Phase 2: AST-Based Structural Similarity

1. Objective of Phase 2

The objective of Phase 2 is to measure structural similarity between two source code files. Unlike token-based comparison, which evaluates lexical similarity, AST-based similarity compares the hierarchical syntactic structure of programs.

- Variable renaming.
- Constant modification.
- Formatting differences.
- Reordering of independent statements.
- Partial copying of functions or code blocks.
- Structural plagiarism even when tokens differ.

2. AST Construction Pipeline

Source Code → ANTLR Lexer → ANTLR Parser → Parse Tree → AST Builder → AST → AST Normalizer

ANTLR generates a parse tree based on the grammar. The ASTBuilder class transforms the parse tree into a simplified AST representation that removes syntactic noise while preserving structural hierarchy.

3. AST Design Decisions

- Identifiers are normalized to a generic node.
- Constants are normalized to 'CONST'.
- Function names are not preserved.
- Structural constructs (IF, WHILE, FOR, ASSIGN, etc.) are preserved.
- Operator values are stored but not used in cost calculation.
- AST focuses strictly on structural representation.

4. AST Normalization

- Nested BLOCK nodes are flattened.
- Single-child BLOCK nodes are removed.
- Children are merged to reduce redundant hierarchy.
- Normalization reduces artificial structural differences caused by formatting.

5. Weighted Tree Edit Distance Algorithm

Tree Edit Distance (TED) computes the minimum cost required to transform one AST into another. Allowed operations include insertion, deletion, and substitution of nodes.

Substitution Cost: If node types are identical, cost = 0. If node types differ, cost = max(weight(A), weight(B)).

Similarity Formula: Similarity = 1 − (TreeEditDistance / MaxWeightedSize).

6. Node Weight Model

- FUNCTION – 6
- IF / WHILE / FOR – 5
- ASSIGN / FUNCTION_CALL – 4
- RETURN – 3
- BLOCK – 2
- IDENTIFIER / CONSTANT – 1

7. Ordered vs Unordered Matching

Ordered matching is applied to nodes where child order is semantically significant (e.g., expressions and control-flow structures). Dynamic programming is used for ordered children.

Unordered matching is applied to PROGRAM, BLOCK, and FUNCTION nodes. A greedy matching strategy pairs children based on minimum distance.

Limitation: Greedy matching does not guarantee globally optimal bipartite matching.

8. Subtree Similarity Detection

All subtrees from both ASTs are compared pairwise. The maximum similarity score found among subtree pairs is reported.

This enables detection of partial plagiarism such as copied functions or blocks.

9. Complexity Analysis

- Tree Edit Distance: Approximately $O(N \times M)$.
- Subtree Similarity: Worst case $O(N^2 \times M^2)$.
- Greedy Unordered Matching: $O(n^2)$ per unordered node.

10. Simplifications and Limitations

- Operator values are not penalized if different.
- Greedy unordered matching instead of Hungarian algorithm.
- No semantic or type analysis.
- No data-flow or control-dependency comparison.

11. Conclusion

Phase 2 implements a structural plagiarism detection mechanism based on AST construction, normalization, weighted Tree Edit Distance, ordered and unordered matching, and subtree similarity detection. Despite intentional simplifications, the system significantly improves detection accuracy over token-based similarity.

Examples:
Token similarity drops because many tokens are different. But with AST:

After normalization:

- Function names ignored
- Identifiers
- Same structure:
    - FUNCTION
        - BLOCK
            - ASSIGN
            - RETURN

Both ASTs become structurally identical.

AST similarity: 100%

```
int sum(int a, int b) {
    int result = a + b;
    return result;
}
```

```
int add(int x, int y) {
    int total = x + y;
    return total;
}
```

```
Tokens in file 1: 21
Tokens in file 2: 21

Token-based Similarity: 66.67%
Decision: Similar
```

```
AST-based Similarity: 100.00%

===== AST STRUCTURAL DIFF =====
[MATCH] PROGRAM
  [MATCH] FUNCTION
    [MATCH] PARAM
    [MATCH] PARAM
    [MATCH] VAR_DECL
      [MATCH] BIN_OP
        [MATCH] IDENTIFIER
        [MATCH] IDENTIFIER
    [MATCH] RETURN
      [MATCH] IDENTIFIER
PROGRAM
  FUNCTION
    PARAM
    PARAM
    VAR_DECL
      BIN_OP
        IDENTIFIER
        IDENTIFIER
    RETURN
      IDENTIFIER
PROGRAM
  FUNCTION
    PARAM
    PARAM
    VAR_DECL
      BIN_OP
        IDENTIFIER
        IDENTIFIER
    RETURN
      IDENTIFIER

Max Subtree Similarity: 100.00%
Weighted AST Similarity: 100.00%
AST Decision: Similar
```