Анализ методов проверки кода программ на плагиат в цифровой образовательной платформе Мирера

М.С. Дьяченко[1], В.А. Домрина[2], А.Г. Леонов[3], К.А. Мащенко[4], И.Г. Райко[5], А.А. Холькина[6]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, Mdyachenko@niisi.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, domrina@niisi.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, МПГУ, Москва, Россия, Государственный университет управления, Москва, Россия, dr.l@vip.niisi.ru;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, kirill010399@vip.niisi.ru;

[5]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, ilya.rayko@niisi.ru;

[6]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, kholkina@niisi.ru

Аннотация. Возможность копировать части кода программ участниками образовательных учебных курсов по программированию затрудняет объективное оценивание уровня знаний каждого обучающегося. В статье излагается вариант решения этой проблемы на примере авторской цифровой образовательной платформы Мирера. Поставлены требования к антиплагиат-анализу с учетом специфики цифровой образовательной платформы Мирера. Рассмотрены известные алгоритмы и методы для реализации антиплагиат-анализа, выделены их достоинства и недостатки. Реализован метод отпечатков на токенизированной программе, описаны примеры работы антиплагиат-анализатора. Отмечается возможность использования антиплагиат-анализа как инструмента для контролирования уровня освоения курса студентами.

Ключевые слова: цифровая образовательная платформа, цифровая образовательная платформа Мирера, антиплагиат, антиплагиат-анализ программного кода, метод отпечатков.

1. Введение

С развитием цифровых технологий образовательных процессов растет число попыток заимствования (плагиата) [8], в том числе в курсах по программированию при составлении студенских программ. При этом учащиеся копируют не столько алгоритмы решения, сколько фрагменты кода программы или даже код целиком, поэтому проблема обнаружения плагиата актуальна практически для любого курса, который сдержит задачи по написанию программ для школ и университетов [1,2,5]. Выделяют несколько уровней списывания [10, 15], от полного копирования до изменения текста программы до неузнаваемости, при этом даже примитивные методы изменения кода, такие как перестановка компонент программы, добавление пробелов и других разделителей, а также замена всех имен переменных и функций способны «обмануть» преподавателя, проверяющего решения на плагиат вручную. Стоит также отметить, что решения, находящиеся в свободном доступе [7, 11, 12], подразумевают загрузку в систему или проверку с компьютера преподавателя, но проверка на заимствование необходима и для платформ с автоматическими проверками решений, так, Codeforces и Яндекс Контест используют проприетарный антиплагиат-алгоритм [16]. В связи с этим возникает необходимость автоматической быстрой проверки всех отправляемых решений, прошедших проверку, на схожесть, которая могла бы предоставить подробный отчет при обнаружении совпадений, включающий данные списывающего, возможного автора оригинального решения и вероятность плагиата, и работала для решения, отправленного в любое время.

Такие аспекты, как необходимость учитывать стиль кода студента, разные объемы решений, язык программирования, используемый в решении и оперативное получение обратной связи вне зависимости от нагрузки на сервер, хранящий решения, формируют особые требования к программе антиплагиат-анализа. Также важной задачей является выявление и акцентирование внимания преподавателя на случаи, когда код совпадает полностью с точностью до изменений, не влияющих на алгоритм программы. В таком случае вероятность плагиата приближается к 100%, то есть списывание с очень высокой вероятностью произошло). Для уменьшения технически и психологической нагрузки преподавателя, важно также минимизировать число ситуаций, когда антиплагиат-анализатор отмечает как

списанные задания, которые на самом деле были выполнены самостоятельно. Все вышеперечисленное усложняет задачу антиплагиат-анализа и требует рассмотрения различных подходов к реализации, позволяющих добиться достоверных результатов в широком круге заданий. Некоторые из методов, с точки зрения авторов представляющие наибольший интерес, разобраны в настоящей статье вместе с анализом их достоинств и недостатков.

