Plagiarism Validation

[Problem Definition 2](#_Toc164724682)

[General Idea 2](#_Toc164724683)

[Main Goals in Details 3](#_Toc164724684)

[Project Requirements 4](#_Toc164724685)

[Required Implementation 4](#_Toc164724686)

[Input 4](#_Toc164724687)

[Output 5](#_Toc164724688)

[Test Cases 5](#_Toc164724689)

[Deliverables 6](#_Toc164724690)

[Implementation (60%) 6](#_Toc164724691)

[Document (40%) 6](#_Toc164724692)

[Allowed Codes 6](#_Toc164724693)

[Milestones 6](#_Toc164724694)

[BONUSES 6](#_Toc164724695)

# Problem Definition

In our project, we leverage MOSS (Measure of Software Similarity), a system designed to detect similarities in software. Our goal is to enhance the review process by making it more efficient and manageable for evaluators. MOSS helps us identify matching pairs that exhibit similarities. However, among these pairs, some matches are deemed insignificant, and cyclical patterns can emerge—for instance, if file 1 is similar to file 2, file 2 to file 3, and file 3 back to file 1, creating a cycle. The aim of our project is to address these issues by filtering out such cycles and prioritizing the matching pairs that show the highest degree of similarity, while disregarding the lower and less consequential pairs that do not significantly impact the analysis.

To achieve this, we propose an optimized approach that not only identifies all connected components within the similarity graph generated by MOSS but also refines these components by removing cycles. This refinement process involves constructing a Maximum Spanning Tree (MST) for each component, focusing on the edges with the highest similarity scores to eliminate redundancy. By doing so, we can ensure that the evaluators are presented with the most pertinent and impactful matches, streamlining the review process, and enhancing the overall efficiency of similarity detection.

This project is expected to significantly reduce the workload on evaluators by automating the process of sifting through numerous matches to identify the ones that truly matter. Through intelligent filtering and prioritization, we can focus on the similarities that warrant further investigation, thereby making the best use of the evaluators' time and resources.

# General Idea

The general idea of our project revolves around the sophisticated analysis of software similarity by employing a two-step optimization process within the framework provided by MOSS (Measure of Software Similarity). Initially, our approach seeks to meticulously identify all connected components within the similarity graph generated by MOSS. A connected component is a subset of files or software snippets where each pair within the subset shares a direct or indirect similarity, forming a cohesive cluster within the larger network of comparisons.

Once these connected components are identified, the next pivotal step involves the exclusion of cyclical redundancy within each component. These cycles can obscure the clarity of results and may lead to inefficiencies in the evaluation process.

To address this challenge, it’s required to construct a Maximum Spanning Tree (MST) for each connected component. The MST is a concept borrowed from graph theory, which in our context, is applied to prune the similarity graph by retaining only the most significant similarity connections, or "edges," within each component. The goal here is to maximize the sum of the similarity scores in the tree, ensuring that the retained similarities are those with the highest values, hence the term "Maximum Spanning Tree." This process effectively eliminates less significant and redundant similarities, streamlining the component to its most essential connections without compromising the integrity of the similarity analysis.

# Main Goals in Details

Your task is **to read in an excel file** containing an N pair of files which each of them contain the hyperlink of the file followed by its similarity. To achieve this concept on the following file we should follow these steps:

1. **Find Groups:** First, we find all groups of files that are similar to each other in some way, making an extensive network of connections. Next, for each group, shows its file IDs and their average matching similarity.
2. **Refine Groups:** In these groups, sometimes things are going in circles - like when A is similar to B, B to C, and C back to A. These circles don't add much value, so we want to get rid of them. To simplify these groups, we only keep the strongest similarities and remove the weaker ones. It's like pruning a tree to keep only the best branches. This step makes sure we focus on the most important similarities.

