Introduction to Algorithms: 6.006
Massachusetts Institute of Technology
Erik Demaine, Ronald L. Rivest, and Nir Shavit

November 5, 2013
Handout # 5

Problem Set 5, Problem 2

Kevin Peng
Collaborators: Genghis Chau, Jodie Chen, Anubhav Jain

This problem set is due **Thursday, November 21** at **11:59PM**.

This solution template should be turned in through our submission site. [1]

For written questions, full credit will be given only to correct solutions that are described clearly *and concisely*.

---

[1] Register an account, if you haven't done so. Then go to Homework, Problem Set 5, and upload your files.

Problem 5-2. [35 points] **Good and Bad Graphs**

(a) [5 points]

We note that graph G' has two times the number of vertices as G and two times the number of edges as V'. G' is effectively a bipartite graph with one set of vertices that are good and one set of vertices that are bad. We run the Bellman-Ford algorithm using the good variant of *Start* as the source and the bad variant of *Start* as the source to get the shortest path from either source to the good or bad variant of any vertex v. Each of these will take $O(2V\dot{2}E)$ time, so both of these together will take $O(8VE) = O(VE)$ time.

(b) [10 points]

We use BFS on G'. We will use each vertex as a source of a BFS. Each time a node is reached through BFS, we will update in our dictionary whether it is reachable in an odd number of steps or an even number of steps. (If a good u reaches a good v or a bad u reaches a bad v through BFS, then there exists an even length path. Otherwise, if a good u reaches a bad v or a bad u reaches a good v, there exist an odd length path.) We repeat this until we have visited all nodes in the graph, for which BFS takes $O(V + E)$ time. Finally, we can go through all pairs of vertices and make sure that they are in the dictionary with the correct information (for example, we need to make sure that pairs of vertices that didn't have paths between them are in the dictionary and have property 1 as true). Since we perform this on every vertex, this takes $O(V^2 + VE)$ time. The extra check on the dictionary takes $O(V^2)$ time. We know that $V \leq E$ since the graph is strongly connected, so we get that this takes $O(VE)$ time.

(c) [10 points]

The negative-weight cycle in G' must have an even length. A path with an odd length is automatically not a cycle because each edge in G' connects two nodes of opposite kind, so the ending node would not be the same parity as the starting node.

We can run Bellman-Ford on this graph and check for the existence of a negative cycle in $O(V'E')$ time, which is $O(4VE)$ time, which is $O(VE)$ time. When running Bellman-Ford, we also generate a corresponding predecessor subgraph. Bellman-Ford will not produce a cycle in the predecessor subgraph if there are no negative-weight cycles (the predecessor subgraph would be a shortest paths tree if there were no negative-weight cycles). The negative-weight cycle corresponds directly to a cycle in the predecessor subgraph. Thus, from the destination node, we backtrack along the predecessor subgraph until we find a node that is repeated twice, and output that node once we find it. That node is on the negative weight cycle of G'.

(d) [10 points]

This problem is synonymous with finding a negative-weight cycle along the path from the good Start to good End. To do this, we go through the list of vertices that we know are on negative-weight cycles and for each vertex, we see if there is a path from good Start to that vertex that is the same parity as a path from good End to that vertex. If there is, then there is a negative cycle, so there is an arbitrary large negative-cost path from good Start to good End. Each lookup in STATEMENTS takes $O(1)$ time and we do at most $O(V)$ lookups, so the total time for this algorithm is $O(V)$.