Project Report for Assignment 6 – Network Graphs & OO design

Introduction:

- Purpose:

This project is used to find out the “best” route to guide a self-driving van between various campus buildings to automate campus mail delivery. It contains a list of buildings and following with their mail codes. User only need to enter the mail codes of two buildings and the shortest path will be calculated automatically.

- Region:

This project selects a region within the campus:
Green Street -> Goodwin Avenue -> Gregory Street -> 6th Street.

Within the region, 20 buildings are selected:
Altgeld Hall 382
Arcade Bldg 303
Armory 532
Burrill Hall 114
Chem & Life Sci Lab (CLSL) 123
Coble Hall 322
Davenport Hall 148
Foreign Language Buildings 164
Gregory Hall 462
Ice Arena 525
Illini Hall 374
Illini Union BookStore 323
Library & Information Sci 493
Lincoln Hall 456
Medical Sciences Building 714
Observatory 190
Psych Bldg 716
SPEECH AND HEARING SCIENCE 482
School of Nursing 186
Undergrad Library 522

- Assumption:

1. Each building has its own unique mail code.
2. Each building only has one spot where the building driveway connect to the street.
3. The distance from a building to its point of road access can be ignored when calculate the “best” routine.
4. Buildings and their point of road access are treated as the same spot.

Task Assignment:

This project is divided into three process: analyze requirements and design the structure of program, write codes to implement functions, test and write report.

Group Members:

- Himanshu P.

I formed a bi-directional graph and defined edges for the nodes and intersections. Then I connected all the nodes and intersections from those edges and allocated weights (distance). I used Google maps for accurate distance. I also updated few edges defined by my project member Ke.

- Jiawei L.

In this project, at beginning, I and my team members analyzed the requirements and selected the region, buildings and streets that will be contained in our project. Then I created the class and identified the functions in the class. I wrote all nodes' information and functions to add nodes and edges in the directed graph and find out the best routine. After all edges were added, I and Shruti did some tests to verify the correctness of our program and wrote the report.

- Ke G.

I created the distribution of selected buildings and intersections by using Google map. Then modified some buildings' access and roads' names. And I finished the first 17 buildings' edges.

- Shruti S.

I contributed to this project by writing function to display the output in the desired format, writing the doctest for print_shortest_path and the docstrings for all functions. I did the unit testing for functions: print_all_buildings, get_building_name and print_shortest_path. After the addition of all the edges as well nodes by Jiawei, Ke and Himanshu, I did the integration test for 10 of the 20 buildings.

Implementation:

This project writes an object-oriented program written in Python 3 and uses Python NetworkX library to create a directed graph that connects all buildings and intersections of the streets within the selected region. The program is consisted by one class and eight functions, which are mainly used to create nodes, edges, find the shortest path, and get the output. The details will be introduced later.

- Data Structure

I. Graph

Using direct graph to simulate the connections of campus buildings and roads around them. Nodes in the graph are represent the buildings and intersections joining the streets. Each edge connects to the node has a direction which represents a one-way street.

II. Nodes

Buildings and intersections are represented as nodes in the graph.
Buildings' information are stored in a dictionary. The key value is the building's name. Each key has its corresponding value pairs which store the address and mail code of that building.

```
self.buildings = {building_name:{mail_address, mail_code}}
```

Intersections' information are stored in a list. Each intersection is named by combining the name of streets which it is joining.

```
self.intersection = [street1_name-street2_name]
```

III. Edges
Buildings and intersections are connected by using directed edges which represent the street between two nodes.
Edges information is stored in a nested dictionary. Keys in the outer dictionary are the names of the buildings and intersections. The value pairs of each key are inner dictionary which contains the information of its adjacent nodes. The key of each inner dictionary is the name of the adjacent node which can be a building or an intersection. Its value pairs are the distance between two nodes, the name of the street and the direction of that street.

self.edge = {node_name:{adjacent_node:{distance, street_name, street_direction}}}

- Class

This project is an OO-solution by implementing a class 'BuildingGraph' that provides basic functions to create and manipulate a graph of buildings and roads between them.