Следует отметить, что цифровая образовательная платформа (ЦОП) Мирера обладает некоторой внутренней спецификой, которая может как усложнить реализацию совместимых с Мирерой алгоритмов антиплагиат-анализа, проверки на заимствование, так и позволить преподавателям и составителям курсов получить более подробную и точную статистику как о студентах, замеченных в списывании решений, и примененных ими способами скрыть факт списывания, так и о стилях оформления программ студентами, работающими без заимствований. Так, хранящаяся в системе информация о количестве попыток решения задачи и времени отправки решения может послужить решающим аргументом в подтверждении гипотезы копирования или его отсутствия, а возможность анализа каждого решения предоставляет большие объемы информации о каждом студенте индивидуально. При этом метод анализа на плагиат, совместимый с платформой, должен учитывать а) наличие шаблонов – частей кода программы, предоставленных преподавателем, которые очевидно совпадают у всех студентов, и б) возможность многофайлового представления программы и наличие большого количества вариантов такого представления. Метод отпечатков на токенизированной программе, учитывающий все особенности ЦОП Мирера, встроен в платформу. В статье рассматриваются результаты реализации этого метода при различных видах копирования и списывания, а также описаны планируемые улучшения.

2. Постановка задач исследования и анализ алгоритмов

Предлагается рассмотреть требования к антиплагиату, возможные подходы к реализации и степень удовлетворения этим требованиям. Цель исследования – выбрать алгоритм для внедрения в ЦОП Мирера по результатам этой оценки.

2.1. Формулировка требований

Для формулировки требований необходимо понимать, как списывающий ученик меняет оригинальную программу и сколько времени занимают эти изменения. По объему проделанной работы списывание можно разделить на типы [9, 13, 15]. Например, студент может, получив решение, отправить его без изменений или за несколько минут поменять названия переменных, функций, заменить строчные буквы на прописные и наоборот, убрать или добавить комментарии, пробелы и другие служебные символы – добавить «шум» [3], что точно не помешает программе пройти проверку. Оба случая антиплагиат должен определять с точностью 100%, причем первый случай должен выделять особо, поскольку в большинстве случаев полное совпадение свидетельствует о списывании. Больше времени занимает перестановка местами функций, строк, циклов, объявлений переменных и других элементов программы, а также изменение условий в циклах таких, как замена while на for, $i > 9$ на $i >= 10$ в случае целого i, и разбиение одной функции на несколько и наоборот, поскольку необдуманные изменения такого типа могут нарушить логику воплощенного алгоритма. Несмотря на то, что подобного рода совпадения возникают даже в самостоятельно выполненных работах, ожидается, что антиплагиат с большой точностью определит процент совпадения. Важно учитывать то, что добавление не относящегося к алгоритму кода, например, объявление не используемых переменных или циклов, не влияющих на задачу, не должно уменьшить возвращаемую вероятность, то есть необходимо анализировать вхождение сравниваемых файлов друг в друга.

Одной из ключевых особенностей ЦОП Мирера являются механизмы поддержки шаблонов – кодов и их фрагментов, написанных преподавателем. Преподаватель может задать неудаляемость и неизменяемость элементов шаблона и Мирера обеспечит выполнение этого требования. Таким образом, элементы шаблона студент не сможет убрать из программы или изменить и от антиплагиата-анализатора требуется учитывать наличие таких шаблонов в расчете процента совпадения и определении порога совпадения – минимального процента совпадения программ, при котором считается, что списывание произошло. Порог совпадения также должен зависеть от размера программы и, при желании, регулироваться преподавателем. Также антиплагиат-анализатор должен обрабатывать решения задачи на разных языках программирования, размещенные в нескольких файлах с различными расширениями, оперативно предоставлять обратную связь, безошибочно или с высокой вероятностью определять автора списанного кода и оповещать преподавателя о подозрительных решениях, показывая, какие части программ схожи.

