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

(S1-21_DSECLZG519)
(Data Structures and Algorithms Design)
Academic Year 2020-2021

Assignment 1 – PS12 - [Airport Connections] - [Weightage 12%]

1. Problem Statement

The CEO of a leading airline company wants to know if his flights are servicing all major cities /airports such that if a passenger wants to travel from a given starting city, they can reach any of the other cities using either a direct flight (0 stops) or an indirect flight (multiple stops at intermediate cities).

If there are any cities that are not reachable, the CEO wants to know the minimum number of flights (one-way flights) that need to be added so that the passenger can reach any destination city from the starting city using either direct or indirect flights.

You are given a list of city airports (three-letter codes like "JFK"), a list of routes (one-way flights from one airport to another like ["JFK" , "SFO"]), and a starting city / airport.

Note that routes only allow you to fly in one direction; for instance, the route ["JFK", "SFO"] only allows you to fly from "JFK" to "SFO" and not from SFO to JFK

Also note that the connections don't have to be direct (take off at the starting airport and land in the destination airport). It is okay if the destination airport can be reached from the starting airport by stopping at other intermediary airports in between.

Requirements

1. Model the following problem as Graphs Edge List Implementation using Python 3.7
2. Read the input from a file **inputPS12.txt**
3. You will output your answers to a file **outputPS12.txt**
4. Perform an analysis for the features above and give the running time in terms of input size: n.

Sample Input:

All airports will be input through a file **inputsPS12.txt**.

The list of cities / airports serviced by the airline are entered as comma separated values with the keyword airports as shown below.

```
airports = BGI, CDG, DEL, DOH, DSM, EWR, EYW, HND, ICN, JFK, LGA, LHR, ORD, SAN, SFO, SIN, TLV, BUD
```

The list of routes will be listed as individual pairs in the subsequent lines. One line will contain two airport codes separated by a comma. Each line will contain only one route. The starting keyword in the input file to indicate routes will be "routes" as shown below.

```
routes
DSM, ORD,
ORD, BGI,
BGI, LGA,
SIN, CDG,
CDG, SIN,
CDG, BUD,
DEL, DOH,
DEL, CDG,
TLV, DEL,
EWR, HND,
HND, ICN,
HND, JFK,
ICN, JFK,
JFK, LGA,
EYW, LHR,
LHR, SFO,
SFO, SAN,
SFO, DSM,
SAN, EYW,
```

The source or starting airport will be entered through the input file with the keyword "Starting Airport"

```
Starting Airport = LGA
```

Sample Output

Below is the sample output of the program. The output needs to be written into a file **outputPS12.txt**

```
The minimum flights that need to be added = 3
The flights that need to be added are:
[LGA, TLV]
[LGA, SFO]
[LGA, EWR]
```

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 **designPS12_<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. **inputPS12.txt** file used for testing
4. **outputPS12.txt** file generated while testing
5. **.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_PS12_AirportConnections.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.
8. Please note that the design document must include

   a. The data structure model you chose with justifications
   b. Details of each operations with the time complexity and reasons why the chosen operations are efficient for the given representation
   c. One alternate way of modelling the problem with the cost implications.
9. Writing good technical report and well document code is an art. Your report cannot exceed 4 pages. Your code must be modular and quite well documented.

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 **Wednesday, 22nd Dec, 2021.**
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. Copy / Paste's from web resources / or your friends' submission will attract severe penalty to the extent of awarding 0 marks. We will not measure the extent of such blatant copy pastes and details of who copied from whom and such details while awarding the penalties. It's the responsibility of the team to solve and protect your original work.**
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