1. Suppose you are given a magic black box that somehow answers the following decision problem in polynomial time:

- **Input:** A boolean circuit $K$ with $n$ inputs and one output.
- **Output:** True if there are input values $x_1, x_2, \ldots, x_n \in \{\text{True}, \text{False}\}$ that make $K$ output True, and False otherwise.

Using this black box as a subroutine, describe an algorithm that solves the following related search problem in polynomial time:

- **Input:** A boolean circuit $K$ with $n$ inputs and one output.
- **Output:** Input values $x_1, x_2, \ldots, x_n \in \{\text{True}, \text{False}\}$ that make $K$ output True, or None if there are no such inputs.

[Hint: You can use the magic box more than once.]

**Solution:** For any boolean circuit $K$ with inputs $x_1, \ldots, x_n$, let $K \land x_i$ be the boolean circuit obtained from $K$ by adding a new AND gate, with one input connected to the output of $K$ and the other to the input $x_i$. Similarly, let $K \land \overline{x}_i$ be the boolean circuit obtained from $K$ by adding a NOT gate, with input connected to $x_i$, and an AND gate, with one input connected to the output of $K$ and the other to the NOT gate. For both of these circuits, the output of the new AND gate is the output of the circuit.

Suppose $\text{CircuitSat}(K)$ returns True if $K$ is satisfiable and False otherwise. Then the following algorithm constructs a satisfying input assignment for $K$ or correctly reports that no such assignment exists.

```python
def SatAssignment(K):
    if CircuitSat(K) = False
        return None
    for i ← 1 to n
        if CircuitSat(K ∧ x_i)
            K ← K ∧ x_i
            A[i] ← True
        else
            K ← K ∧ \overline{x}_i
            A[i] ← False
    return A[1..n]
```

The correctness of this algorithm follows by induction from the following observation:

**Claim 1.** The circuit $K \land x_i$ is satisfiable if and only if $K$ has a satisfying assignment where $x_i = \text{True}$.

**Proof:** First, if $K \land x_i$ has a satisfying assignment, then that input assignment must satisfy $K$ and must have $x_i = \text{True}$, because otherwise the AND gate would output False.

On the other hand, if $K$ has a satisfying assignment where $x_i = \text{True}$, then that input assignment also satisfies $K \land x_i$, because that’s how AND gates do. □
The algorithm runs in polynomial time. Specifically, suppose \textsc{CircuitSat}(K) runs in \( O(N^c) \) time, where \( N \) is the total number of vertices and edges in dag representing \( K \). (The vertices consist of the inputs, the internal gates, and the output; the edges are the wires between those points.) Then \textsc{SatAssignment}(K) runs in time

\[
O(N^c) + \sum_{i=1}^{n} O((N + 5i)^c) \leq (N + 1) \cdot O((6N)^c) = O(N^{c+1}),
\]

which is polynomial in \( N \).
2. A Hamiltonian cycle in a graph $G$ is a cycle that goes through every vertex of $G$ exactly once. Deciding whether an arbitrary graph contains a Hamiltonian cycle is NP-hard.

A tonian cycle in a graph $G$ is a cycle that goes through at least half of the vertices of $G$. Prove that deciding whether a graph contains a tonian cycle is NP-hard.

**Solution (duplicate the graph):** I’ll describe a polynomial-time reduction from HAMILTONIANCYCLE. Let $G$ be an arbitrary graph. Let $H$ be a graph consisting of two disjoint copies of $G$, with no edges between them; call these copies $G_1$ and $G_2$. I claim that $G$ has a Hamiltonian cycle if and only if $H$ has a tonian cycle.

$\implies$ Suppose $G$ has a Hamiltonian cycle $C$. Let $C_1$ be the corresponding cycle in $G_1$. $C_1$ contains exactly half of the vertices of $H$, and thus is a tonian cycle in $H$.