Как было сказано выше, важное требование к антиплагиату – работать на доступных данных и

предоставлять результаты работы в виде, поддерживаемом ЦОП Мирера, однако, не все элементы ограничивают алгоритм антиплагиат-анализа. Так, режим строгого контроля, запрещающий студенту копировать и вставлять код, замедлит возможное списывание и позволит преподавателю при необходимости отслеживать процесс решения по черновикам студентов, возможность фиксировать копирование позволит выводить предупреждение «было скопировано … строчек в файл …», свидетельствующее о почти гарантированном списывании при высоком проценте совпадения, а возможность зафиксировать время отправки решения в систему позволяет не только определить автора списанного решения, но и с большей вероятностью определить, списана работа или нет. Например, многие копирования с небольшими изменениями (перестановка строк, замена имен переменных или отсутствие изменений) происходят под конец контеста – набора задач для занятий (семинаров, домашней или контрольной работы), и, если программа не прошла проверку на антиплагиат и сильно отличается от других попыток студента, скорее всего, он списал, предположив, что не успеет поправить свое решение. Однако, если разница между отправкой двух решений небольшая, в зависимости от задачи можно заключить, что, несмотря на процент выше порога, списывания не было, ведь студент бы физически не успел столько поправить.

2.2. Общие шаги алгоритма антиплагиат-анализа при внедрении в ЦОП Мирера

Перед тем, как оценить алгоритмы и их степень соответствия вышесказанным требованиям, рассмотрим общие детали при реализации и критерии, которые могут автоматически выполняться благодаря такому подходу. Каждый алгоритм будет получать на вход полный текст программы и преобразовывать каждую строку к стандартному виду: убирать все комментарии и пробелы, заменять все прописные буквы на строчные, заменять имена переменных на V и имена функций на F (последнее слегка видоизменено для алгоритмов, привязанным к переменным (или функциям) и синтаксису вокруг них). Степень совпадения, и, соответственно, вероятность списывания, для двух программ будем рассчитывать следующим образом: из двух вероятностей, каждая из которых равна количеству совпадающих объектов (для каждого алгоритма эти объекты свои), деленному на количество всех объектов первой и второй программы соответственно и умноженному на 100%, выбираем наибольшее значение. Таким образом, обеспечена независимость от «шума» и добавления не играющих роли в алгоритме элементов, например, при добавлении лишней строки с определением переменной или цикла и переименовании всех переменных алгоритм получит вероятность 100%. О работе в реальном времени, определении автора и информации для преподавателей будет сказано позже, в разделе про реализацию метода отпечатков на токенизированной программе.

Все рассматриваемые алгоритмы удовлетворяют самому первому критерию – безошибочно определять полностью совпадающие, в том числе с точностью до «шума», программы. Следовательно, остаются следующие требования: поддержка большого количества языков, возможность работы с несколькими файлами, поддержка функциональности шаблонов, независимость от изменения порядка функций, строк, циклов и т.д. в программе, возможность передачи результатов таким образом, что по ним легко предоставить информацию преподавателю, о которой было сказано выше. Также от алгоритма требуется не иметь false positive ответов – самостоятельно выполненных решений, которые антиплагиат отметил заимствованными.

2.3. Алгоритмы и их степень соответствия поставленным требованиям

Приведем обзор алгоритмов, используемых в настоящем исследовании.

1. Побайтовое сравнение. Алгоритм собирает строки стандартного вида в одну и кодирует, сравнивая каждый байт первой программы с каждым байтом второй программы по порядку. Этот метод подходит для нахождения идентичных с точностью до «шума» программ, но чувствителен к перестановке элементов программы. Его модификации, например, сравнивание только определенных байтов, не представляют интерес, поскольку сравниваются слишком малые части программы, что приводит к неэффективности по времени работы или завышению процента совпадения (в каждом решении одной задачи с большой вероятностью есть одинаковые небольшие конструкции, однако о заимствовании это не свидетельствует). Таким образом, этот метод прост в реализации, но не удовлетворяет важным критериям.

