ASSIGNMENT 3: GRAPH SUBSET MAPPING

Goal: The goal of this assignment is to take a complex new problem and formulate and solve it as a SAT problem. Formulation as SAT is a valuable skill in AI that will come in handy whenever you are faced with a new problem in NP class. SAT solvers over the years have become quite advanced and are often able to scale to decently sized real-world problems.

Scenario: You are an investigative agency working on uncovering a drug mafia. You have got telephone records of various telephone numbers which are believed to be associated with this mafia. You have also got a set of emails related to the mafia. However, you do not know which telephone number corresponds to which email address. The goal is to automatically figure out the mapping between emails and phones if it exists. To solve this problem (for our assignment), you make a few assumptions.

1. Some people are net savvy and use emails. All people know how to use phones. People who use emails regularly use both emails and phones to communicate with each other.
2. If a person X emailed a person Y, he also called Y on phone at some point. If X did not email a person Y (and both of them have email addresses), he did not call Y either.
3. Each person has exactly one email address and exactly one phone number.

You abstract out the problem by creating two graphs – Gphone and Gemail. There exists a directed edge between two nodes in Gphone (or in Gemail) if the first phone number (or email address) called (or emailed) the second. Your goal is find a mapping from emails to phone numbers. Gemail is the smaller graph because fewer people are net-savvy.

Problem Statement: There are two directed graphs G and G'. The graphs do not have self-edges. Find a one-one mapping M from nodes in G to nodes in G' such that there is an edge from v1 to v2 in G if and only if there is an edge from M(v1) to M(v2) in G'. Sample cases are shown here.

We will use miniSAT, a complete SAT solver for this problem. Your code will read two graphs in the given input format. You will then convert the mapping problem into a CNF SAT formula. Your SAT formula will be the input to miniSAT, which will return with a variable assignment that satisfies the formula (or an answer "no", signifying that the problem is unsatisfiable). You will then take the SAT assignment and convert it into a mapping from nodes of G to nodes of G'. You will output this mapping in the given output format.

You are being provided a problem generator that takes inputs of the sizes of G and G' and generates random problems with those parameters.

Input format:

Nodes are represented by positive integers starting from 1. Each line represents an edge from the first node to the second. Both graphs are presented in the single file, the larger first. The line with "0 0" is the boundary between the two. The input file that represents the last example in the slide is:

1 2

1 3

1 4

2 4

3 4

0 0

1 2

3 2

Output format:

The mapping will map each node of G into a node id for G'. The first numbers on each line represent a node as numbered in the smaller graph G, and the second number represents the node of the larger graph G' to which it is mapped. The output of the same problem is

1 2

2 4

3 3

If the problem is unsatisfiable output a 0.

Code

1. You may program the software in any of C++, Java or Python. The versions of the compilers that will be used to test your code are
   JAVA: java version "1.7.0_79" (OpenJDK)
   Python 3.6
   g++ 4.8.1

2. Executing the command "./run1.sh test" will take as input a file named test.graphs and produce a file test.satinput – the input file for minisat. You can assume that test.graphs exists in the present working directory. ('test' is a parameter and can be changed when running).
3. Executing the command "./run2.sh test" will use the generated test.satoutput, test.graphs (and any other temporary files produced by run1.sh) and produce a file test.mapping – the mapping in the output format described above. You can assume that test.graphs, test.satoutput (and other temp files) exist in the present working directory.
4. The TA will execute your scripts as follows:
   ./compile.sh
   ./run1.sh test
   ./minisat test.satinput test.satoutput
   ./run2.sh test

   When we call "./run1.sh test", you can assume that test.graphs exists in the present working directory. When we call "./run2.sh test", you can assume that test.graphs, test.satinput and test.satoutput exist in the present working directory, along with any other temporary files created by "./run1.sh test".

Please note that you are NOT allowed to call minisat within run1.sh or run2.sh. The TA will call minisat. Your code will be killed after the given problem cutoff time, so please write good code. However, cutoff time will be not be too short (like 1-2 minutes only) – it will be larger for large problems, so you need not overoptimize I/O code.

Useful resources

1. http://minisat.se/MiniSat.html: The MiniSat page
2. http://www.dwheeler.com/essays/minisat-user-guide.html: MiniSat user guide

What is being provided?

A problem generator for outputting G and G' where G is a subset of G' and therefore a mapping between the two exists. A check function to test whether your output is accurate or not, i.e., the mapping is indeed an accurate graph mapping.

What to submit?

1. Submit your code in a .zip file named in the format **<EntryNo>.zip.** If there are two members in your team it should be called <EntryNo1>_<EntryNo2>.zip. Make sure that when we run "unzip yourfile.zip" it contains a directory with the same name as the zip file and the following files are present in that directory:
   compile.sh

run1.sh
run2.sh
writeup.txt

You will be penalized for any submissions that do not conform to this requirement.

We will run your code on a few sample problems and verify the ability of your code to find solutions within a cutoff limit. The cutoff limits will be problem dependent and your translation does not need to depend on the cutoff limit, therefore it is not part of the input format. Of course, better translations will scale better, fetching more points.

2. The writeup.txt should have two lines as follows
   First line should be just a number between 1 and 3. Number 1 means C++. Number 2 means Java and Number 3 means Python.

   Second line should mention names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say None.

   After these first two lines you are welcome to write something about your code, though this is not necessary.

Code verification before submission: Your submission will be auto-graded. This means that it is absolutely essential to make sure that your code follows the input/output specifications of the assignment. Failure to follow any instruction will incur significant penalty. The details of code verification will be shared on Piazza (similar to A1).

Evaluation Criteria

1. Final competition on a set of similar problems. The points awarded will be your normalized performance relative to other groups in the class.
2. Extra credit may be awarded to standout performers.

What is allowed? What is not?

1. You may work in teams of two or by yourself. We do not expect a different quality of assignment for 2 people teams. At the same time, please spare us the details in case your team cannot function smoothly. Our recommendation: this assignment may be a little hard for students with limited prior exposure to logic. If you are such a student, work in teams if you can

find a workable partner. If you are good at logic, the assignment is quite easy and a partner will not be required.

2. You can use any language from C++, Java or Python for translation into and out of Minisat, as long as it works on our test machines. We will NOT be responsible for differences in versions leading to execution failures.
3. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.
4. Please do not search the Web for solutions to the problem.
5. Your code will be automatically evaluated against another set of benchmark problems. You get a zero if your output is not automatically parsable.
6. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.