Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «**Информатика, искусственный интеллект и системы управления**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии**»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Классификация известных методов определения заимствований в исходных кодах программ»

Студент группы **ИУ7-54Б** _____ **Чепиго Д.С.**

(Подпись, дата) (И.О. Фамилия)

Руководитель _____ **Майков К.А.**

(Подпись, дата) (И.О. Фамилия)

***2022** г.*

РЕФЕРАТ

Расчетно-пояснительная записка 17 с., 1 рис., 1 табл., 16 ист.

ЗАИМСТВОВАНИЯ, ПЛАГИАТ, ПРОГРАММНЫЙ КОД, ИСХОДНЫЙ КОД, АЛГОРИТМЫ ПОИСКА ПЛАГИАТА.

Объектом исследования являются алгоритмы определения заимствований в исходных кодах программ.

Цель работы – провести обзор существующих алгоритмов определения заимствований в исходных кодах программ.

В процессе исследования проанализированы предметные области применения алгоритмов определения заимствований в исходных кодах программ и изучены существующие алгоритмы.

В результате исследования предложены критерии сравнения алгоритмов определения заимствований в исходных кодах программ, разработаны критерии их сравнения и классифицированы 5 алгоритмических реализаций известных методов.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Введем понятие «плагиат» (от англ. *plagiarism*) – процесс или практика использования идей или работ другого человека и приписывания их себе [1]. В данной работе будем использовать слово «плагиат» как синоним слову «заимствование».

Введем понятие «контест» (от англ. *contest*) – соревнование, в котором каждый участник выступает без прямого контакта с конкурентами или вмешательства со стороны них [2]. Будем использовать словосочетание «контест по программированию», как синоним к «соревнование по программированию».

ВВЕДЕНИЕ

Заимствование материала является распространенной проблемой как в сфере образования, так и в сферах научной и культурной деятельности. Зачастую это может повлечь за собой нарушение авторско-правового и патентного законодательства [3]. Соответственно, возникает необходимость установления авторских прав на результаты интеллектуального труда.

Основная область применимости проверки на заимствования исходных кодов программ – сфера образования, а также соревнования по программированию. Это не только нарушает авторские права, но и отрицательно сказывается на качестве образования.

Цель работы – провести обзор существующих алгоритмических реализаций известных методов определения заимствований в исходных кодах программ Для достижения этой цели требуется решить следующие задачи:

— провести анализ предметной области и обзор существующих решений;
— установить критерии принадлежности программного кода к категории плагиата;
— сформулировать критерии сравнения методов;
— классифицировать существующие методы определения заимствований в исходных кодах программ.

1 Анализ предметной области

1.1 Области применения определения заимствований в исходных кодах программ

1.1.1 Сфера образования

Академический плагиат – очень распространенное явление, рассмотрим его на примере университета. Иногда для определения заимствований не нужны специальные алгоритмы и автоматизированные системы, например:

— исходный код проверяется преподавателем в присутствии студента и акт заимствования можно определить при разговоре;

— в образовательных целях студенту необходимо реализовать уже существующий алгоритм, например, конкретную сортировку массива целых чисел. На такое задание заведомо есть верный ответ и решения студентов не будут сильно отличаться;

— количество студентов на курсе мало, преподаватель знает кто потенциально мог использовать чужой код и реализовывает проверку вручную.

Не всегда преподаватели считают, что плагиат это плохо, ведь иногда важнее понимание работы определенного кода, а не его реализация.

В остальных случаях, если преподаватели заинтересованы в выявлении плагиата, перед ними стоит задача найти оптимальный способ его определения, например – автоматизация проверки исходных кодов.

1.1.2 Контесты по программированию

На сегодняшний день существует множество различных соревнований по программированию, начиная от соревнования по самому запутанному коду на языке Си [4] и заканчивая крупнейшей международной олимпиадой по программированию ICPC [5]. Также контесты иногда проводятся при отборе на работу в крупные компании.

Плагиат в таких случаях часто играет решающую роль, ведь только на его основе можно определить уровень участника.

1.1.3 Разработка программного обеспечения

Плагиат при разработке программного обеспечения встречается намного реже, так как существует юридический документ, определяющий его использование и распространение – лицензия на программное обеспечение. Такие лицензии делятся на два типа: несвободные (исходный код закрыт) и свободные (открытое программное обеспечение). Их различия сильно влияют на права конечного пользователя в отношении использования программы.

1.2 Критерии принадлежности кода к плагиату

Опираясь на [6] были выделены 4 основных типа заимствования исходного кода.

