

DETECCIÓN DE PLAGIO

A través de Control Flow Graphs

Ian Padrón | A01708940

Ramona Najera | A01423596

Armando Rosas | A01704132



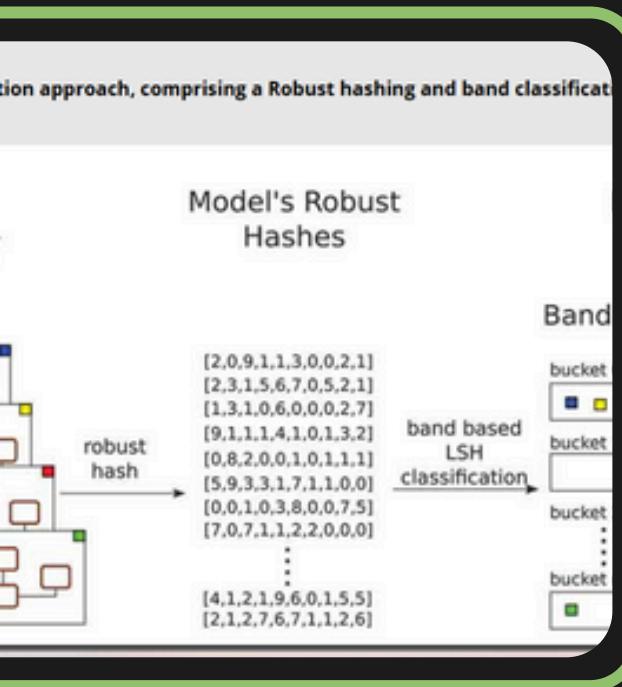
INTRODUCCIÓN



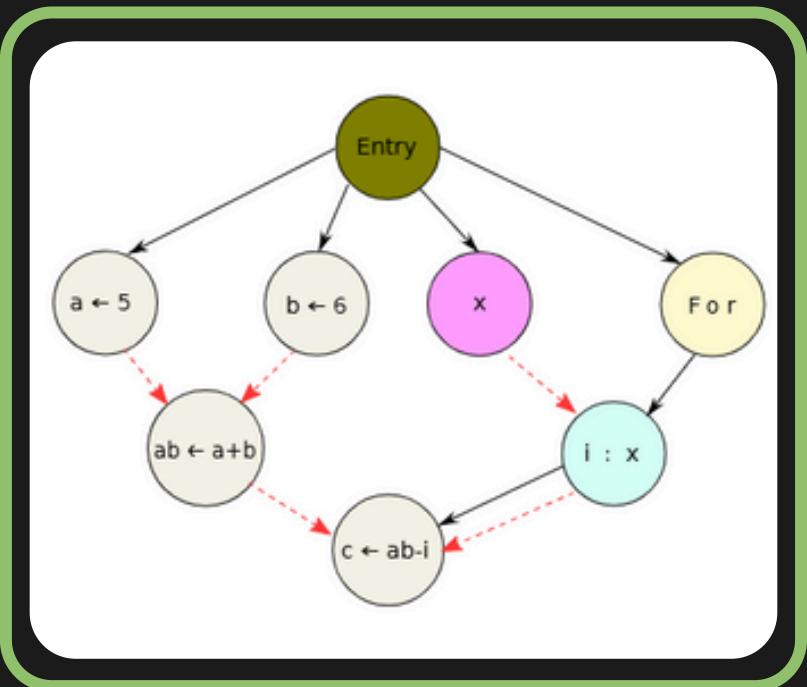


MARCO TEÓRICO

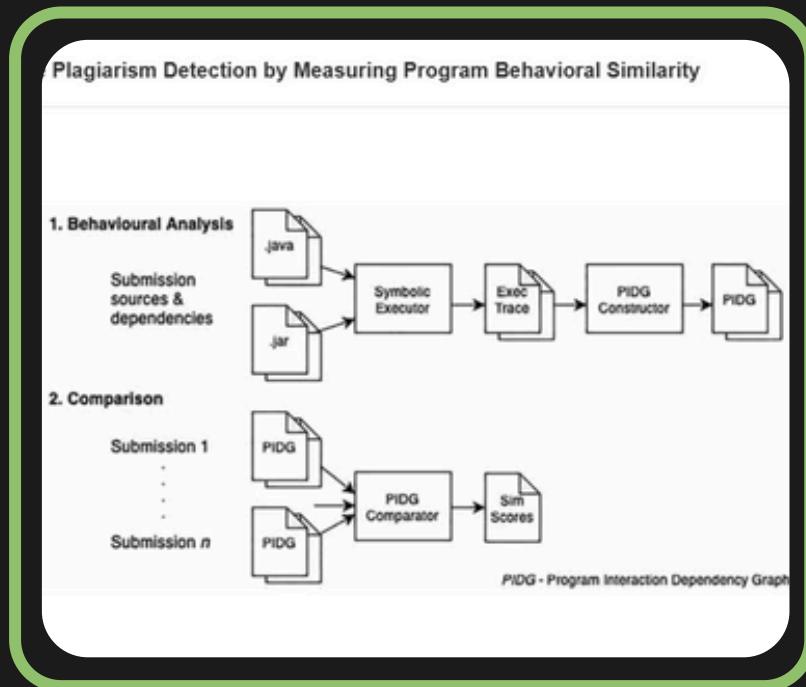
Hashing



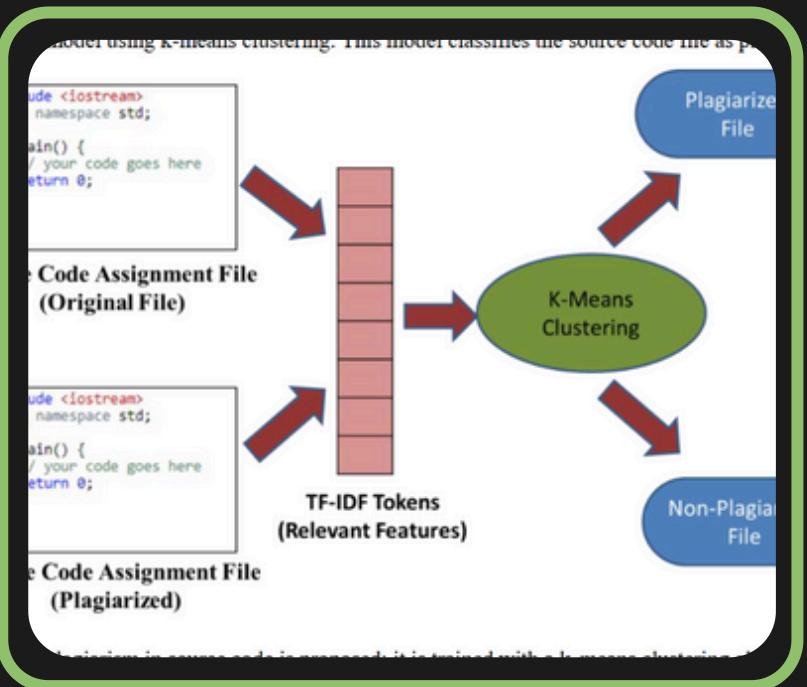
Abstract Syntax Tree
Program Dependence Graphs



