

CS 333 - Homework #1

Due To

13 April 2023 Thursday, by 11.59 pm.

Homework 1

There are 2 independent tasks in this homework.

Task1 (50 Points)

This task is intended to provide you experience in building/manipulating adjacency list representations of graphs, and to reinforce your understanding of BFS and DFS by adapting them to solve a novel problem. This programming assignment is to be completed individually in either Java or Python (your choice) and submitted to LMS.

Zip file "KidneyDonation" is attached with some starter code (in Python and Java both) and test cases. Modify the DonorCycle.java or donor_cycle.py file (you don't need to do both!) to solve the problem described in the writeup below.

Here is a summary of the test cases provided:

- example.txt is the example file in the writeup. The answer should be True.
- slides.txt is the directed graph from the slides with the right-most node as the query. The answer should be False.
- straightline.txt is just a line of vertices. The answer should be False.
- biggerloop.txt is 10 vertices in a circle. The answer should be True.
- almostcycle.txt is 10 vertices in a cycle, but with one edge in the opposite direction from the rest. The answer should be False.
- complete.txt is 10 vertices that are each connected to all the other vertices. The answer should be True.
- lollypop0.txt is a graph with vertices in a lollypop shape, meaning there is a straight line of nodes connected to nodes in a loop. The query node is at the start of the stick of the lollypop, so the answer should be False.
- lollypop5.txt is the same lollypop graph as above, but now the query node is in the candy portion of the lollypop. The answer should be True.

To give input to your program, you should just run the program. At first nothing will appear to happen, but then if you copy-paste the contents of a file into the console (the place that things print to in your IDE) and hit enter the code should run. The starter code parses the input for you and invokes a function using those parameters. To provide output, simply print to the console. Make sure you do not have any print statements besides just your answer. There will be additional tests run when you submit that you will not have direct access to, so make sure you do some debugging on your own, too

Description

Kidneys are organs in the human body whose function is critical for maintaining life. Put simply, the kidneys primarily serve to regulate the salinity, water content, and pH of one's blood, and also remove toxins of various kinds from the body. If someone has no working kidneys then they must either receive frequent kidney dialysis (which cycles blood through a machine to perform the function of a kidney), or else receive a new kidney from an organ donor. Humans are generally born with two kidneys, and one functioning kidney is sufficient for survival, so it is possible (though not risk-free) for a living person to donate a kidney to a patient in need. To protect from disease, the human immune system has evolved to recognize and attack cells from other individuals. As such, before donation, we must check whether a donor and recipient are compatible. Donor-patient compatibility is determined by weighing many factors, but one important factor is HLA match compatibility.

HLA Matching

Human Leukocyte Antigens (HLAs) are proteins that attach to the membranes of cells to help the immune system identify which cells are "self" cells and which are "invader" cells. Each person has a collection of different HLAs, and the more HLAs that are shared between the donor and the recipient the safer the transplant will be. The similarity is measured by a "HLA match score", and we will consider a patient and donor to be compatible in the case that their match score is at least 60. Because of all the factors at play for donor-recipient compatibility, it may be the case that a recipient does not know any compatible donors. To help that patient find a donor, donation networks have been set up. The idea is that if Recipient A needs a kidney, but their friend Donor A is not compatible, then perhaps Donor A could give a kidney to compatible Recipient B in exchange for Donor B giving a kidney to Recipient A. So in short, if Recipient A has no known donors, they can try to identify a "kidney trade" with another recipient using one of their donors. These trades could also be larger "kidney donation cycles", where Donor A gives to Recipient B, Donor B gives to recipient C, and Donor C gives to recipient A. For the sake of this assignment, our cycles can be of any length, but they must be cycles (the idea being that Donor A is only willing to donate a kidney if the result is Recipient A receiving one).

Problem Statement

For this assignment you will implement an algorithm to identify whether a given recipient can receive a kidney through a donation cycle.

Kidney Donation Cycle Definition

A sequence of recipients $(r_0, r_1, r_2, \dots, r_{n-1})$ is a kidney donation cycle of length n provided:

- all of r_1, \dots, r_{n-1} are kidney recipients,
- for each choice of $0 \leq i < n - 1$ there exists a donor associated with r_i is compatible with r_{i+1} ,
- and a donor associated with r_{n-1} is compatible with r_0 .

Input

Your input will be string (which you can think of as the contents of a file) which contains the following information:

- One row giving the number of recipients (call this n). Each recipient will be identified by a number between 0 and $n - 1$ (inclusive)
- One row giving the number of donors (call this m). Each donor will be identified by a number between 0 and $m - 1$ (inclusive)
- One row containing m comma-separated integers between 0 and $n - 1$ indicating the recipient that each donor is associated with.
- A table giving the HLA match points for every Donor-Recipient pair, represented as a sequence of m rows each with n comma-separated integers.
- One row giving a “query recipient”.
- Overall, the input will have $m + 4$ lines.

Output and Running Time

Your code will print just a single Boolean value (True or False in python, true or false in java) indicating whether there exists a kidney donation cycle which includes the query recipient.

The worst-case asymptotic running time of your program should belong to $O(nm)$, where n is the number of recipients and m is the number of donors (if it helps you to verify your running time, observe that it must be that $m \geq n$).

