CSE 505, Fall 2022

HW2: Propositional Satisfiability

Due: **Thu., Sep 15**

Version 1.1.2
Sep 4, 2022

This is an assignment that uses Propositional SAT Solvers.

Background

For this assignment, you will use a widely available SAT solver picoSAT. Use of other SAT solvers may be possible, but discuss that with me first.

Simplified DIMACS form: Most SAT solvers use this format for formulae, specified as a text file. In this form, the propositions are numbered from 1..M. Positive literals are represented by that proposition's number, while negative literals are represented by the corresponding negative number. Formulas are written in CNF using a special notation, as explained below.
The first line of the file has:
p cnf *M N*
("p" and "cnf" have to appear as is.) Here, "*M*" is an integer representing the **number of distinct propositions** in the formula, and "*N*" as an integer representing the **number of clauses** in the formula. The remaining lines of the file represent clauses of the formula, each clause appearing in a single line. The literals in the clause are listed on that line (separated by a space). End of a clause is marked by "0" (note that "0" does not represent a proposition). The entire file is taken as a formula in conjunctive normal form.
For instance, the formula in readable form as (q => r), (r,s => q) may be represented as

```
p cnf 3 2
-1 2 0
-2 -3 1 0
```

(Here q,r,s are propositions 1,2,3 respectively).

When fed this input (as a file) to picosat, it will print out:

```
s SATISFIABLE
v -1 2 -3 0
```

Read the output as “the given formula is satisfiable with an assignment where the first and third propositions are false, and the second is true”.

What to Submit

Each of the problems ask you to generate an CNF formula that is amenable for satisfiability checking. For each, you should submit two things:

I. First, a clear and complete description of your solution (PDF)
II. Second, a program (in a language of your choice) that takes a file in the given input format, and generates another file, in simplified DIMACS form, that encodes the satisfiability instance. Your program should be well-documented enough that I can easily feed input files and inspect output files.

You may combine all the answers to (I) into a single PDF file, as long as individual answers are clearly marked. Answers to (II) are expected to be individual programs, but if convenient, you may combine all answers to (II) also into a single program, with a clearly-documented command-line option to specify which question it solves.

Warm-Up (Q1, Q2)

The following programs ask you to pose properties of a given CNF formula φ in terms of satisfiability of another formula φ'.

1. **At least one true:** Given a CNF formula φ over propositions P = {p_1, p_2, …, p_N}, generate another CNF formula φ' such that φ' is satisfiable if and only if φ has a satisfying assignment that makes **at least one** of p_1, p_2, …, p_N true. *Constraints:* (1) The size of φ' (sum of sizes of all clauses in φ', where the size of a clause is the number of literals in it) should be no more than O(N) larger than the size of φ. (2) The time it takes to generate φ' should be polynomial in N.
2. **At most one true:** Given a CNF formula φ over propositions P = {p_1, p_2, …, p_N}, generate another CNF formula φ' such that φ' is satisfiable if and only if φ has a satisfying assignment that makes **at most one** of p_1, p_2, …, p_N true. *Constraints:* (1) The size of φ' (sum of sizes of all clauses in φ', where the size of a clause is the number of literals in it) should be no more than ~~O(N)~~ O(poly(N)) larger than the size of φ where poly(N) is some polynomial in N. (2) The time it takes to generate φ' should be polynomial in N.
   For extra credit, you may consider a solution where the size of φ' is O(N) larger than the size of φ.

Q3. Count

Given a CNF formula φ over propositions P = {p_1, p_2, …, p_N}, and a number K, 1 <= K <= N, generate another CNF formula φ' such that φ' is satisfiable if and only if φ has a satisfying assignment that makes **at most K of** p_1, p_2, …, p_N true. *Constraints:* (1) The size of φ' (sum of sizes of all clauses in φ', where the size of a clause is the number of literals in it) should be no more than O(KN) larger than the size of φ. (2) the time it takes to generate φ' should be polynomial in KN.