2. Метод отпечатков для n-грамм и токенов, алгоритм просеивания.

Разобьем программу на n-граммы (последовательность n идущих подряд символов) или токены (минимальная текстовая единица), рассмотрим последовательность их хешей. Определим окно размера w как последовательность из w хешей, и будем рассматривать окна размеров t-k+1, где t - размер для гарантированного подтверждения совпадения, а n-граммы при $n < k$ не рассматриваются. t и k – параметры, которые

подбираются на соответствующем наборе данных, от их выбора зависит степень чувствительности к перестановке элементов программы и вероятность пропустить небольшие совпадения. Количество отпечатков [2],[3] также играет ключевую роль в алгоритме, будем выбирать хотя бы один отпечаток в каждом окне, чтобы ограничить максимальное расстояние между отпечатками. Алгоритм состоит в том, чтобы выбирать как отпечаток в каждом окне элемент с минимальным хешем, поскольку эти элементы с большой вероятностью одинаковы в соседних окнах. [3, 14]

Следует обратить внимание на то, что метод отпечатков не зависит от синтаксической логики программы, поэтому может использоваться в программах с шаблонами любого размера и вида и несколькими файлами, которые можно объединить в один. Алгоритм также независим от перестановки элементов, хорошо работает с заменами условий циклов и разбиением части программы на несколько [3,5]. Несмотря на наличие минусов, о которых будет сказано в следующем разделе, грамотная реализация способна использовать их для более качественной проверки в некоторых случаях или снизить негативное влияние до минимума.

Вышеописанный метод для n-грамм лежит в основе системы MOSS (Measure Of Software Similarity) [5,6], разработанной в 1997 году, которая используется многими американскими университетами как основное средство проверки, в том числе благодаря достаточно быстрому и качественному результату и отсутствию отмеченных как плагиат честно выполненных работ.

3. Метод отпечатков на токенизированной программе на основе переменных.

Метод похож на предыдущий, однако учитывает соседние к токену переменной токены и информация о них и становится отпечатком [4]. Поскольку метод опирается на синтаксическую составляющую программы, он обязан оценивать программу только с шаблоном и часто не может оценить один шаблон для справедливой оценки совпадения кода. Более того, исследования показывают меньшую эффективность, чем похожие методы [4].

4. Метод отпечатков на синтаксическом дереве на основе переменных.

Использование синтаксического дерева приводит к более сложным в реализации, но одним из самых точных методов для антиплагиата. Например, существует подход, аналогичный методу на основе переменных на токенизированной программе: на основе кода создается синтаксическое дерево, в котором выделяются вершины переменных, и рассматриваются вершины выше вершины данной переменной, содержащие больше одного потомка и с учетом информации о них создаются отпечатки [4]. Таким образом, достигается полная независимость от способа объявления и использования переменной. Метод способен предоставить более подробную и наглядную информацию о логике и синтаксической составляющей программы, однако для оценивания необходим полный код программы, и получить информацию о совпадении частей, не входящих в шаблон, не получится. Таким образом, этот метод, несомненно, имеет огромные плюсы, но несовместим с ключевыми элементами ЦОП Мирера.

Ввиду вышеперечисленных плюсов, успешном использовании в одном из самых известных антиплагиатов, а также наличии в общем доступе библиотек Python по токенизации [6] и коду основы алгоритма в общем доступе [3] было принято решение ввести в ЦОП Мирера метод отпечатков на токенизированной программе.

3. Реализация метода отпечатков на токенизированной программе

Рассмотрим детали реализации антиплагиата, его поведение в различных типах ситуаций, механизм работы в реальном времени и результаты подбора порога совпадения и параметров по итогам тестирования антиплагиата на датасете из массива решений студентов ЦОП Мирера.

