Lab 7: Cliques

The questions below are due on Monday October 30, 2017; 10:00:00 PM.

You are not logged in.

If you are a current student, please Log In (https://6009.csail.mit.edu/fall17/labs/lab7?loginaction=redirect) for full access to the web site.
Note that this link will take you to an external site (`https://oidc.mit.edu`) to authenticate, and then you will be redirected back to this page.

Table of Contents

- 1) Preparation (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_1)
- 2) Introduction (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_2)
- 3) Testing your lab (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_3)
- 4) Representation (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_4)
    - 4.1) Friendship (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_4_1)
- 5) Setting Up the School (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_5)
    - 5.1) Insertion (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_5_1)
    - 5.2) Deletion (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_5_2)
    - 5.3) Updating (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_5_3)
- 6) Cliques (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_6)
    - 6.1) Independent Set (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_6_1)
- 7) Code Submission (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_7)
- 8) Checkoff (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_8)
    - 8.1) Grade (https://6009.csail.mit.edu/fall17/labs/lab7#catsoop_section_8_1)

1) Preparation

This lab assumes you have Python 3.5 or later installed on your machine.

The following file contains code and other resources as a starting point for this lab: `lab7.zip` (https://6009.csail.mit.edu/fall17/lab_distribution.zip?path=%5B%22fall17%22%2C+%22labs%22%2C+%22lab7%22%5D)

Most of your changes should be made to `lab.py`, which you will submit at the end of this lab. Importantly, you should not add any imports to the file. You may submit portions of the lab late (see the grading page (https://6009.csail.mit.edu/fall17/grading) for more details), but the last day to submit this lab will be the Friday after the due date.

This lab is worth a total of 4 points. Your score for the lab is based on:

- correctly answering the questions throughout this page (0.1 points)
- passing the test cases from `test.py` under the time limit (1.9 points), and
- a brief "checkoff" conversation with a staff member to discuss your code (2 points).

Your points for `test.py` are based on how quickly your code runs on the server:

- all tests correct and complete in less than 2 seconds each: 1.9 points
- all tests correct and complete in less than 5 seconds each: 1.5 points
- all tests correct and complete in less than 30 seconds each: 1 point

Note that each of the tests will be run in its own process.

Please also review the collaboration policy (https://6009.csail.mit.edu/fall17/collaboration) before continuing.

2) Introduction

The students of North Shore High School are notoriously cliquey. Tina Fey and Tim Meadows try their best to forge relationships between students, but students from one group rarely communicate with students from another.

In this lab, your goal is to model the students of North Shore High School as a graph, and to explore different ways to model the school's social network based on queries to the graph.

3) Testing your lab

We've included a user interface to help visualize the problem. Run `server.py` and open your browser to localhost:8000 (http://localhost:8000/) to see the UI.

As in the previous labs, we also provide you with a `test.py` script to help you verify the correctness of your code.

4) Representation

You will need to devise an appropriate model of the high school's social network. To get started, we will give you a list of students, where each student is represented by a list `[name, interest1, interest2, interest3, ...]`.

4.1) Friendship

Two students are considered to be *friends* if they have at least one interest in common (however, students are not considered to be friends with themselves). The *weight* of a friendship is how many interests that friendship has in common.

For example, if Adam is interested in `['fishing', 'video games', 'programming', 'fast food', 'football', 'music']`, and Glen Coco is interested in `['programming', 'music', 'coffee', 'fishing', 'video games']`, then Adam and Glen Coco are friends, and the weight of their friendship is 4. That's 4 for you, Glen Coco. You go, Glen Coco.

In `tiny.json`, find and identify two people who are friends. Write a test case in `test.py` (in the `TestTiny` class) to make sure those students are considered friends if they are both added to the school.

In `tiny.json`, what is the weight of the friendship between Cady and Gretchen? (After answering, add a corresponding test case to `test.py`.)

We have provided two data sets for the school, found in the `resources` folder of the lab: `tiny.json` and `school.json`. We recommend using the `tiny.json` data set to test your understanding of the concepts presented here, and to write your own test cases for the lab.

