Quiz 2 Solutions

Problem 1. True or False [20 points] (4 parts)

For each of the following questions, circle either T (True) or F (False). **Explain your choice.** (No credit if no explanation given.)

(a) T F Topological sort requires $\Omega(V \lg V)$ if the edge weights are unbounded.
Explain:

Solution: False. This statement is almost complete nonsense; topological sort has nothing to do with weights. It takes $\Theta(V + E)$ time on any (unweighted) DAG.

(b) T F A set of n integers whose values are in the range $[0, n^8)$ can be sorted in $O(n)$ time.
Explain:

Solution: True. Use radix sort with a radix of size n. Then each invocation of counting sort takes $O(n + n) = O(n)$ time. Each element has 8 "digits", so the total time for radix sort is $O(8n) = O(n)$.

(c) T F If a depth-first analysis of a graph contains at least one back edge, any other depth-first analysis of the same graph will also contain at least one back edge.
Explain:

Solution: True. This follows from the fact that an analysis contains a back edge if and only if the graph contains a cycle.

(d) T F Any DFS forest of an undirected graph contains the same number of trees.
Explain:

Solution: True. In an undirected graph, each connected component of the graph will be a single tree in a DFS.

Problem 2. Short Answer [32 points] (4 parts)

(a) In the graph of 2x2x2 Rubik's cube positions (as in Problem Set 4), there are exactly 6 edges incident on each vertex. We want to use a bi-directional BFS to find a shortest path between two vertices s and t. If the shortest path from s to t contains d edges, approximately how many vertices must be explored to find a shortest path, in the worst case? How does this compare to the number of vertices required for a single BFS starting from s?

Solution: Each vertex has 5 children (the sixth edge leads to its predecessor). Exploring n levels deep then requires visiting about 5^n vertices.

We must explore $d/2$ levels deep from each side, so we visit approximately $2 \cdot 5^{d/2} + 1$ vertices. This is approximately the square root of the number required for a single BFS.

(b) Each edge in a connected, unweighted graph G is colored either red or blue. Present an algorithm to compute a path between s and t that traverses the fewest number of red edges. Analyze its running time.

Solution: Assign each blue edge a weight of 0, and each red edge a weight of 1. Then run Dijkstra's algorithm, for a total time of either $O((V + E) \lg V)$ or $O(E + V \lg V)$. This solution received full credit.

It is possible to achieve $O(V + E)$, if you're really clever. Observing that all shortest paths have length of at most $V - 1$, you can use an idea from counting sort to implement a priority queue that gives $O(1)$ DECREASE-KEY and amortized $O(1)$ EXTRACT-MIN ($O(V)$ total over V operations). But we didn't really expect anyone to come up with this solution under quiz conditions.

(c) An efficiency-crazed (complexity minded) algorithms developer tries to reduce the amount of work done in the Bellman-Ford algorithm. Her modified Bellman-Ford first runs a modified BFS starting from s that keeps track of the BFS step $d(e)$ in which it discovers each edge e. Thus, $1 \leq d(e) < |V|$. In the Bellman-Ford algorithm itself, instead of relaxing *all* the edges in every iteration, at iteration i (i starts at 1 and goes up to $|V| - 1$), she only relaxes the edges with $d(e) \leq i$. Is this be guaranteed to give the shortest distance from the s to all the vertices or detect a negative cycle? Explain why or provide a small counterexample.

Solution: The intended answer for this problem is that the algorithm does work. The reason is because BFS ensures that we are relaxing edges on the shortest path from s to every other vertex *in order*. That is, in iteration i, we will relax the i-th edge along the shortest path, so by the $|V| - 1$-th iteration, all the edges on the shortest paths will be relaxed. The path relaxation theorem then tells us that the the distance labels of the vertices on these paths will be the shortest distance.

Unfortunately, due to the problem's wording, it could also be interpreted that the cycle-check iteration is *not* run. This leads to a failure in finding negative cycles, so students who pointed out this specific issue also got credit for the problem.

A common mistake for this problem was to just say that edges with $d(e)$ greater than i cannot be relaxed. While this is a true observation of the modified algorithm, this alone does not prove that the algorithm terminates with the correct results. Also note that this observation is not true for the original Bellman-Ford algorithm. It is possible to relax all the edges in the graph in a single iteration.

(d) Describe a strategy for analyzing a directed acyclic graph using depth first search, choosing the search order so that *all* the edges in the analysis will be cross edges. (In other words, the search will produce *no* tree edges, back edges, or forward edges.)

Solution: Search the nodes in reverse order by topological sort. This guarantees that every edge traversed will point to a node that is already finished.

Problem 3. A Vacation Home in Hawaii [24 points]

Beverly owns a vacation home in Hawaii and wishes to rent the place out for n days beginning on May 1st (on the $n + 1$st day, she plans to take a vacation there herself). She has obtained m bids, each of which has a starting day s_i and ending day e_i (between 1 and n), and the amount $\$_i$ that the bidder is willing to pay. Beverly can only rent the house to a single bidder on any given day. (That is, she may not accept two bids b_i and b_j such that the intervals (s_i, e_i) and (s_j, e_j) overlap.)

Beverly decides to model this problem as a directed graph with weighted edges so that she can use a standard graph algorithm (or a minor variation of a standard algorithm) to find the bids to accept which maximizes her revenue.

