Exercise 1: Consider the boolean expression

\[ E = (x_3 \land \neg((x_1 \lor x_2) \land (\neg x_1 \lor \neg x_2))) \lor \\
(\neg x_2 \land (x_1 \lor x_3) \land (\neg x_1 \lor \neg x_2)) \]

Construct a boolean circuit that computes the value of \( E \), given inputs for \( x_1, x_2, x_3 \).

What is the size of this circuit? 6

What is the depth? 4

How does the size compare to the length of \( E \)?
Exercise: Reduction from Reachability to Circuit Value

Reachability: given a directed graph $G = (V, E)$
with $V = \{1, \ldots, n\}$ and $E \subseteq V \times V$,
is there a path from node $1$ to node $n$ in $G$.

Circuit Value: given a boolean circuit $C$ without input
variables, is the output of $C$ equal to $T$?

We show how to reduce Reachability to Circuit Value, i.e.
how to construct from a graph $G$ a circuit $R(G)$ such
that:

1 is reachable from $n$ in $G$ iff
the value of $R(G)$ is $T$.

In $R(G)$ we use gates of two forms:

1) $f_{i,j,k}$, with $1 \leq i, j \leq n$ and $0 \leq k \leq n$

Intuitively, $f_{i,j,k}$ is true iff \[ i \rightarrow j \rightarrow \cdots \rightarrow k \rightarrow \cdots \rightarrow j \]
all $\leq k$

i.e. there is a path from $i$ to $j$ not using any
intermediate node bigger than $k$.

2) $h_{i,j,k}$, with $1 \leq i, j, k \leq n$ iff

\[ i \rightarrow j \rightarrow k \rightarrow \cdots \rightarrow j \]
all $\leq k$\]
i.e. there is a path from $i$ to $j$ not using any
intermediate node bigger than $k$, but using $k$
as intermediate node.
We describe now the gates and how they are connected.

- For $h = 0$, all $g_{ij}$ gates are constant gates

\[ g_{ij} = T \quad \text{if} \quad i = j \quad \text{or} \quad i \rightarrow j \quad \text{in} \quad R(G) \]

(note that there are no $k_{ij}$ gates)

- For $h = 1, 2, \ldots, n$
  - $h_{ijk}$ is an AND gate with predecessors $g_{ij}, k, h-1$ and $g_{k,j,h-1}$

  - $g_{ijk}$ is an OR gate with predecessors $g_{ij}, k, h$ and $h_{ijk}$

The output gate is $g_{1n,n}$

Note that $R(G)$ can be computed from $G$ in logarithmic space.

Note that the circuit $R(G)$ is legitimate, since it contains no cycles; we can reverse the gates $1, 2, \ldots, 2n^3 + n^2$, in non-decreasing order of the third index, and with $h_{ijk}$ preceding $g_{ijk}$

We have to show that the value of the output gate of $R(G)$ is $T$ if and there is a path from $1$ to $n$ in $G$.

We prove by induction on $h$ that the values of the gates correspond to the informal meaning we gave them:

- For $h = 0$: this holds

- If it is true up to $h-1$, the definitions of $g_{ij}$ and $h_{ijk}$ guarantee that it is true also for $h$. 
Exercise: A boolean function if is said to be monotone if it has the following property: if one of the values changes from 0 to 1, then the value of if does not change from 1 to 0.

We show that if is monotone iff it can be expressed by a circuit with only AND and OR gates.

\[ \Leftrightarrow \] Consider a circuit C with only AND and OR gates expressing if.

We show by induction on the depth of a node M:
- if the value of an input \( x \) changes from 0 to 1
  then the value of M does not change from 1 to 0.

Base: depth \((M) = 0\), then M is either an input or a constant node.
- if it is a constant, its value does not change
- if it is an input different from \( x \)
- if it is input \( x \), its value changes from 0 to 1 (and not from 1 to 0)

Induction: suppose that for all nodes at least \( k \) deep, the value does not change from 1 to 0.

Consider a node \( M \) at least \( k+1 \) deep. We show that the value of \( M \) does not change from 1 to 0.

Case 1: \( M \) is an AND node

\[
\begin{array}{c}
\overline{Y} \\
\downarrow \text{AND}
\end{array}
\]

Case 2: \( M \) is an OR node

\[
\begin{array}{c}
\overline{Y} \\
\downarrow \text{OR}
\end{array}
\]
We need to consider various situations corresponding to the changes of \( y_1, y_2 \) from 0 to 1.