3.1. Реализация алгоритма

За основу была взята библиотека copydetect [7], позволяющая сравнить несколько файлов с заданными параметрами методом, аналогичным выбранному. Данная библиотека была модифицирована для лучшей совместимости с реализованными алгоритмами работы с решениями в ЦОП Мирера.

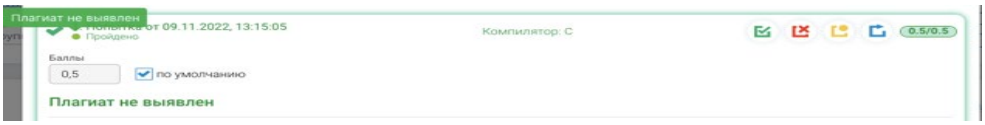

Рис. 1. Плагиат не найден

Рис. 2. Результаты проверки на антиплагиат в случае обнаружения заимствования

Так, чтобы обеспечить работу в реальном времени и обнаружить автора оригинального кода, антиплагиату передается id ученика, задачи, попытки и время отправки правильного решения вместе с кодом, при этом каждое новое решение сравнивается с уже зафиксированными для данной задачи; если список решений с процентом совпадения больше порога пуст, заключается, что заимствования не было (рис. 1), иначе автором считается ученик с самым большим процентом совпадения из этого списка (рис. 2), который по окончанию проверки преподаватель увидит в статистике с процентами схожести, временем отправления, по которому сортируются строки, и возможностью просмотреть код обеих сравниваемых программ с подчеркнутыми совпадающими строками – если в строке есть совпавшие символы, она подсвечивается желтым, если совпадает полностью с точностью до «шума» – красным (рис. 3). В статистике курса в зависимости от вероятности, возвращенной антиплагиатом, каждой задаче присваивается кружок соответствующего цвета – зеленого, если плагиат не обнаружен, желтого, если вероятность от 65% до 80%, и красного, если вероятность до 100% (рис. 4).

