**Software Similarity Test System**

**For**

**<Team Name>**

Written by:

Date:

Table of Contents

[1. Introduction 2](#_Toc121140647)

[2. General Framework of the System 3](#_Toc121140648)

[3. Overall Strategy and Approach 4](#_Toc121140649)

[3.1 Similarity Testing Strategy (Basic) 4](#_Toc121140650)

[3.2 Similarity Test Data (Basic) 4](#_Toc121140651)

[4. Execution Plan 5](#_Toc121140652)

[4.1 System Execution Explanation (Basic) 5](#_Toc121140653)

[5. Software Similarity Reporting 6](#_Toc121140654)

[5.1 Software Similarity Definitions (Basic) 6](#_Toc121140655)

[6. System Installation Environment 7](#_Toc121140656)

[6.1 Environment (Basic) 7](#_Toc121140657)

[7. User Manual (Basic) 8](#_Toc121140658)

[8. Appendices 9](#_Toc121140659)

# Introduction

Basic Points:

* The Programming Languages the system supports to test ？

python

* The Size of System it supports to test (How many lines of codes it support? How many modules it supports? etc.,)

200MB

* The Strategy used for the system (Line by Line Character Matching? Control Flow Matching? Data Flow Matching?)

Two ways:

1. DIFFLIB

Due to the characteristic of python , Programmers have a lot of freedom in programming ,so we can compare different modules. For example, code A includes module a ,code B includes module b. we just use a function “SequenceMatcher” in “Difflib”. And use function “Levenshtein” distance to calculate the similarity between to strings，and return a value between 0 and 1.

And the whole similarity will calculate by “the copied module length/ the whole module length”

1. Winnowing algorithm

Through a combination of Hashing methods and sliding window to create a fingerprint. And we compare the fingerprints of two code with Jaccard similarity coefficient.

The details of algorithm based on “[Winnowing: local algorithms for document](https://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf)

[Fingerprinting](https://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf)”

Extra Points:

* S/C Structure
* Visualized Result

# General Framework of the System

Basic Points:

* The functional structure of the system

Extra Points:

* Defined by your team

# Overall Strategy and Approach

## Similarity Testing Strategy (Basic)

**1.Difflib**:

Due to the characteristic of python , Programmers have a lot of freedom in programming ,so we can compare different modules. For example, code A includes module a ,code B includes module b. we just use a function “SequenceMatcher” in “Difflib”. And use function “Levenshtein” distance to calculate the similarity between to strings，and return a value between 0 and 1.And the whole similarity will calculate by “the copied module length/ the whole module length”

**2. Winnowing algorithm**

Through a combination of Hashing methods and sliding window to create a fingerprint. And we compare the fingerprints of two code with Jaccard similarity coefficient.

The details of algorithm based on “Winnowing: local algorithms for document Fingerprinting”

## Similarity Test Data (Basic)

# Execution Plan

## System Execution Explanation (Basic)

# Software Similarity Reporting

## Software Similarity Definitions (Basic)

|  |  |
| --- | --- |
| **Critical** |  |
| **Medium** |  |
| **Low** |  |

# System Installation Environment

## Environment (Basic)

# User Manual (Basic)

# Appendices