Work Integrated Learning Programmes Division
M.Tech (Data Science and Engineering)

(S1-20_DSECFZG519)

(Data Structures and Algorithms Design)

Academic Year 2020-2021

Assignment 1 – PS22 - [Freight Booking] - [Weightage 12%]

1. Problem Statement

A startup company is venturing into the space of online railway freight booking and wants your help in developing a system which can record all railway freight routes that are available between cities in India. The data is captured such that each freight train and its associated cities are captured as vertices and the association as edges. Assume that the trains are bi-directional, that means the same train number is used for the onward and return journey.

As a first phase, they want to identify the following information.

1. List of unique freight trains and list of unique cities that have freight service.
2. Find out which city is the main transport hub. (City which is visited by the greatest number of trains)
3. Find out which cities are connected by a single train.
4. If a package needs to be sent directly from city a to city b, which train should they book?
5. Can a package be sent from city a to city b even if it must be transferred (change trains) at an intermediary city c (any number of transfers).

Requirements:

1. **Implement the above problem statement in Python 3.7 using graph data structures and adjacency matrix. Do not use the inbuilt Python data structure.**
2. **Perform an analysis for the features above and give the running time in terms of input size: n.**

Operations:

1. **def readCityTrainfile(self, inputfile)**: This function reads the input file **inputPS22.txt** containing the name of the cities and the freight trains between them in one line separated by a slash. A sample input file entry is shown below. The Train number is the first entry in each row followed by the different cities it services separated by a slash '/'

   ```
   T1235 / Chennai / New Delhi
   ```

   The function should create relevant vertices for both the freight train and its associated cities and relevant edges to indicate the association of a train and its connecting cities. Ensure that the vertices are unique and there are no duplicates.

2. **def showAll(self):** This function displays the count of unique freight trains and cities entered through the input file. It should also list out the unique freight trains and cities that have freight service stored. This function is called after all input data has been captured. The output of this function should be pushed into **outputPS22.txt** file. The output format should be as mentioned below.

   ```
   --------Function showAll --------

   Total no. of freight trains: 5

   Total no. of cities: 8

   List of Freight trains:

   T1235

   T2342

   T1122

   T2341

   T5623


   List of cities:

   Chennai

   New Delhi

   Ahmedabad

   Mumbai
   ```

```
Nagpur

Hyderabad

Calcutta

Vishakhapatnam

---------------------------------------
```

Note: This is only an indicative output and not the actual output of the program.

3. **def displayTransportHub(self):** This function displays the name of the city which is visited by the greatest number of trains. The function also displays the names of the incoming freight trains to the **outputPS22** file. The function is triggered when the 'searchTransportHub' tag is found in the file **promptsPS22**.txt file.

   searchTransportHub:

   The output of this function should be appended into **outputPS22.txt** file. The output format should be as mentioned below.

```
--------Function displayTransportHub --------

Main transport hub: New Delhi

Number of trains visited: 3

List of Freight trains:

T1235

T2342

T2341

-----------------------------------------
```

Note: This is only an indicative output and not the actual output of the program.

4. **def displayConnectedCities(self, train):** This function displays all the cities are connected by a single train. The function reads the input freight train number from the file **promptsPS22.txt** with the tag as shown below.

   searchTrain: T1122

   searchTrain: T1235

The output of this function should be appended into **outputPS22.txt** file. If a train is not found, an appropriate message should be output to file. The output format should be as mentioned below.

```
--------Function displayConnectedCities --------

Freight train number: T1122

Number of cities connected: 3

List of cities connected directly by T1122:

Ahmedabad

Mumbai

Nagpur

-----------------------------------------
```

Note: This is only an indicative output and not the actual output of the program

5. **def displayDirectTrain(self, city a, city b):** This function displays the freight train name which can be booked to send a package directly from city a to city b. The function reads the input cities from the file **promptsPS22.txt** with the tag as shown below.

   ```
   searchCities: Calcutta: New Delhi

   searchCities: Chennai: Hyderabad
   ```

   The output of this function should be appended into **outputPS22.txt** file. If there is no direct train or a city is not found, an appropriate message should be output to the file. The output format should be as mentioned below. If there is more than one train that can be booked, the train number you encounter first can be output.

   ```
   --------Function displayDirectTrain --------

   City A: Calcutta

   City B: New Delhi

   Package can be sent directly: Yes, T2342 (if no, display appropriate message)

   -----------------------------------------
   ```

Note: This is only an indicative output and not the actual output of the program

6. **def findServiceAvailable(self, city a, city b):** This function finds whether a package can be sent from city a to city b with any number of stops/transfers (ie to deliver the package from

city a to city b it might even get transferred on another train at an intermediary city c). The function reads the input cities from the file **promptsPS22.txt** with the tag as shown below.

ServiceAvailability: Calcutta: Mumbai

ServiceAvailability: Nagpur: Vishakhapatnam

Also display the entire route to transfer the package from city a to city b. The output of this function should be appended into **outputPS22.txt** file. If the package can't be transferred or a city is not found, an appropriate message should be output to the file. The output format should be as mentioned below.

```
--------Function findServiceAvailable --------

City A: Calcutta

City B: Nagpur

Can the package be sent: Yes, Calcutta > T2342 > New Delhi > T2341 > Ahmedabad > T1122 > Nagpur (if no, display appropriate message)

-----------------------------------------
```

Note: This is only an indicative output and not the actual output of the program

7. Include all other functions that are required to support these basic mandatory functions.

Sample file formats

Sample Input file

The input file **inputPS22**.**txt** contains names of the trains and the connected cities in one line separated by a slash (/).

