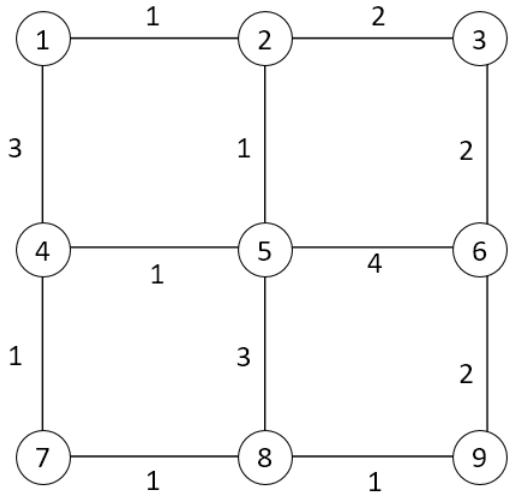
COMPSCI330 Design and Analysis of Algorithms
Assignment 6

Due Date: Wednesday, November 1, 2017

Guidelines

- **Describing Algorithms** If you are asked to provide an algorithm, you should clearly define each step of the procedure, and then analyze its overall running time. There is no need to write pseudo-code; an unambiguous description of your algorithm in plain text will suffice. If the running time of your algorithm is worse than the suggested running time, you might receive partial credits.
- **Typesetting and Submission** Please submit each problem as an *individual* pdf file for the correct problem on Sakai. LaTeX is preferred, but answers typed with other software and converted to pdf is also accepted. Please make sure you submit to the correct problem, and your file can be opened by standard pdf reader. **Handwritten answers or pdf files that cannot be opened will not be graded.**
- **Timing** Please start early. The problems are difficult and they can take hours to solve. The time you spend on finding the proof can be much longer than the time to write. If you submit within one week of the deadline you will get half credit. Any submission after that will not receive any credit.
- **Collaboration Policy** Please check this page for the collaboration policy. You are **not** allowed to discuss homework problems in groups of more than 3 students. **Failure to adhere to these guidelines will be promptly reported to the relevant authority without exception.**

Problem 1 (Dijkstra)**.** (10 points) We are going to perform Dijkstra's algorithm on the following graph



This is an undirected graph, the numbers close to the edges are the edge weights. The numbers on the vertices are the indices of the vertices. We will perform Dijkstra using vertex 1 as the starting vertex.

(a) (5 points) Give the ordering in which the vertices are visited in Dijkstra's algorithm. (When there are multiple options, the algorithm chooses the vertex with smaller index.)

(b) (5 points) After vertex 7 is visited by Dijkstra's algorithm, what are the distance values for all the vertices (if a vertex cannot be reached yet its distance value is $+\infty$)?

Problem 2 (Negative Edges)**.** (25 points) In this problem we will analyze the Bellman-Ford algorithm we discussed in class. Recall this is an algorithm for finding shortest paths when the graph can have negative edges.

We are given a graph G, edge weights $w[u, v]$, starting vertex s and ending point t. The algorithm defines states as: $d[u, i]$ is the shortest path from source s to u that uses at most i edges (if there is no path with at most i edges from s to u, $d[u, i] = +\infty$). The transition function is:

$$d[u, i+1] = \min\{d[u, i], \min_{(v,u)\in E} d[v, i] + w[v, u]\}.$$

The base case is given by $d[s, 0] = 0$ and $d[u, 0] = +\infty$ for all $u \neq s$.

We say a cycle $(u_1, u_2, ..., u_l)$ is a negative cycle, if the total weight of the cycle $w[u_l, u_1] + \sum_{i=1}^{l-1} w[u_i, u_{i+1}]$ is negative.

In this problem, we assume it is possible to reach all other vertices from s. To establish the correctness of the algorithm, the key steps are the following two properties:

(a) (10 points) Prove that if the graph G does not have any negative cycles, for any vertex u, there exists a shortest path from s to u that contains at most $n - 1$ edges.

(Hint: Prove that the shortest path should not visit the same vertex twice.)

(b) (15 points) If the graph has a negative cycle $(u_1, u_2, ..., u_l)$, prove that there exists a vertex u such that $d[u, n] < d[u, n-1]$.

(Hint: Focus on the vertices in the cycle.)

Problem 3 (Public Transit)**.** (25 points) Alice wants to go to Durham downtown by the public transit system. She has a transit map of the area, which in this problem is abstracted as a graph. Alice is at a vertex s and she is going to vertex t. The edges on the graph represents possible bus schedules. Each directed edge $e = (u, v)$ is described by three numbers: $l(e) > 0$ is the amount of time to get from u to v by bus; $t(e) \geq 0$ is the time of the first bus; $int(e) > 0$ is the interval of the bus. Starting at time $t(e)$, there will be a bus from u to v every $int(e)$ minutes (so the bus leaves u at times $t(e), t(e) + int(e), t(e) + 2int(e), ...$). In this problem you need to design an algorithm to help Alice figure out what is the fastest time to get to t. Alice leaves s at time 0 and she does not care about how many transfers to make. The graph is connected. If Alice arrives some vertex u at time t, and at the same time there is a bus leaving, Alice will be able to catch the bus.

(a) (10 points) Design an algorithm for Alice that finds the earliest time that she can reach vertex t (starting at time 0 at vertex s). Your algorithm should run in time $O(m + n \log n)$.

(Hint: You cannot apply Dijkstra's algorithm directly. If you modify Dijkstra's algorithm, you only need to describe the difference.)

(b) (15 points) For every vertex u, let $d[u]$ be the earliest time that Alice can reach u from s (starting at time 0). Suppose there is a subset of vertices S, we have already computed $d[u]$ for $u \in S$, and we know for any $v \notin S$, $d[v] \geq d[u]$.

Now, let v be a vertex not in S that has the minimum $d[v]$ value, prove that the best route to get to v only leaves the set S in the last step.