<table>
<thead>
<tr>
<th>( y_1 )</th>
<th>( y_2 )</th>
<th>( y_3 )</th>
<th>( y_4 )</th>
<th>( y_5 )</th>
<th>( y_6 )</th>
<th>( y_7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

We see that both for an AND node and for an OR node, the output value never changes from 1 to 0.

\[ \Rightarrow \]

We show by induction on \( n \) that every monotone boolean function \( f(x_1, \ldots, x_n) \) of \( n \) variables can be represented by a circuit with AND and OR gates only.

Base case: 0 arguments: \( f \) is constant, and hence monotone.

Inductive case: assume the claim holds for \( n \).

We show it holds for a function \( f(x_1, \ldots, x_{n+1}) \).

We exploit the fact that

\[
f(x_1, \ldots, x_n, x_{n+1}) = (f(x_1, \ldots, x_n) \land (x_{n+1} \lor f(x_1, \ldots, x_n, 0))) \lor (f(x_1, \ldots, x_n, \neg 1) \land \neg (x_{n+1} \lor f(x_1, \ldots, x_n, 1)))
\]
Hence, we can construct a circuit \( C_f \) computing \( f(x_1, \ldots, x_n, x_{n+1}) \) as follows:

![Diagram](image)

Observe that, since \( f(x_1, \ldots, x_{n+1}) \) is monotone, we have that also \( f(x_1, \ldots, x_n, 0) \) and \( f(x_1, \ldots, x_n, 1) \) are monotone.

Hence, since \( f(x_1, \ldots, x_n, 0) \) and \( f(x_1, \ldots, x_n, 1) \) are \( n \)-variable monotone functions, by inductive hypothesis they can be represented by circuits with AND and OR gates only.

Hence, it suffices to show that we can get rid of the only remaining NOT gate.

Consider the following circuit \( C'_f \), in which we have eliminated the NOT gate.
Let us consider the possible values of $f$ in $C_f$ and $C'_f$.
It depends on the values of $r_0$, $r_0$, $r_1$.

<table>
<thead>
<tr>
<th>$r_m$</th>
<th>$v_0$</th>
<th>$v_1$</th>
<th>$C_f$</th>
<th>$C'_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Note that $C_f = C'_f$, except for case (7).

However, since $f(r_1, \ldots, r_{m+1})$ is monotone, cases (3) and (7) cannot occur, since they would mean that $f(r_1, \ldots, r_m, r_{m+1})$ changes from $v_0 = 1$ for $r_{m+1} = 0$ to $v_0 = 0$ for $r_{m+1} = 1$.

Hence, $C'_f$ is the correct circuit consisting of AND and OR gates only, and computing $f(r_1, \ldots, r_{m+1})$. 
Exercise 11.11 b)

Consider the problem \textsc{FALSE-SAT}.

Given a boolean expression \( E \) that is false when all its variables are made false, is there some other truth assignment that makes \( E \) false, besides all-false?

 Decide whether the problem is in \( \text{NP} \) or \( \text{coNP} \).

 Describe its complement.

 If the problem or its complement is \( \text{NP-complete} \), prove it.

\textbf{Proof:}

The problem is \( \text{NP-complete} \).

\textbf{In \( \text{NP} \)}: given a boolean expression \( E \), we need to check:

1) that \( E \) is false when all variables are assigned false
2) that there is some other truth assignment making \( E \) false

(1) can be done in poly-time by a DTM
(2) can be done in poly-time by a NTM

Guess a truth assignment \( T \) different from all false, and answer yes if and only if \( T \), \( E \) evaluates to false

\textbf{NP-hard:} by a reduction from \( \text{SAT} \)

Let \( E \) be a boolean expression with variables \( x_1, \ldots, x_n \),

we construct an expression \( E' \) s.t. \( E \in \text{SAT} \) iff \( E' \in \text{FALSE-SAT} \)

1) test if \( E \) is true when all variables are false (polynomial)

If so, \( E \in \text{SAT} \), and we convert it to a fixed expression

that is in \( \text{FALSE-SAT} \), \( e.g. \) \( \neg \forall x \).
2) Otherwise, let $E' = \neg E \land (x_1 \lor x_2 \lor \cdots \lor x_n)$. Clearly, the reduction is poly-time.

We know that $E'$ is false when all of $x_1, \ldots, x_n$ are false. Notice that in case (2) $E'$ is false when all variables are false. Hence, if $E \in \text{SAT}$, then it is satisfied by a truth assignment $T$ different from all-false. Thus, $\neg E$ is made false by $T$, and $E' \in \text{FALSE-SAT}$.

Conversely, if $E' \in \text{FALSE-SAT}$, then since $x_1, \ldots, x_n$ is false only for the all-false truth assignment, there must be some other truth assignment $T$ that makes $\neg E$ false. Then $T$ makes $E$ true, and $E \in \text{SAT}$.