Assignment 7

COMPSCI 2CO3: Data Structures and Algorithms–Fall 2025

Deadline: November 21, 2025

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. We provide a general *shared grace period* for all assignments of 5 days (120h) that students can utilize whichever way they seem fit without further notice or approval. For example, a student can hand in *one* assignment 100h late, or hand in 4 assignments 20h late (as long as the total stays below 120h). Beyond the above grace period, we require students to *submit on time*. Late submissions are not accepted and will result in a grade of zero. The above grace period *does not stack* with MSAFs or SAS accommodations.

Problem 1. A directed graph is semi-connected if, for every pair of nodes m, n, there is either a path from node m to node n or a path from node n to node m (or both).

P1.1. Show that every strongly connected graph is semi-connected.

P1.2. Provide a connected graph that is not semi-connected.

P1.3. Provide an algorithm to determine whether a directed *acyclic* graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is semi-connected in $O(|\mathcal{N}| + |\mathcal{E}|)$.

P1.4. Provide an algorithm to determine whether a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is semi-connected in $O(|\mathcal{N}| + |\mathcal{E}|)$. You may assume to have an algorithm A that can determine whether a directed *acyclic* graph is semi-connected (e.g., algorithm A could be your solution for Problem P1.3).

P1.5. Which of the two *graph representation* we saw in the course material did you use to represent the graphs in the previous two questions? What would the complexity of your algorithms be if you used the other graph representation?

Problem 2. 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	2	4
J	2	3	1	4	1	2
K	3	3	2	1	4	2

We play a two-player game in which two players pick a distinct initial cell (e.g., player one picks cell GA and player two picks cell IE). Next, each player places their token on the initial cell they picked. In each round, each 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 a 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 player that reaches the initial cell of the other player first wins. For example, starting at cell GA, player one can reach cell IE in *four rounds*: move *right* from GA to GB, *up* from GB to JB, *right* from JB to JE, and *up* from JE to IE. Hence, player one is guaranteed to win if player two needs more than four rounds to reach cell GA starting at cell IE.

A player A has a *strong* starting position if player A can find a strategy that *guarantees a win*: a strategy that lets player A reach the initial cell of their opponent O in less rounds than O needs, via any strategy, to reach the initial cell of A. We want to find an algorithm to determine whether player A has a strong starting position.

P2.1. 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?

P2.2. Provide an efficient algorithm that, given a $m \times n$ game board, will determine whether a given player has a strong starting position.

P2.3. Explain why your algorithm is correct and is efficient.

P2.4. Which of the two *graph representation* we saw in the course material did you use to represent the game board? What would the complexity of your algorithm be if you used the other graph representation?

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.