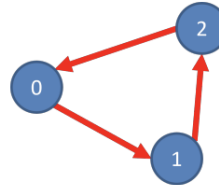
Example

This graph would be the result of the following text. We have an edge from recipient 0 to 1 because donor 2 has a match score of 71 with recipient 1, and donor 2 is the friend of recipient 0. There is an edge from recipient 1 to recipient 2 because donor 6 has a match score of 100 with recipient 2, and donor 6 is friends with recipient 1. There is an edge from recipient 2 to recipient 0 because donors 7 and 8 have a match score of 90 with recipient 0.

```

3
9
0,0,0,1,1,1,1,2,2
10,15,20
20,20,9
0,71,20
2,2,2
3,3,3
4,4,4
5,5,100
90,0,0
90,0,0
0

```



Task2 (50 Points)

This task is intended to provide you experience in building/manipulating adjacency list representations of weighted graphs, and to reinforce your understanding of Dijkstra's algorithm by adapting it to solve a novel problem. This programming assignment is to be completed individually in either Java or Python (your choice) and submitted to LMS.

Attached "SubprimeDelivery" is a zip file with some starter code (in python and java both) and test cases. Modify the Subprime.java or subprime.py file (you don't need to do both!) to solve the problem described in the writeup below.

There are two test cases provided:

- test0.txt is the test from the writeup. Its shortest path should be 0,2,3 and have cost 120
- test1.txt is a larger graph. Its shortest path should be 0,4,5,6,7,3 and have cost 70

To give input to your program, you should just run the program. At first nothing will appear to happen, but then if you copy-paste the contents of a file into the console (the place that things print to in your IDE) and hit enter the code should run. The starter code parses the input for you and invokes a function using those parameters. To provide output, simply print to the console. Make sure you do not have any print statements besides just your answer. There will be additional tests run when you submit that you will not have direct access to (in fact the majority are hidden for this assignment), so make sure you do some debugging on your own, too!

Motivation

Premium service subscription rates are plummeting for tech companies across the board. Online shopping companies such as ONLINESHOP is now coming out with its answer to this trend: sub-premium service subscriptions. ONLINESHOP company's sub-prime subscribers will receive a small discount on all

products bought from ONLINESHO[, but you must agree to receive only the worst customer treatment for all things. One example of this intentionally poor treatment is that ONLINESHOP company will now send your packages to you via routes that are not guaranteed to be the most efficient. Instead, they will try to use sub-prime orders to do their best to minimize the excess capacity of all of their vehicles.

Problem Statement

Your algorithm will receive two weighted graphs as input. The first graph will represent the transportation capacities of their vehicles. The nodes represent cities, edges represent trucks travelling between cities, and the weight of the edge represents the maximum weight that can be carried by each truck. The second graph will represent the current load on all vehicles. The same nodes and edges will be present as for the first graph, but now the edges represent the current weight of all cargo in each truck. Your goal will be to write an algorithm which finds the path from a given start city to a destination city which minimizes the sum of capacity percentages across all trucks used. Note that the number of trucks is irrelevant, we only care to minimize the sum of percentages. At-capacity trucks may not be used.

Input

Your input will be a multi-line string which contains the following information:

- One row giving the number of total cities (call this c)
- c rows each containing a list of pairs of integers. Each pair of integers represents a destination city and the carrying capacity of the truck going there. Values within a pair are separated by a comma, the pairs themselves are separated by semicolons. If the node associated with a particular line has no outgoing edges then we will write only a period on that line.
- c more rows each containing a list of pairs of integers. Each pair of integers represents a destination city and the current load of the truck going there. Values within a pair are separated by a comma, the pairs themselves are separated by semicolons. If the node associated with a particular line has no outgoing edges then we will write only a period on that line.
- One row containing an int identifying the start city one row containing an int identifying the destination city.

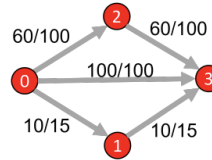
Output and Running Time

Your code will print a list of integers (one integer per line) indicating a sequence of cities which starts in the start city and ends in the destination city that also minimizes the cumulative sum of percentages of truck capacities. The worst-case asymptotic running time of your program should belong to $O(ct)$, where c is the number of cities and t is the number of truck.

```

4
3,100;1,15;2,100
3,15
3,100
.
3,100;1,10;2,60
3,10
3,60
.
0
3

```



Example

The graph would be the result of the text above (note lines with just a dot because 3 has no outgoing edges). In this case the path your algorithm should print would be 0,2,3. This path has cost 120 (because it is 60% + 60%). The path 0,1,3 would have cost 133.34 (66.67% + 66.67%). The path 0,3 is not valid because the edge is already at capacity. After your program finishes, the following text should be printed to the console:

```

0
2
3

```

Submission Rules

You will submit this homework via the LMS system. You should follow the file-naming conventions and guidelines below:

- You should submit your source files as a ZIP archive file (NOT RAR or other formats). The name of the file should be in format “<USER-ID>_hw<HOMEWORK-NR>.zip”. For example, if your username is vy1043, then the name of the submitted file should be “vy1043_hw1.zip”. Pay attention that all the letters are in lower-case. ZIP archive is supposed to contain just the source files, under two folders corresponding to the two tasks (“KidneyDonation” and “SubprimeDelivery” folders).
- Late submissions and files that do not compile are not accepted.
- You can resubmit your homework (until the deadline) if you need to.
- You are not allowed to use any external libraries and/or functions. Everything must be implemented by you from scratch.
- Any type of plagiarism will not be tolerated. Your submitted codes will be compared with other submissions and also the codes available on internet and violations will have a penalty of -100 points. (In case of copying from another student both parties will get -100)