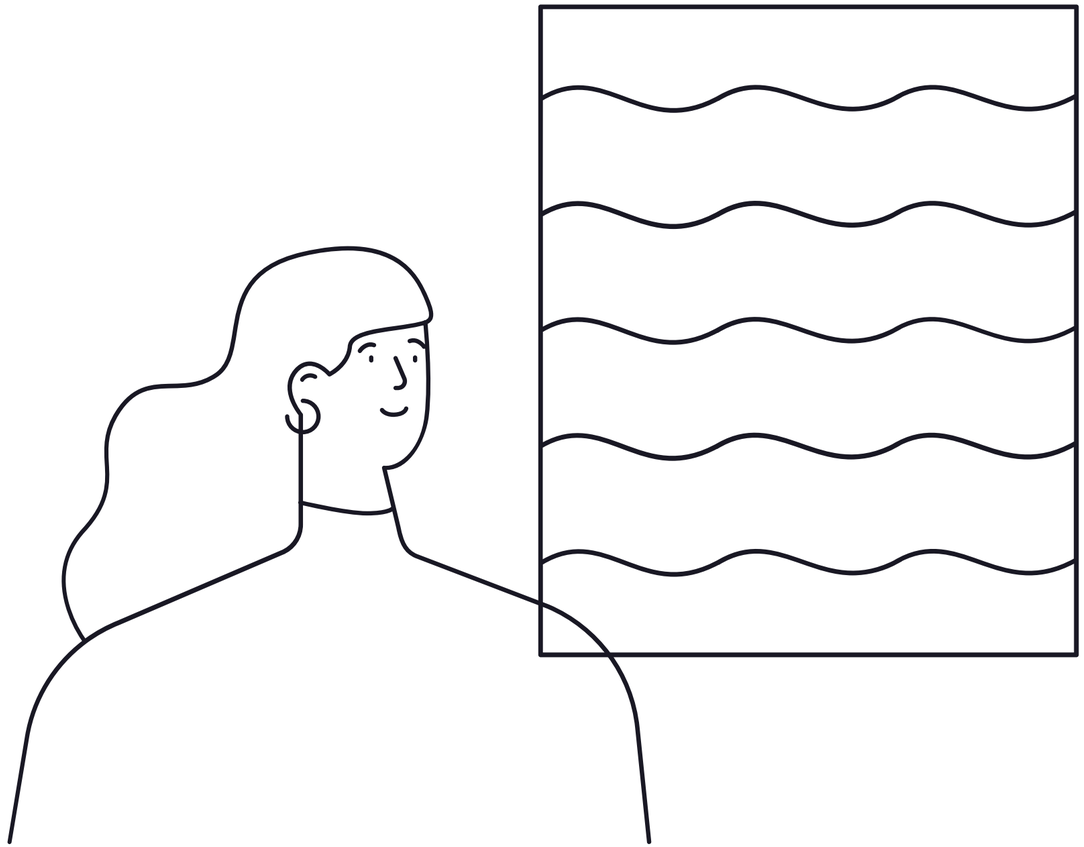
Plagiarism Detection

Kareem Elnaghy

What is the problem?



PLAGIARISM

What is Plagiarism?

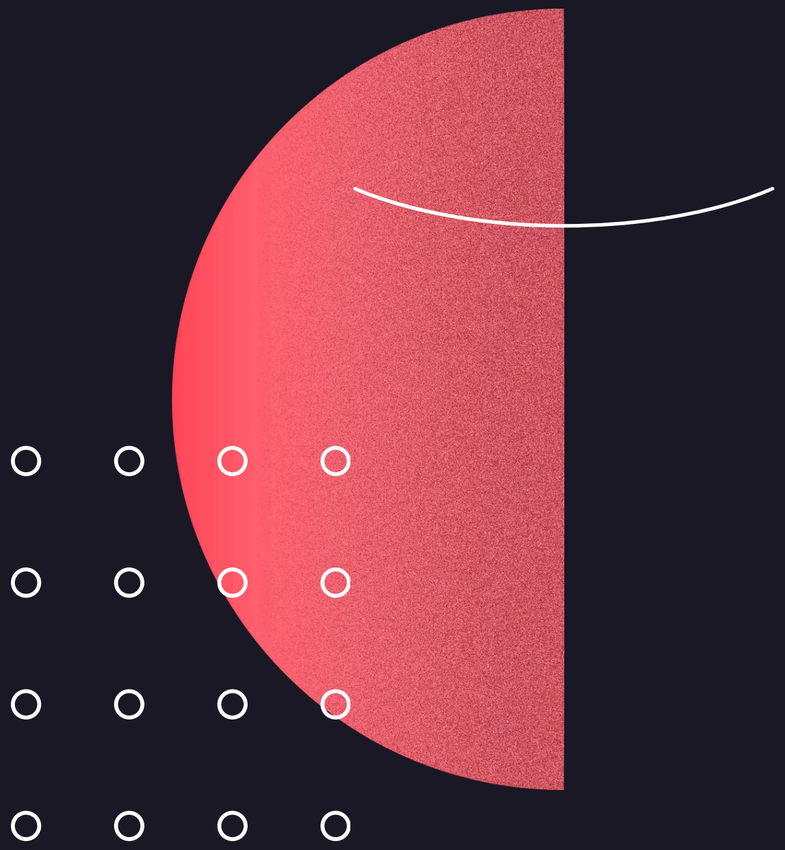
- Plagiarism is the act of taking someone else’s work or ideas and taking credit for it.