**NOTE:** during the step2 (MST), consider the max similarity of each pair as its representative similarity.

The following figures show the expected input & output of such idea.

|  |  |  |
| --- | --- | --- |
| **File 1** | **File 2** | **Lines Matched** |
| [D:/Source/1/ (25%)](http://moss.stanford.edu/results/0/4012916089008/match0.html) | [D:/Source/2/ (36%)](http://moss.stanford.edu/results/0/4012916089008/match0.html) | 84 |
| [D:/Source/2/ (75%)](http://moss.stanford.edu/results/0/4012916089008/match1.html) | [D:/Source/3/ (76%)](http://moss.stanford.edu/results/0/4012916089008/match1.html) | 52 |
| [D:/Source/3/ (64%)](http://moss.stanford.edu/results/0/4012916089008/match2.html) | [D:/Source/1/ (13%)](http://moss.stanford.edu/results/0/4012916089008/match2.html) | 28 |
| [D:/Source/4/ (64%)](http://moss.stanford.edu/results/0/4012916089008/match3.html) | [D:/Source/5/ (13%)](http://moss.stanford.edu/results/0/4012916089008/match3.html) | 28 |
| [D:/Source/5/ (47%)](http://moss.stanford.edu/results/0/4012916089008/match4.html) | [D:/Source/6/ (49%)](http://moss.stanford.edu/results/0/4012916089008/match4.html) | 40 |
| [D:/Source/6/ (47%)](http://moss.stanford.edu/results/0/4012916089008/match5.html) | [D:/Source/4/ (48%)](http://moss.stanford.edu/results/0/4012916089008/match5.html) | 41 |

.

Figure1: Input File

|  |  |
| --- | --- |
| **FIRST: Groups Stats** | **SECOND: Refined Pairs (MST)** |
| |  |  |  | | --- | --- | --- | | **Group Index** | **Files** | **Average Similarity** | | 1 | 2, 1, 3 | 48.2 | | 2 | 4, 5, 6 | 44.7 | | |  |  |  | | --- | --- | --- | | File 1 | File 2 | Line Matches | | [D:/Source/2/ (75%)](http://moss.stanford.edu/results/0/4012916089008/match1.html) | [D:/Source/3/ (76%)](http://moss.stanford.edu/results/0/4012916089008/match1.html) | 52 | | [D:/Source/3/ (64%)](http://moss.stanford.edu/results/0/4012916089008/match2.html) | [D:/Source/1/ (13%)](http://moss.stanford.edu/results/0/4012916089008/match2.html) | 28 | | [D:/Source/4/ (64%)](http://moss.stanford.edu/results/0/4012916089008/match3.html) | [D:/Source/5/ (13%)](http://moss.stanford.edu/results/0/4012916089008/match3.html) | 28 | | [D:/Source/5/ (47%)](http://moss.stanford.edu/results/0/4012916089008/match4.html) | [D:/Source/6/ (49%)](http://moss.stanford.edu/results/0/4012916089008/match4.html) | 40 | |

Figure 2: The Expected result.

# Project Requirements

## Required Implementation

|  |  |
| --- | --- |
| **Requirement** | **Performance** |
| 1. **Read** a **html file (or excel)** containing N matching pairs. Each pair consists of file1 path, file2 path, file1 hyperlink, file2 hyperlink, similarity percentage (%) of each file | **Time:** should be **bounded by O(N)**, N is the number of matching pairs |
| 1. **Construct** the Graph | **Time:** should be **bounded by O(N)**, N is the number of matching pairs |
| 1. Find **ALL groups and their statistics** | **Time:** should be **bounded by O(N)**, N is the number of matching pairs |
| 1. Refine each group, **by** finding its **max spanning tree** | **Time:** should be **bounded by**  Where are number of pairs & files of each group |
| 1. **Output**    1. Group statistics file.    2. Refined matching pairs file. |  |

## Input

**Matching Excel File:**

Given an Excel file with N rows, each consisting of three columns, as shown below

1. file one, which includes hyperlink1 and similarity1,
2. file two, which also includes hyperlink2 and similarity2,
3. Lines matched.