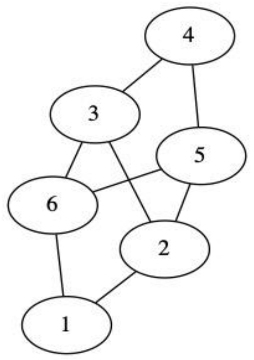
Q4. Subgraph Isomorphism

Given two graphs, G, and H, this problem is to determine if G has a subgraph that is isomorphic to H. See Wikipedia for details. This problem is NP-complete. Your task is to pose this problem as an instance of the satisfiability problem.

Input: Two files describing the graphs in the subgraph isomorphism problem instance. For each graph, V = {1, 2, …, N} and |E| = M. Then the input file starts with

```
N  M
```

followed by M lines, with each line containing a pair of integers representing an edge.



For instance, the graph on the left is encoded as:

```
6 8
1 2
2 3
3 4
4 5
5 6
6 1
2 5
3 6
```

Output: A CNF formula, encoded as a file in simplified DIMACS form, that is satisfiable if and only if the graph in the first input file (say, G) has a subgraph that is isomorphic to the graph in the second file (say, H). *Constraints:* (1) The size of your generated formula should be polynomial in the sizes of the input graphs. (2) The time it takes to generate the output should be polynomial in the sizes of the input graphs.

Q5. Graph Partitioning

Given a graph G=(V,E) with an even number of vertices, and a number K, we want determine if it is possible to partition the vertex set V into two disjoint sets (partitions) of equal size such that the number of edges that span the two partitions is less than or equal to a given number K. (We say an edge spans two partitions if its end points lie in different partitions). For example, for the graph pictured in Q4, it is possible to partition such that the number of edges spanning the partitions is bounded by 4. But there is no way to partition that graph such that the number of edges spannin the partitions is bounded by 3.

Inputs: (i) a file describing the graph to be partitioned; and (ii) a number K

Output: A CNF formula, encoded as a file in simplified DIMACS form, that is satisfiable if and only if the graph in the input file can be partitioned with K or fewer edges spanning the partition. *Constraints:* (1) The size of your generated formula should be polynomial in the size of the input graph and K. (2) The time it takes to generate the output should be polynomial in the size of the input graph and K.

Grading:

Problems 1 and 2 will be graded out of 5 points. Problems 3, 4, and 5 will be graded out of 10 points.

Submission:

Your submission will consist of a **zip or tar** file (name of the file does not matter for the submission) containing all the PDF(s) and programs. Please make sure that individual solutions are clearly marked so that there is no confusion as to which program/PDF file corresponds to which question. Submit the file as an attachment to the HW2 form on Blackboard.

Important: Work on the warm-up problems individually. You can work on Q. 3, 4, and 5 with **one** other team member. In that case, both have to submit complete solutions separately. Make sure that your submission has a clear marking of who your teammate is.

Extra Credit:

Q6. Consider the decision version of the MAX-SAT problem: given a CNF formula and a number *k*, determine if there is an assignment that satisfies *k* or more clauses of the formula. Pose this problem as an instance of SAT.

Errata/Changes:

1. **Sep 1:** The "constraint" part of each question only specified the complexity constraint on the size of the generated formula. There is also a constraint that the formula should be generated in time that is polynomial in the size of the input. I added this constraint.
2. **Sep 4:** The graph input format had a spurious "K" which is not necessary for the problems in this homework. I've now removed the "K".
    a. **Sep 9:** while I had removed "K" from the specification, I had not removed this spurious value from the example. This is fixed now.

3. **Sep 4:** Q2 had a constraint that the resulting formula should be only O(N) larger than the input formula. If you have difficulty finding such a solution, I’ll accept any solution that increases the size only by a polynomial factor. Solutions meeting the O(N) constraint will get extra credit.