5) Setting Up the School

We have provided a class called `School` in `lab.py`, to represent the school. Your job will be to complete the methods therein according to the specification below.

5.1) Insertion

When a student enters North Shore, they are immediately categorized based on their hobbies and interests. Based on these criteria, the student is then assigned to the appropriate cliques.

You may assume that student names are unique (i.e. no two students share a name). Your method of inserting students should ensure that you find and create the appropriate friendships between students.

5.2) Deletion

When a student leaves the school, it is as though they never existed: they are wiped from the school database and all cliques, and they leave no trace of any unique interests.

5.3) Updating

Students' interests may change over time. We would like a means of representing these changes in our school. When a student changes their interests, the weights of their friendships should change appropriately.

If Cady was updated so that her interests were `['sports', 'math']`, who would her friends be? Enter your answer as a Python list below (in any order). (Also, add a corresponding test case to `test.py` in `TestTiny`.)

6) Cliques

In high school, cliques are a hard problem to solve. In graph theory, they are, too (NP-hard (https://en.wikipedia.org/wiki/NP-hardness), this is). In essence, the upshot is that there is no known way to solve this problem efficiently.

A *clique* is defined as a subset of vertices in a graph such that every two vertices in the clique share an edge. A *maximal clique* is a clique that cannot be extended any further by adding another vertex (i.e., a clique that is not a subset of another clique).

It is your job to categorize students into cliques, where a school clique is a group of friends who are all friends with each other. Here, we will only look for maximal cliques, as defined above. We will implement this categorization through three methods:

- `get_cliques_for_student` should return a list of the maximal cliques to which a given student belongs
- `get_cliques` should find all the maximal cliques in the whole school
- `get_cliques_of_size_n` should find all the maximal cliques in the school that have a given size

Importantly, you should make sure that the values returned by these methods account for students being added and deleted from the school, or being updated.

Given the intractability of this problem, try to make your implementation of these methods as efficient as possible (in particular, try to avoid recomputing the result to the same problem more than once).

Add at least one nontrivial test case for each of the methods above to the `TestTiny` class in `test.py`. Your tests should test that the correct cliques are computed, even if students are added/removed/updated.

6.1) Independent Set

You were tasked with solving the clique problem faced by the North Shore High School. One idea you have is to promote friendships between students in different cliques by introducing them to one another.

We hope that by targeting specific students and mixing them into new groups, we may be able to alleviate our problem.

Luckily, graph theory will come to your aid once again! In graph theory, an *independent set* is the complement of a clique: that is, it is a set of vertices in a graph, none of which are adjacent. A *maximal independent set* is the complement of a maximal clique.

Implement `find_independent_set`, which should return a maximal independent set for a given student (i.e., a set of students, none of whom are friends) that contains the given student. In particular, it should return the ***largest possible*** **independent set that contains that student**.

As a note: once we had that independent set, we could, in principle, add a new shared interest between the given student and the students in the independent set, creating a new maximal clique without expanding the size of any existing clique.

Add at least one nontrivial test case for `find_independent_set` to the `TestTiny` class in `test.py`.

7) Code Submission

Select File `No file selected`

8) Checkoff

Once you are finished with the code, please come to a tutorial, lab session, or office hour and add yourself to the queue asking for a checkoff. **You must be ready to discuss your code and test cases in detail before asking for a checkoff.**

You should be prepared to demonstrate your code (which should be well-commented, should avoid repetition, and should make good use of helper functions). In particular, be prepared to discuss:

- the additional test cases you added to `TestTiny`
- what values you stored in attribute variables of the `School` class. How did you represent students? Friendships?
- how the values you stored are updated when a student is added, deleted, or updated
- Your implementation of `get_cliques`
- Your implementation of `get_cliques_for_student`
- Your implementation of `get_cliques_of_size_n`
- Your implementation of `find_independent_sets`

8.1) Grade

You have not yet received this checkoff. When you have completed this checkoff, you will see a grade here.

```
\
/   /\__/\
\__=( o_o )=
(________)
 |_ |_ |_ |_
```