1) Программный код скопирован без каких-либо изменений (идентичен оригиналу с точностью до комментариев).
2) Код скопирован с «косметическими» заменами идентификаторов (имен функций и переменных, типов данных, строковых литералов).
3) Код может включать заимствования второго типа и модифицирован путем добавления, редактирование или удаления его фрагментов или бесполезных участков кода. Также возможны изменения порядка, не влияющие на логику самой программы.
4) Программа некоторым образом переписана с общим сохранением логики работы и функциональности, однако синтаксически она может абсолютно отличаться от оригинала.

Заимствования четвертого типа крайне затруднительны для выявления, зачастую это приводит к нахождению плагиата алгоритмов, а не исходного кода. Так, например, в приведенном выше примере со студентами и сортировкой массива целых чисел, плагиатом будут считаться только заимствования первого типа.

1.3 Критерии сравнения методов определения плагиата

На основе приведенного выше анализа предметной области были выделены следующие критерии сравнения методов определения плагиата.

1) Поддержка нескольких языков программирования.
2) Определение каждого из типов заимствования.
3) Точность на небольшом объеме исходного кода.
4) Низкоуровневое сравнение кода (ассемблерный листинг или байткод).

Первый критерий важен в ситуациях, когда работы, которые необходимо сравнить, написаны на разных языках программирования. Определение каждого из типов заимствований важно, так как на этой основе можно оценить уровень заимствования программного кода. Критерий «точность на небольшом объеме исходного кода» был выбран из-за специфики сферы образования и контестов по программированию (как правило размер кода в таких задач не является большим). Низкоуровневое сравнение кода было выделено, потому что в настоящее время это новое и очень перспективное направление в сфере определения заимствований.

2 Обзор существующих решений

2.1 Метод сравнения текста

Такие методы являются классическими для определения плагиата в обычном тексте, то есть не нацелены на программный код. Они заключаются в сравнении текстовых представлений программ на основе таких метрик, как:

— расстояние Жаккара [7];

— расстояние Джаро – Виклера [8];

— расстояние Левенштейна [9];

— Колмогоровская сложность [10].

В отличие от обычного текста, программный код нельзя маскировать с помощью замены кириллицы на латиницу, допущения орфографический ошибок или использовать изменение регистра (исключая строковые переменные).

Методы, основанные на сравнении текстов находят только заимствования первого типа. Такую проверку легко обойти, например, переименовав переменные и добавив свои комментарии. Такие методы не поддерживают несколько языков программирования.

Текстовые методы можно использовать для первоначальной оценки плагиата, так как они работают значительно быстрее, чем остальные [11]. Также, их можно применять при проверке небольших программных кодов, которые реализуют заведомо известный алгоритм. Потому что в таком случае единственный плагиат, который в нем можно найти – особенность структурирования кода и название переменных.

2.2 Метод сравнения токенов

Метод основан на преобразовании ключевых слов программы в токены – последовательности лексических единиц с определенным значением [12]. То-

кенизация осуществляется по следующему алгоритму.

1) Каждому оператору языка программирования или группе операторов, имеющих схожее назначение, присваивается уникальный идентификатор. Значения идентификаторов назначаются заранее и для всех классов операторов.
2) По полученным идентификаторам строится строка. В данной строке порядок токенов соответствует порядку следования их в исходном коде.
3) Полученные токены сравниваются любым доступным способом

Такой подход находит плагиат первых двух типов, также он может выявлять заимствованные фрагменты кода, которые расположены в различных местах программы, но делает это с неточностями. Поддержка разных языков программирования возможна только при использовании нескольких лексических анализаторов.

Методы, использующие токены, как и текстовые методы можно использовать для первоначальной проверки плагиата. Также если все программные коды, которые необходимо проверить на плагиат написаны на одном языке программирования и реализуют одинаковый функционал, то такой подход дает хорошую точность.

2.3 Метод сравнения метрик

Метод дополняет алгоритмы токенизации [13]. Помимо сравнения токенов он на их основе определяет метрики, например:

— количество условных конструкций;
— количество используемых циклов;
— количество глобальных переменных.

Также, зная специфику действий пользователей, можно выдвигать свои метрики, например – учитывать время обновления кода или количество пройденных с первого раза тестов, если они имеются.

На основе таких метрик высчитывается схожесть программных кодов. Такие методы достаточно легко обмануть, изменив структуру программы, напри-

мер добавив лишние функции или пустые циклы. Все остальные характеристики аналогичны методу, использующему токены.

