Birla Institute Of Technology & Science, Pilani
Work Integrated Learning Programmes Division
M.Tech (Data Science and Engineering)

(S1-21_DSECCZG519)

(Data Structures and Algorithms Design)

Academic Year 2021-2022

Assignment 1 – PS07 - [Instructor Mapping] - [Weightage 12%]

1. Problem Statement

At the university there is a requirement to hire course instructors who can handle most of the courses offered in the university so that they can easily map the course instructors to a corresponding course. The university has received a number of resumes from different applicants who can handle a certain number of courses. Our job is to hire the minimum number of candidates that can handle the courses.

For the sake of this exercise, let us assume that an applicant can handle at most 5 courses

Requirements:

Model the following problem as a graph based problem. Clearly state how the vertices and edges can be modeled such that this graph can be used to answer the following queries efficiently

The queries are **:**

1. List the names of candidates and the unique list of courses handled by all of them combined.
2. What is the least number of candidates that need to be hired to cover most of the courses? List the names of the candidates.
3. Generate a list of candidates who can handle a particular course.
4. If a course has two parts Part A and Part B, but there is no single faculty who knows both parts, can it be handled by two course instructors instead? If so, draw the path that enables this mapping
5. Perform an analysis for the questions above and give the running time

Operations:

1. **def readApplications(self, inputfile)**:This function reads the input file **inputPS07.txt** containing the name of the candidate and the languages they know separated by a slash.

   ```
   Manasa / Data structures / Software Engineering / Compiler Design
   ```

   The function should create relevant vertices for both the candidates and the courses they know and relevant edges to indicate the connection between the candidate and the courses. Ensure that the vertices are unique and there are no duplicates

2. **def showAll(self):** This function displays the total number (count) of unique candidates and courses that have been fed into the system. It should also list out the candidate names and unique list of courses stored. The output of this function should be pushed into **outputPS25.txt** file

   ```
   --------Function showAll--------
   Total no. of candidates: 12
   Total no. of Courses: 11

   List of candidates:
   Manasa
   Harish
   Paul
   Amjad
   Parul
   Nisha
   Rohan
   Rahul
   Sanjay
   Surya
   Venkat
   Zubin

   List of Courses:

   Data structures
   Software Engineering
   Compiler Design
   Algorithms Design
   ```

```
Cloud Computing
Software Testing
Data Structures and Algorithm design
Software Architecture
Distributed Computing and Data Storage Technologies
Distributed Computing
Data Storage Technologies
```

3. **def displayHireList(self):** This function displays the minimum number of candidates that need to be hired to cover most of the courses that are fed into the system. The function also displays the names of the candidates and the courses they can handle individually to the **outputPS07.txt** file. The function is triggered when the following tag is found in the file **promptsPS07.txt**

   ```
   showMinList
   ```

   The output of this function should be appended into **outputPS7.txt** file. The output format should be as mentioned below

   ```
   --------Function displayHireList--------
   No of candidates required to cover most of the courses: 3

   Manasa / Data Structures / Compiler Design / Software Engineering
   Venkat / Data Structures / Software Architecture / Cloud Computing / Distributed
   Computing
   Paul / Algorithm Design / Software Testing/ Data Storage Technologies
   ```

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

4. **def displayCandidates(self, course):** This function displays all the candidates who can handle a particular course. The function reads the input course from the file **promptsPS07.txt** with the tag as shown below

   ```
   searchCourse: Data Strcutures
   searchCourse: Algorithm Design
   ```

   The output of this function should be appended into **outputPS07.txt** file

   ```
   --------Function displayCandidates --------

   List of Candidates who can handle Data Structures:

   Manasa
   Venkat
   ```

5. **def findCourseInstructors(self, PartA, PartB):** Use one of the traversal techniques to find out which instructor can handle part A and which instructor can handle part B. The function reads the input courses from the file **promptsPS07.txt** with the tag as shown below

```
--------Function findCourseInstructors --------

Part A: Data Structures

Part B: Algorithm Design

Course Instructors:

Part A: Manasa, Venkat

Part B: Paul
```

Sample file formats

Sample inputPS07.txt

The input file **inputPS07.txt** contains names of the candidates and the languages they speak in one line separated by a slash (/).

```
Manasa / Data Structures / Compiler Design / Software Engineering

Venkat / Data Structures / Software Architecture / Cloud Computing / Distributed Computing

Paul / Algorithm Design / Software Testing/ Data Storage Technologies

Harry/Cloud Computing / Distributed Computing

Mary/Software Architecture/ Software Engineering
```

Sample promptsPS07.txt

```
showMinList
searchCourse: Data Strcutures
searchCourse: Algorithm Design
Part A: Data Structures
Part B: Algorithm Design
```

Sample outputPS07.txt

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

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

Total no. of candidates: 12

Total no. of Courses: 11

List of candidates:

Manasa
Harish
Paul
Amjad
Parul
Nisha
Rohan
Rahul
Sanjay
Surya
Venkat
Zubin

List of Courses:

Data structures
Software Engineering
Compiler Design
Algorithms Design
Cloud Computing
Software Testing
Data Structures and Algorithm design
Software Architecture
Distributed Computing and Data Storage Technologies
Distributed Computing
Data Storage Technologies
-----------------------------------------------------------------------------------------------------

--------Function displayHireList--------

No of candidates required to cover most of the courses: 3

Manasa / Data Structures / Compiler Design / Software Engineering
Venkat / Data Structures / Software Architecture / Cloud Computing / Distributed
Computing
Paul / Algorithm Design / Software Testing/ Data Storage
Technologies-----------------------------------------------------------------------------------------
----------------

---------------------Function displayCandidates --------
```

List of Candidates who can handle Data Structures:

Manasa
Venkat

-------Function findCourseInstructors --------

Part A: Data Structures

Part B: Algorithm Design

Course Instructors:

Part A: Manasa, Venkat

Part B: Paul

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 **designPS6_<group id>.docx** detailing your design and time complexity of the algorithm and a contribution table as below

Roll Number	Name	Contribution (%)

** Note: Each group should upload one submission only (Individual submissions are not valid)

2. **inputPS7.txt** file used for testing
3. **outputPS7.txt** file generated while testing
4. **.ipynb** (python notebook) file containing the python code. Create a single notebook. Do not fragment your code into multiple files

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

[Group id] _A1_PS7_StudentList.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 or above
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 notebook file for code. Do not fragment your code into multiple files.
4. Read the input file and create the output file in the root folder itself along with your notebook file. Do not create separate folders for input and output files

4. Deadline

1. The strict deadline for submission of the assignment is **10th Dec, 2021 11:55 PM**

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