Task to do:

1. our variable(x) might be different datatype like

1. Int

2. Float

3. bool

4. another variable with char+num like "var22"

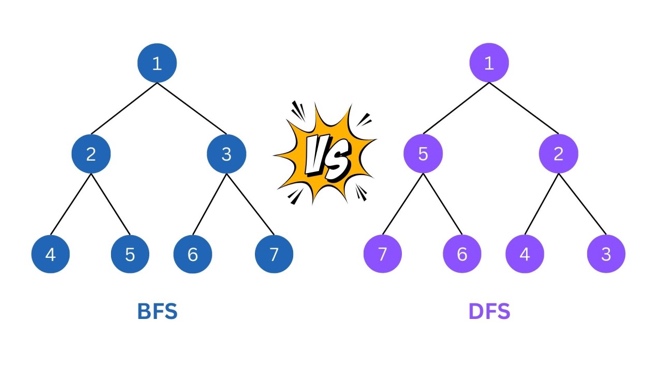
2. Get input as SMT Solver

3. output must be in SMT-lib format (using to\_smt2() )

4. mutation count

Order of mutation also matter

We are using depth first



A formula/constraint F is **valid** if F always evaluates to true for any assignment of appropriate values to its uninterpreted symbols. A formula/constraint F is **satisfiable** if there is some assignment of appropriate values to its uninterpreted symbols under which F evaluates to true. Validity is about finding a proof of a statement; satisfiability is about finding a solution to a set of constraints

Different type of operators

Arithmetic Operators:

* Addition (+): 3 + 5 evaluates to 8.
* Subtraction (-): 7 - 2 evaluates to 5.
* Multiplication (\*): 4 \* 6 evaluates to 24.
* Division (/): 10 / 2 evaluates to 5.0.
* Modulus (%): 10 % 3 evaluates to 1 (remainder of division).

Comparison Operators:

* Equal to (==): 5 == 5 evaluates to True.
* Not equal to (!=): 3 != 5 evaluates to True.
* Greater than (>): 8 > 5 evaluates to True.
* Less than (<): 2 < 7 evaluates to True.
* Greater than or equal to (>=): 6 >= 6 evaluates to True.
* Less than or equal to (<=): 4 <= 3 evaluates to False.

Logical Operators:

* AND (and): True and False evaluates to False.
* OR (or): True or False evaluates to True.
* NOT (not): not True evaluates to False.

Assignment Operators:

* Assignment (=): x = 5 assigns the value 5 to variable x.
* Addition assignment (+=): x += 3 is shorthand for x = x + 3.
* Subtraction assignment (-=): x -= 2 is shorthand for x = x - 2.
* Multiplication assignment (\*=): x \*= 4 is shorthand for x = x \* 4.
* Division assignment (/=): x /= 2 is shorthand for x = x / 2.

Bitwise Operators (for integers):

* Bitwise AND (&): 5 & 3 evaluates to 1.
* Bitwise OR (|): 5 | 3 evaluates to 7.
* Bitwise XOR (^): 5 ^ 3 evaluates to 6.
* Bitwise NOT (~): ~5 evaluates to -6.
* Left shift (<<): 5 << 1 evaluates to 10.
* Right shift (>>): 5 >> 1 evaluates to 2.

Membership Operators:

* In (in): 3 in [1, 2, 3] evaluates to True.
* Not in (not in): 4 not in [1, 2, 3] evaluates to True.

Identity Operators:

* is: x is y evaluates to True if x and y refer to the same object.
* is not: x is not y evaluates to True if x and y refer to different objects.