University of Waterloo
CS 341 — Algorithms
Spring 2014
Assignment 4

Written Portion Due: Fri July 18, 2014 at 3:00pm.
Programming Portion Due: Mon July 21, 2014 at 3:00pm.

Assignment Guidelines

- Use A4-coversheet.pdf for the first page of your assignment or format the first page of your assignment in *exactly* the same manner.
- Answers the questions in the same order that they are given in the assignment.
- Make it very clear to the marker where the answer to one question ends and the answer to the next one begins.
- You may lose up to 20% on a question because it is difficult to read or difficult to understand.
- You may only use the techniques discussed in class.
- Your whole assignment is considered late if either the programming question or the written portion is submitted late.

Assignment Questions

1. [**5 points**] Give an example of a graph $G = (V, E)$, a source vertex $s \in S$, and a set of tree edges $E_T \subseteq E$ such that for each vertex $v \in V$, the unique shortest path in the graph (V, E_T) between s and v is a shortest path in G, yet the set of edges E_T cannot be produced by calling $BFS(G, s)$, no matter how the vertices are ordered in each adjacency list. Justify corretness of your example by drawing E_T as well as all possible breadth-first search trees that $BFS(G, s)$ might produce.

   Hint: There exists an example graph with 5 vertices.

2. [**8 points**] Suppose that A is an algorithm that computes the weight of the minimum spanning tree of a weighted undirected graph $G = (V, E)$, but only if G has edge weights that are non-negative. Suppose that H is a weighted undirected graph that contains at least one edge of negative weight. Describe how to use algorithm A to compute the weight of the minimum spanning tree of H. Justify the correctness of your solution.

3. **[10 points]** Given a directed acyclic graph $G = (V, E)$ and a source vertex $u_0 \in V$ (no edges are directed into u_0). Design an algorithm to determine the number $N[v]$ of directed paths in G from u_0 to v, for all $v \in V$. The complexity of the algorithm should be $O(n + m)$.

   (a) Give a short high-level description of the the main idea of your algorithm, as well as pseudocode, and explain why the complexity is $O(n + m)$. The first step is topological sort (you do not need to include the details of this step). In the second step, process the vertices in topological order.

   (b) Prove that your algorithm is correct. This will be a proof by induction making use of an appropriate loop invariant.

4. **[10 points]** Many maps include a table that gives approximate driving times between cities using the most direct route. Unfortunately, the driving time may vary greatly depending on the time of day (e.g. during rush hour the table may grossly underestimate the time and it may be advantageous to take a detour through a couple of intermediate cities, avoiding the direct route). Consider the problem of finding an optimal travel route if we know how the travel times change throughout the day.

   We can formalize the problem as follows. You are given an undirected graph $G = (V, E)$, where vertices represent the cities and edges represent connections between the cities. Also provided is a table D, where for every $e \in E$ and every integer $h \in [0, 23]$, integer $D[e, h]$ represents the driving time in hours from one end-point of edge e to the other, assuming we start at hour h. Note that, for simplicity, we are assuming all times are given in integer (hour) incrememnts.

   Given two vertices u and v and the starting hour h, you must design an algorithm which will find the fasted route from vertex u to vertex v provided we start driving at u at hour h or some hour later than h. Note that it may actually be advantageous to wait for one or more hours in a particular city. Represent the problem using a different graph where a shortest path algorithm will find the optimal solution.

5. **[7 points]** Create and fill in an appropriate size table for a dynamic program that solves the 0-1 Knapsack where the weight capacity of the knapsack is 6 and the value (or profit) and weight of five items are as follows:

item	1	2	3	4	5
value p_i	5	4	3	2	1
weight w_i	5	2	2	2	1

   Consider the items in the same order that they are listed in the table, i.e. consider item 1 first, item 2 next, etc. Indicate both your answer and how you read off your answer from the dynamic programming table.

6. [**15 points**] Programming Question

   Implement the algorithm for topological sort as described in class. The input will be a directed graph $G = (V, E)$. You can assume there are no self-loops (edge (v, v) for some $v \in V$) in G.

   The input to your algorithm is as follows: the number of vertices n in G, followed by the adjacency list for each vertex (one per line).

   ```
   n
   Adjacency list for vertex 1
   Adjacency list for vertex 2
   ...
   Adjacency list for vertex n
   ```

   You should store the graph using the adjacency list structure and perform a DFS in increasing order of vertices.

   You algorithm should output one of the following:

   - A topological ordering of the vertices in V, if one exists (this will occur if and only if G is a DAG). The ordering is determined by listing the vertices in decreasing order of finishing time.
   - A directed cycle in G, if G is not a DAG. Recall that a cycle exists if you find a back edge (u, v) in G during the DFS. If this is the case, print the cycle starting at v, followed by the endpoints of the tree edges (that are part of the cycle) until you reach u, then print v again (i.e. the directed cycle starting at v).

   Print each output on one line, separating vertices with a space.