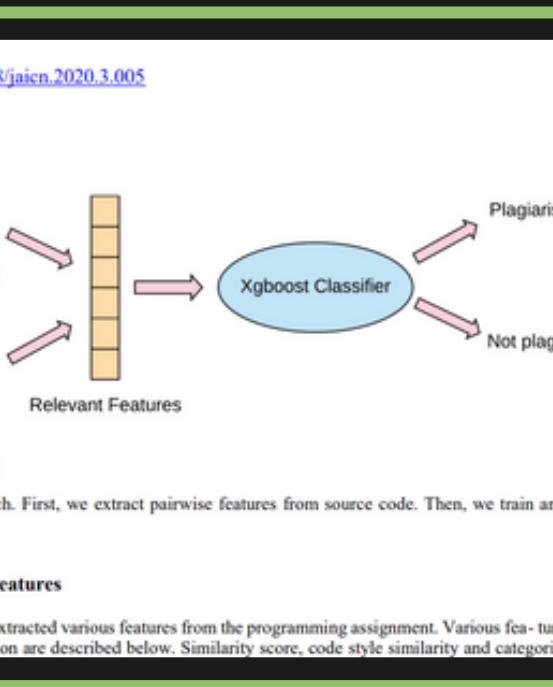
Interaction Dependence Graphs



TF-IDF + clustering techniques



XGboost model

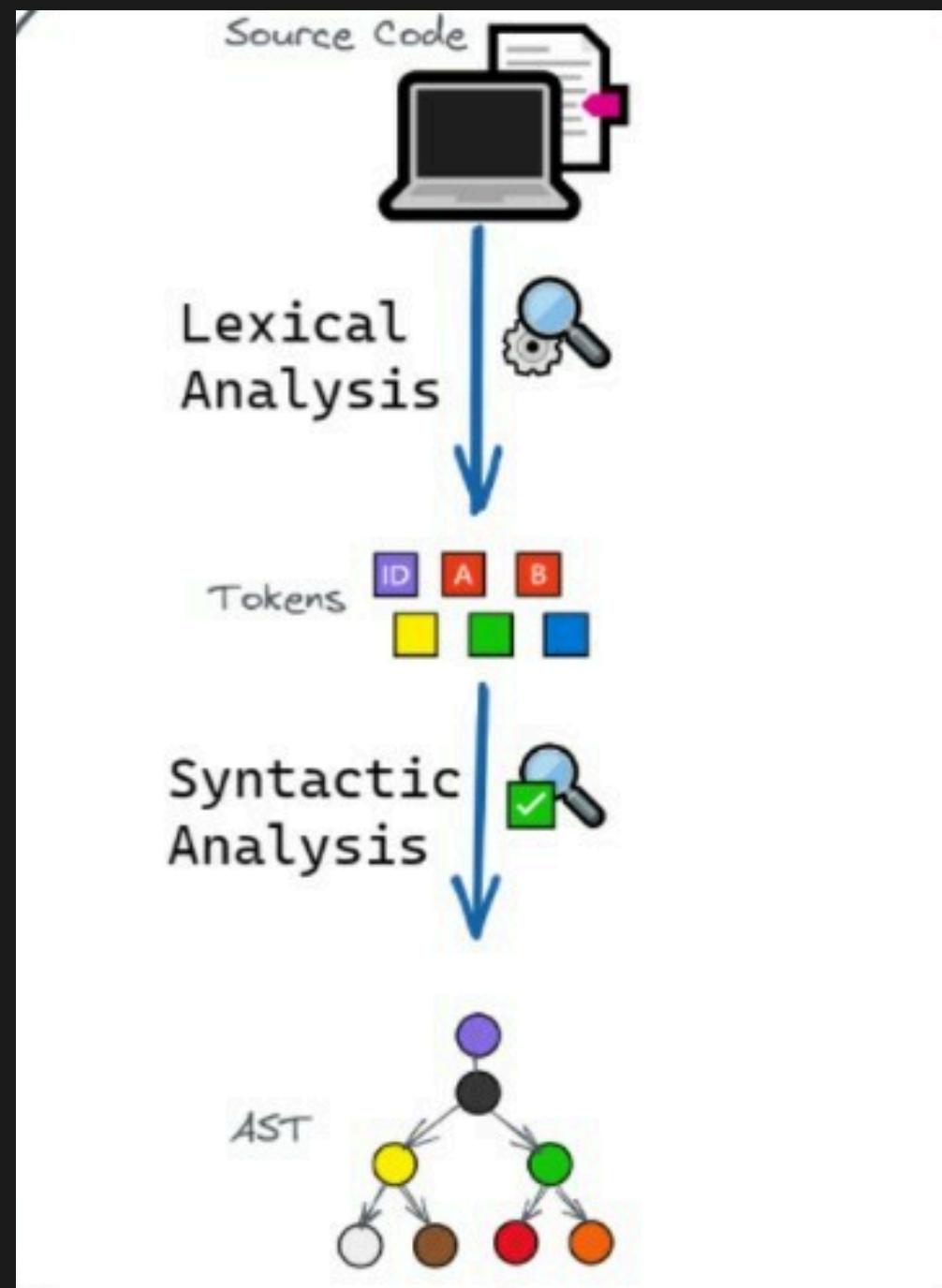


Model Driven
Engineering

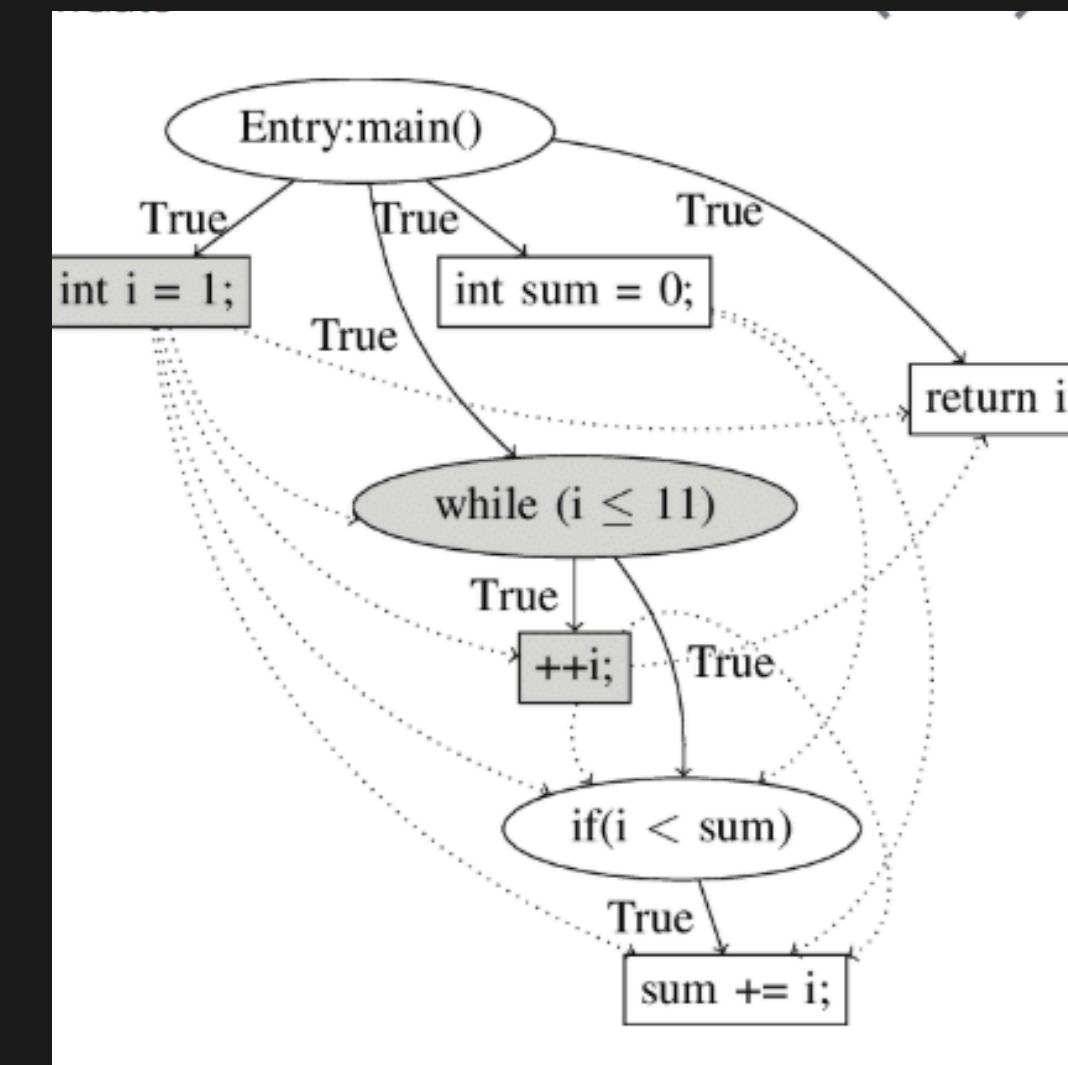
Control and data
dependency

Symbolic execution

Classification tasks with
feature extraction



Program Dependency Graph



The diagram illustrates a system with three states, S_1 , S_2 , and S_3 , represented by colored circles (blue, pink, and cyan respectively). The transitions between states are labeled with matrices A :