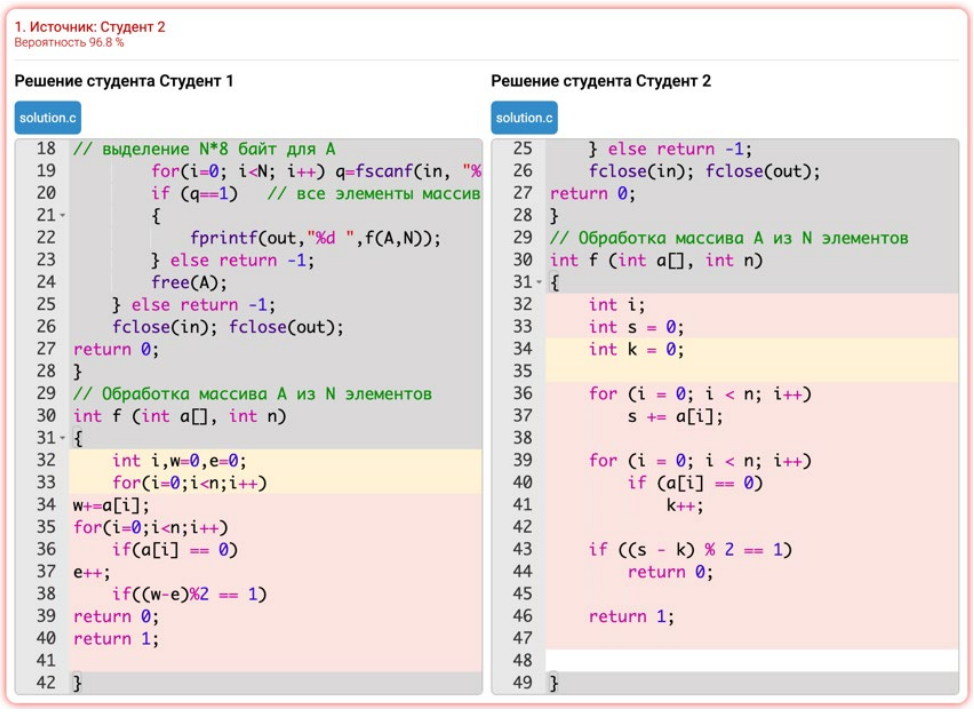

Рис. 3. Подсветка совпадающих элементов решения

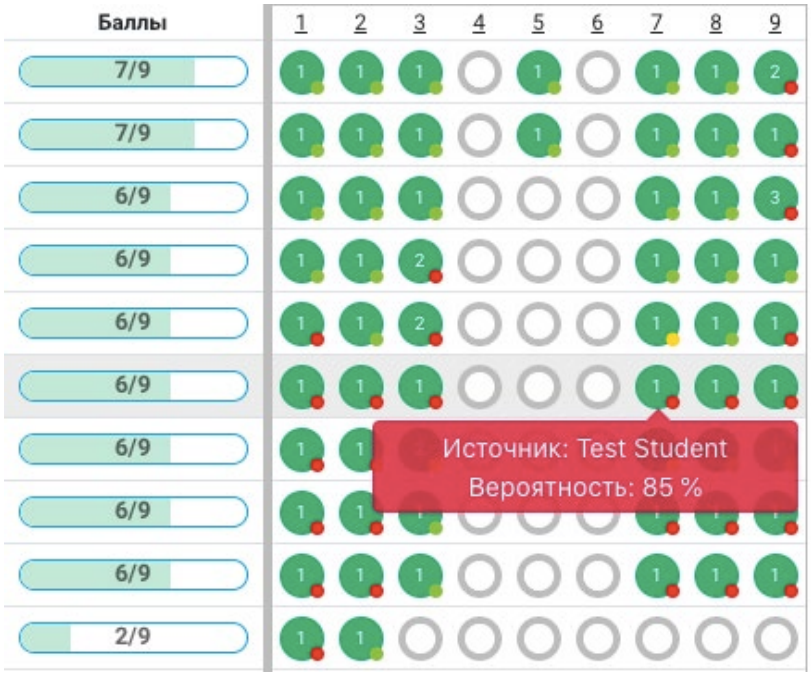

Рис. 4. Статистика решений

Особенности базы данных ЦОП Мирера позволяют сравнивать решения с решениями групп прошлых лет, у которых также было это задание, что убирает возможность незаметно получить у старшекурсников готовый код.

3.2. Подробности использования и применение антиплагиата в работе ЦОП Мирера

Поскольку метод достаточно чувствителен к сильным изменениям стиля кода, таким, как определение переменных одним оператором или разными, и использованию нескольких функций вместо одной, который обычно не меняется в попытках одного ученика, с помощью проверки всех решений студента с учетом времени между попытками при подозрении на списывание антиплагиат позволяет отследить, не было ли в какой-то момент копирования у ученика с отличающимся стилем кода или алгоритмом. Также отсутствие анализа логики кода позволяет проверять любые попытки, даже синтаксически неправильные или вызывающие ошибки компиляции, без противоречий и влияния на результат, а невозможность предоставить преподавателю подробный отчет о логике копирования и работы программы компенсируется простотой реализации и временем на получение результата. Так, даже недостатки метода могут позволить преподавателю лучше понять ситуацию в определенных случаях.

Таким образом, чаще всего антиплагиат применяется для сравнения всех правильных попыток в процессе отправки и в конце контеста для получения полной картины выполнения заданий, а при подозрении на списывание – для более тщательной проверки, в том числе не обязательно правильных попыток, и проверки попыток одного студента на сохранение стиля кода.

