CS 180: Algorithm and Complexity
Lab 1

Down with Trojans 1.0.0

Matt Ferland, Yadi Cao, and Noor Nakhaei
July 20, 2023
Due: August 1, 2023 @ 11:59 PM PT

In this lab, you'll be writing the implementation for the following Dynamic Programming problem. You can choose either C, C++, or Python 3 to implement the solution. You'll be given a basic skeleton that contains the code and the makefile (if you choose to use C). You're expected to understand how to implement a Dynamic Programming problem using either the Top-down or Bottom-up approach.

Problem. You are playing an indie RPG dungeon crawler video game called Down with Trojans. In it, you play as a UCLA student attempting to sabotage USC's football team before the upcoming UCLA vs USC football game (as if the Bruins need it to win).

The game is played on an n by n grid, and you start on the top left. Your goal is to reach the square on the bottom right, where the coaching records for the Trojan team are stored. You can only move down and right, not up and left. You start with H HP, and if it ever goes below 0 after moving to a tile, the player will be caught and you get a game over. There 4 types of tiles: damage tiles (which come with number d), healing tiles (which come with number h, protection tiles, and multiplier tiles. If you land a damage tile, USC's UPD is on to you, you will lose d HP. If you land on a healing tile, then you bribe a USC student to help you, and you will gain h HP. If you land on a protection tile, Matt helps protect you, so whenever you land on a damage tile, you can choose to blame Matt instead, throwing the UPD off and making so you don't lose any HP. Finally, a multiplier tile contains a ton of cash, so you can choose, the next time you land on a healing tile, to gain double the HP instead. Note that while you can hold both a multiplier token and a protection token at the same time, you can only hold up to one of each at a time.

Your dynamic programming algorithm should output a boolean that reports whether it is possible for the player to reach bottom right without running out of HP. The runtime should be $O(n^2)$

Starting the lab. Download the skeleton code from BruinLearn for this lab's manual and skeleton code archive, which contains the skeleton codes and makefile for this lab.

Files to modify.

1. If you choose to use C or C++, modify `kill_Down_with_Trojans.c` only. You are not allowed to modify the makefile to include any advanced additional libraries.
2. If you choose to use Python 3, modify `kill_Down_with_Trojans.py` only. You are allowed to use NumPy and SciPy libraries. You are not allowed to include any additional advanced external libraries.
3. Basic external libraries such as these related to IO in C and C++, or `math`, `random`, etc. in Python 3, are allowed.

Input and output format Your input format must be a text file named `*.txt`, where `*.txt` will be parsed as the 2nd argument to your program. The input file describes the grid configurations in the following format:

1. The first line of the file will contain 2 integers, n and H, where n is the size of the grid (as with the problem description), and H is the starting HP.
2. We agree that the starting point is always at (0, 0), and the bottom right is at $(n-1, n-1)$.
3. The next $n^2 - 1$ lines (the top left square will be empty) will contain 4 integers each, x, y, t, and v, where x and y are the coordinates of the tile, t is the type of the tile (0 for damage, 1 for healing, 2 for protection, and 3 for multiplier), and v is the value of the tile (only applies for damage and healing tiles, it will be 0 for others).
4. You can assume that $n \leq 100$

Your output must be a file named `*_out.txt`, which contains a single integer, 1 if it is possible to reach the bottom right without running out of HP, and 0 otherwise. You function must compute this in no more than a few seconds.

Submission.

1. All lab submissions will take place on BruinLearn. You will find submission links for all labs under the Assignments page.
2. Your submission should be a single tarball that includes only the following files:
3. The tarball should be named `YOURUID_APPEND.tar`, where `YOURUID` is your 9-digit UID without any separators, `APPEND` is `P` if you choose Python or `C` if you choose C.
4. Your submission will be graded solely based on your latest submission to BruinLearn, without any exceptions. Please double-check your submission to make sure you submit the correct version of your implementation.
5. If you choose to use C, it's your responsibility to make sure your submission can be compiled and run on a Linux machine using the provided makefile. You can use Seasnet or your own virtual machine (recommended: Ubuntu 22.04) to test your code.
6. If you choose to use Python 3, it's your responsibility to make sure your submission can be run on a Linux machine with any Python 3.10 environment. You can use Conda and virtual environment with Python 3.10 to test your code.
7. Any submission in the 1st trial that does not follow the format will automatically be marked as incorrect, but feedback will still be provided for your 2nd trial. Any submission in the 2nd trial that does not follow the format will be permanently marked as a fail.

Hint Try all healing tiles to make sure your code isn't completely busted, only damage and healing to test basic functionality, and then when that works move on to working on including one of the tokens, before making sure you can do both at the same time. Then, ensure your code works efficiently up to size n=100.