Project Presentation
Plagiarism Detector

Team 9
-Siddhant Pasari
-Siddhant Benadikar
-Vivek Nair
-Pallav Gupta

Introduction

- Popular technologies of plagiarism detection are mostly based on text, token and syntax tree.

- Tree based plagiarism detection technology can effectively detect the code plagiarism compared to the other two.

- We propose a more effective plagiarism detection algorithm based on Abstract syntax tree by computing the hash values of the syntax tree nodes and comparing them using Longest Common Subsequence.

Programming Language Targeted

First Phase: **Java**

Future Work: Python

System Features and Functionalities

- Detection between multiple files of two JAVA projects.
- Handling edge cases of empty file, non-Java files.
- Smart Plagiarism checker
    - Converts the code into AST
    - Compares hash values of the nodes of the AST
    - Handling changes in order, variable names etc.
- Easy to use **UI**

Algorithm for Plagiarism Detection

We are using 2 algorithms for the detection of plagiarism:-

1) Longest Common Subsequence
2) Jaccard’s Similarity measure

```
LCS(Node r1, Node r2, M ap < Node, List < Node >> B)

List < Node > subtree1 ← Preorder traversal of the subtree rooted at r1
List < Node > subtree2 ← Preorder traversal of the subtree rooted at r2
m ← Size of the subtree rooted at r1
n ← Size of the subtree rooted at r2
Array < INT, INT > c

for i = 0 to m do
    c(i, 0) ← 0
end for
for j = 0 to n do
    c(0, j) ← 0
end for
```

Reference:
Detection of plagiarism in computer programming using abstract syntax trees
Olav Skjelkvåle & Ligaarden
9th November 2007

```
for i = 1 to m do
    for j = 1 to n do
        v1 ← (i-1)-th element in subtree1
        v2 ← (j-1)-th element in subtree2
        if (label(v1) = label(v2)) and ((label(parent(v1) = label(parent(v2)) or label(v1) = BLOCK) then
            c(i, j) = c(i - 1, j - 1) + 1
        else
            c(i, j) = max(c(i, j - 1), c(i - 1, j))
        end if
    end for
end for
```

Infrastructure/Tools

- **Java Parser** to parse the given files and generate an AST of it
- **Eclipse JDT Parser** for parsing the given files and generating AST
- **Intellij** to write the java program that will perform all the backend processing
- **JUnit 5** for testing
- **HTML, CSS, Javascript, AngularJs** for Frontend and UI.
- **Spring Boot** for backend
- **Github** for version control and tracking the project progress
- **Slack** for team communication

Future Additions

- Side by side file comparison with similar code highlighted.
- Handling python files
- Hosting the project online.
- Download report

Demo!

THANK YOU