- Functions

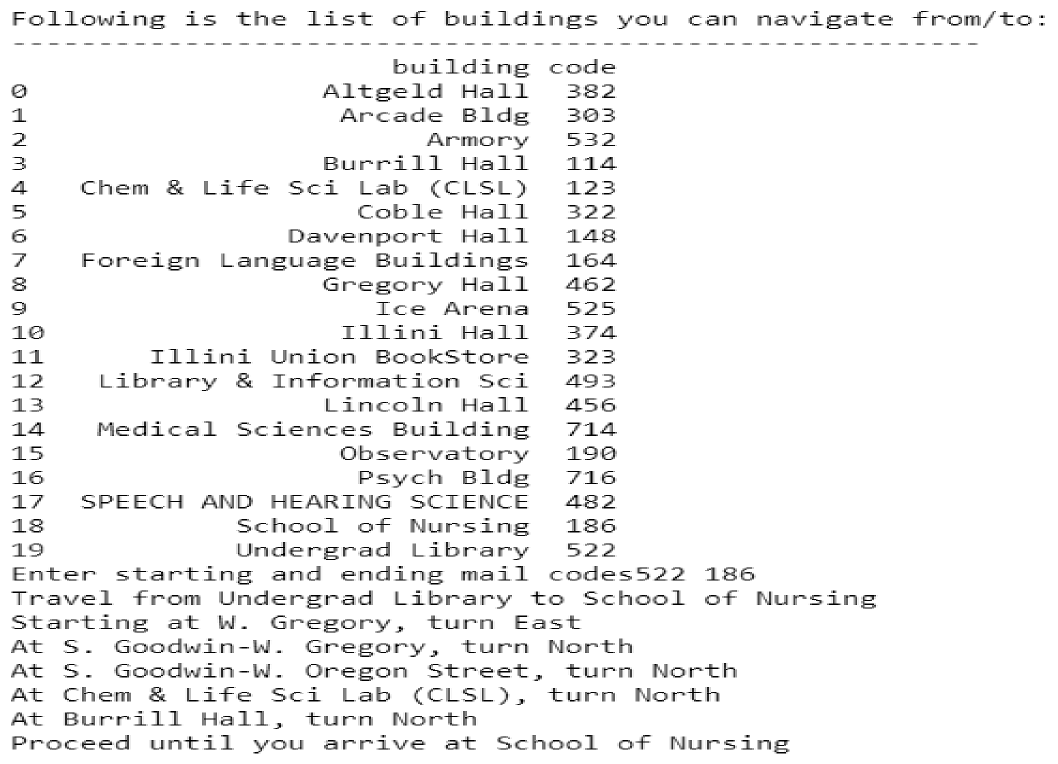
Our solution has 6 functions.

I. add_building_nodes: This method takes a directed graph as input and adds all the buildings as the nodes. It then returns a graph having all the buildings added to it.
II. add_intersection_nodes: This method takes a directed graph as input and adds all the intersections as the nodes. It then returns a graph having all the intersection nodes added to it.
III. add_edges: This method takes a directed graph as input and adds all the possible edges between the nodes. It then returns a graph having all the edges between nodes added to it.
IV. print_all_buildings: This function takes the complete graph with all the nodes and edges and constructs a dataframe from it having the building name and mailcode. The output is a dataframe sorted on basis of building name.
V. get_building_name: This method prints out the building names corresponding to the input mail code.
VI. print_shortest_path: This method takes a directed graph and a list of edges on the shortest path between source and target and prints out the shortest path in a desired path.

