# CSE 202
# ALGORITHMS II
# Spring 2025

## ASSIGNMENT
***Due Date:*** *Saturday, May 10th, 2025, 23:59*

***Assignment Submission:*** ***<u>You can do this assignment as a team of 2.</u>*** *Turn in your assignment by the due date through LMS. For your answer,* ***you should use the template_assignment.ipynb*** *file and should follow the format provided in the template file. Save the file as '****<first name1>_< last name1>_<first name2>_<last name2>_assignment.ipynb****' (<u>Only one team member should submit the assignment</u> ). Put your names and student IDs to the top cell of the file as comment.Run all the cells and upload the file to LMS.*

***All work in your submission must be your own; you must neither copy from (including online resources) nor provide assistance to anybody else. Do not show or share your code with others. <u>You will be equally responsible if your solution ends up in somebody else's assignment.</u>***

If you need guidance for this assignment and have questions, stop by my office.

***READ CAREFULLY!!!*** ***You are <u>required to use graph algorithm codes that are shared in the lectures</u> in the implementation of this assignment. Appropriate graph algorithm code has been given to you in template_assignment.ipynb. Start with this file and make the necessary additions to complete this assignment. Graph code can be used as it is. Use it to create a graph representation of the problem below. Provided graph code can already calculate the shortest distance. However, you need to make additions to the graph code to find the complete shortest path.***

***<u>If you do not use the provided graph code (with necessary additions) to implement this assignment, your assignment will NOT be accepted and you will not get any credit.</u>***

You are given a text file as input (***cities_and_neighbors.txt***). In this text file, you will find all 81 provinces in Turkiye and their neighbors (There may be some mistakes in the list of neighbors. Use the input file as it is for this assignment). In each line in text file, first value is province name and subsequent values (separated by commas) are the neighbors.

In this assignment, we will assume that there is a direct road from each province to each of its neighbors. Given two provinces (fore example, departure is Istanbul and destination is Mugla), your goal is to find the path between them that goes though ***minimum number of provinces.***

***You are required to solve this question with a graph algorithm. And you are also required to use the graph code (bfs - Breadth First Search algorithm implementation) provided to you.*** Represent each province and its neighbours with a graph. Essentially, each province can be represented with a node (use addNode() of Graph class) and the neighbours can be represented with edges (use addEdge() of Graph class). After you get the graph representation, you can use the *BFS* shortest path algorithm in your solution. ***bfs*** method of Graph class already calculates the shortest distance. Modify graph code to find the actual shortest path as well. *Hint: You can use* ***prev*** variable (previous node in the shortest path) in Node class. You can look at *Dijkstra's* shortest path algorithm in lecture slides to see how *prev* is updated as discussed in class as well. Apply same method to BFS. Then use *prev* values to trace back from destination city to departure city that will be the shortest path from departure city to destination city.

# Part A

Write a function named '***find_shortest_path(departure, destination)'*** that finds the shortest path (that goes through minimum number of provinces) from the **departure** province to the **destination** province. If there is more than one possible shortest path, only reporting one is sufficient.

*Input: departure and destination provinces*

*Output:* **Print** the *minimum number of* provinces has to be visited (including destination) to go from departure to destination and the *actual path followed*.

Example: function can called as follow:

find_shortest_path("Istanbul", "Mugla")

Output is: (This is an example output. May not be the correct answer)

***Minimum number of provinces*** : 6

***Shortest Path***: Istanbul, Kocaeli, Bursa, Balikesir, Izmir, Aydın, Mugla

# Part B

In the second part of this assignment, you have the same goal as Part A (and you will use the same graph and BFS algorithm); but you have to ***make a stop in a city*** while traveling from departure city to destination city. Again you have make sure that you complete the trip by visiting minimum number of cities.

Write a function named '***find_shortest_path_with_stop(departure, destination, stop)'*** that finds the shortest path (goes through minimum number of provinces) from the **departure** province to the **destination** province while making sure visiting **stop** province (You have to pass through stop). If some cities are visited multiple times, count each visit as an additional step.

*Input: departure, destination and stop provinces*

*Output:* **Print** the *minimum number of* provinces has to be visited (including destination) to go from departure to destination while making sure passing through stop **and** the *actual path followed*.

Example: function can called as follows:

find_shortest_path_with_stop("Istanbul", "Mugla", "Kutahya")

Output is: (This is an example output. May not be the correct answer)

***Minimum number of provinces*** : 6

***Shortest Path***: Istanbul, Kocaeli, Bursa, Kutahya, Manisa, Aydın, Mugla

find_shortest_path_with_stop("Istanbul", "Mugla", "Konya")

Output is: (This is an example output. May not be the correct answer)

***Minimum number of provinces*** : 7

***Shortest Path***: Istanbul, Kocaeli, Sakarya, Bolu, Ankara, Konya, Antalya, Mugla

Your ***find_shortest_path and find_shortest_path_with_stop*** functions should run (should not give an error) and print an output to get any credit in this assignment. However, you can get partial credit based on the completeness and correctness of output.