CS102L Data Structures & Algorithms
Lab Week 10

Department of Computer Science

Dhanani School of Science and Engineering

Habib University

Spring 2024

Contents

Introduction

1.1 Instructions

- This lab will contribute 1% towards your final grade.
- The deadline for submission of this lab is at the end of lab time.
- The lab must be submitted online via CANVAS. You are required to submit a zip file that contains all the *.py* files.
- The zip file should be named as *Lab_10_aa1234.zip* where *aa1234* will be replaced with your student id.
- **Files that don't follow the appropriate naming convention will not be graded.**
- **Your final grade will comprise of both your submission and your lab performance.**

1.2 Marking scheme

This lab will be marked out of 100.

- 50 Marks are for the completion of the lab.
- 50 Marks are for progress and attendance during the lab.

1.3 Lab Objectives

- To learn how to create graphs in Python
- To learn how to manipulate graphs in Python
- To learn how to create graph applications using the different helper functions of graphs

1.4 Late submission policy

There is no late submission policy for the lab.

1.5 Use of AI

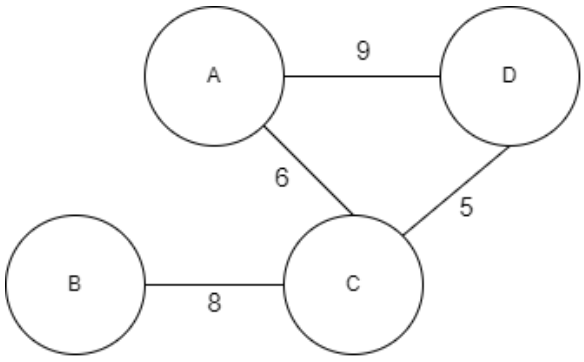
Taking help from any AI-based tools such as ChatGPT is strictly prohibited and will be considered plagiarism.

1.6 Viva

Course staff may call any student for Viva to provide an explanation for their submission.

1.7 Prerequisites: Dictionaries

For this lab, you will be working in Python Structure Dictionary. Please recap your concepts about dictionaries from CS101 as they will be required for this lab. Have a look at following undirected weighted graph:



A possible implementation of this Graph using dictionary is given below:

```
G= {
"A": [("D",9),("C",6)],
"B": [("C",8)],
"C": [("A",6),("B",8),("D",5)],
"D": [("A",9),("C",5)],
}
```

Here, each Key is representing a node and each value is a list of tuples where each tuple is storing the node and weight to which the node in Key is connected to.

Lab Exercises

2.1 Helper Functions

Helper Function are really important for working on graphs. Please make the helper functions given below. Please note that Graph in each of the function given below is in adjacency list format which is stored in a python dictionary. Please utilize helper_functions.py file for implementing them:

- **addNodes(G, nodes):** This function will take a graph G and a list of nodes as parameters. It will add the nodes in the list in G.

```
>>>G = {}
>>>nodes = [0, 1, 2, 3, 4, 5]
>>>addNodes(G, nodes)
>>>print(G)
{ 0: [], 1: [], 2: [], 3: [], 4: [], 5:[] }
```

- **addEdges(G, edges, directed = False):** This function will take a graph G and a list of edges E (list of tuples where each tuple represents an edge and has 3 values: node1, node2, weight) as parameters. It will add the edges in the graph G.

- **listOfNodes(G):** This function will take a graph G as a parameter and return a List of the Nodes in the graph G.

```
>>>print(listOfNodes(G))
[0, 1, 2, 3, 4, 5]
```

- **listOfEdges(G, directed = False):** This function will take a graph G as a parameter and return a List of the Edges in the graph G.

```
>>>print(listOfEdges(G))
[(0, 1, 1), (0, 2, 1), (1, 2, 1), (1, 3, 1), (2, 4, 1),
   (3, 4, 1), (3, 5, 1), (4, 5, 1)]
```

- **getNeighbors(G, node):** The function will take an undirected graph G and a node as a parameter and will return the list of its neighboring nodes.

```
>>>G = { 0: [(1, 1), (2, 1)], 1: [(0, 1), (2, 1), (3, 1)
   ], 2: [(0, 1), (1, 1), (4, 1)], 3: [(1, 1), (4, 1),
   (5, 1)], 4: [(3, 1), (2, 1), (5, 1)], 5: [(3, 1), (4,
    1)] }
>>>print(getNeighbors(G, 0))
[1, 2]
```

- **getNearestNeighbor(G, node):** The function will take a weighted undirected graph G and a node and return its nearest neighboring node (neighbor node having lowest weight). This function will return just 1 node which can be anyone.

- **removeNode(G, node):** This function will take a graph G and a node as a parameter. It will remove that node and all edges of that node. You may need to update other values in dictionary to remove a particular node.

