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.
3. Each person has exactly one email address and exactly one phone.

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

Your code must compile and run on machine named 'todi' or any machine with similar configuration present in GCL. Please supply a compile.sh script. Also supply two shell scripts run1.sh, run2.sh:

1. 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.

2. 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.
3. The TA will execute your scripts as follows:
   ./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 and the minisat process will be killed after the given problem cutoff time, so that the cutoff time is only relevant for the time minisat takes to process your output, making the testing independent of time requirements for I/O.

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" the following files are produced in the present working directory:

   compile.sh
   run1.sh
   run2.sh
   Writeup.pdf

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 (as mentioned earlier, the cutoff time only measures the time required for minisat to run). 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.

2. Submit at-most 1 page writeup (10 pt font) describing your algorithm for translation into SAT and, if you performed, any insights or experiments reporting the scalability of your code. This is not graded but failure to submit a satisfactory writeup will incur negative penalty of 20% of total score. Your writeup will help us identify any common misconceptions and particularly good ideas for discussion in the class.

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 must use C++ for translation into and out of miniSAT.
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.