Homework 5 Solutions

Maxwell Young

December 5, 2003

Question 22.2-6

Create a graph where every wrestler is represented by a vertex and rivalries are represented as edges between the corresponding wrestlers. Start with all the nodes initially coloured grey.Then run a modified BFS on this graph which colours the parent red, the children green, the children's children red, and so on. If at any point, this modified BFS changes a red node to a green node (or vice versa), then return FALSE (since a designation of good guys and bad guys is not possible). Else, return true when the algorithm halts. Then to find the actual designation, we can just run through the graph again and record the coloring. All of this can be done in $O(n + r) = O(|V| + |E|)$ time since that is the running time of BFS.

Question 22.2-8

There were two main answers that people gave me; I gave marks to both. However, the easiest way to solve this problem is to run a DFS on the graph. We are then guaranteed to cover each edge exactly once in each direction. If you wish to be picky, you could have said something about using print out statements in order to illustrate the path.

The other way is to simply use the normal BFS which will create a BFS tree in the graph. But we have to deal with the non-tree edges. We can simply traverse the graph a second time (with BFS say) and everytime we meet a node v with a non-tree edge, we simply traverse that edge in both directions coming back to the original node v, then continue on. Again, you could use print out statements on your second traversal through the graph in order to get the actual path you are taking.

Both of these algorithms run in $O(|V| + |E|)$ time.

To use pennies to get out of the maze, we can simply place pennies heads up as we walk down a path. If we ever find ourselves travelling down a path with pennies heads up, we should flip the pennies over as we go to show the tails. Faced with a choice between different paths, always choose the leftmost path without pennies. Such an algorithm will always get you out of the maze eventually.

Question 24.1-1 and Question 24.3-1

See the links on my website just below the one you used to get to this page.

Question 5

Base Case: Let us consider a path from s with zero edges (ie. an isolated vertex). Depending upon how you wish to define things, $w(s \rightsquigarrow s) = \infty$ or 0. In either case, we have $0 = dist(s) \leq w(s \rightsquigarrow s = \infty$ or 0.

Inductive Hypothesis: for paths, $s \rightsquigarrow v$, of length $j < l$, assume that $dist(v) \leq w(s \rightsquigarrow v)$ when the algorithm halts.

Inductive Step: Consider a path $s \rightsquigarrow v'$ of length l. Also, say that u lies just prior to v' along this path (ie. u is connected to v' just before it). After relaxation, we have:

$dist(v') = dist(u) + w(u, v')$
$\leq w(s \rightsquigarrow u) + w(u, v')$ by the induction hypothesis
$= w(s \rightsquigarrow v')$ as desired.

Since the base case and induction steps hold, we have proved our result.

Question 24.3-6

The answer to this question shares much in common with the question from the previous HW involving Prim's algorithm and ways of speeding it up when all the edge weights in the graph are bouned between 1 and W.

Here we use an array, indexed 0 to W, of linked lists as our data structure. The linked list at index $i < W$ holds the vertices with distance i from the source vertex. We store the vertices not reachable from the source vertex. Now EXTRACT-MIN will only require $O(1)$ time to perform, same with edge relaxation. Hence, the algorithm now runs in $O(|V|W + |E|)$ time as desired.