**IWLS 2017 programming contest**
Mathias Soeken, *EPFL*

**The $Y$ function.** Let $G_n = (V_n, E_n)$ be a triangular grid of order $n$. Triangular grids up to order 3 are the following.

$$G_0 = \bullet \qquad G_1 = \triangle \qquad G_2 = \triangle\!\triangle \qquad G_3 = \triangle\!\triangle\!\triangle \tag{1}$$

The grid of $G_n$ has $(n+2)(n+1)/2$ nodes and three sides, each of which is having $n+1$ nodes. We call these sides left, right, and bottom. From a triangular grid $G_n$ we can obtain triangular grids of height $n-1$ by removing all vertices of the left, right, or bottom side. We call these grid $G_n^l$, $G_n^r$, and $G_n^b$, respectively.

Let $Y$ be a function that maps a triangular grid $G_n = (V_n, E_n)$ into a Boolean function using the following recursive procedure:

$$Y(G_n) = \begin{cases} x_v & \text{if } n = 0 \text{ and } V_0 = \{v\}, \\ \langle Y(G_n^b) Y(G_n^r) Y(G_n^l) \rangle & \text{otherwise.} \end{cases} \tag{2}$$

Here, $\langle xyz \rangle = xy \vee xz \vee yz$ is the majority-of-three function. Then $Y_n = Y(G_n)$ is the $Y$ function of size $n$.

For example, $Y_0$ is the identity function, $Y_1$ is the majority function, and if we label the nodes in $G_2$ with $x_1$, $x_2$, $x_3$, $x_4$, $x_5$, $x_6$ from top to bottom and from left to right, we obtain $Y_2 = \langle \langle x_1 x_2 x_3 \rangle \langle x_2 x_4 x_5 \rangle \langle x_3 x_5 x_6 \rangle \rangle$.

We highly recommend to read Exercise 67 in Section 7.1.1. of Donald E. Knuth's *The Art of Computer Programming*. A link to an online version of the exercise is provided on the contest homepage.

**Task.** Implement a logic synthesis algorithm that takes as input a combinational benchmark (provided in Verilog, Aiger, or Blif format) and outputs a YIG ($Y$-inverter graph) that is composed of $Y$-gates, which implement the $Y$ function, and inverters. The output format should be clear from the following example.

```
.i 8
.o 2
.w 2
w1 = Y1(i1, ~i2, 0)
w2 = Y2(i1, ~i1, 1, i3, i4, w1)
o1 = Y2(w1, w2, 0, 1, ~i5, i6)
o2 = Y1(i7, i8, w2)
.e
```

The first three lines give the number of primary inputs, primary outputs, and internal wires, which are implicitly named `i1`, `i2`, ..., `o1`, `o2`, ... and `w1`, `w2`, ... Then gates are defined, which store their result either in a wire or in an primary output. Gates take as input previously defined wires, primary inputs, or constants. All inputs can be inverted using ˜. We provide converters for YIG files into Verilog on the contest homepage. The goal is to find a YIG with a small number of gates. Each gate has unit cost. For example, the given example above has cost 4.

**Hints.** Since $Y_1$ is the majority-of-three function, YIGs naturally contain majority-inverter graphs (MIGs). However, one can do better when using larger $Y$-gates. Note that $Y_1(x_1, x_2, 0) = x_1 \wedge x_2$ and $Y_1(x_1, x_2, 1) = x_1 \vee x_2$. There is a popular 3-input function contained in $Y_2$ in a similar way.