3.3. Проверка антиплагиата на реальных данных

Из задач определенных типов, размеров решений и шаблонов было выделено вручную несколько оригинальных и списанных решений разных типов списывания. Результаты проверки антиплагиатом с лучшим подбором параметров можно видеть в таблице ниже, где показан средний процент, было выбрано 20 задач с плагиатами следующих типов:

1. Полностью скопированное решение
2. Добавление шума (пробелы, комментарии, служебные символы)

3. Изменение названий переменных, функций и т.д.

4. Изменение условий в if с использованием отрицания и т.д.

5. Изменение порядка инициализаций переменных и функций, строк

6. Изменение циклов while и for друг на друга

7. Вынесение и внесение в функции строк, инициализаций переменных и т.д.

Таблица 1. Результаты проверки антиплагиатом с лучшим подбором параметров

	1	2	3	4	5	6	7
1	-	100%	100%	96%	100%	91%	84%
2	-	-	100%	96%	100%	91%	84%
3	-	-	-	96%	100%	89%	84%
4	-	-	-	-	96%	88%	81%
5	-	-	-	-	-	91%	82%
6	-	-	-	-	-	-	76%
7	-	-	-	-	-	-	-

По результатам видно, что типы 1-3 и 5 обнаружены с точностью 100%, 4, 6 и 7 с высокой точностью. В будущем планируется тестирование на более объемных выборках.

4. Заключение

Антиплагиат-анализ позволяет преподавателю не только оценить сходство двух отправленных решений быстрее, удобнее и точнее, чем ручная проверка, но и лучше оценить нарушения типичного стиля кода для списавшего ученика, определить момент, в который произошло списывание, указать предполагаемых авторов оригинальных решений и закономерности, по которым может происходить копирование, узнать, какие задачи вызывают наибольшую сложность и какие конструкции каких размеров чаще всего списывают студенты. Вся эта информация, несомненно, поможет преподавателю не только пресекать попытки списывания и объективно оценивать знания своих учеников, но и улучшить методику преподавания, понимая, какие задания относятся к наиболее сложным и какой материал вызывает трудности в усвоении, что поможет скорректировать курс и повысить объективность результатов проверки знаний.

В дальнейшем планируется активно апробировать антиплагиат-анализатор системы Мирера на большей выборке, включающей в себя курсы прошлых лет, поработать над наглядностью вывода результатов преподавателю, в том числе введением предупреждений вида «до конца контеста менее n минут от времени отправки решения, было скопировано более m строк, разница во времени отправки попытки менее l минут», где n, m, l могут изменяться преподавателем и отображения в статистике контестов и курсов, а также внедрить в ЦОП Мирера антиплагиат-алгоритмы, работающие с тестами и заданиями со свободным вводом, и модифицированный метод, использующий синтаксическое дерево, с учетом шаблонов.

Anti-Plagiarism Tool for Code in the Digital Educational Platform of Mirera

M.S. Diachenko, V.A.Domrina, A.G. Leonov, K.A. Mashchenko, Ilya G. Raiko, A.A. Kholkina

Abstract. The possible copying of parts of code in the solution by participants in educational programming courses makes it difficult to objectively estimate the level of knowledge of each student. The article describes a solution to this problem using the example of the author's digital educational platform Mirera. The requirements for anti-plagia-

rism tool which take the specifics of the digital educational platform Mirera into account are set. Algorithms and methods for anti-plagiarism are considered, their advantages and disadvantages are highlighted. The details of the implementation of the fingerprinting method on a tokenized program and examples of the work of anti-plagiarism tool are described. The possibility of using the tool as an instrument for better understanding the students and the degree of mastering the course is noted.

Keywords: digital educational platform, digital educational environment, Mirera, anti-plagiarism tool, tokenization, fingerprinting.

