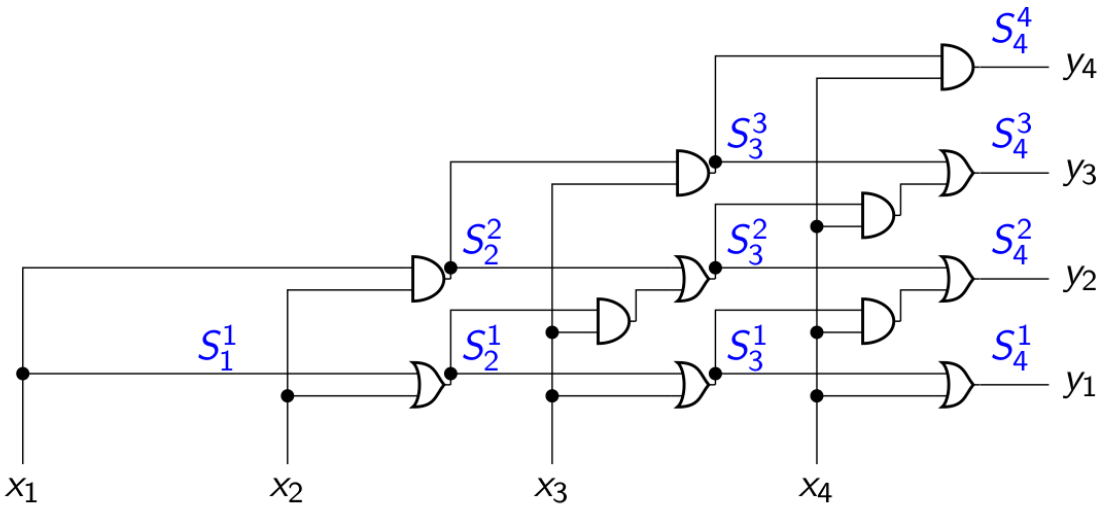
Cardinality Constraints — X37043_en

Write a program in Python that, using the **optilog** library, finds all solutions of a set of cardinality constraints.



A set of cardinality constraints is a set of inequalities like:

$$\begin{aligned} &x_1 + x_2 \geq 1 \\ &x_2 + x_3 + x_4 \geq 2 \\ &x_1 + x_2 + x_3 + x_4 \leq 2 \end{aligned}$$

They can be translated into SAT using sequential counters:

$$\begin{array}{ll} S_1^1 \leftrightarrow x_1 & \\ S_i^i \leftrightarrow S_{i-1}^{i-1} \wedge x_i & i = 2, \ldots, k \\ S_i^1 \leftrightarrow S_{i-1}^1 \vee x_i & i = 2, \ldots, n \\ S_i^j \leftrightarrow S_{i-1}^j \vee (x_i \wedge S_{i-1}^{j-1}) & i = 3, \ldots, n, j = 2, \ldots, \min\{i-1, k\} \end{array}$$

The clause S_n^k plus the clauses defining the **right** implications of the previous equivalences enforce $x_1 + \cdots + x_n \geq k$. In the last equivalence, for instance, the clauses are:

$$S_i^j \rightarrow S_{i-1}^j \vee (x_i \wedge S_{i-1}^{j-1}) \equiv \left\{ \begin{array}{l} \overline{S_i^j} \vee S_{i-1}^j \vee S_{i-1}^{j-1} \equiv \overline{S_i^j} \vee S_{i-1}^{j-1} \\ \overline{S_i^j} \vee S_{i-1}^j \vee x_i \end{array} \right.$$

The constraint $x_1 + \cdots + x_n \geq k$ is interpreted as: the number of variables assigned to true is at least k. The minus sign is interpreted as negation. Therefore, $x_1 - x_2 \geq 2$ is interpreted as: both x_1 and $\overline{x_2}$ are both true. Therefore, as inequality, it is in fact $x_1 + (1 - x_2) \geq 2$ that has a unique solution $x_1 = 1, x_2 = 0$.

Input

The input is a text (in the stdin) with pairs of connected nodes. For instance, the following text in the case of our example:

```
x1 + x2 > 1
x2 + x3 + x4 > 2
x1 + x2 + x3 + x4 < 2
```

Output

The output is also a text (in the stdout) where in every line there is a variable assignment (variables assigned to 1 (true) as possitive, and those assigned to 0 (false) as negative. In this example:

```
{ -x1 x2 x3 -x4 }
{ -x1 x2 -x3 x4 }
```

Sample input 1

```
x1 + x2 + x3 > 2
x1 + x2 + x3 < 2
```

Sample output 1

```
{ -x1 x2 x3 }
{ x1 -x2 x3 }
{ x1 x2 -x3 }
```

Sample input 2

```
x1 + x2 > 1
x2 + x3 + x4 > 2
x1 + x2 + x3 + x4 < 2
```

Sample output 2

```
{ -x1 x2 x3 -x4 }
{ -x1 x2 -x3 x4 }
```

Sample input 3

```
x1 - x2 > 2
```

Sample output 3

```
{ x1 -x2 }
```

Scoring

Problem information

Author: Jordi Levy

Generation: 2026-01-25T15:28:24.413Z