, explain exactly why you cannot and explain under what circumstances you can topologically sort.

3. Consider a sequence of K MAKESETS, M FINDS, and N UNIONS, not necessarily in that order, where $K = O(N)$. (Assume that UNION is as presented in class, and does not include an implicit FIND. That is, the arguments to UNION are always the representative elements of the sets involved.) We discussed two ways of representing sets in class; for each one, we considered good ways of implementing UNION-FIND. Which representation would you use and what would be the amortized runtime if $M = N$? Which representation would you use and what would be its complexity if $M = N^2$?

4. Consider the special case of Dijkstra's algorithm for graphs where all edge weights are either 1 or 2.

   (a) Argue that no shortest path can have weight more than $2|V|$ in this case.

   (b) What are the special properties of the priority queue data structure that is used in this case of Dijkstra (what are the possible keys)?

   (c) We want to implement Dijkstra to run in $O(E)$ time in this special case. To do this we need a priority queue implementation for the special case that does DECREASE-KEY and EXTRACT-MIN operations in $O(1)$ time, and which can be initialized with all the vertices in $O(V)$ time. Give such an implementation for the special case needed.

5. Explain how BFS(G) can be modified to identify the connected components of a directed graph G. (An English description is OK here.) What is the running time of this modification?

6. Do Problem 23.3 - 9.

7. You have a hash table of size $m = 11$ and hash functions h_1 and h_2

$$h_1(k) = k \bmod m$$

$$h_2(k) = 1 + (k \bmod (m-1)).$$

(a) Show the contents of the hash table after inserting, in order, the keys 10, 22, 32, 4, 15, using chaining and h_1 only.

(b) Show the contents of the hash table after inserting, in order, the keys 10, 22, 32, 4, 15, using **open addressing with double hashing**, and $h_1(k)$ and $h_2(k)$ above, as hash functions.

8. Assume e is the unique least expensive edge in the graph. Explain why e is in **all** minimum spanning trees.

9. Suppose you have an application of priority queues where you know the that there can be only k different priorities, called priorities $1, \ldots, k$. You know that there will be a large number N of items in your priority queue, where N is much bigger than k, so many items will have the same priority.

   (a) Suppose you use a standard heap-based priority queue and put the N elements in this priority queue. What is the worst case running time of `delMax` and `insert` in this implementation?

   (b) Design a data structure for this application that allows the `Insert` and `delMax` operations to run in $O(\log k)$ time. You may use any data structure that we learned about without writing code for it, but be careful in explaining any modification you must make for this problem. Explain clearly how you do `insert` and `delMax`. You may use any data structure operations that we learned about without writing code for them.

10. For each of the following problems either show the problem is NP-complete or describe the best (least-time) polynomial time algorithm that you can find. For problems that are NP-complete, you are welcome to make use of any of the NP-complete problems discussed in class or in the text (e.g., SAT, 3SAT, Vertex-Cover, Clique, Hamiltonian Cycle, Traveling Salesman). For an algorithm, you should describe your algorithm at a very high level, with just enough detail for someone familiar with the course to understand how the algorithm works (i.e., we don't want to see code). You are welcome to make use of any of the algorithms discussed in class or in the text.

    (a) Input: a bipartite graph G and an integer k.

    Question Is there an edge-cover of G of size $\leq k$? (An *edge-cover* is a set of edges such that each vertex in G is the endpoint of some edge in this set.)

    (b) Input: a graph G with n vertices. Question: Does G have a simple cycle of length $n-1$?

    (c) Input: strings S and T over the usual 26-letter alphabet and an integer k

    Question: Is there a way to align the strings so that $\geq k$ characters match? (The order of characters within a string cannot be changed, but gaps are allowed. For example, the strings DOG and DGB can be aligned as DOG# and D#GB to get two matching characters; the symbol # is being used to represent a gap.)

    (d) Input: a bound b and a group of n people with information about which pairs of people are currently not speaking to each other.

Question: Is it possible to choose a subgroup of size b so that each member of this subgroup is willing to speak to all other members of this subgroup?

11. What is a *cut* in a flow network? How are cuts related to flows?

12. Suppose that we have computed a maximum flow f from s to t in the flow network G. Now suppose that G is modified; a new edge is added to G from u to v. The new edge has capacity 3. Call the udpated flow network G'. How would you compute the maximum flow in G' without redoing the algorithm from scratch. How many iterations of the algorithm would you need, in the worst case?

13. Consider the problem of storing n books on shelves in a library. The order of the books is fixed by the cataloging system and so cannot be rearranged. Therefore, we can speak of a book b_i, where $1 \leq i \leq$, that has a thickness t_i and height h_i. The length of each bookshelf at this library is L. Suppose all the books have the same height h (i.e. $h_i = h_j$ for all i, j) and the shelves are all separated by a distance of greater than h, so any book fits on any shelf. The greedy algorithm would fill the first shelf with as many books as we can until we get the smallest i such that h_i does not fit, and then repeat with subsequent shelves. Show that the greedy algorithm always finds the optimal shelf placement, and analyze its time complexity.

14. If $L_1 = \{11\}$ and $Ł_2 = \{00\}$, what is $(L_1L_2)^*$.

15. Let A, B, and C represent decision problems. Suppose we know that $A \leq_P B$ and $B \leq_P C$. Mark each statement below as either True or False, and explain why.

    (a) $B \leq_P A$

    (b) $A \leq_P C$.

    (c) If B is NP-complete and C is in NP then C is NP-complete.

    (d) If B is NP-complete and A is in NP then A is NP-complete.

16. Do 36.4-5. Why doesn't this say that SAT is in PTIME? (This, of course, is the problem I meant to assign rather than assigning 36.4-4 twice.)