Литература

1. И. А. Посов, В. Е. Допира. Методы поиска плагиата в кодах программ. Известия СПбГЭТУ «ЛЭТИ» 2019; (6): 61-66 https://izv.etu.ru/assets/files/izvestiya-6_2019_p061-066.pdf

2. Анализ алгоритмов выявления плагиата в кодах программ, написанных на языках высокого уровня [Электронный ресурс] / Д. А. Чепрасов; Национальный исследовательский Томский политехнический университет (ТПУ) ; науч. рук. Ю. Я. Кацман // Современные техника и технологии сборник трудов XVIII международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 9-13 апреля 2012 г: в 3 т.: / Национальный исследовательский Томский политехнический университет (ТПУ) . — 2012 . — Т. 2 . — [С. 431-432]
https://www.lib.tpu.ru/fulltext/v/Conferences/2012/C2/V2/v2_212.pdf

3. A. Aiken, S. Schleimer, D. Wikerson. Winnowing: local algorithms for document fingerprinting. In Proceedings of ACMSIGMOD Int. Conference on Management of Data, San Diego,CA, June 9–12, pp. 76–85.ACMPress, New York, USA, 2003. https://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf

4. Nick Moone. Comparing variable fingerprints to indicate plagiarism in code, June 18, 2021 https://scripties.uba.uva.nl/download?fid=682144

5. Kevin W. Bowyer and Lawrence O. Hall. Experience Using ”MOSS” to Detect Cheating On Programming Assignments. https://www3.nd.edu/~kwb/nsf-ufe/1110.pdf

Сайт библиотеки «Pygments» [Электронный ресурс]. https://pygments.org/ (дата обращения: 01.09.2022)

Библиотека «copydetect» [Электронный ресурс]. https://github.com/blingenf/copydetect (дата обращения: 01.09.2022)

6. Никитов А.В., Орчаков О.А., Чехович Ю.В. Плагиат в работах студентов и аспирантов: проблема и методы противодействия. Университетское управление: практика и анализ. 2012;(5):61-68.
https://www.umj.ru/jour/article/download/506/507

7. Perkins, Mike & Basar Gezgin, Ulas & Gordon, Raymond. (2019). Plagiarism in higher education: classification, causes and controls. Pan-Pacific Management Science, Vol. 2, 3-21 https://www.researchgate.net/publication/354143709_Plagiarism_in_higher_education_classification_causes_and_controls

8. Sraka, Dejan and Branko Kaucic. “Source code plagiarism.” Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces (2009): 461-466. https://www.semanticscholar.org/paper/Source-code-plagiarism-Sraka-Kaucic/ac10405ce5a9421b2e2dd33836350f259614931a

9. Сайт сервиса «MOSS» [Электронный ресурс]. https://theory.stanford.edu/~aiken/moss/ (дата обращения: 01.09.2022)

10. Сайт сервиса «Copyleaks» [Электронный ресурс]. https://copyleaks.com/ru/проверка-кода-на-плагиат/ (дата обращения: 01.09.2022)

11. T. Vrbanec, A. Meštrović: Taxonomy of academic plagiarism methods, Zbornik Veleučilišta u Rijeci, Vol. 9 (2021), No. 1, pp. 283-300

12. Яшин Гліб Євгенович, Хмелюк Марина Сергіївна Метод просеивания для выявления плагиата в программном коде на языке C# // Перший Незалежний Науковий Вісник. 2015. №1-1. URL: https://cyberleninka.ru/article/n/metod-proseivaniya-dlya-vyyavleniya-plagiata-v-programmnom-kode-na-yazyke-c (дата обращения: 29.07.2022).

13. Смирнова Ю.В. МЕТОДЫ ПРОВЕРКИ УНИКАЛЬНОСТИ ПРОГРАММНОГО КОДА // Вестник магистратуры. 2019. №4-2 (91). https://cyberleninka.ru/article/n/metody-proverki-unikalnosti-programmnogo-koda

14. Статья о работе антиплагиата в Яндекс Контест [Электронный ресурс]. https://yandex.ru/ /support/lyceum-teachers/common-concepts/check-out-solutions.html#plagiarism.