Problem Set 4

All parts are due April 25, 2017 at 11:59PM.

Name: Faaya Abate Fulas

Collaborators: Ebenezer Nkwate

Part A

Problem 4-1.

-Topologically sort the DAG.

-Assign each edge in the DAG a weight of -1 and run the Bellman-Ford algorithm where the relaxations are performed according to the topologically sorted ordering of the nodes to find the longest path between all pairs of nodes in $\emptyset(V + E)$ time. Since a uniform weight of -1 is applied to all the edges in the graph, the longest path between two nodes must also traverse the most number of nodes.

-Insert an edge between the two furthest nodes in the graph. The edge will be directed from the right most node to the left most node in the topologically sorted order. This will form a cycle containing the most number of nodes maximizing the size of the largest SCC in the graph

Problem 4-2.

-Make T copies of all the nodes in G for all the possible times at which Judy arrives at the nodes, $u_0, u_1, ..u_{T-1}$

For some node u_i,

-If $i > r_u$, the cost of an edge from u_i to its neighbor, v_j , is $d_i w_i$. In this case, she arrives at the next node v_j at time, $i + w_i$ where the traffic light timer is in state $(i + w_i) mod T$. Therefore, draw an edge with weight $d_i w_i$ from u_i to v_j where $j = (i + w_i) mod T$

-Else, the cost of the edge from u_i to its neighbor,v_j, is $s_u(r_u - i) + d_i w_i$. In this case, she arrives at the next node v_j at time, $(r_u - i) + i + w_i$ where the traffic light timer is in state $((r_u - i) + i + w_i) mod T$. Therefore, draw an edge from u_i to v_j with weight $s_u(r_u - i) + d_i w_i$ where $j = ((r_u - i) + i + w_i) mod T$

-For all the possible states of destination node, t_i, draw an edge to a single destination node, t', with an edge weight of 0.

-The new graph, G', will have $nT + 1$ nodes and $(m + 1)T$ edges. Run Dijkstra on G' from s_0 to t' in $O(mT + nTl\log(nT))$. The resulting path will use the minimum amount of fuel to get from s to t.

Problem 4-3.

(a)
- Modify the Floyd-Warshall algorithm by keeping track of the maximum probability of failure encountered when traversing an edge in the graph so far. The path through the $k + 1^{th}$ node is taken if the edge with the maximum probability of failure in the path containing node $k + 1$ has a smaller probability of failure than the p_i of the edge with the maximum probability of failure in the current path.
- Since this is just a modified Floyd-Warshall, it will run in $O(n^3)$

(b)
- The probability of failure for a path between u and v is the product of the probability of failure of the edges in the path since the probabilities are independent. The $\log(1 - p_i)$ sum of the probabilities of the edges is equal to the $\log$ of the total probability of success for the path. Since $1 - p_i < 0$, for $1 - p_i > 1 - p_j$, $\log(1 - p_i) < \log(1 - p_j)$. Therefore, minimizing the $\log(p_i)$ sum will maximize the probability of success and minimize the probability of failure.
- Since $\log(1 - p_i)$ is a negative number, to find the path with the maximum probability of success between any two nodes in the network, a weight of $w' = -\log(1-p_i)$ is assigned to the edges in the graph.Then, running Floyd Warshall's algorithm on the graph with the modified edge weights will return the paths with the highest probability of success in $O(n^3)$ time.

Problem 4-4.

-Traverse the graph level by level, visiting each node only once. After all the salaries of the the current node's neighbors are known, update the current node's salary to the max(s+w for all its neighbors, current salary).

-After every node has been visited, add all the current salaries of the nodes to a counter, initialized to 0, that keeps track of every employees salary per month.

-Do this twelve times for every month, and return the final value of the counter to find the total salary paid to employees after a year in $O(m + n)$ time.

Part B

Problem 4-5. Submit your implementation on `alg.csail.mit.edu`.