Assignment 7

SFWRENG 2CO3: Data Structures and Algorithms–Winter 2024

Deadline: March 17, 2024

Department of Computing and Software
McMaster University

Please read the *Course Outline* for the general policies related to assignments.

Plagiarism is a *serious academic offense* and will be handled accordingly.
All suspicions will be reported to the *Office of Academic Integrity*
(in accordance with the Academic Integrity Policy).

This assignment is an *individual* assignment: do not submit work of others. All parts of your submission *must* be your own work and be based on your own ideas and conclusions. Only *discuss or share* any parts of your submissions with your TA or instructor. You are *responsible for protecting* your work: you are strongly advised to password-protect and lock your electronic devices (e.g., laptop) and to not share your logins with partners or friends!

If you *submit* work, then you are certifying that you are aware of the *Plagiarism and Academic Dishonesty* policy of this course outlined in this section, that you are aware of the Academic Integrity Policy, and that you have completed the submitted work entirely yourself. Furthermore, by submitting work, you agree to automated and manual plagiarism checking of all submitted work.

Late submission policy. Late submissions will receive a late penalty of 20% on the score per day late (with a five hour grace period on the first day, e.g., to deal with technical issues) and submissions five days (or more) past the due date are not accepted. In case of technical issues while submitting, contact the instructor *before* the deadline.

Problem 1. Consider an $m \times n$ game board in which each cell has a numeric value, e.g.,

	A	B	C	D	E	F
G	1	2	2	3	4	2
H	3	4	4	4	4	1
I	1	4	1	3	1	4
J	2	3	1	4	1	2
K	3	3	2	1	4	2

A player starts the game with a token in the top-left cell (the cell GA in this example) and the player finishes the game by moving to the bottom-right cell (the cell KF in this example). In each round of the game, the player can move in *four* directions (up, down, left, and right). The distance of each move is determined by the value of the cell. When going over the border of the game board, one ends up on the other side. For example, if the player is in the cell JB, which has value 3, then the player can move 3 steps up (reaching GB), 3 steps right (reaching JE), 3 steps down (reaching HB), and 3 steps left (reaching JE).

The score of a player is determined in the total number of rounds the player needs to reach the bottom-right cell.

P1.1. Model the above problem as a graph problem: what are the nodes and edges in your graph, do the edges have weights, and what problem are you trying to answer on your graph?

P1.2. Provide an efficient algorithm that, given a $m \times n$ game board, will find an optimal solution (a minimum number of steps to reach the bottom-right cell) if such a solution exists. If the game board has no solution, then the algorithm should report that the game board is invalid. The runtime of your algorithm should be worst-case $O(mn)$.

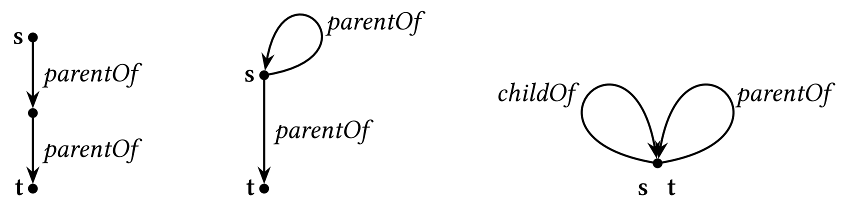
P1.3. Explain why your algorithm is correct and has a complexity that is worst-case $O(mn)$.

P1.4. Which of the two *graph representation* we saw in the course material did you use to store the game board? What would the complexity of your algorithm be if you used the other graph representation?

Problem 2. Edge-labeled graphs are graphs in which edges have labels that represent the type of relationship that is expressed by that edge. For example, in a social network graph, the edges could be labeled *parentOf*, *friendOf*, and *worksWith*.

One way to express *graph queries* (that express how new information can be derived from edge-labeled graphs) is via a *query graph* that expresses how new relationships between source node **s** and target node **t** can be derived from existing information. Consider, for example, the following three example query graphs:

Query 1: *grandParentOf*(**s**, **t**)	Query 2: *ancestorOf*(**s**, **t**)	Query 3: *relatedTo*(**s**, **t**)



The first query relates nodes that represent grandparents and their grandchildren, the second query relates nodes that represent ancestors and their descendants, and the third query relates everyone with a direct family relationship.

Let Q be a graph query and G be an edge-labeled graph representing a data set. The *graph query evaluation* problem is the problem of computing the derived relationship in G expressed by Q.

Typically, queries are *small* and data graphs are enormous. Hence, here we will assume that the size of a query graph is *constant*.

P2.1. Model the *graph query evaluation* problem as a graph problem: what are the nodes and edges in your graph, do the edges have weights, and what problem are you trying to answer on your graph?

P2.2. Provide an efficient algorithm that, given a a graph G, a source node n in graph G, and query Q, will find all nodes m such that the pair (n, m) is in the derived relationship in G expressed by Q. Assuming Q has a constant time, the runtime of your algorithm should be worst-case $O(|G|)$ in which $|G|$ is the total number of nodes and edges in G.

P2.3. Explain how you represented your graph G, why your algorithm is correct, and why your algorithm has a complexity that is worst-case $O(|G|)$.

Assignment Details

Write a report in which you solve each of the above problems. Your submission:

1. must start with your name, student number, and MacID;

2. must be a PDF file;

3. must have clearly labeled solutions to each of the stated problems;
4. must be clearly presented;
5. must *not* be hand-written: prepare your report in LaTeX or in a word processor such as Microsoft Word (that can print or export to PDF).

Submissions that do not follow the above requirements will get a grade of zero.

Grading

Each problem counts equally toward the final grade of this assignment.