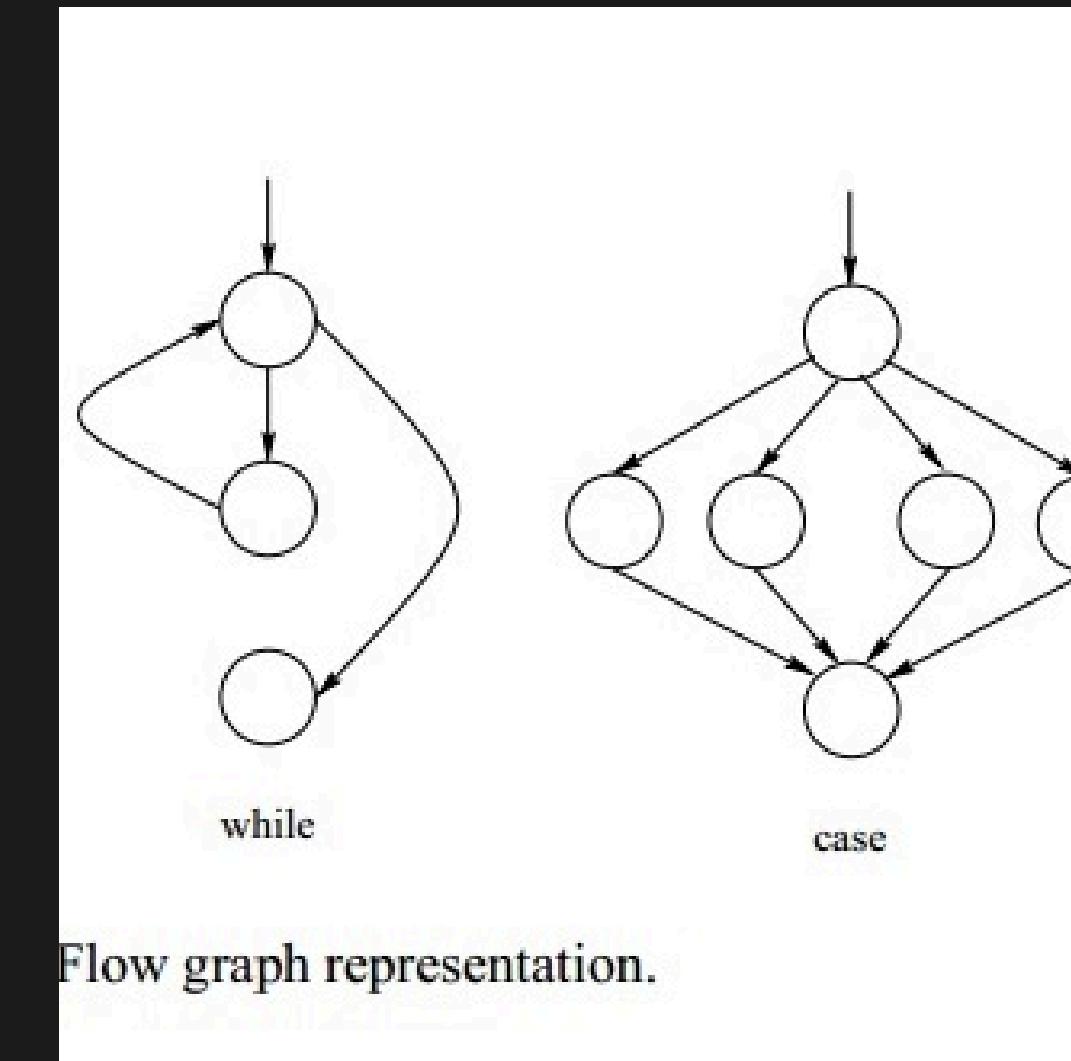
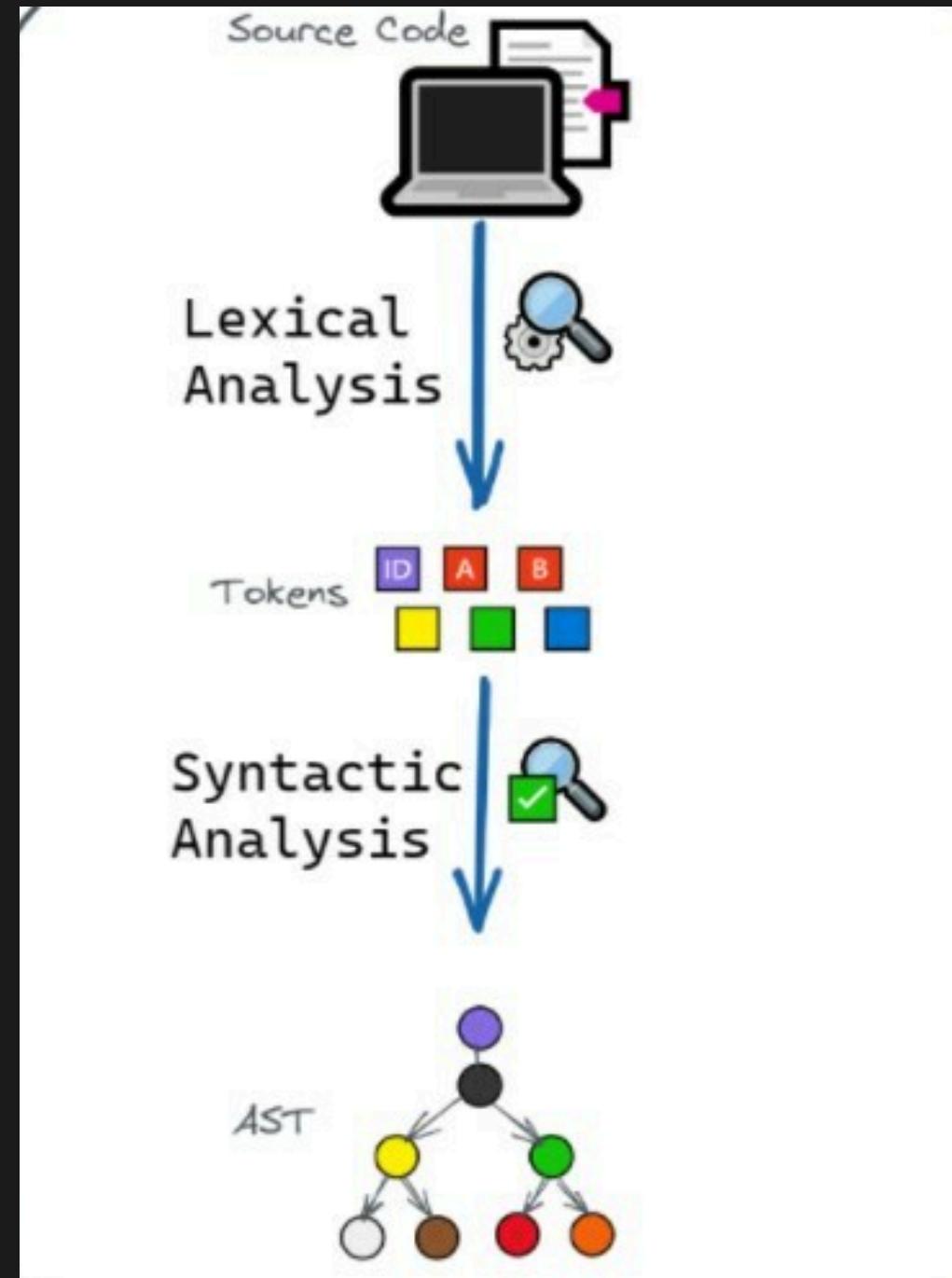
- $A_{(1,1)}$: Self-loop on S_1 .
- $A_{(2,1)}$: Transition from S_2 to S_1 .
- $A_{(1,3)}$: Transition from S_1 to S_3 .
- $A_{(1,2)}$: Transition from S_1 to S_2 .
- $A_{(3,1)}$: Transition from S_3 to S_1 .
- $A_{(3,2)}$: Transition from S_3 to S_2 .
- $A_{(2,3)}$: Transition from S_2 to S_3 .
- $A_{(2,2)}$: Self-loop on S_2 .
- $A_{(3,3)}$: Self-loop on S_3 .

