**COS 226 – Data Structures and Algorithms**

**Fall 2014 – Flipped Lecture Section**

**Group worksheet**

**Week 9 – 11.13.14**

**Topics covered: shortest path, maxflow-mincut**

**Solution**

**Instructions:** This worksheet covers shortest path algorithms and flow diagrams. Answer questions as a group (3-4 students)

1. **Fattest Path** : Given an edge-weighted digraph and two vertices s and t, design an algorithm to find a fattest path from s to t. The bottleneck capacity of a path is the minimum weight of an edge on the path. A fattest path is a path such that no other path has a higher bottleneck capacity.

**First create a pathExists(T) algorithm that determines whether or not a path exists of fatness T. We can do this by simply removing all edges of weight less than T, then running BFS from s to t, taking E+V time. Given this routine, we then need to simply perform a binary search on our edge weight values. Sorting the edge weights is time ElogE, and running log(E) pathExists is also ElogE.**

1. **Modified Dijkstra’s algorithm (fin-f13)**

The standard version of Dijkstra’s algorithm does not consider a vertex once it is removed from the minPQ. However a modified version of Dijkstra’s algorithm that may reconsider a vertex (even after removing from the minPQ) is given below.

private void relax(Graph G, vertex v)

for (Edge e: G.adj(v)) {

w = e.to();

if (distTo[w] > distTo[v] = e.weight()) {

distTo[w] = distTo[v] + e.weight();

edgeTo[w] = e;

if (pq.contains(w)) pq.change(w, distTo[w])

else pq.insert(w, distTo[w]);

}

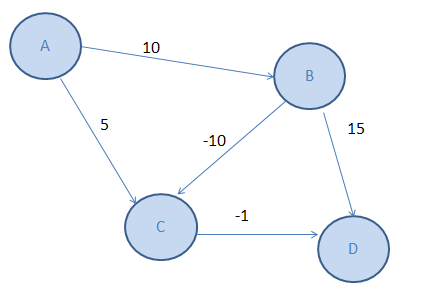
}

}

Will this code work even if there are negative edges in the graph? What is the complexity of the algorithm?

Yes, the standard Dijkstra’s version (based on the correctness proof) assumes that once a node is removed from the PQ, its min distance is determined and will not be considered again. In this version of Dijkstra’s, the code works even if there is a negative edge (not a negative cycle). The main code calls the relax() method as shown below.

insert(s, minPQ); /\* insert source into min PQ \*/

while (!minPQ.empty())

relax (G, minPQ.delMin());

Consider the example below. Apply the Dijkstra’s algorithm using the version above.

Put A to PQ. Delete A from PQ. set A=0. Relax C=5, B=10.

Choose C and delete C from PQ. Relax D to 4. So we have in minPQ: B=10, D=4

Choose D and delete D from PQ. No relaxations. So we have in minPQ: B=10

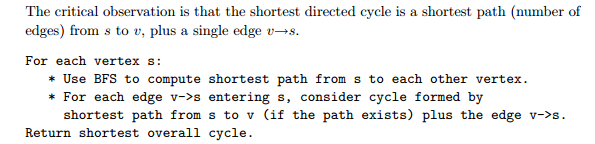
Choose B and delete B from PQ. No relax of D. But there is a C distance can now be set to 0. And C is added back to PQ.

Hence a vertex can go back to PQ many times. but no more than E times. hence in the worst case the code runs EV

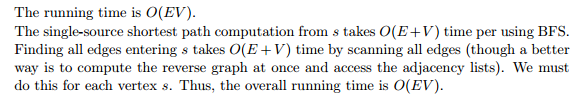
1. **Shortest directed cycle (fin-f08)**

Given a directed graph with V vertices and E edges, design an efficient algorithm to find a directed cycle with the minimum number of edges (or report that the graph is acyclic).

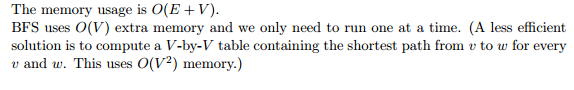
1. Describe your algorithm in the space below



1. What is the order of growth of the worst-case running time of your algorithm?



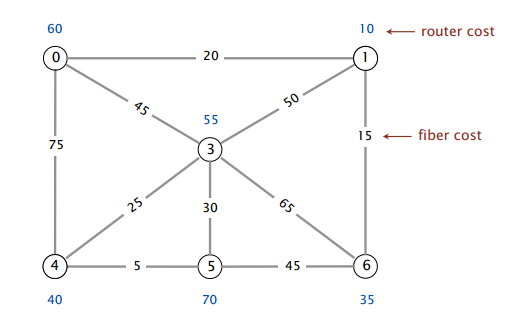
1. What is the order of growth of the memory usage of your algorithm?



1. **Algorithm Design [fin-s14]**

There are N dorm rooms, each of which needs a secure internet connection. It costs wi > 0 dollars to install a secure router in dorm room i and it costs cij > 0 dollars to build a secure fiber connection between rooms i and j. A dorm room receives a secure internet connection if either there is a router installed there or there is some path of

fiber connections between the dorm room and a dorm room with an installed router. The goal is to determine in which dorm rooms to install the secure routers and which pairs of dorm rooms to connect with fiber so as to minimize the total cost.



Formulate the problem as a minimum spanning tree problem. To demonstrate your formulation, modify the figure above to show the MST problem that you would solve to find the minimum cost set of routers and fiber connections.

Solution

