Data structures and algorithms Grado Ciencia e Ingeniería de Datos Universidad Carlos III de Madrid Lab Case.	uc3m | Universidad Carlos III de Madrid

Author: Isabel Segura-Bedmar

Creating a social network for students

Our university has a project to create a new social network for the students at the university by integrating existing social networks managed by departments and faculties.

The goal of the current exercise is to define and implement the different data structures and algorithms supporting the mentioned social network according to the requirements specified at each phase.

Phase I (Linked lists)

Considering that, in a first stage, the social network will include users of existing social networks and considering that doubly linked lists can represent those existing networks, you must implement the following data structures:

- A **Student** data structure that stores information about each student. Its attributes are:
  - **email**: this is the email for the student in the social network. Each email is unique. That is, there cannot be different students with the same email
  - **city:** city of residence.
  - **campus:** It has one of the following values: GETAFE, LEGANES, COLMENAREJO.
  - **blocks**: Students in the network can block other students to forbid them communicating with them. This attribute stores the number of times this student has been blocked by other users.
  - **date_sign_in**: The date when the user registered in the social network. The Python **datetime** module allows you to work with dates and times in a simple way. In this link, https://www.programiz.com/python-programming/datetime#date, you can find information and examples about how to use datetime.
- A **StudentsList** data structure to contains all the students registered in a social network. This data structure must be a linked list and implement all its methods. A social network must never contain repeated students. For

this reason, the methods *addLast*, *addFirst* and *insertAt* must always check if the student exists or not, before adding it into the list.

Moreover, you must implement a new data structure, **ManageNetworkList,** with the following methods:

1. Create a method, named **merge**, that merges two social networks into one by alternating their students. That is, the method takes two objects of the StudentsList and returns a new list which alternates the students from both social networks. For example, *L1={A,B,C,D}, L2={E,F}, merge(L1,L2)={A,E,B,F,C,D}.* The new list must not contain repeated students.

2. Create a method **getCampusCity** taking a social network (that is, an object of the StudentsList class) as input and an integer parameter opc so that:
   - If opc =1: the method must return a StudentsList object containing all the students residing in the same city that the campus where they are studying.
   - If opc =2: the method must return a StudentsList object containing all the students residing in cities different that the one where their campus is located.

Note: The order in the resulting list must be the same that in the input list.

3. Create a method **locateByCity** taking a social network (that is, an object of the StudentsList class) and a city name as input and returning a list containing all the students (that is, an object of the StudentsList class) who live in that city.

Note: The order in the resulting list must be the same that in the input list.

4. Create a method **orderBy** having a social network (that is, an object of the StudentsList class) as input and an integer parameter opc so that:
   - If opc=1, the method returns a new list of students (that is, an object of the StudentsList class) sorted by ascending order according to their email.
   - If opc=2, the method returns a new list of students (that is, an object of the StudentsList class) sorted by descending order according to their email.

Note 1. You must implement your own sort method based on some of the sorting algorithms (such as bubble or insertion)[1].

Note 3. The input list cannot be modified. The method must return a new list where the students are sorted.

5. Create a method **getStudentsByDateInterval**, which takes a social network (an object of the StudentsList class) and two dates (start and end), and returns the list of all students from the input social network

[1]https://www.tutorialspoint.com/python/python_sorting_algorithms.htm

whose registration dates are in this interval of dates. Please, consider the following comments:
    a. start <= end.
    b. Both dates are included into the interval.
    c. The order in the resulting list must be the same that in the input list.
6. Write a table with the Big-Oh functions for the StudentList's and **ManageNetwork**'s methods. Please, include a brief explanation for each method. Moreover, you should discuss for every method its best and worst case.

Phase II (binary search trees)

As the number of students increases, access time when searching for students by username is increasing to unacceptable limits. In order to improve access time, you must implement a binary search tree, **StudentsTree**, to store students from a given social network. This class must implement the common methods of a binary search tree: insert, find and remove. Moreover, you must add the following methods:

1. Create a method, **copySocialNetwork**, that takes an object of the StudentsList class (phase 1) and inserts each student from the list into the invoking tree. The method does not return anything.
2. Create a method **getOrderedList** that returns an object of the StudentsList class containing all the students in the tree ordered by their email. In order to obtain this list, you cannot use any sort-algorithm (such as bubble, etc), you must traverse the tree in a recursive way.
3. Create a method **deleteByNumberOfBlocks** having an integer n as parameter. The method must remove all students having a number of blocks equal or greater than n. Hint: You can use an auxiliary and recursive method. Moreover, you can use the method remove from the StudentsTree to remove the nodes.
4. Members of the social network could also be searched by their sign-in date. Is the StudentsTree class an efficient data structure for this kind of searchers? Reason your answer. If you consider that it is not efficient, please, describe a more efficient data structure for supporting this kind of searches.

Phase III (graph)

One of the capabilities of a social network is allowing for users to follow their friends or people sharing interesting contents. There is a need to provide a data structure supporting this capability. As you can imagine, the most adequate data structure is a graph. This phase asks to provide an implementation of the social network based on the graph data structure. Please, name it as **StudentGraph**. To simplify your task, the graph only has to store the emails of

the students (see Note). You must consider that the university has more than 20 thousand students.

1. What graph representation is the most suitable for such a large number of possible users? Explain your answer.
2. Create a **constructor** for the StudentGraph taking a Python list with the students' emails registered in the social network. When an object of StudentGraph class is just created, there will not be any relationship between the students. In other words, the constructor method only stores the emails (students) into the graph, without creating any relationship between them.
3. Create a method **addStuden**t taking an email as input and adding it to the social network.
4. Create a method **areFriends** taking two students (emails) as input and creating a friendship relation between them. Keep in mind that friendship relation is a symmetric relationship.
5. Create a method **getDirectFriends** that, given a student (email), returns a Python list containing the emails of his/her direct friends.
6. Create a method **suggestedFriends** that, given a student, returns a Python list containing the emails of his/her potential friends. That is, the method must find those students connected with the given student but having no direct link with him/her. Hint: Some of the graph traversal algorithms may be useful to implement the solution.