Three small plots in dashed boxes show frequency distributions:

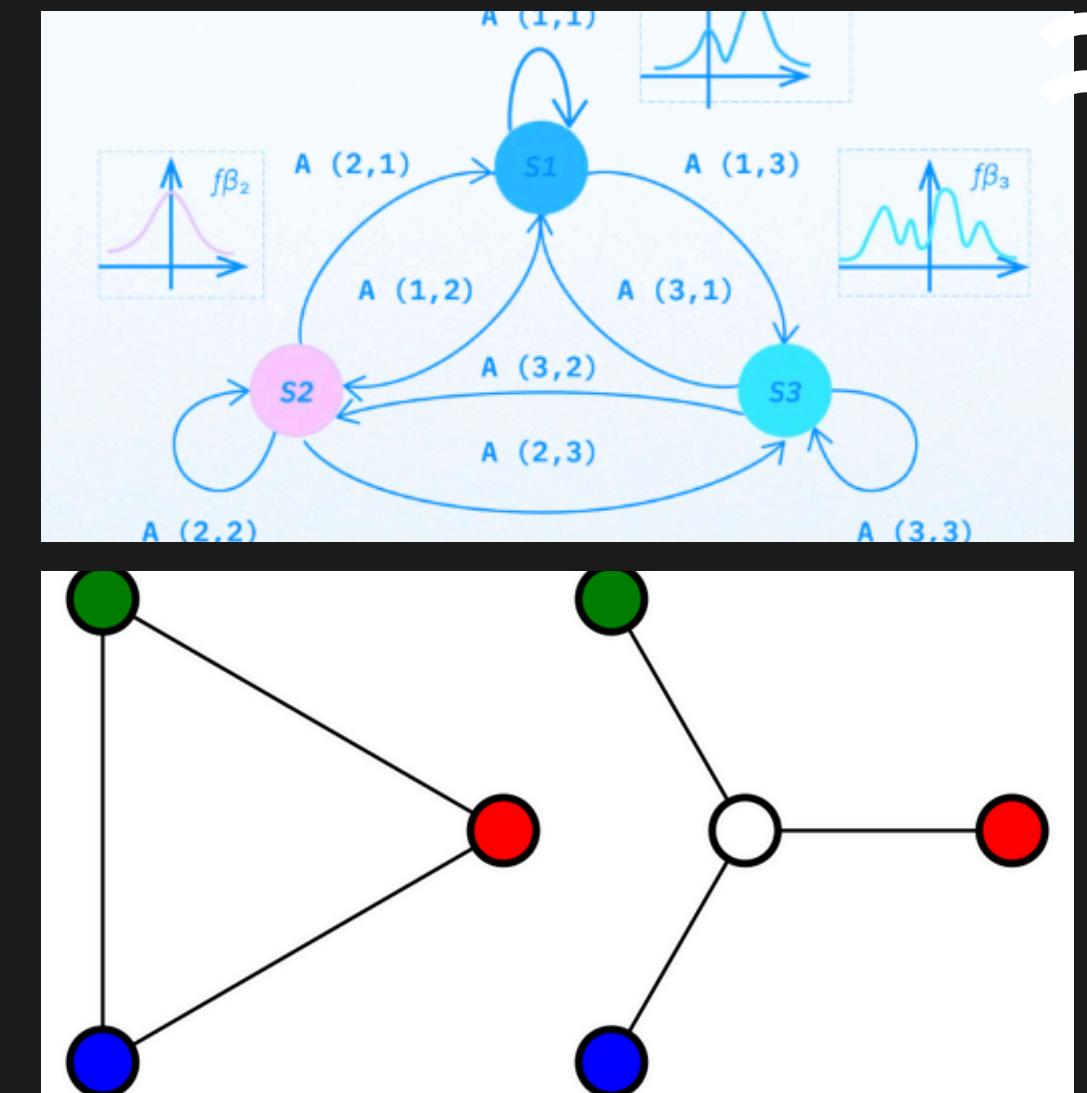
- $f\beta_1$: A plot with two peaks.
- $f\beta_2$: A plot with one peak.
- $f\beta_3$: A plot with two peaks.

Análisis de similitud con cadenas de Markov

PROUESTA INICIAL



Control Flow Graph



*Cadenas de Markov
+
Graph Edit Distance (GED)*

Abstract Syntax Tree

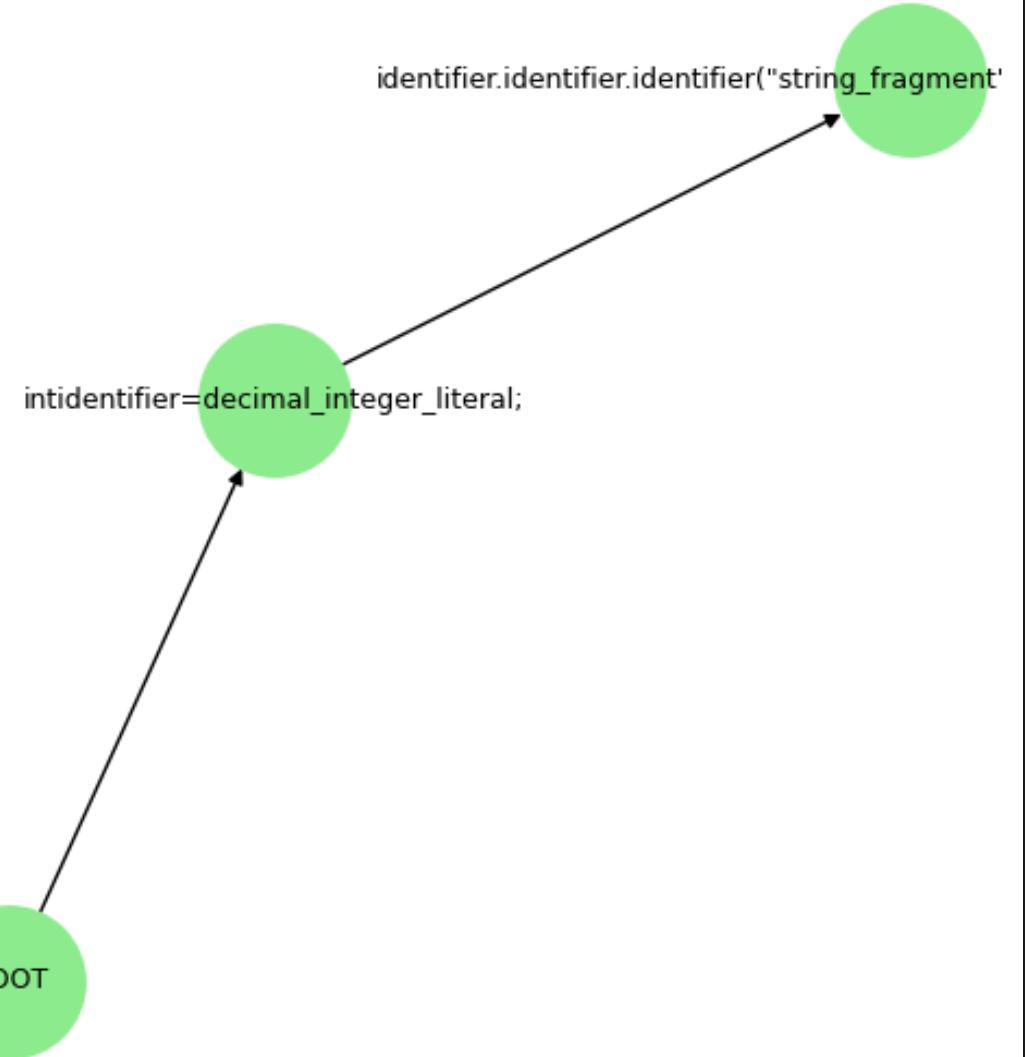
i

PROPUESTA FINAL



VARIABLES

Java Control Flow Graph (CFG)



