CS 6301.502. Implementation of advanced data structures and algorithms

Fall 2017; Fri, Oct 20.

Long Project LP4: Counting and enumeration problems in graphs

Ver 1.0: Initial description (Fri, Oct 20).

Ver 1.1: Part f: output has to be a simple cycle (Mon, Nov 6).

Due: 11:59 PM, Sun, Nov 12 (1st deadline), Sun, Nov 19 (2nd deadline).

Max score: 200. Max excellence credits: 10.

Initial code base: ArrayIterator, Graph, GraphAlgorithm, XGraph, LP4, Timer

You can also use code from other examples shown by the instructor.

Starter code for project is provided. Do not change the driver programs,

except for replacing g00 by your group number.

\* Submit before the first deadline to be eligible for excellence credit.

\* For each group, only its last submission is kept and earlier submissions are discarded.

\* Your code must be of good quality, and pass all test cases to earn excellence credits.

\* All groups satisfying above conditions will be assigned excellence credits.

Project Description:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Max Score for Parts a+b: 50. Max EC: 2.

Part a. Count the number of topological orders of a given DAG.

Part b. Enumerate all topological orders of a given DAG.

Sample input:

7 8

1 2 1

1 3 1

2 4 1

3 4 1

4 5 1

4 6 1

5 7 1

6 7 1

Output for part a:

4

Output for part b (in any order):

1 2 3 4 5 6 7

1 2 3 4 6 5 7

1 3 2 4 5 6 7

1 3 2 4 6 5 7

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Max score for Parts c+d: 50. Max EC: 2.

Part c. Number of simple shorest paths from a source node to a target node.

The input is a directed graph G=(V,E) as input, with edge weights

W:E-->Z (negative weights are possible). The output is the number of

shortest paths in the graph (not necessarily disjoint), from the source

to the target. If the graph has a negative or zero cycle, reachable from the

source, then print a message "Non-positive cycle in graph. Unable to solve problem".

Part d. Enumerate all shortest paths from source to target.

Note that in the following input graph, there is a negative cycle 6-->7-->6,

but the cycle is not reachable from the source 1. Therefore, the algorithm

reports 2 shortest paths from 1 to 5.

Sample input:

7 8

1 2 2

1 3 3

2 4 5

3 4 4

4 5 1

5 1 -7

6 7 -1

7 6 -1

1 5

Output for part c:

2

Output for part d:

1 2 4 5

1 3 4 5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Max score for Parts e: 50. Max EC: 1.

Part e. Shortest path that uses at most given number of edges.

The input is a directed graph G=(V,E) as input, with edge weights

W:E-->Z (negative weights are possible). In addition, a limit on

the number of edges in a shortest path is specified. Your program

should return the length of a shortest path from a source vertex

to a target vertex, using no more than the number of edges stated

in the limit. Excellence credits will be assigned based on quality of code,

running time of algorithm and space used by it.

Sample input:

10 11

1 2 1

1 7 3

1 9 6

2 3 1

3 4 1

4 5 1

5 6 1

6 10 1

7 8 3

8 10 3

9 10 6

1 10

3

Output (Shortest path from 1 to 10 with at most 3 edges):

9

Output with limit of 2 edges: 12

Output with limit of 6 or more edges: 6

Output with less than 2 edges: Infinity

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Max score for Parts f: 50. Max EC: 5.

Part f. Reward collection problem.

The input is an undirected graph G=(V,E), with weights W:E-->Z+, a

source node s in V. In addition each node has a reward amount

associated with it, R:V-->Z+. This amount can be claimed by visiting

that node (representing a node that is dirty and cleaning it with a

vacuum cleaner). Starting from s, find a traversal that starts and

ends at s, in which the reward collected is a maximum.

Rules: You are not allowed to visit any intermediate node more than once.

[Clarification added on Mon, Nov 6: traversal must be a simple cycle).

The reward at a node can be collected only if the traversal got there using

a shortest path from s to that node.

Excellence credits will be based on quality of code, its correctness, and

running times on test cases.

Note that in the following input, node 7 has a reward of $1,000, but

if we go there to collect it, there is no way to get back to s,

because the traversal is not allowed to revisit intermediate nodes for

a second time. In addition, no reward is collected when nodes 5 and 4

are visited, because the traversal did not get there using a shortest

path from s.

Sample input:

7 7

1 2 2

1 4 4

2 3 5

4 5 4

3 6 2

5 6 4

6 7 4

1

0 10 10 5 5 10 1000

Output:

30

1 2 3 6 5 4 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_