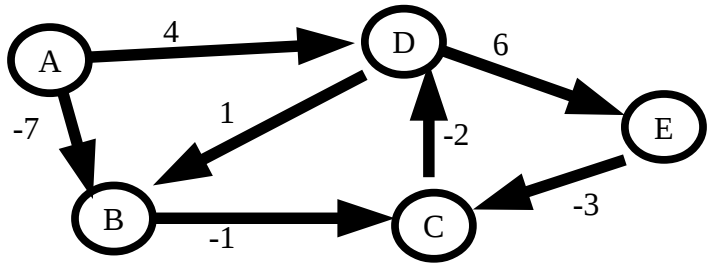
UNIVERSITY OF MINNESOTA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
4041H: HONORS ALGORITHMS AND DATA STRUCTURES
FALL 2015

"LAST MIDTERM" EXAM (180 points):

This is a open book, open notes (including e-book and e-notes) exam. You may not use the internet or get assistance from others. You have 75 minutes to complete the exam.

Problem 1. (30 points)
Illustrate Dijkstra on the following graph starting from node A (the first) and explain the problem that occurs:

Adjacency matrix = $\begin{bmatrix} 0 & -7 & \infty & 4 & \infty \\ \infty & 0 & -1 & \infty & \infty \\ \infty & \infty & 0 & -2 & \infty \\ \infty & 1 & \infty & 0 & 6 \\ \infty & \infty & -3 & \infty & 0 \end{bmatrix}$



Problem 2. (40 points)
Design an algorithm to determine whether n given points in the plane are all on one line. What is the complexity of your algorithm? Write some pseudocode and a short description of your approach. Finally, show the different steps of your algorithm on a simple example.

Problem 3. (30 points)
A maximal spanning tree of a connected weighted undirected graph is a spanning tree with the largest possible weight. Devise an algorithm similar to Prim's algorithm for constructing a maximal spanning tree of a connected weighted graph. Write some pseudocode and show a small example.

Problem 4. (40 points)
Suppose we have found the maximal flow through a network and have a solution. Then one of the capacities on a single edge is decreased (i.e. road construction). Describe an algorithm to re-compute a new maximal flow from the previous now-invalid solution (you may assume that the old solution is indeed invalid). What is the running time of this algorithm? Describe a way to increase the efficiency (not necessarily the asymptotic behavior) of your algorithm by storing additional information when first computing the optimal flow (you do not need to maintain these data structures on the re-computed graph). This additional storage must not increase the asymptotic running time of the original algorithm.

(Problem 5 on back!)

Problem 5. (40 points)
A way to solve many mazes is to simply place your right hand on a wall and always taking right hand turns at intersections (similar solution for lefties). What algorithm is this in disguise? One problem in real hedge/corn mazes is the difficulty recognizing whether you have seen an intersection before or not (due to the similarity in visual queues and difference in perspective). Clarify under what conditions this is problematic for the algorithm (i.e. when will this algorithm fail if you cannot remember/correlate any past information) and illustrate it with a small example.

[Hint: Draw an example and see the behavior if you are having trouble identifying the algorithm.]