**Text1.xlsx:**

|  |  |  |
| --- | --- | --- |
| **File 1** | **File 2** | **Lines Matched** |
| [D:/Source/1/ (25%)](http://moss.stanford.edu/results/0/4012916089008/match0.html) | [D:/Source/2/ (36%)](http://moss.stanford.edu/results/0/4012916089008/match0.html) | 84 |
| [D:/Source/2/ (75%)](http://moss.stanford.edu/results/0/4012916089008/match1.html) | [D:/Source/3/ (76%)](http://moss.stanford.edu/results/0/4012916089008/match1.html) | 52 |
| [D:/Source/3/ (64%)](http://moss.stanford.edu/results/0/4012916089008/match2.html) | [D:/Source/1/ (13%)](http://moss.stanford.edu/results/0/4012916089008/match2.html) | 28 |
| [D:/Source/4/ (64%)](http://moss.stanford.edu/results/0/4012916089008/match3.html) | [D:/Source/5/ (13%)](http://moss.stanford.edu/results/0/4012916089008/match3.html) | 28 |
| [D:/Source/5/ (47%)](http://moss.stanford.edu/results/0/4012916089008/match4.html) | [D:/Source/6/ (49%)](http://moss.stanford.edu/results/0/4012916089008/match4.html) | 40 |
| [D:/Source/6/ (47%)](http://moss.stanford.edu/results/0/4012916089008/match5.html) | [D:/Source/4/ (48%)](http://moss.stanford.edu/results/0/4012916089008/match5.html) | 41 |

## Output

Two excel files, as shown in the example

1. **Groups Statistics file:** with three columns
   1. the group index.
   2. ids (submissions) of every group.
   3. the average matching similarity of each group.

|  |  |  |
| --- | --- | --- |
| **Group Index** | **Files** | **Average Similarity** |
| 1 | 2, 1, 3 | 48.2 |
| 2 | 4, 5, 6 | 44.7 |

1. **Refined Matches file:** containing the pairs after MST, with three columns:
   1. file one, which includes hyperlink1 and similarity1,
   2. file two, which also includes hyperlink2 and similarity2,
   3. Lines matched.

|  |  |  |
| --- | --- | --- |
| File 1 | File 2 | Line Matches |
| [D:/Source/2/ (75%)](http://moss.stanford.edu/results/0/4012916089008/match1.html) | [D:/Source/3/ (76%)](http://moss.stanford.edu/results/0/4012916089008/match1.html) | 52 |
| [D:/Source/3/ (64%)](http://moss.stanford.edu/results/0/4012916089008/match2.html) | [D:/Source/1/ (13%)](http://moss.stanford.edu/results/0/4012916089008/match2.html) | 28 |
| [D:/Source/4/ (64%)](http://moss.stanford.edu/results/0/4012916089008/match3.html) | [D:/Source/5/ (13%)](http://moss.stanford.edu/results/0/4012916089008/match3.html) | 28 |
| [D:/Source/5/ (47%)](http://moss.stanford.edu/results/0/4012916089008/match4.html) | [D:/Source/6/ (49%)](http://moss.stanford.edu/results/0/4012916089008/match4.html) | 40 |

## Test Cases

### Sample Test:

* **Goal:** test the correctness
* **Given:** six small-size pairs with the expected O/P
* **Location:** \Test Cases\Sample

### Complete Test: (TBD…)

* **Goal:** test the efficiency (beside correctness)
* **Given:** 3 levels
  1. Small: # pairs O(100’s)
  2. Medium: # pairs O(1000’s)
  3. Large: # pairs O(100000’s)

# Deliverables

## Implementation (60%)

1. **Read** in an Excel file containing an N row of three columns (file 1, file 2, lines matched) – representing the matching pairs.
2. **Find all the groups** and calculate their statistics**.**
3. **Refine the groups by finding MST** of each group.
4. **Output** two Excel files:
   1. Groups Statistic File
   2. Refined Matches File

## Document (40%)

1. Entire Source code.
2. Detailed analysis of your code.
3. The efficiency of your code through recording execution time of complete cases.

## Allowed Codes

1. Code for dealing with Excel files (read, write…etc)
2. Open-source code for the **disjoint sets** and/or **priority queue**. You **MUST** **understand** and **analyze** it!

# Milestones

|  |  |  |
| --- | --- | --- |
|  | **Deliverables** | **Due to** |
| **Milestone1** | 1. **Read** and parse the Excel file 2. **Find all the groups** and calculate their statistics**.** 3. **Refine the groups** by finding MSTof each group. 4. **Output** two Excel files: 5. Groups Statistic File 6. Refined Matches File | **(START of Practical Exam Week)** |
| **For Milestone1:**   * + **MUST** deliver the required tasks and **ENSURE** it’s worked correctly.   + **MUST** deliver in your scheduled time (TO BE ANNOUNCED) | | |

# BONUSES

1. Compare two (or more) MST algorithms and demonstrate the differences in time.
2. Any enhancement that significantly reduces the time improves upon the existing algorithm's performance.