2-Satisfiability
DUE: Friday, January 12, 2024

Write a program to solve the 2-Satisfiability Problem: Given a statement in Propositional Logic in Conjunctive Normal Form, where every clause has exactly two literals, your program must determine if the statement is satisfiable or not. If it is, it must output a model satisfying the statement, otherwise it must indicate that the statement is not satisfiable.

Recall the algorithm discussed in class: The statement is converted into a digraph G consisting of a vertex for each variable and its negation, and two arcs for each clause to represent the equivalent implication and its contrapositive. Kosaraju’s algorithm is used to determine the strong components of G, say G_1, …, G_k. Construct the condensation of G, the DAG whose k vertices are the strong components of G, in which there is an arc between vertices iff there is an arc between any two vertices in the corresponding (distinct) strong components. (If parallel arcs are suppressed, there will never be arcs in two different directions between the same two strong components.) If two contradictory literals occur in the same strong component, the original input statement is unsatisfiable. If not, the statement is satisifiable, and a satisfying model is obtained by considering the strong components in reverse topological order (with respect to the condensation DAG) and labeling every unlabled variable so that the corresponding literal is true.

There will be at most 5000 variables and at most 100,000 clauses in any one input statement. Your program should run in at most 5 seconds. The standard input and output will be used.

INPUT:

nv nc
u_1 v_1
. . .
u_nc v_nc

where

nv = # of variables
nc = # of clauses
|u_i| and |v_i| are the variables in the ith clause
u_i < 0 and/or v_i < 0 if the variable is negated in the ith clause

OUTPUT:

Not satisfiable if the input is not satisfiable

x_1 if the input is satisfiable, in which case x_i is either ’T’ or ‘F’ depending on whether the
x_2 corresponding varaible is true or false.
. . .
x_nv