Estructuras de control procesadas

Tags de los nodos en el AST

⌚Específicos

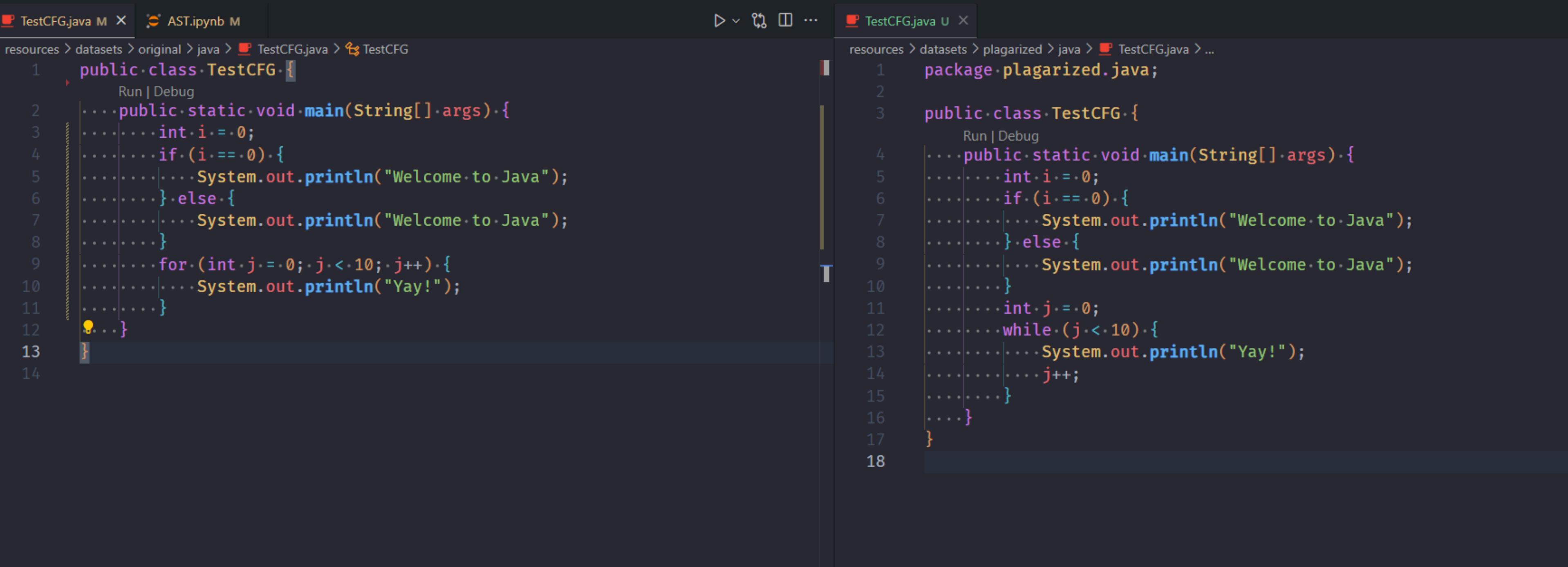
🔒No dependen del programador

✓Consistentes

```
translation_unit [0, 0] - [4, 0]
function_definition [0, 0] - [3, 1]
type: primitive_type [0, 0] - [0, 3]
declarator: function_declarator [0, 4] - [0, 10]
declarator: identifier [0, 4] - [0, 8]
parameters: parameter_list [0, 8] - [0, 10]
  "(" [0, 8] - [0, 9]
  ")" [0, 9] - [0, 10]
body: compound_statement [0, 11] - [3, 1]
  "{" [0, 11] - [0, 12]
declaration [1, 1] - [1, 17]
  type: primitive_type [1, 1] - [1, 4]
declarator: init_declarator [1, 5] - [1, 16]
  declarator: identifier [1, 5] - [1, 9]
  "=" [1, 10] - [1, 11]
  value: number_literal [1, 12] - [1, 16]
  ";" [1, 16] - [1, 17]
expression_statement [2, 1] - [2, 32]
binary_expression [2, 1] - [2, 31]
  left: binary_expression [2, 1] - [2, 24]
  left: binary_expression [2, 1] - [2, 16]
    left: identifier [2, 1] - [2, 5]
    operator: "<<" [2, 6] - [2, 8]
    right: string_literal [2, 9] - [2, 16]
      """ [2, 9] - [2, 10]
      string_content [2, 10] - [2, 15]
      """ [2, 15] - [2, 16]
    operator: "<<" [2, 17] - [2, 19]
    right: identifier [2, 20] - [2, 24]
operator: "<<" [2, 25] - [2, 27]
right: string_literal [2, 28] - [2, 31]
  """ [2, 28] - [2, 29]
  string_content [2, 29] - [2, 30]
  """ [2, 30] - [2, 31]
  ";" [2, 31] - [2, 32]
  }" [3, 0] - [3, 1]
```