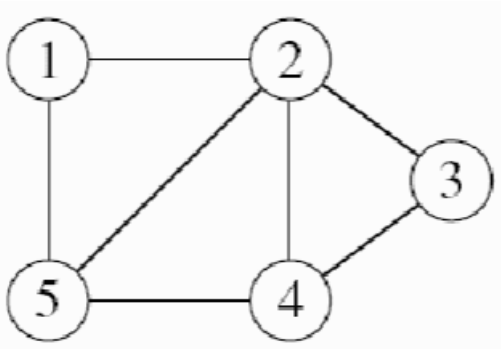
- **removeNodes(G, nodes):** This function will take a graph G and a list of nodes as parameters. It will remove the nodes in the list in G.

- **displayGraph(G):** This function will take a graph G as a parameter and print Adjacency list representation of the graph G.

```
>>>G = { 0: [(1, 1), (2, 1)], 1: [(0, 1), (2, 1), (3, 1)
   ], 2: [(0, 1), (1, 1), (4, 1)], 3: [(1, 1), (4, 1),
   (5, 1)], 4: [(3, 1), (2, 1), (5, 1)], 5: [(3, 1), (4,
    1)] }
>>>DisplayGraph(G)
G = { 0: [(1, 1), (2, 1)], 1: [(0, 1), (2, 1), (3, 1)],
   2: [(0, 1), (1, 1), (4, 1)], 3: [(1, 1), (4, 1), (5,
   1)], 4: [(3, 1), (2, 1), (5, 1)], 5: [(3, 1), (4, 1)]
    }
```

2.2 Question 1

Given the following undirected graph, let's call it UG = (V, E)

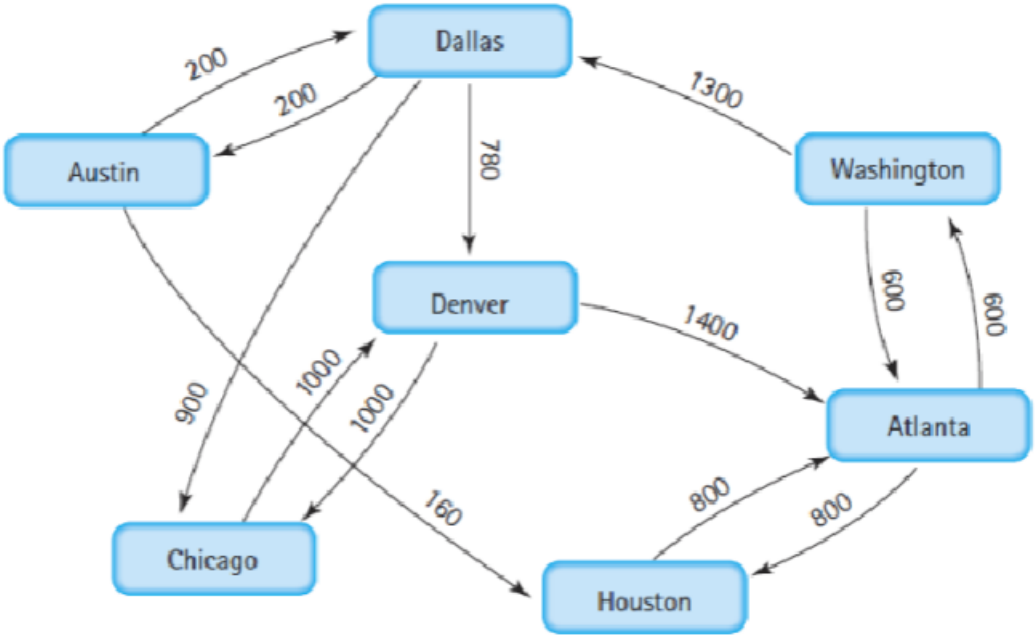


Write a Python program to:

1. Build the adjacency list representation of the graph using appropriate helper functions.
2. Display the list of nodes/vertices in the graph.
3. Display the list of edges in the graph.
4. Display the neighbors of the above graph.

2.3 Question 2

The following graph is an example from Rosen (2011). It shows the flights and distances between some of the major airports in the United States.



1. Write a program to create adjacency list representation of the given directed and weighted graph using appropriate helper functions.

2. Write a function that returns all pair of airports (A1, A2) having only one-way connection either from A1 to A2 or A2 to A1.

3. Write a function that returns the nearest airport from a given airport A.

4. Write a function that returns all airports that are connected to a given airport A with not more than one intermediate airport. This function will take Graph G and node A as input parameters and return the list of airports.

5. Aliens decided to invade US and they captured Washington. Hence, Washington is no more part of United States's graph (your current graph). US decided to build another root from Atlanta to Dallas and vice versa in absence of Washington whose weight will be 1700. Update your graph with this information and display it.