ST. CLOUD STATE UNIVERSITY

CSCI 404/504: Design and Analysis of Algorithms (Fall 2002)
Instructor: Pranava K. Jha

2SAT is in P

2SAT is a special case of the classical SATISFIABILITY problem. It admits an efficient solution.

Instance: Collection $C = \{c_1, \ldots, c_m\}$ of clauses on a set U of n Boolean variables such that $|c_i| = 2$ for $1 \leq i \leq m$.

Question: Is there a truth assignment for the variables in U that satisfies all clauses in C?

2*SAT* can be solved by setting the value of a literal and by following the resulting constraints on other variables in order to satisfy each clause. Consider, for instance, the formula:

$$(x_1 + x_2)\ (x_2' + x_3)\ (x_1' + x_2')\ (x_3 + x_4)\ (x_3' + x_5)\ (x_4' + x_5')\ (x_3' + x_4).$$

Number of variables $n = 5$ and number of clauses $m = 7$. (In general, $m \geq n$.)

Start by making x_1 TRUE. This makes the clause $(x_1 + x_2)$ TRUE; accordingly, this clause may be removed. Under this assignment, x_2' must be TRUE (i.e., x_2 must be FALSE) in order to satisfy the clause $(x_1' + x_2')$. Making x_2 FALSE satisfies the clause $(x_2' + x_3)$ as well, leaving only the shorter formula

$$(x_3 + x_4)\ (x_3' + x_5)\ (x_4' + x_5')\ (x_3' + x_4)$$

to be satisfied. Since each clause consists of only two literals, truth assignments are propagated simply and deterministically among clauses. When such a propagation stops, the remaining clauses contain variables whose values have not yet been set and which are independent of the previous variables. Accordingly, satisfiability of the remaining clauses is independent of any previous computation.

Choose an arbitrary literal from an arbitrary clause, and begin a new truth-assignment propagation. Continuing the foregoing example, if we make x_3 TRUE, we propagate to find x_4 TRUE and x_5 FALSE. In the process, we end up with $(x_3' + x_5)$ that is FALSE, and hence the formula is not satisfied. On the other hand, if we make x_3 FALSE, then we propagate to find x_4 TRUE and x_5 FALSE, and the formula is indeed satisfied. This step of truth assignment and propagation may be repeated before all clauses have been satisfied.

The algorithm can be formally described as repeatedly decomposing a formula Φ until no unsatisfied clauses remain.

Algorithm 2*SAT* (Φ):
// Φ is an instance of 2*SAT* in conjunctive normal form, i.e.,
// a set of clauses, each of which consists of two literals;
1. choose a variable v in Φ;
2. set v to TRUE;
3. remove from Φ all clauses of the form $(v + w)$;
4. for each clause of the form $(v' + w)$ in Φ, set the literal w to TRUE and remove the clause $(v' + w)$ from Φ;
   // w is forced to be TRUE, since v' in the clause $(v' + w)$ is FALSE;
   // note that w itself is a literal, not necessarily a (positive) variable
   // in particular, if $w = x'$ where x is a variable,
   // then setting w to TRUE is equivalent to setting x to FALSE;
5. continue to set the values of the "forced" variables, and remove clauses that are satisfied by the corresponding truth assignment until
   either (i) there are no more "forced" variables,
   or (ii) a problem arises where a variable is forced to be both TRUE and FALSE;
   in case (i), Φ has reduced to a smaller 2*SAT* formula;
   if it is empty, then conclude that Φ is satisfiable and STOP,
   otherwise perform the algorithm on the smaller formula;
   // in case (ii), proceed as follows:
6. set v to FALSE;
7. remove from Φ all clauses of the form $(v' + w)$;
8. for each clause of the form $(v + w)$ in Φ, set the literal w to TRUE and remove the clause $(v + w)$ from Φ;
9. continue to set the values of the "forced" variables, and remove clauses that are satisfied by the corresponding truth assignment until
   either (i) there are no more "forced" variables,
   or (ii) a problem arises where a variable is forced to be both TRUE and FALSE;
   in case (i), Φ has reduced to a smaller 2*SAT* formula;
   if it is empty, then conclude that Φ is satisfiable and STOP,
   otherwise perform the algorithm on the smaller formula;
   in case (ii), conclude that Φ is not satisfiable, and STOP

With appropriate data structures, the algorithm can be implemented in time that is $O(m^2)$.

