Midterm 2: Sample solutions and comments

1. The TRUE/FALSE questions did not require any explanation

   (a) TRUE. Interpolation search was the method discussed in Assignment 5.

   (b) FALSE. Johnson's algorithm uses edge re-weighting to remove negative weight edges, not produce small integer weights.

   (c) TRUE. As discussed, after the i-th phase of Bellman-Ford the exact shortest path values have been determined for all nodes whose link-distance from the source in the shortest path tree is at most i.

   (d) FALSE. The node at the root of an optimal binary search tree might not split the probabilities of the remain nodes at all evenly. An example of this appeared in Assignment 7 Q.2.

   (e) TRUE. This was discussed in class, based on both Euclidean distance estimates and "landmark"-based estimates.

2. (a) This comes directly from Assignment 6, Q1 (c) and (d).

   (b) The key thing here is to recognize, just as INSERT can be viewed as being analogous to incrementing a binary counter, UNION can be viewed as being analogous to adding two binary counters. We proceed from low order bits to high order bits (small blocks to large blocks), doing bit-wise addition (merging of corresponding blocks) with a carry in the case where both bits are 1 (both blocks are non-null). Since merging takes linear time, it takes $O(2^i)$ time to process all if the i smallest size blocks.

3. Since *amortized_cost* = *actual_cost* - *drop_in_potential*, it must be the case that when *amortized_cost* < *actual_cost* , the *drop_in_potential* is positive. Since potential can be viewed as a measure of the balance of the associated tree, when all keys are given the same weight, and lower potential corresponds to increased balance, it follows that "whenever the actual cost of splaying exceeds the amortized cost, the splay tree must have become more balanced as a result."

4. (a) The hint suggested the following adversary strategy: Respond to queries of the form "is $A[i, j] = 1$?" (equivalently, " is the edge (i, j) in the graph G?", by YES, if $i < j$, and NO otherwise. If the algorithm

stops making queries before all of the entries $A[i,j]$ with $i \geq j$ then it can not distinguish between the case where the un-probed such entries are all 0 (corresponding to an acyclic input graph), and the case where at least one of the un-probed such entries is 1 (corresponding to a cyclic input graph).

(b) Suppose that G has a directed cycle. Then *for any assignment of distinct labels* to the vertices of G there must be at least one edge in the cycle that has a smaller label on its head than on its tail. Thus the labeling does not constitute a linearization of G. Since this is true for all legal label assignments, this implies that G is not linearizable.

(c) Suppose that every vertex of G has at least one in-coming edge. Then consider a walk in G that starts at any vertex and continues in the *backward* direction from each vertex that it encounters. Since every vertex has an in-coming edge this walk need never terminate. Thus eventually some vertex v must be revisited. But the walk from the first visit to v to its second visit must be a (reverse) cycle.

(d) The invariant holds, by construction, before the loop starts (i.e. when $X = V$ and $i = 1$). Assuming that it holds when $i = i_0$, it suffices to show that it holds when $i = i_0 + 1$. If we denote by v_{i_0} the vertex extracted in the i_0-th iteration then, for all $w \in X - \{v_{i_0}\}$, either
(i) $(v_{i_0}, w) \in E$, in which case key$[w]$ has decreased by 1, as has $\mathrm{d}^{\mathrm{in}}_X(w)$, or
(ii) $(v_{i_0}, w) \notin E$, in which case key$[w]$ has not changed, as has $\mathrm{d}^{\mathrm{in}}_X(w)$.

(e) On termination $i > |V|$ and so, by invariant (i), label$[u]$ < label$[v]$, for all $(u,v) \in E$. The uniqueness of the labeling is a consequence of the fact that the label i is assigned exactly once – during the i-th iteration of the while loop.

(f) Keep one list L_j for all vertices v with $\mathrm{d}^{\mathrm{in}}_X(v) = j$. By the claim, EXTRACT_MIN always removes a vertex from list L_0. DECREMENT_KEY moves a vertex from its current list L_j to list L_{j-1}. It follows that both EXTRACT_MIN and DECREMENT_KEY have cost $O(1)$.
Since EXTRACT_MIN is performed $|V|$ times and DECREMENT_KEY is performed at most once per edge, the total cost is $O(V + E)$.

(g) To start $i = 1$ and so the invariant holds trivially. Suppose that it holds when $i = i_0$. It suffices to show that following the i_0-th iteration $\mathrm{d}[v_{i_0}] = \delta(s, v_{i_0})$. Suppose that the shortest path from s to v_{i_0} (if such a path exists) ends with the edge (v_j, v_{i_0}). By the linearization property, we know that $j < i_0$ and hence, by the induction hypothesis, $\mathrm{d}[v_j] = \delta(s, v_j)$ before the start of the i_0-th iteration. But during the i_0-th iteration we assign $\mathrm{d}[v_{i_0}] \leftarrow \min\{\mathrm{d}[v_{i_0}],\ \mathrm{d}[v_j] + wt(v_j, v_{i_0})\}$. Thus, $\mathrm{d}[v_{i_0}] \leq \mathrm{d}[v_j] + wt(v_j, v_{i_0})\} = \delta(s, v_{i_0})$.

(h) Simply change all of the edge weights from $wt(u,v)$ to $-wt(u,v)$. Since there are no directed cycles in G, this cannot introduce any negative

cycles and so shortest paths remain well-defined. But any path in the re-weighted graph of weight W corresponds to a path of weight $-W$ in the original graph, and vice-versa. Thus, a shortest path in the re-weighted graph must correspond to a longest path in the original.