Solution to midterm exam 2 December 7, in class

ENEE641/ENME808X, Fall 2016 (72 total points, each question 12 points)

Problem 1 Using n as our base, there will be $\lceil \log_n k \rceil$ digits in the integers, and thus $\lceil \log_n k \rceil$ rounds of count sort are needed.

Problem 2

(a)

(i) If $w'(e) \leq w(e)$, do nothing. Otherwise, remove e from T. This divides the graph into two trees, T_1 and T_2. Perform DFS starting at any vertex, following only edges in T to find all vertices in T_1. Any unvisited vertices are in T_2. Find the lightest edge between a vertex in T_1 and a vertex in T_2, and add this edge to T to make T'.

(ii) If $w'(e) \geq w(e)$, do nothing. Otherwise, add e to T. This will create a cycle in the graph. Starting at u, one of the vertices incident to e, perform a DFS, following only edges in T. When you find the back edge to u, backtrace the path back to the root to find all the edges in the cycle. Remove the lightest of these edges from the T to make T'.

(b)

(i) If an edge in T has its weight decreased, obviously it can remain in T, as no other edge will do better. If e has its weight increased, we must remove it from T because there may be a lighter edge which completes the MST. We search for the light edge which crosses the cut which respects the partition of T into two trees, which will be safe to add to our MST according to theorem 23.1.

(ii) If an edge in T has its weight increased, obviously it cannot be added to T, as it will not do better than existing edge in T. If e has its weight decreased, we add it to T because it may make another (heavier) edge redundant. By adding e, T is no longer a tree but will have one cycle which includes e. Removing any edge in this cycle will make it a tree again, and by removing the heaviest edge, we make it a MST.

(c)

(i) $O(m + n) = O(m)$

(ii) $O(n)$

Problem 3

(a) We search the graph for vertices with in-degree zero. We add all these vertices to a queue. Then we operate in rounds. In each round, we have two queues, one for vertices which have in-degree zero at the start of the round (Q_1), and one for vertices which have in-degree zero by the end of the round (Q_2). In each round, we remove all vertices from Q_1, and examine their outgoing edges to see if any successors now have in-degree zero. If so, we add the successor to Q_2. We report the number of rounds required to remove all vertices as our result. If at any point Q_1 is empty but vertices remain, then there is a cycle in the graph and the curriculum is invalid.

(b) By removing all in-degree zero vertices in every round, we take all possible classes in every semester. Therefore the number of rounds is the minimum number of semesters required.

(c) $O(m + n)$

Problem 4

(a) Given a graph G and an integer k, is there an independent set of k or more vertices in G?

(b)

1. Prove $L \in$ NP The verification token is the list of vertices in the clique V'. Verify in $O(|V|^2)$ time that all the vertices in V are connected by an edge.

2. Choose a known NP-complete language Independent set problem.

3. Describe reduction algorithm Take a graph G that we want to show has at an independent of at least size k. Use for the input to the clique problem the graph's complement, G', and k.

4. Prove correctness of reduction algorithm If there is a clique V' of size k in G, then V' will be an independent set in G', also of size k. Because they were all connected in G, they will not share any edges in G', and so each edge will be incident on at most one vertex in V'. Likewise, if there is an independent set of size k in G', then it will be clique in G of the same size.

5. Prove reduction algorithm runs in polynomial time Creating the complementary graph will take $O(|V|^2)$ time, because you must consider each pair of vertices.

Problem 5

(a) We replace every undirected edge (u, v) in G with a pair of directed edges (u, v) and (v, u). Now, we add a weight to every directed edge, such that:

$$w(u, v) = max \begin{cases} 0 \\ w(v) - w(u) \end{cases}$$

Then use Dijkstra's shortest path algorithm to calculate the path with the smallest total edge weight between u and v.

(b) We give each edge a weight corresponding to the uphill altitude biked over that edge. Thus, by finding the shortest path between u and v over these weights, we find the path which minimizes total altitude gained.

(c) $O((m + n) \log n)$

Problem 6

Chaining Maintain a linked list for each slot in the hash table. Add colliding entries to the linked list. An advantage of this is that expected lookup time will be $O(1 + n/m)$ where n is the number of entries and m is the number of hash slots, whereas in open addressing expected lookup time tends toward infinity as n approaches m. Related, chaining can store more than m elements, whereas open addressing cannot.

Open Addressing Entries occupy the hash table slot itself. If there is a collision, a secondary probing function tells you which slots to use. An advantage is that it requires less memory if the entries are small, possibly allowing a larger table and therefore have few collisions.