The computation of 2*SAT* can alternatively (and more efficiently) be formulated as a graph algorithm. To that end, let Φ be an instance of 2*SAT*. Construct a directed graph $G(\Phi)$ as follows:

- Vertices of $G(\Phi)$ are the variables of Φ and their negations.
- There is an arc (x, y) in $G(\Phi)$ if and only if there is a clause $(x' + y)$ or $(y + x')$ in Φ.

Note that $a + b$ is equivalent to each of $a' \Rightarrow b$ and $b' \Rightarrow a$. Thus a 2*SAT* formula may be viewed as a set of implications. Accordingly, if we have a formula $(a' + b)\ (b' + c)\ (c' + d)$, then we have a string of implications $a \Rightarrow b \Rightarrow c \Rightarrow d$, which leads to $a \Rightarrow d$, since implication is transitive.

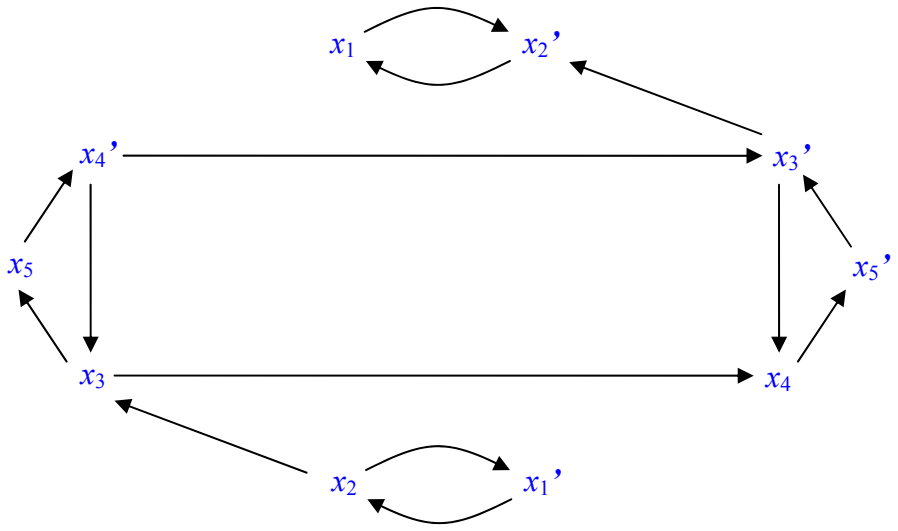
- If for some variable a, there is a string of implications $a \Rightarrow \ldots \Rightarrow a'$, and another string of implications $a' \Rightarrow \ldots \Rightarrow a$, then the formula is *not* satisfiable, otherwise the formula is satisfiable.

Theorem: *A Boolean formula Φ in 2SAT is unsatsfiable if and only if there is a variable x such that there are directed paths from x to x' and from x' to x in G*(Φ).

For the formula

$$(x_1 + x_2)\ (x_2' + x_3)\ (x_1' + x_2')\ (x_3 + x_4)\ (x_3' + x_5)\ (x_4' + x_5')\ (x_3' + x_4).$$

the implication graph is as follows:



The present discussion shows that 2*SAT* reduces to the graph problem of finding strongly connected components (*SCC*) in the implication graph, since a 2*SAT* formula is unsatisfiable if and only if some variable and its complement reside in the same *SCC*. As *SCC* is known to have a linear-time solution and the implication graph is constructible in linear time, it is clear that 2*SAT* may be decided under the same time bound.

While 2SAT is in P, 3SAT is NP-complete. This shows that innocent changes in a problem may lead to drastic changes in the complexity.

———o———