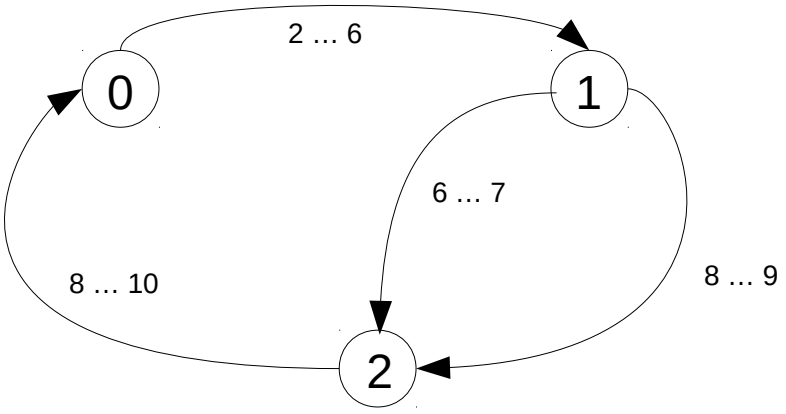
CISC/CMPE-365
Fall 2019
Week 2 Lab Problem: Connecting Flights

Your assignment is to solve a travel-planning problem.

Given a set of cities and a set of flights between them (each flight with a departure time and arrival time) your task is to find the route that starts in specified city A and ends in specified city B, and has the earliest possible arrival time in B. The flight plan may contain any number of flights but each flight in the route must have a departure time **strictly greater** than the arrival time of the previous flight in the route.

For example, suppose the set of cities is {0,1,2} and the set of flights is given by this table:

Source	Destination	Departure Time	Arrival Time
0	1	2	6
1	2	6	7
1	2	8	9
2	0	8	10



All times are given in a universal time frame – no need to worry about time zones etc.

If the task is to fly from City 0 to City 2, the route is 0 – 1 – 2, and the arrival time at City 2 is 9. We cannot take the earlier flight from City 1 to City 2 because its departure time is not greater than the arrival time of the flight from City 0 to City 1. If the task is to fly from City 1 to City 0, the route is 1 – 2 – 0, and the arrival time at City 0 is 10.

If the task is to fly from City 2 to City 1, there is no solution: we can fly from City 2 to City 0, but the only flight from City 0 to City 1 departs before we arrive in City 0.

Your program should be based on a modified version of Dijkstra's Algorithm. The modifications include

- allowing for multiple edges between vertices (as in the example above, in which there are two flights from City 2 to City 3)
- redefining the total weight of a path to be the arrival time of the final flight in the path, instead of the total of the weights of the edges in the path
- making sure no connections are chosen that violate the rule that there must be time between the arrival of one flight and the departure of the next flight in the path

Input to your program consists of the flight information, and possibly the desired source and destination cities. All data values are integers.

Output from your program should list the cities on the flight route (if one exists), and give the arrival time in the target city. If there is no route from the specified start city to the specified end city, your program should print a message to that effect.

The details of input and output are given in the file 2019_Lab_2_IO_Specifications.pdf

Here is a small but non-trivial set of cities and flights on which you can test your program:

Source	Destination	Departure Time	Arrival Time
0	1	1	2
0	1	3	6
0	2	2	8
0	3	4	8
1	2	7	9
1	3	3	4
2	0	1	2
2	1	2	4
2	3	1	4
2	3	7	8
3	0	1	3
3	0	6	8
3	1	2	4
3	2	5	6

This data set is also available as one of the data files provided:
2019_Lab_2_flights_test_data.txt

The optimal route from City 0 to City 2 is 0 – 1 – 3 – 2, arriving at time 6 (unless I have missed a better one). You may want to work out other optimal routes between cities and confirm that your program finds the correct solutions.

Practical considerations:

Your solution is due at 11:59 PM on September 21 . Late assignments will be penalized 10% per day. No assignments will be accepted after September 25.

You should write your program in Java, C, C++ or Python.

To complete the assignment, run your program on the data file 2019_Lab_2_flights_real_data.txt and report the answer to the following question:

What is the optimal route (if any) from the city corresponding to the last two digits of your student number to the city corresponding to the third-last and fourth-last digits of your student number?

Example: If your student number is 56821194, you must report the optimal route from City 94 to City 11.

If the two cities happen to be the same, that's ok! Your program does not need to figure out which cities to use – you can simply write these numbers into your code if you wish. In the given data file, approximately 2% of city combinations do not have valid routes.

Your submitted solution must include

- Your source code, which must contain your name, student number, and the following statement: "I certify that this submission contains my own work, except as noted." Any part of your code which was written by someone else must have a comment identifying the author and/or source.

- A text file giving your answer to the question stated above.

You should combine the pieces of your submission in a .zip file and upload the .zip file to OnQ. The name of the .zip file should start with your student number.

You may upload multiple submissions. Only the last one will be graded.