RESULTADOS OBTENIDOS

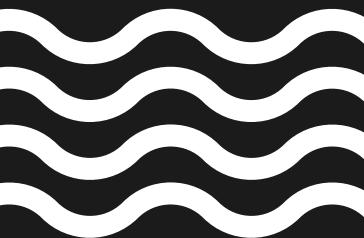


```
TestCFG.java M X AST.ipynb M resources > datasets > original > java > TestCFG.java > TestCFG
1 public class TestCFG {
2     Run | Debug
3     public static void main(String[] args) {
4         int i = 0;
5         if (i == 0) {
6             System.out.println("Welcome to Java");
7         } else {
8             System.out.println("Welcome to Java");
9         }
10        for (int j = 0; j < 10; j++) {
11            System.out.println("Yay!");
12        }
13    }
14 }
```

```
TestCFG.java U X resources > datasets > plagiarized > java > TestCFG.java > ...
1 package plagiarized.java;
2
3 public class TestCFG {
4     Run | Debug
5     public static void main(String[] args) {
6         int i = 0;
7         if (i == 0) {
8             System.out.println("Welcome to Java");
9         } else {
10            System.out.println("Welcome to Java");
11        }
12        int j = 0;
13        while (j < 10) {
14            System.out.println("Yay!");
15            j++;
16        }
17    }
18 }
```

Markov-based similarity: 0.6000

GED-based similarity: 0.9118



COMPARACIÓN CON EL ESTADO DEL ARTE

PROPUESTA IDEADA

ESTADO DEL ARTE

| | |
|---|--|
| Menor complejidad en los algoritmos a implementar y utilizar | Mayor eficiencia y capacidad para reconocer códigos complejos |
| Escalabilidad del código. Mayor área de mejora | Algoritmos robustos. Mayor precisión al reconocer plagio |
| Reproducible con los conocimientos del curso | Requiere de un gran trasfondo de conocimientos |
| = Posibilidad de agregar nuevos lenguajes | Usualmente enfocados a un lenguaje en particular |



CONCLUSIÓN

Desarrollamos un sistema de detección de plagio inspirado en el artículo SimilaR ([Bartoszuk et al., 2020](#)), adaptando su enfoque a una versión más sencilla y accesible. Si bien nuestras técnicas son menos precisas que las propuestas en el artículo original, su implementación es más directa y práctica, lo que permite una aplicación funcional con menor complejidad técnica.

- 
- Bartoszuk M, Gagolewski M. **SimilaR**: R Code Clone and Plagiarism Detection. *R Journal* [Internet]. 2020 Jun 1 [cited 2025 Apr 4];12(1):367–85. Available from: <https://research.ebsco.com/linkprocessor/plink?id=1115fc70-4a08-3234-a931-e12b27951f62>
- Cheers, Y. Lin and S. P. Smith, Academic Source Code Plagiarism Detection by Measuring Program **Behavioral Similarity**, in *IEEE Access*, vol. 9, pp. 50391-50412, 2021 <https://ieeexplore.ieee.org/abstract/document/9388651>
- Raddam Sami Mehsen, Majharoddin M. Kazi, Hiren Joshi. Detecting Source Code Plagiarism in Student Assignment Submissions Using **Clustering Techniques**. *Journal of Techniques* [Internet]. 2024 Jun 1 [cited 2025 Apr 4];6(2). Available from: <https://research.ebsco.com/linkprocessor/plink?id=cc5a02a7-6061-3ef1-849e-472a140c63f4>
- Viuginov N, Grachev P, Filchenkov A. A Machine Learning Based Plagiarism Detection **in Source Code**. 2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence [Internet]. 2020 Dec 24 [cited 2025 Apr 4];1-6. Available from: <https://research.ebsco.com/linkprocessor/plink?id=e35db71f-fb12-35c7-bbf1-00e587be4554>
- Duracik M, Mikusova M, Callejas-Cuervo M. Optimized method based on the K-means **clustering algorithm** as a tool to detect source code plagiarism. *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao* [Internet]. 2020 May 1 [cited 2025 Apr 4];2020(E29):620–32. Available from: <https://research.ebsco.com/linkprocessor/plink?id=04dba5ad-6bb6-3db6-9d8c-e77f6c3a4c39>
- Awale, N., Pandey M., Dusal A., Timsina B. *Plagiarism Detection in Programming Assignments using Machine Learning*. *Journal of Artificial Intelligence and Capsule Networks* (2020) Vol.02/ No. 03 Pages: 177-184 https://web.archive.org/web/20220228023930id_ /<https://www.irojournals.com/aicn/V2/I3/05.pdf>
- Ganguly D, Jones GJF, Ramírez-de-la-Cruz A, Ramírez-de-la-Rosa G, Villatoro-Tello E. Retrieving and classifying instances of **source code plagiarism**. *Information Retrieval Journal* [Internet]. 2018 Feb 1 [cited 2025 Apr 4];21(1):1–23. Available from: <https://research.ebsco.com/linkprocessor/plink?id=da238657-0580-31ec-8282-453368d3b0c1>
- Martínez S, Wimmer M, Cabot J. Efficient plagiarism detection for **software modeling** assignments. *Computer Science Education* [Internet]. 2020 Jun 1 [cited 2025 Apr 4];30(2):187–215. Available from: <https://research.ebsco.com/linkprocessor/plink?id=d6d17a31-4b09-3f06-abbd-4546587a63ba>