2.4 Метод сравнения деревьев

Подход основан на представлении кода в виде абстрактного синтаксического дерева.

Абстрактное синтаксическое дерево (англ. *Abstract Syntax Tree*) – конечное помеченное ориентированное дерево, в котором внутренние вершины сопоставлены с операторами языка программирования, а листья – с соответствующими операндами, то есть переменными и константами.

На рисунке 1 изображено дерево для выражения while (x > 0) x = x - 1.

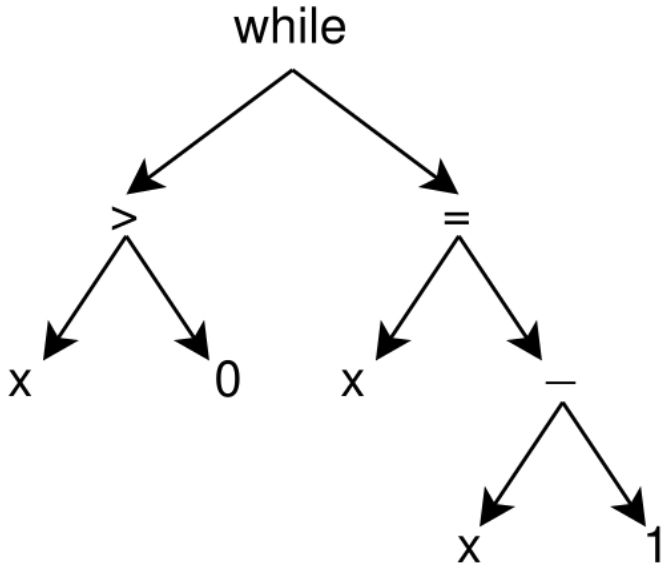

Рисунок 1 – Представление структуры исходного кода в виде дерева

В таком формате программные коды сравниваются любым доступным способом. Таким образом, методы, основанные на абстрактных синтаксических деревьях состоят из двух этапов: построение деревьев и их анализ. Из наиболее популярных алгоритмов сравнения – алгоритм Zhang-Shasha, который позволяет вычислять редакционное расстояние между деревьями [14], или сравнение деревьев с помощью строковых ядер [15].Таким образом можно выявить плагиат первых трех типов и осуществить поддержку нескольких языков програм-

мирования.

2.5 Метод низкоуровневого сравнения кода

Данный метод сравнивает код после этапа компиляции или интерпретации [16]. В зависимости от специфики языка программирования, такой подход может показывать разную точность. При этом заимствования первых двух типов наблюдается высокая точность, так как компилятор часто оптимизирует мелкие структурные изменения, например, замена for на while и наоборот в си-подобных языках программирования. Также такой метод дает хороший результат при модификациях третьего и четвертого типа, но его также можно легко понизить, например, поменяв типы данных (хранить числа в строковом представлении).

Такой подход можно успешно использовать, когда все программы компилируются одинаково и имеют заведомо оговоренные типы данных. В таком случае можно определить заимствования всех типов.

3 Классификация существующих методов определения заимствований в исходных кодах программ

На основе [6] была дана характеристика точности на небольшой объеме исходного кода. Все остальные характеристики оценивались на основе самых распространенных алгоритмических реализаций из каждой группы методов.

В таблице 1 приведена классификация ранее рассмотренных методов определения заимствований в исходных кодах программ. По горизонтали расположены критерии сравнения методов определения плагиата.

1) Поддержка нескольких языков программирования.
2) Определение типов заимствований.
   2.1) Изменение комментариев и добавление пустых строк.
   2.2) Изменение имен функций, переменных и типов данных.
   2.3) Изменение порядка выполнения кода.
   2.4) Выделение частей кода в функции.
3) Точность на небольшом объеме исходного кода. Находится как среднее из точности определения каждого из типа заимствований.
4) Низкоуровневое сравнение кода (ассемблерный листинг или байткод).

Таблица 1 – Классификация существующих методов определения заимствований в исходных кодах программ

	1	2.1	2.2	2.3	2.4	3	4
Текст	Нет	100%	80%	76%	65%	**80%**	Нет
Токены	Да	100%	92%	87%	74%	**88%**	Нет
Метрики	Да	100%	87%	95%	72%	**89%**	Нет
Деревья	Да	100%	93%	91%	60%	**84%**	Нет
Низкоуровневый код	Да	99%	100%	85%	80%	**91%**	Да

Выявлено, что алгоритмы сравнения текста дают высокую точность при определении плагиата $\approx 100\%$, когда исходный код был видоизменен только с помощью комментариев и пустых строк . Такой метод применим в сферах, когда исходный код может быть точной копией других кодов. Например, если один студент скопировал работу другого студента и поленился что-то менять. Также его можно использовать в качестве первичной проверки, так как он работает быстрее других алгоритмов.

