University of Waterloo
CS 341 Winter 2026
Written Assignment 3

Elena Grigorescu, Mark Petrick, Luke Schaeffer
Copyright © 2026 Distribution (except by the authors) is prohibited.

Due Date: Friday, February 27 at 11:59pm to Crowdmark
All work submitted must be the student's own.

- Make sure to read the Assignments section on the course webpage for instructions on submission and question expectations ("Instructions for Assignments"): `https://student.cs.uwaterloo.ca/~cs341/#Assignments`

Question 1 [10 marks] Communication Conundrum

Suppose you are managing a team of people, and for each person you have list of the languages they speak fluently. We say two people on the team can *communicate directly* if they share a common language. Two people can communicate *indirectly* if there is a series of intermediaries $A = C_0, C_1, \ldots, C_{k-1}, C_k = B$ such that C_i, C_{i+1} can communicate directly for $0 \leq i < k$.

a) Design an algorithm to determine whether everyone on the team can communicate with each other. The input is a bipartite graph of people and languages where person p and language ℓ are adjacent if and only if p speaks ℓ fluently.

b) Design an algorithm to test whether the loss of any individual member of the team would disrupt communication.

Question 2 [10 marks] Palindromic Walks

Let $G = (V, E)$ is a graph on $|V| = n$ vertices and $|E| = m$ edges. Suppose that every edge $e \in E$ has a label $\ell(e) \in \{1, \ldots, k\}$ such that for every u, all edges (u, v) incident to u have distinct labels. We say a walk $v_0, \ldots, v_r \in V$ is *palindromic* if the sequence of edge labels is the same forwards as backwards:

$$\ell((v_0, v_1)) \cdots \ell((v_{r-1}, v_r)) = \ell((v_{r-1}, v_r)) \cdots \ell((v_0, v_1)).$$

(More generally, a *palindrome* is a string that is the same forwards and backwards.)

a) Describe an efficient algorithm taking input $G = (V, E)$ (in adjacency list representation, with labels), vertices $s, t \in V$, and outputs the shortest palindromic walk from s to t, or reports that no such walk exists.

b) Show that G has $m = \Theta(nk)$ edges.

c) Justify the correctness of your algorithm, and analyze the runtime as a function of n and k.

d) Find a small (≤ 5 vertices) graph and vertices s, t such that

1. there is a path from s to t,
2. all vertices are incident to edges with the same set of labels, and
3. there is no palindromic walk from s to t.

Question 3 [15 marks] Alice and Bob play a game of Not Chess

Alice and Bob are having lunch and the break room has a chess board. Neither of them really know how to play chess, so they make up their own game with the following rules.

They scatter some pawns on the board, and one queen. The players take turns moving the queen one or more steps in a straight line either left, down, or diagonally left and down,[1] provided it is unobstructed by the pawns. After some number of moves, the queen will reach the bottom left corner or otherwise blocked by pawns, leaving no legal moves. The player who cannot make a move loses.

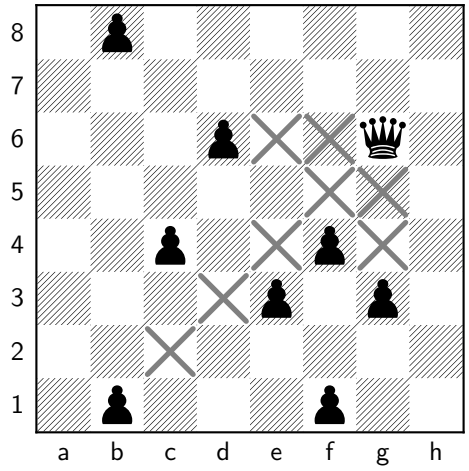

Figure 1: Available moves in the example instance if the queen is on g6.

Alice and Bob play a few rounds. They find it is easiest to leave the pawns where they are, and reset the queen to an arbitrary unoccupied square. Once the queen is placed, the winner (under optimal play) is determined from the starting configuration, since there is no randomness or hidden information. Alice and Bob are naturally curious when the first player

[1] Like a queen normally moves, but only in three directions.

Input:

```
8 8
-P------
--------
---P----
--------
--P--P--
----P-P-
--------
-P---P--
```

Output:

```
1P111211
11121111
111P1111
11211111
11P12P11
1211P1P2
11112111
2P211P21
```

Figure 2: Example input and output

or second player has a winning strategy; for each unoccupied square, if the queen starts there is the game won by the 1st or 2nd player under optimal play.

Design an algorithm that gets the configuration of the pawns as input. For each unoccupied square, the algorithm computes the winner of the game (under optimal play) if the queen starts in that position, and outputs a representation of the board with either `1` (for first player winner) or `2` (for second player winner) filled into each spot that is not already occupied by a pawn.

The input consists of $m + 1$ lines. The first line contains integers m and n separated by a space ($1 \leq m, n \leq 1000$) — we generalize the problem to $m \times n$ chess boards. The following m lines represent the board with lines containing n characters that are either "`-`" (for a blank space) or "`P`" (for a pawn). The first character of the last line represents the bottom left corner of the board, as you might expect.

The algorithm should output a similar board, but where each `-` replaced with either `1` (if the first player wins under optimal play) or `2` (if the second player wins under optimal play).

For example, the bottom left character in the example output is a `2` because if the queen is in the bottom left corner then it has no moves, so the first player is unable to play and immediately loses. On the other hand, the square just above the bottom left corner is `1` because the first player can move to the corner, and then the second player has no move.

a) Design an algorithm for this problem and include pseudocode.

b) Justify the correctness of your algorithm, and analyze the runtime. For full marks, it should run in $\Theta(mn)$ time.

c) Implement your algorithm in C++ and submit to Marmoset. Further instructions about Marmoset submission to follow on the course website.

Question 4 [10 marks] Study Schedule

Suppose that you are preparing for n incoming midterm exams numbered as $1, 2, \ldots, n$. Each exam will be graded on a scale from 0 to 100. You have decided to spend a total of H hours to study for the exams and you want to divide up the time for studying the exams. For simplicity, assume that H is a positive integer, and you will spend a non-negative integer number of hours studying for each exam. To find out how to best divide up your time, you roughly estimate that if you spend h hours on studying for the exam i you will get a grade of $f_i(h)$. You may assume that each function f_i is non-decreasing (i.e. $f_i(h) \geq f_i(h')$ for every $h \geq h'$). In other words, the harder you study for the exam, the better grade you will get.

Given the set of estimated score functions $\{f_1, f_2, \ldots, f_n\}$ (explicitly, as an array $F[1..n, 0..H]$ of positive integers), devise an efficient algorithm to figure out how many hours (in integer values only) to spend on studying for each exam so that your total sum of all the grades is maximized. Prove the correctness of your algorithm and analyze its time and space complexity. Note that to get full credit your algorithm needs to output both the optimal total grade and the corresponding distribution of time.