Why is it important?

- Academic Integrity
- Content Authenticity
- Legal and Ethical Compliance
- Efficiency

Complexities

- Textual Complexity
- Efficiency
- Balancing Trade-offs
- Ambiguities in matching



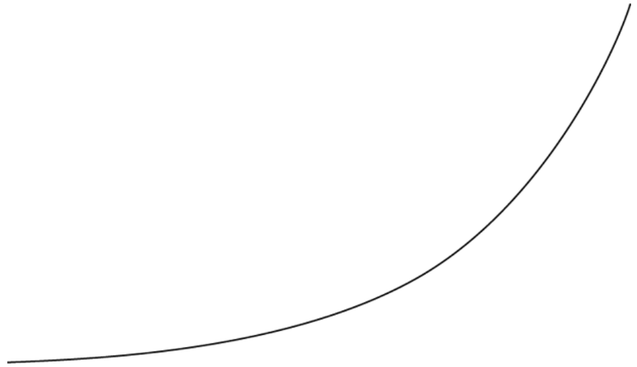
Why is it worth solving?

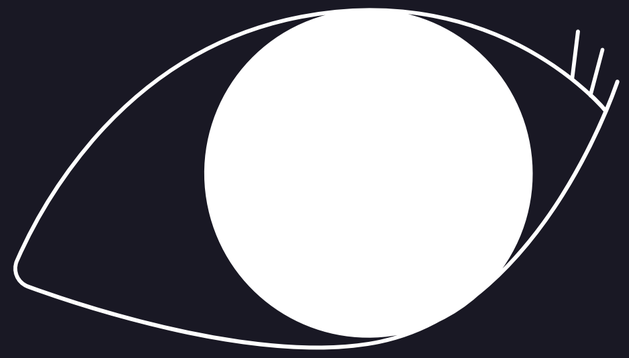
BECAUSE ...

Educational Impact
Contribute to better educational outcomes by helping students understand the importance of originality and ethical writing.

Economic Implications
Protects original content ensuring creators, researchers and writers receive credit and financial benefits for their work.

Growing Need for Automation
With the exponential growth of digital content, manual methods for detecting plagiarism are becoming impractical.





ABSTRACTING THE PROBLEM

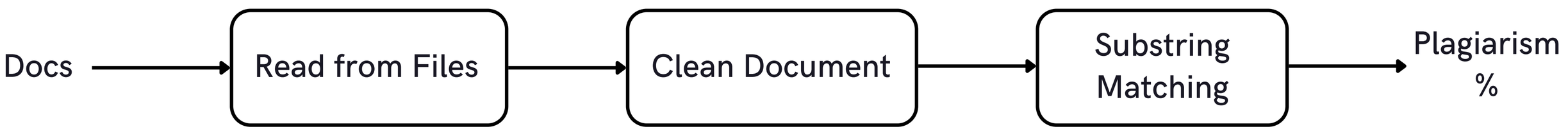
STRING-MATCHING PROBLEM

- The goal is to compare two documents and quantify their similarity by identifying overlapping substrings

Key Components

- Inputs are two text documents
- Output is the % of the source document that matches the target document.
- Constraints
  - Ignoring non-alphabetic characters
  - Case-insensitive comparison
  - Account for overlapping substrings

ALGORITHM


Docs
Read from Files
Clean Document
Substring Matching
Plagiarism %

ALGORITHMIC APPROACHES

Brute Force Approach

- Checks every possible substring.
- Ensures Correctness since it examines all possible matches
- Trade-offs
  - Simple to implement
  - Inefficient due to redundant comparisons

Dynamic Programming Approach

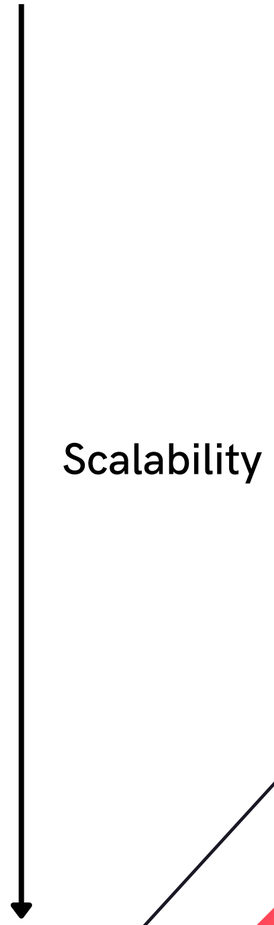
- Uses a 2D table to store the length of the longest common substrings.
- Trade-offs
  - Better Time complexity vs. BF
  - Worst Space Complexity

Transform and Conquer

- Rabin-Karp Algorithm uses hash-based techniques
- Trade-offs
  - Best time and space complexity
  - Complexity in Implementation

EFFICIENCY

Approach	Time Complexity	Space Complexity	Best Use Case
Brute Force	O(N*M*min(N,M))	O(1)	Small Datasets
Dynamic Programming	O(N*M)	O(N*M)	Medium datasets
Transform and Conquer	O(N*M)	O(1)	Large datasets



Thank you