Describe such a model, and give the asymptotic cost of finding the set of bids that maximizes the revenues. Assume that the bids are given as list of tuples, not necessarily sorted in any particular order. Try to find an algorithm that is as efficient as possible.

Solution: There are several solutions. More efficient solutions received more points.

Create one vertex for each of the n days. A bid (s_i, e_i) is an edges from day s_i to day e_i, with weight equal to $\$_i$. We also need zero-weight edges from each day i to day $i + 1$, to cover no-occupancy periods (without them, the graph might not even be connected!).

Now find the longest path in this graph. The best way to do this is to negate each edge and run the DAG shortest path algorithm. Because there are n vertices and $O(m + n)$ edges, this requires $O(V + E) = O(m + n)$ time. The time to construct the graph is also $O(V + E)$, so the total time required is $O(m + n)$.

Problem 4. Clusters [24 points]

At Podunk State University, there are 100 courses, and n students. The number of students n varies from term to term and is virtually unbounded because Podunk State is required by law to admit all students who apply for admission (and even a few who don't).

Since there are few courses and many students, it is common to find groups of students who are all taking the same set of courses at the same time. In order to optimize its available resources in light of the current economic crisis, the administration wants to know how many clusters there are, where a *cluster* is defined as a maximal set of students who are all registered for the same set of courses.

The administration hires you to solve this problem. You are given a table that lists all n students and the courses each is taking. Design an efficient algorithm to determine the number of clusters. Analyze its running time as a function of n.

Solution: The problem can be solved optimally by associating a 100-bit vector with each student, where the ith bit is 1 if the student is taking the ith course and 0 if not (the courses are listed in some consistent order). Thus, two students are in the same cluster if they have identical vectors. Constructing these vectors is easy, given the lists of courses each student is taking. After determining all n vectors, apply radix sort to the list of vectors. This is possible because the vectors can be treated as base-2 integers with at most 100 digits. The running time of radix sort is $O(d(n+k)) = O(100(n+2)) = O(n)$. After running radix sort, group consecutive vectors into the same cluster if they are identical. This takes $O(n)$ time, for a total running time of $O(n)$.

Counting sort achieves the same asymptotic runtime, but the constant factors are so much larger as to be impractical (it requires declaring an array of size 2^{100}).

Problem 5. Cap-tain Jean-Luc Pi-card, U-S-S En-ter-prise [20 points] (3 parts)

The Enterprise is in orbit around Starbase 6006. An urgent medical situation requires that it travel as quickly as possible to the distant planet *Vertex T*. In fact, the situation is so urgent that the Federation requests that the Enterprise arrive *yesterday*. Your goal is to determine if it is possible to get the Enterprise from s to t in negative total time, so that it arrives before it leaves.

You have a directed graph G representing the Universe. Each known location in Federation space is a vertex in the graph. There are a number of hyperspace bypasses between these points; a hyperspace bypass from point u to point v is represented by the edge (u, v). The weight of the edge $w(u, v)$ is equal to the amount of time that passes while traversing it.

Notice that edges may have negative weights! Some bypasses run through some sort of crazy time-warp thing, causing the ship to travel back in time.

(a) [2 points] Ensign Dijkstra volunteers to lay in the course. In one sentence, explain why Dijkstra's algorithm can't be used to solve this problem.

Solution: Dijkstra's algorithm cannot be used on graphs that contain negative edge weights.

(b) [2 points] You ask Ensign Bitdiddle to plot the course instead. Supposing that G contains no negative-weight cycles, describe an algorithm that returns TRUE if there exists a path from s to t with negative total weight, and FALSE otherwise.

Analyze the running time of your algorithm.

Solution: Run Bellman-Ford starting from s in order to determine the weight of the shortest path from s to t. This weight is negative if and only if there exists a path with negative total weight.

(c) [16 points] Lieutenant Data tells you that there is exactly one negative-weight cycle present in G. Describe an algorithm that returns TRUE if there exists a path from s to t with negative total weight, and FALSE otherwise. (You may assume that you know which vertices are in the cycle.)

Analyze the running time of your algorithm.

Solution: First, determining whether the cycle is reachable from s. This can be done either by running either BFS or Bellman-Ford, starting at s. Then, determine whether t is reachable from the cycle. This can be done by reversing every edge in the graph, and proceeding as for s.

If the cycle is reachable from s, and t is reachable from the cycle, then return TRUE—simply go from s to the cycle, then go around the cycle as many times as needed, then go from the cycle to t.

If the cycle is *not* reachable from s, or t is *not* reachable from the cycle, then *no* path from s to t can contain any edge in the cycle. Simply remove the cycle from the graph, and solve the problem as in part (b).

(d) [0 points] (This part is optional.) Ensign Treaps, in his bright red shirt, suggests you use a treap (from Quiz 1) to solve this problem. Fortunately, he is destined to die on arrival at Vertex T. Describe his glorious death.

Solution: There were a lot of entertaining answers (and accompanying illustrations) to this part. Answers of any kind were rewarded with arbitrary and large numbers of bonus points. Unfortunately, these bonus points don't count towards any grade, and have no cash value.

The most popular answer was that Ensign Treaps is forced to take Quiz 1, whereupon he either dies from the trauma, or kills himself. The next-most popular answer involved being assaulted by a mob of 6.006 students. The third-most frequently given answer was "velociraptors!", or some variant.