

# CS 2420-001 ALGORITHMS AND DATA STRUCTURES

Spring Semester, 2016

## Assignment 9: Graph Algorithms II

**Due Date:** 9:30 a.m., Wednesday, Apr. 27, 2016 (at the beginning of CS 2420 class)

(**Note:** This assignment has two programming exercises. Please do not miss the note at the end of this document, which contains important information that can help you finish this assignment.)

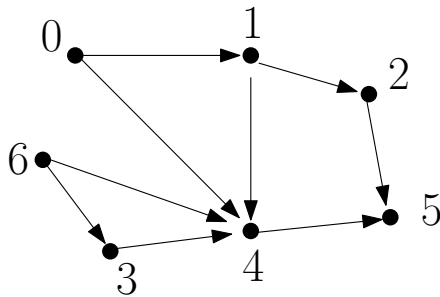


Figure 1: A directed acyclic graph (DAG) in the input file for Question 1.

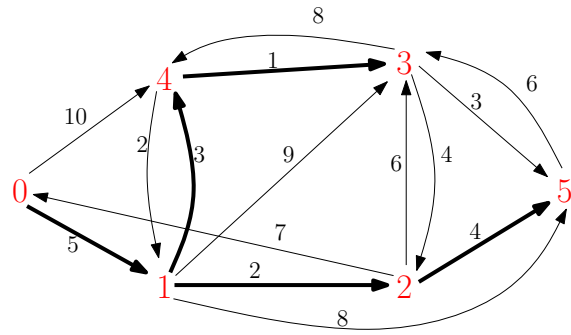


Figure 2: A weighted directed graph for Question 2: the edges have weights and the thick edges form a shortest path tree from vertex 0.

1. In this exercise, we will implement the topological sort algorithm. **(25 points)**

On Canvas, go to the following directory: homework/hw9/question1. There are a starter cpp file “hw9\_Q1.cpp” and an input file “hw9\_Q1\_input.txt”.

As in HW 8, in the input file, the first line is the number of vertices of the input graph, followed by an adjacency matrix of the graph, which is the one in Figure 1. The graph is a directed acyclic graph (DAG). The program first reads the graph information from the input file. The adjacency matrix will be stored in an array  $M$ , and then the program will construct the adjacent lists from the matrix.

Your task is to complete the function `topSort()` and output a topological order of the graph. The file “solution\_hw1\_Q1\_output.txt” gives a correct output. Since the topological order may not be unique, you may give different answers. We discussed two algorithms in class for the topological sort. You may use either one of them.

2. In this exercise, we will implement Dijkstra’s shortest path algorithm. **(25 points)**

On Canvas, go to the following directory: homework/hw9/question2. There are a starter cpp file “hw9\_Q2.cpp” and an input file “hw9\_Q2\_input.txt”.

Here the input file is not the same as before because the graph edges have weights. The number in the first line is still the number of vertices. For the adjacency matrix, for each element  $M[i, j]$ , if it is zero, then it means that there is no edge from  $i$  to  $j$ ; otherwise there is an edge and its weight is  $M[i, j]$ . The graph in the input file is the one in Figure 2.

When the program constructs the adjacent lists, the weights of the edges will also be stored in the adjacency lists. For this, as discussed in class, there is a data member “weight” in the Vertex struct.

Your task is to complete the function `Dijkstra()`. You should use vertex 0 as the source vertex. Essentially the function `Dijkstra()` is used to compute the predecessor array *pre* and the shortest path distance array *dis*. Both arrays have been defined in the constructor of the Graph class. After you compute the two arrays in the function `Dijkstra()`, you can use the two functions `printSP()` and `getSPdis()` I provided to print the shortest paths from vertex 0 to all other vertices and their shortest path distances. The code has already been given in the cpp file for this part. The file “solution\_hw9\_Q2\_output.txt” contains the correct output.

As we discussed in class, there are two ways to implement the priority queue  $Q$  in the Dijkstra’s algorithm, either using an array or using a heap. Although a heap is preferred since it is more efficient, you may also use an array, which is much easier to write the program.

**Note:** On the one hand, the topological sort and Dijkstra’s shortest path algorithm are important and fundamental topics, and I believe it is better for you to have a chance to implement them. On the other hand, I understand that it is the end of the semester and you may be quite busy. Therefore, I would like to do some extra work to help you on this assignment. Here is the plan. You go ahead and work on the assignment. In order to help you, I will upload my programs to Canvas (in the “hw9” directory) by the end of Saturday. You can take a look at my programs and then finish yours.

My solutions to HW8 will be posted on Canvas soon. They might be helpful too.

**Total Points: 50**