Questions and Exercises to work out and turn in:

Grading Guidelines:

* A right answer will get full credit when:

1. It is right (worth 25%)
2. It is right **AND** neatly presented making it easy and pleasant to read. (worth an **extra** 15%)
3. There is an **obvious and clear link** between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth an **extra** 60%).
4. Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.

You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, **personal** writing is expected.

* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **DO NOT DELETE ANYTHING FROM THIS FILE:** JUST **INSERT** YOUR ANSWERS.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), **USE THIS FILE** BY CREATING SUFFICIENT SPACE AND WRITE IN YOUR ANSWERS.

FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST **A 30% PENALTY.**

Objectives of this assignment:

* to use and manipulate the concepts presented in this module
* to propose and write algorithms in pseudocode
* to analyze the time complexity of algorithms
* to analyze the space complexity of algorithms
* to learn autonomously new concepts

What you need to do:

Answer the questions and/or solve the exercises described below.

Exercise 1 (25 points)

1. Run the Bellman-Ford algorithm on the directed graph of Figure 1.1, using vertex x as the source. In each pass, relax edges in the order provided on Figure 1.2, and show the ***d*** and values after each pass.
2. Change the weight of edge (z,x) to 4 and run the algorithm again, using s as the source. In **each** pass, relax edges in the order provided on Figure 1.2, and show the **d** and values after **each** pass.

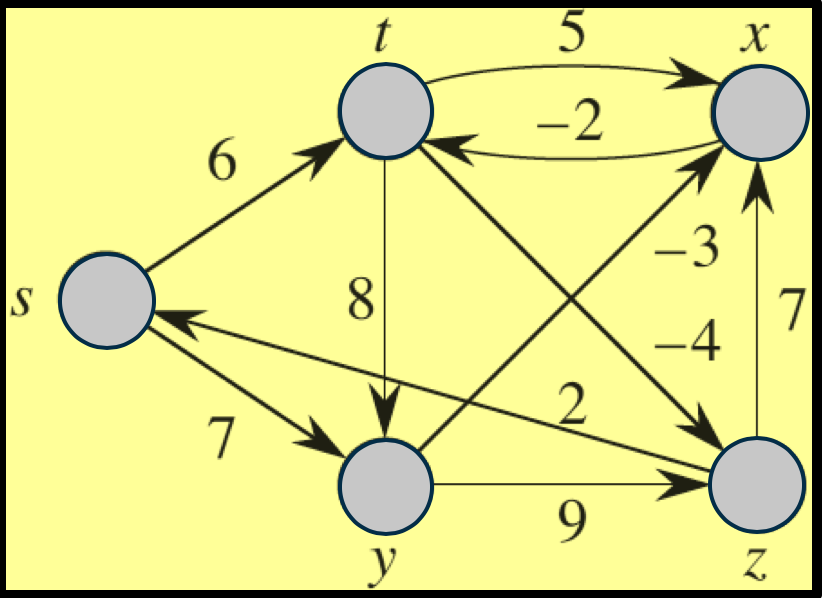


Figure 1.1 Graph

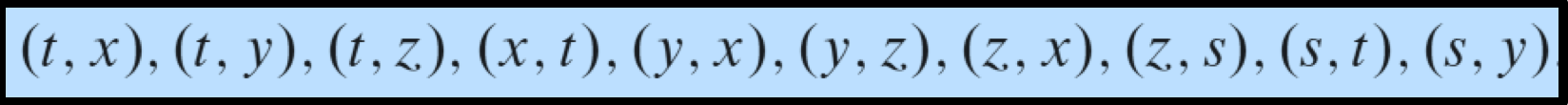
****

Figure 1.2 Edges Order

Exercise 2 (25 points)

Given a weighted directed graph G = (V,E) with no negative-weight cycles, let ***m*** be the maximum number of edges from the source s to any vertex . (Here, the shortest path is by weight, not the number of edges.) Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in ***m + 1*** passes, even if ***m*** is not known in advance. **Prove** that your modified algorithm will work.

BELLMAN-FORD-MODIFIED(G, w, s)

1. INITIALIZE-SINGLE-SOURCE(G, s)
2. changes = 1
3. while changes > 0
4. changes = 0
5. for each edge (u, v) in G.E
6. vdFirst = v.d
7. RELAX(u, v, w)
8. vdSecond = v.d
9. if vdFirst > vdSecond
10. count = count + 1
11. Return TRUE

This new algorithm has a few slight changes from the original BELLMAN-FORD(G, w, s) algorithm. First, since there is guaranteed to be no negative weight cycle, return FALSE statement at the condition check is unnecessary, so that has been removed. Second, a changes variable has been added, and the while loop now depends on whether or not changes is above 0, rather than depending on the number of vertices. The proof for the Bellman-Ford algorithm can be used to prove that this method works, all except for the addition of the changes variable. The reason the changes variable can be used, is because it always increases whenever there is a change in v.d, or in other words when vdFirst > vdSecond. If changes is greater than 0, that means the while loop will continue. However, in the Bellman-Ford algorithm, there is the possibility of wasted calls to RELAX(u, v, w), since if no changes were made after an iteration of the while loop, the next iteration of the while loop is calling RELAX(u, v, w) on the exact same edges and vertices with no changes. Thus, the results will be the same every time it is called. In this new algorithm, if there is no change made to any of the vertices, changes = 0, and the while loop will stop running. Changes relates to m in that changes will be above 0 for m iterations, and when the while loop should terminate, at m + 1, changes will finally equal 0.

Exercise 3 (25 points)

Change the weight of edge (z,x) to 4 and run Dijkstra’s algorithm on the directed graph G = (V, E) of Figure 3.1, first using vertex ***s*** as the source and then using vertex x as the source. **In the style of Figure 24.6**, show the d and values and the vertices in set S **after each iteration** of the while loop.

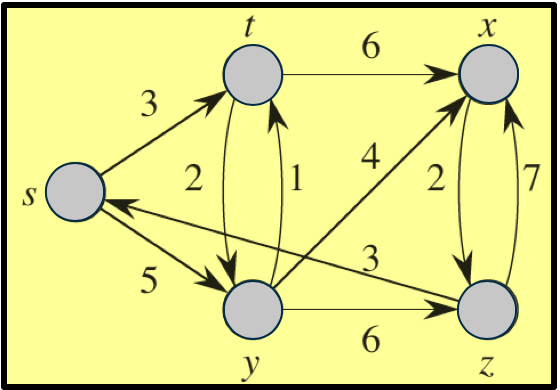


Figure 3 Graph G

Exercise 4 (25 points)

Suppose that we are given a weighted, directed graph G = (V,E) in which edges leaving the source vertex ***s*** may have negative weights while all other edge weights are nonnegative, and there are no negative-weight cycles. Argue that Dijkstra’s algorithm correctly finds shortest paths from s in this graph.

What you need to turn in:

* Electronic copy of this file (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.
* Recall that answers must be well written, documented, justified, and presented to get full credit.
* How this assignment will be graded:
* A right answer will get full credit when:
* It is right (worth 25%)
* It is right AND neatly presented making it easy and pleasant to read. (worth 15%)
* There is an obvious and clear link between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth 60%).
* Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.
* You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, personal writing is expected.