Sample inputPS22.txt

```
T1235 / Chennai / New Delhi
T2342 / Calcutta / New Delhi
T1122 / Ahmedabad / Nagpur / Mumbai
T2341 / Ahmedabad / New Delhi
T5623 / Vishakhapatnam / Hyderabad
```

Sample promptsPS22.txt

```
searchTransportHub
```

```
searchTrain: T1122
searchTrain: T1235
searchCities: Calcutta: New Delhi
searchCities: Chennai: Hyderabad
ServiceAvailability: Calcutta: Mumbai
ServiceAvailability: Nagpur: Vishakhapatnam
```

Sample outputPS22.txt

```
--------Function showAll --------
Total no. of freight trains: 5
Total no. of cities: 8

List of Freight trains:
T1235
T2342
T1122
T2341
T5623

List of cities:
Chennai
New Delhi
Ahmedabad
Mumbai
Nagpur
Hyderabad
Calcutta
Vishakhapatnam
-----------------------------------------

--------Function displayTransportHub --------
Main transport hub: New Delhi
Number of trains visited: 3
List of Freight trains:
```

```
T1235

T2342

T2341

----------------------------------------

--------Function displayConnectedCities --------

Freight train name: T1122

Number of cities connected: 3

List of cities connected directly by T1122:

Ahmedabad

Mumbai

Nagpur

----------------------------------------

--------Function displayDirectTrain --------

City A: Calcutta

City B: New Delhi

Package can be sent directly: Yes, T2342 (if no, display appropriate message)

----------------------------------------

--------Function findServiceAvailable --------

City A: Calcutta

City B: Nagpur

Can the package be sent: Yes, Can the package be sent: Yes, Calcutta > T2342 > New Delhi >
T2341 > Ahmedabad > T1122 > Nagpur (if no, display appropriate message)

----------------------------------------
```

Note that the input/output data shown here is only for understanding and testing, the actual file used for evaluation will be different.

2. Deliverables

1. Word document **designPS22_<group id>.docx** detailing your design and time complexity of the algorithm.
2. **[Group id]_Contribution.xlsx** mentioning the contribution of each student in terms of percentage of work done. Download the Contribution.xlsx template from the link shared in the Assignment Announcement.
3. **inputPS22.txt** file used for testing
4. **promptsPS22.txt** file used for testing

5. **outputPS22.txt** file generated while testing
6. **.py file** containing the python code. Create a single *.py file for code. Do not fragment your code into multiple files

Zip all of the above files including the design document and contribution file in a folder with the name:

[Group id]_A1_PS22_Freight.zip and submit the zipped file.

Group Id should be given as **Gxxx** where xxx is your group number. For example, if your group is 26, then you will enter G026 as your group id.

3. Instructions

1. It is compulsory to make use of the data structure(s) / algorithms mentioned in the problem statement.
2. Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full. Also ensure basic error handling is implemented.
3. For the purposes of testing, you may implement some functions to print the data structures or other test data. But all such functions must be commented before submission.
4. Make sure that your read, understand, and follow all the instructions
5. Ensure that the input, prompt and output file guidelines are adhered to. Deviations from the mentioned formats will not be entertained.
6. The input, prompt and output samples shown here are only a representation of the syntax to be used. Actual files used to evaluate the submissions will be different. Hence, do not hard code any values into the code.
7. Run time analysis is to be provided in asymptotic notations and not timestamp based runtimes in sec or milliseconds.

Instructions for use of Python:

1. Implement the above problem statement using Python 3.7.
2. Use only native data types like lists and tuples in Python, do not use dictionaries provided in Python. Use of external libraries like graph, numpy, pandas library etc. is not allowed. The purpose of the assignment is for you to learn how these data structures are constructed and how they work internally.
3. Create a single *.py file for code. Do not fragment your code into multiple files.
4. Do not submit a Jupyter Notebook (no *.ipynb). These submissions will not be evaluated.
5. Read the input file and create the output file in the root folder itself along with your .py file. Do not create separate folders for input and output files.

4. Deadline

1. The strict deadline for submission of the assignment is **27th Dec, 2020.**
2. The deadline has been set considering extra days from the regular duration in order to accommodate any challenges you might face. No further extensions will be entertained.
3. Late submissions will not be evaluated.

5. How to submit

1. This is a group assignment.
2. Each group has to make one submission (only one, no resubmission) of solutions.
3. Each group should zip all the deliverables in one zip file and name the zipped file as mentioned above.
4. Assignments should be submitted via Canvas > Assignment section. Assignment submitted via other means like email etc. will not be graded.

6. Evaluation

1. The assignment carries 12 Marks.
2. Grading will depend on
   a. Fully executable code with all functionality working as expected
   b. Well-structured and commented code
   c. Accuracy of the run time analysis and design document.
3. Every bug in the functionality will have negative marking.
4. Marks will be deducted if your program fails to read the input file used for evaluation due to change / deviation from the required syntax.
5. Use of only native data types and avoiding libraries like numpy, graph and pandas will get additional marks.
6. Plagiarism will not be tolerated. If two different groups submit the same code, both teams will get zero marks.
7. Source code files which contain compilation errors will get at most 25% of the value of that question.

7. Readings

Text book: Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, Wiley (Students Edition). **Chapters:** 6