

## Question

Given a Boolean circuit instance whose output evaluates to TRUE, explain how the correctness of the result can be verified in polynomial time using Depth First Search (DFS).

## Answer

### Background: Boolean Circuits and Verification

A Boolean circuit is a directed acyclic graph (DAG) where:

- internal nodes are logic gates (AND, OR, NOT),
- leaves are input variables or constants,
- there is a single designated output gate.

A Boolean circuit instance is considered a YES-instance if, under a given input assignment, the output gate evaluates to TRUE.

The task here is verification, not computation. Given that the output is claimed to be TRUE, we must verify this claim efficiently.

### Key Idea of Polynomial-Time Verification

The correctness of a Boolean circuit's output can be verified by:

- traversing the circuit from the output gate to the input gates,
- recursively checking that each gate produces the correct output value,
- ensuring consistency with the logical semantics of each gate.

This process can be naturally implemented using Depth First Search (DFS).

### Circuit as a Graph

The Boolean circuit is modeled as a directed graph:

- vertices represent gates or input literals,
- directed edges represent signal flow from inputs to outputs.

Since Boolean circuits are acyclic, the resulting graph is a DAG, which is ideal for DFS traversal.

## Verification Using DFS

### DFS Strategy

A DFS is started from the output gate. The algorithm recursively verifies the correctness of the output by checking its children.

At each visited node:

- all input gates (children) are visited,
- their values are verified,
- the gate's logical operation is checked against the claimed output.

### Gate-by-Gate Verification Rules

Suppose a gate claims to evaluate to TRUE.

- **AND gate:** All input gates must evaluate to TRUE. DFS verifies each input recursively.
- **OR gate:** At least one input gate must evaluate to TRUE. DFS verifies that at least one child evaluates to TRUE.
- **NOT gate:** Its single input must evaluate to FALSE. DFS verifies this condition.
- **Input literal:** The value is directly checked against the given input assignment.

If all checks succeed, the correctness of the output gate is verified.

### DFS-Based Verification: Pseudocode

Below is a high-level DFS procedure to verify that the output of a Boolean circuit evaluates to TRUE under a given input assignment.

```
Verify(gate):
    if gate is an input literal:
        return assigned truth value of gate

    if gate is AND:
        for each input child c:
```

```

        if Verify(c) == false:
            return false
        return true

    if gate is OR:
        for each input child c:
            if Verify(c) == true:
                return true
        return false

    if gate is NOT:
        let c be the single input child
        return NOT Verify(c)

```

The verification starts by calling `Verify(output_gate)`. If this call returns TRUE, the claimed output of the circuit is correct.

## Why DFS Is Sufficient and Efficient

### Correctness

DFS guarantees that:

- every gate contributing to the output is visited,
- the logical correctness of each gate is verified bottom-up,
- no gate is processed before its dependencies.

Because the circuit is acyclic, DFS does not revisit nodes unnecessarily.

### Time Complexity

Let:

- $n$  be the number of gates,
- $m$  be the number of wires (edges).

DFS visits each gate and each wire exactly once. Therefore, the verification runs in:

$$O(n + m),$$

which is polynomial in the size of the circuit.

## Relation to NP Verification

The Boolean Circuit Value Problem (BCVP) is a canonical **NP**-complete problem.

This verification procedure demonstrates that:

- given a certificate (the input assignment),
- the correctness of the output being TRUE

can be verified in polynomial time.

This aligns precisely with the definition of **NP**: YES-instances have efficiently verifiable certificates. DFS provides a structured way to perform this verification.

## Relation to Circuit-SAT and NP-Completeness

The *Circuit-SAT* problem asks whether there exists an input assignment that makes a given Boolean circuit evaluate to TRUE. Circuit-SAT is a canonical **NP-complete** problem.

In the context of NP, an input assignment serves as a *certificate* for a YES-instance of Circuit-SAT.

The DFS-based verification described above shows that:

- given a candidate assignment,
- the correctness of the circuit's output being TRUE,
- can be verified in polynomial time.

This directly satisfies the definition of NP: YES-instances admit polynomial-time verifiable certificates.

Thus, DFS provides a concrete and efficient mechanism for verifying certificates in NP-complete problems such as Circuit-SAT.

## Why BFS Is Not Necessary

Although Breadth First Search could traverse the circuit, DFS is preferable because:

- it naturally mirrors recursive gate evaluation,
- it supports bottom-up verification,
- it requires minimal auxiliary memory via recursion or a stack.

Thus, DFS is both conceptually and practically well-suited for this task.

## Final Conclusion

The correctness of a Boolean circuit's TRUE output can be verified in polynomial time using DFS.

By performing a depth-first traversal from the output gate and verifying the logical consistency of each gate with its inputs, correctness is ensured in linear time with respect to the circuit size.

## Intuition

Evaluating a Boolean circuit is like checking a proof tree:

- DFS follows the reasoning backward,
- verifies each step,
- and confirms that the final conclusion (TRUE output) is justified.