Подход сравнения токенов показал высокую точность при изменении как комментариев, так и переменных или типов данных. Его также можно использовать в сфере образования, например, если все работы, которые необходимо проверить, написаны на одном языке программирования и реализуют одинаковый функционал. В таком случае средняя точность $\approx 88\%$.

Алгоритмы, использующие сравнение метрик аналогичны токенам. Но они дают более высокую точность $\approx 95\%$ (у токенов $\approx 87\%$) при изменении порядка выполнения кода. Такой подход применим, когда исходные коды разных людей решают одинаковую задачу и добавление пустых или лишних циклов недопустимо.

Сравнение синтаксических деревьев – единственный алгоритм, которые дает точность более $\approx 90\%$ на первых трех типах заимствований. Он хорошо поддерживает несколько языков программирования и в зависимости от выбранного алгоритма сравнения двух деревьев может применяться для разных задач. Хорошей областью применимости являются соревнования по программированию, когда участники решают одинаковые задачи, но на разных языках.

Низкоуровневое сравнение кода показало самую высокую среднюю точность $\approx 91\%$, это связано с тем, что этот алгоритм находит модификации четвертого типа. Такой подход применим, когда все программы одинаково компилируются и имеют заведомо оговоренных типы данных, например, в курсе программирования на Си с автоматическим тестированием.

ЗАКЛЮЧЕНИЕ

Во время выполнения работы была выполнена цель – проведен обзор существующих алгоритмических реализаций известных методов определения заимствований в исходных кодах программ

Были определены 4 критерия принадлежности кода к плагиату. На их основе были предложены критерии сравнения алгоритмов определения заимствований в исходных кодах программ.

Предложена классификация 5 алгоритмических реализаций известных методов:

- метод сравнения текста;
- метод сравнения токенов;
- метод сравнения метрик;
- метод сравнения деревьев;
- метод низкоуровневого сравнения кода.

Для каждой из алгоритмических реализаций известных методов были выявлены сферы применимости.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Cambridge dictionary [Электронный ресурс]. – Режим доступа: https://dictionary.cambridge.org/ свободный – (12.12.2022)

2. Merriam dictionary [Электронный ресурс]. – Режим доступа: https://www.merriam-webster.com/, свободный – (12.12.2022)

3. Бобкова О.В., Давыдов С.А., Ковалева И.А., Плагиат как гражданское правонарушение // Патенты и лицензии. 2016. № 7. С. 31

4. The International Obfuscated C Code Contest [Электронный ресурс]. – Режим доступа: https://www.ioccc.org/, свободный – (12.12.2022)

5. Международная студенческая олимпиада по программированию [Электронный ресурс]. – Режим доступа: https://icpc.global/, свободный – (12.12.2022)

6. A Survey on Software Clone Detection Research, Chanchal Kumar Roy and James R. Cordy, 2007

7. Comparing sets of patterns with the Jaccard index, Sam Fletcher, Md Zahidul Islam, 2018

8. A Comparison of String Metrics for Matching Names and Records, William W. Cohen, 2013

9. A Normalized Levenshtein Distance Metric, Li Yujian, Liu Bo, 2007

10. Колмогоровская сложность и алгоритмическая случайность, Верещагин Н. К., Шень В. А., 2013

11. Plagiarism: Taxonomy, Tools and Detection Techniques, Hussain A Chowdhury and Dhruba K Bhattacharyya, Dept. of CSE, Tezpur University, 2012

12. Применение метода токенизации для поиска схожих фрагментов кода в студенческих работах, Шиков Алексей Николаевич, Сорокин Дмитрий Сергеевич, 2014

13. Demonstration of Text Similarity Metric for Plagiarism Detection, Ohnmar Htun, Yoshiki Mikami, 2014

14. Zhang K., Shasha D. Simple fast algorithms for the editing distance between trees and related, problems // SIAM Journal of Computing, 1989. — No. 18. — Pp. 1245–1262.

15. Comparison of Abstract Syntax Trees using String Kernels, Raul Torres, Thomas Ludwig, Julian M. Kunkel, 2018

16. Semantics-Based Obfuscation-Resilient Binary Code Similarity Comparison with Applications to Software Plagiarism Detection, Lannan Luo, Jiang Ming, Dinghao Wu, Peng Liu, Sencun Zhu, The Pennsylvania State University