Results

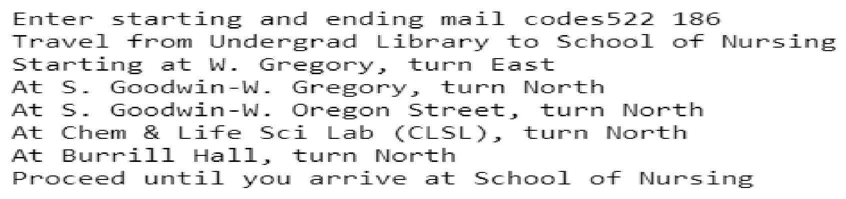
The output of this solution is as follows:

I. The list of buildings that you can navigate to/from using this solution:



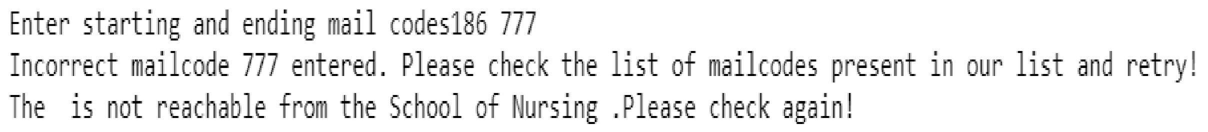
```
Following is the list of buildings you can navigate from/to:
--------------------------------------------------------
                          building code
0                    Altgeld Hall  382
1                     Arcade Bldg  303
2                          Armory  532
3                    Burrill Hall  114
4    Chem & Life Sci Lab (CLSL)  123
5                      Coble Hall  322
6                  Davenport Hall  148
7    Foreign Language Buildings  164
8                    Gregory Hall  462
9                      Ice Arena  525
10                   Illini Hall  374
11        Illini Union BookStore  323
12     Library & Information Sci  493
13                   Lincoln Hall  456
14      Medical Sciences Building  714
15                    Observatory  190
16                      Psych Bldg  716
17    SPEECH AND HEARING SCIENCE  482
18              School of Nursing  186
19              Undergrad Library  522
Enter starting and ending mail codes522 186
Travel from Undergrad Library to School of Nursing
Starting at W. Gregory, turn East
At S. Goodwin-W. Gregory, turn North
At S. Goodwin-W. Oregon Street, turn North
At Chem & Life Sci Lab (CLSL), turn North
At Burrill Hall, turn North
Proceed until you arrive at School of Nursing
```

II. Shortest Path from source to target



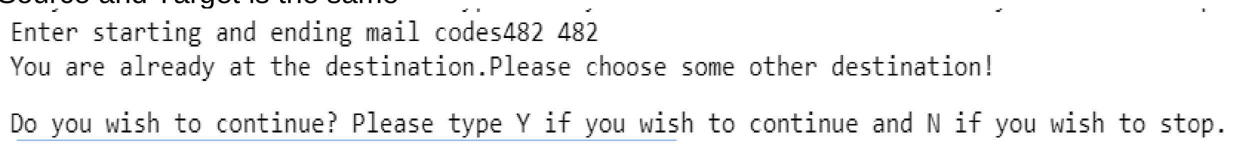
```
Enter starting and ending mail codes522 186
Travel from Undergrad Library to School of Nursing
Starting at W. Gregory, turn East
At S. Goodwin-W. Gregory, turn North
At S. Goodwin-W. Oregon Street, turn North
At Chem & Life Sci Lab (CLSL), turn North
At Burrill Hall, turn North
Proceed until you arrive at School of Nursing
```

III. Invalid Mail code entered



```
Enter starting and ending mail codes186 777
Incorrect mailcode 777 entered. Please check the list of mailcodes present in our list and retry!
The  is not reachable from the School of Nursing .Please check again!
```

IV. Source and Target is the same



```
Enter starting and ending mail codes482 482
You are already at the destination.Please choose some other destination!

Do you wish to continue? Please type Y if you wish to continue and N if you wish to stop.
```

Conclusion:

The project has been implemented successfully.