$\impliedby$ On the other hand, suppose $H$ has a tonian cycle $C$. Because there are no edges between the subgraphs $G_1$ and $G_2$, this cycle must lie entirely within one of these two subgraphs. $G_1$ and $G_2$ each contain exactly half the vertices of $H$, so $C$ must also contain exactly half the vertices of $H$, and thus is a Hamiltonian cycle in either $G_1$ or $G_2$. But $G_1$ and $G_2$ are just copies of $G$. We conclude that $G$ has a Hamiltonian cycle.

Given $G$, we can construct $H$ in polynomial time by brute force.

**Solution (add $n$ new vertices):** I’ll describe a polynomial-time reduction from HAMILTONIANCYCLE. Let $G$ be an arbitrary graph, and suppose $G$ has $n$ vertices. Let $H$ be a graph obtained by adding $n$ new vertices to $G$, but no additional edges. I claim that $G$ has a Hamiltonian cycle if and only if $H$ has a tonian cycle.

$\implies$ Suppose $G$ has a Hamiltonian cycle $C$. Then $C$ visits exactly half the vertices of $H$, and thus is a tonian cycle in $H$.

$\impliedby$ On the other hand, suppose $H$ has a tonian cycle $C$. This cycle cannot visit any of the new vertices, so it must lie entirely within the subgraph $G$. Since $G$ contains exactly half the vertices of $H$, the cycle $C$ must visit every vertex of $G$, and thus is a Hamiltonian cycle in $G$.

Given $G$, we can construct $H$ in polynomial time by brute force.
To think about later:

3. Let $G$ be an undirected graph with weighted edges. A Hamiltonian cycle in $G$ is heavy if the total weight of edges in the cycle is at least half of the total weight of all edges in $G$. Prove that deciding whether a graph contains a heavy Hamiltonian cycle is NP-hard.

**Solution (two new vertices):** I'll describe a polynomial-time a reduction from the Hamiltonian path problem. Let $G$ be an arbitrary undirected graph (without edge weights). Let $H$ be the edge-weighted graph obtained from $G$ as follows:

- Add two new vertices $s$ and $t$.
- Add edges from $s$ and $t$ every other vertex (including each other).
- Assign weight 1 to the edge $st$ and weight 0 to every other edge.

The total weight of all edges in $H$ is 1. Thus, a Hamiltonian cycle in $H$ is heavy if and only if it contains the edge $st$. I claim that $H$ contains a heavy Hamiltonian cycle if and only if $G$ contains a Hamiltonian path.

$\implies$ First, suppose $G$ has a Hamiltonian path from vertex $u$ to vertex $v$. By adding the edges $vs$, $st$, and $tu$ to this path, we obtain a Hamiltonian cycle in $H$. Moreover, this Hamiltonian cycle is heavy, because it contains the edge $st$.

$\impliedby$ On the other hand, suppose $H$ has a heavy Hamiltonian cycle. This cycle must contain the edge $st$, and therefore must visit all the other vertices in $H$ contiguously. Thus, deleting vertices $s$ and $t$ and their incident edges from the cycle leaves a Hamiltonian path in $G$.

Given $G$, we can easily construct $H$ in polynomial time by brute force.

**Solution (smartass):** I'll describe a polynomial-time a reduction from the standard Hamiltonian cycle problem. Let $G$ be an arbitrary graph (without edge weights). Let $H$ be the edge-weighted graph obtained from $G$ by assigning each edge weight 0. I claim that $H$ contains a heavy Hamiltonian cycle if and only if $G$ contains a Hamiltonian cycle.

$\implies$ Suppose $G$ has a Hamiltonian cycle $C$. The total weight of $C$ is at least half the total weight of all edges in $H$, because $0 \geq 0/2$. So $C$ is a heavy Hamiltonian cycle in $H$.

$\impliedby$ Suppose $H$ has a heavy Hamiltonian cycle $C$. By definition, $C$ is also a Hamiltonian cycle in $G$.

Given $G$, we can easily construct $H$ in polynomial time by brute force.