PA2: Amazon Sub-prime

This programming assignment is to be completed in either Java or Python (your choice) and submitted on Gradescope.

Collaboration is restricted to just 2 people (other than yourself), and it must be "Whiteboard Only". In summary, this means that you may discuss the assignment with 2 others, but you cannot preserve any artifacts (digital or physical) from these discussions. In particular, this means you may not take and keep photos, notes, video, audio recordings, whiteboard contents, etc. from any discussions.

All collaborators and other resources consulted should be listed as a comment at the top of your code submission. Course staff and official course materials do not need to be listed.

Motivation

Premium service subscription rates are plummeting for tech companies across the board. Amazon is now coming out with its answer to this trend: sub-premium service subscriptions. Amazon sub-prime subscribers will receive a small discount on all products bought from Amazon, but you must agree to receive only the *worst* customer treatment for all things. One example of this intentionally poor treatment is that Amazon 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

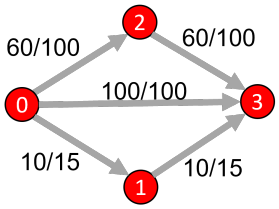
– one row containing an int identifying the destination city.

Output 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

Running Time The worst-case asymptotic running time of your program should belong to $O(c \cdot t)$, where c is the number of cities and t is the number of trucks.

Example

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



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
```