Assignment 20

Alex Clemmer
Student number: u0458675

1

Cycles that have net negative weight screw up algorithms because you can indefinitely make the cost of your path smaller. Making all paths positive, however, changes your best path: the reason is that if you have negative weights, you can take some paths and they actually bring your total down, where if they're all positive, you wouldn't take these paths at all, because all positive weight *contributes* to your total path length.

2

Suppose you have a graph with two paths; one of them is composed of 2 edges with a weight of 10, and one with a weight of -30. Now suppose the other path has, say, 12 edges, each with a weight of -1.

Increasing the weight of all edges by w, in this case, clearly gives all 12 edges in the second branch a weight of 29 each, compared to the 2 edges in the first branch which will have a weight of 40 each. This *clearly* fails to find the shortest acyclic path.

3

Suppose you have a graph with two negative cycles; start as one cycle, then branch into two cycles, then merge back into the same cycle again. Suppose also that these two branches have different negative weights. Now suppose that at the point of merge, you have an exit. In this case, removing the back edge will cause one of the paths to no longer lead to the sink. If this is excluded path is the more efficient path, then you have just ruined your result.