CSCI 4041, Fall 2018, Programming Assignment 11

Due Tuesday, 11/20/18, 10:30 AM (submission link on Canvas)

This is not a collaborative assignment; you must design, implement and test the solution(s) on your own. You may not consult or discuss the solution with anyone other than the course instructor or TAs. In addition, you may not include solutions or portions of solutions obtained from any source other than those provided in class. Obtaining or sharing solutions to any programming assignment for this class is considered academic misconduct. If you are not sure what this means, consult the class syllabus or discuss it with the course instructor.

(Note: the first page is entirely fluff information. If you don't like ridiculous stories used to justify doing the problem, you're free to skip to page 2 where we start talking about implementation)

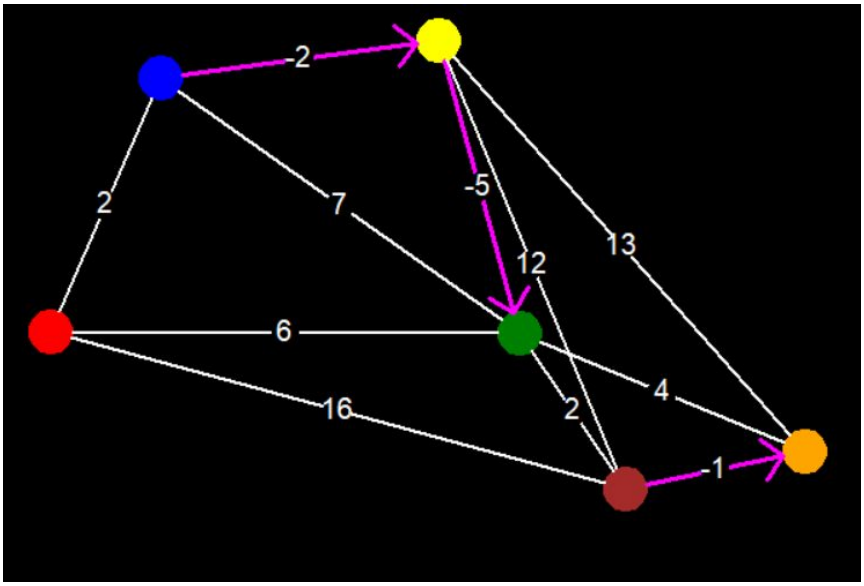
As one of the top engineers of the Gray Legion, your life recently has been nothing but humiliation. Twice now, your superior technology has failed the Gray Legion: first they were prevented from invading the Earth by a bunch of ridiculous colorful warriors using the power of friendship, and then your incredible transdimensional beam was reflected back against the legion's battleship as they attempted to assimilate the Orange Beta system.

You have been placed on hyperspace time table computation, grunt work which is a duty far beneath your cognitive prowess. However, you've noticed something odd about a few of the hyperspace jump signatures detected from a few scouting ships attempting to forge new hyperspace routes: even taking into account relativity and how hyperspace travel interacts with it, the ships appear to be arriving at their destination before they depart. Interestingly, anywhere this phenomenon is detected, attempting to traverse the potential hyperspace route in reverse has resulted in catastrophic failure: the scouting ship simply disappears, never to be heard from again.

You suspect that the scouts have accidentally stumbled upon one-way wormholes that not only break relativity by transporting objects faster than light, but actually send the objects back in time by a few hours as well.

You intend to use this knowledge to shock the Gray Legion by giving them the greatest improvement in intragalactic travel time they have seen for centuries.

Unless...you manage to find some sequence of hyperspace jumps that starts and ends at the same location but requires negative time. A negative time loop would go beyond just increasing the speed of transportation, it would allow true backwards time travel. That is something you're not willing to give to the Gray Legion, not when you can use it yourself to rewrite history.



Updated star map for PA11: one-way wormholes shown as magenta arrows

Download the PA11.py template from the course website.

The template contains most of the same classes and structure as PA10: see the PA10 writeup for details.

You must implement the `optimize_hyperspace_routes` function, which takes as input a list of Star objects (essentially a list of vertices), and a matrix of integers representing the jump times (edge weights) between any pair of Stars, and uses either the Floyd-Warshall algorithm, or Johnson's algorithm to compute the shortest path from every star to every other star. This function will output one of two things, depending on whether or not there is a negative cycle present in the graph.

If there is a negative cycle, then the function will return a list of Star objects, representing the sequence of Stars you must visit to form the negative cycle (this means that the list should start and end with the same Star object).

If there is not a negative cycle, then the function will return a matrix (i.e. a list of lists) of integers, where each entry matrix[i][j] represents the minimum total weight of any path from the Star at index i to the Star at index j (remember that you can get the index of a Star object by using its .num attribute).

Requirements:

- You must download the template file PA11.py and edit the `optimize_hyperspace_routes` function. You can create your own helper functions, but don't edit the code beyond the "DO NOT EDIT" line. There is one exception to this: you are permitted to switch the flag on turtle.tracer(1) to a 0 if you want to speed up testing.
- You must complete the all-pairs shortest paths functionality of `optimize_hyperspace_routes` by using either the Floyd-Warshall algorithm, or Johnson's algorithm. You may use whatever technique you want to find negative cycles.
- Your program must run without errors on the version of Python installed on the CSELabs machines, Python 3.5.2. (if you're testing this on CSELabs, you need to type python3 or idle3 instead of python or idle to start the correct version from the terminal)
- You are not permitted to use the input() function as this will break the grading script.
- You must implement either the Floyd-Warshall or Johnson's algorithm, as found in Chapter 25 of the textbook. Any other algorithm will receive very little credit.
- This assignment will be graded automatically based on the number of test cases your program passes. There will be several secret test cases in addition to the ones included in the template to ensure you're not hard-coding in the solutions.
- This program will only run test cases until you fail one, avoiding the problem of having to scroll through test output to find the one broken test case.
- The grading breakdown for this assignment is as follows:
    - 30%: File runs without syntax errors
    - 70%: Passing test cases without breaking any requirements.
- The unedited template file already runs without syntax errors. This means that if your program causes syntax errors, you will get a better score by just submitting the original template unedited.
- Submit your edited PA11.py file to the Programming Assignment 11 link on Canvas before 10:30 AM on 11/20/18. No credit will be given for late submissions.