УДК 004

# Система локального антиплагиата в контексте обучения программированию

*Фатеенков Данила Витальевич*
*Приамурский государственный университет имени Шолом-Алейхема*
*Студент*

**Аннотация**
В данной статье рассматривается применение локального антиплагиата в контексте обучения программированию на примере информационной системы, созданной для организации процесса обучения программированию. В статье рассмотрены популярные алгоритмы поиска совпадений в тексте, описаны их достоинства и недостатки, а также рассмотрен пример внедрения системы локального антиплагиата в систему обучения программирования.
**Ключевые слова:** локальный антиплагиат, поиск совпадений в тексте, программирование, обучение программированию, алгоритмы поиска

# Local anti-plagiarism system in the context of programming education

*Fateenkov Danila Vitalievich*
*Sholom-Aleichem Priamursky State University*
*Student*

**Abstract**
This article considers the application of local anti-plagiarism in the context of programming education on the example of an information system created to organise the process of programming education. The article considers popular algorithms of searching for matches in the text, describes their advantages and disadvantages, and also considers an example of implementation of the local anti-plagiarism system in the programming training system.
**Keywords:** local anti-plagiarism, search for matches in text, programming, learning programming, search algorithms

## 1. Введение

### 1.1 Актуальность

Автоматизация процесса обучения актуальным IT направлениям с каждым годом становится всё более актуальной задачей в разработке обучающих информационных систем. Стандартные методы организации методики обучения программированию в настоящее время представляют собой комплексный и длительный процесс, который можно автоматизировать во многих аспектах. Автоматизации можно подвергнуть, например, процесс публикации отчётов о выполнении работы учеников, а также процесс распределения отчётов по соответствующим каталогам с

целью эффективного поиска необходимой информации для работы со стороны преподавателя.

Одним из самых распространённых методов обучения, помимо изучения теоретического материала, является закрепление знаний путём решения задач практической направленности. Такой подход к организации обучения программированию актуален во многих IT сферах: начиная с изучения основ алгоритмизации и продолжая изучением прикладных навыков (например, реализация сложных структур данных, либо создание полноценных приложений и информационных систем). Такой подход к обучению эффективен, но представляет собой сложный и зачастую длительный процесс. Проблема заключается в необходимости проверки созданных проектов и программ учениками: данный процесс занимает много времени, особенно если заданий и обучающихся в группе много. Например, решений к заданиям может быть до 100 штук, если группа состоит из 10 учеников (при условии, что все из них посещают занятия и вовремя выполняют выданные им задания) и в курсе также 10 заданий.

К описанной проблеме также добавляется тот факт, что ученики могут списывать решения, что усложняет объективную проверку и поиск уникальных ответов со стороны обучающихся. Для решения поставленной проблемы стоит использовать системы локального антиплагиата.

### 1.2 Обзор исследований

Использование систем антиплагиата представляет собой стандартную практику при проверке различных работ в высших образовательных учреждениях (например, проверка курсовых или дипломных работ) не только в России, но и в других странах. Т. В. Хованская и М. Н. Сандирова опубликовали научную работу, в которой рассмотрели процесс проверки работ с помощью антиплагиата и описали основные виды заимствований, модули систем локального антиплагиата и возможности таких систем [1].

К. Ф. Сафин и Ю. В. Чехович предложили комбинированный подход к обнаружению текстовых заимствований, основанный на использовании внутренних методов для выявления высоко оригинальных документов [2].

Е. С. Кузьминых, С. П. Ильина и М. А. Маслова в своей научной работе рассмотрели проблему заимствования кода в сфере программирования [3]. В статье описаны способы анализа кода и проведено исследование эффективных методов и инструментов для поиска заимствований в коде.

О. Н. Половикова и В. Е. Иванова описали разработку автоматического детектора проверки на плагиат программного кода для проверки студенческих работ [4]. Для первичной оценки схожести двух токенизированных представлений используется метрика Левенштейна, которая легла в основу локального антиплагиата в рассматриваемой в рамках данной статьи информационной системе обучения программированию.

Е. А. Часов, М. А. Марина, А. В. Киптенко и Д. А. Слепнев провели анализ методов поиска заимствований в исходном коде программ [5]. Были

выделены достоинства, недостатки рассмотренных методов, а также описан поиск оптимального подхода к решению проблемы заимствований.

П. А. Яковлев представил метод эффективного сравнения символьной последовательности со всеми строками из некоторого множества, работающий существенно быстрее, чем наивный перебор сравнений со всеми строками подряд [6]. Алгоритм основан на поиске редакционного расстояния и построении префиксного дерева строк.

### 1.3 Цель исследования

Цель – рассмотреть существующие алгоритмы нахождения совпадений в текстах и внедрить систему локального антиплагиата в информационную систему, предназначенную для организации обучения программированию.

## 2 Материалы и методы

Для реализации системы локального антиплагиата используется PHP, а также происходит обращение к таблицам реляционной СУБД MySQL для получения информации (в таблицах не хранятся сами текста, которые используются в алгоритме) для работы локального антиплагиата.

## 3 Результаты и обсуждения

Для решения проблемы скопированных решений, в ноябре 2023 года было внедрено обновление, основным нововведением которого стал локальный антиплагиат, направленный на минимизацию случаев копирования кода между студентами.

На этапе проектирования системы локального антиплагиата были проанализированы следующие алгоритмы поиска заимствований:

1. Алгоритм шинглов – алгоритм, предполагающий разбиение текста на отдельные шинглы. Предварительно текст проходит этап канонизации (очистка текста от знаков препинаний, союзов, предлогов, HTML-тегов и других составляющих, которые могут повлиять на объективность результатов проверки текстов на заимствование). Принцип алгоритма шинглов заключается в сравнении случайной выборки контрольных сумм шинглов двух текстов между собой.


Рисунок 1. Пример разбиения текста на шинглы

2. Алгоритма поиска подстрок в строках – к таким алгоритмам относятся алгоритм Кнута-Морриса-Пратта, алгоритм нахождения подстрок

на основе суффиксных массивов, Алгоритм Рабина-Карпа. Этот класс алгоритмов один из самых распространённых и имеет реализации во многих популярных в настоящее время языках программирования.

3. Алгоритм нахождения редакционного расстояния – метрика похожести двух строковых последовательностей. Чем больше расстояние, тем более различны строки. Для двух одинаковых последовательностей расстояние равно нулю. Особенно эффективен алгоритм в ИИ и компьютерной лингвистике, но также часто применяется для сравнения строк.

Проведя анализ существующих алгоритмов, выбор был сделан в пользу алгоритма нахождения редакционного расстояния.

Работу антиплагиата можно описать следующим образом: выбранное решение форматируется (из него удаляются все пробелы и табуляции, чтобы получилась одна цельная строка), после чего происходит обращение к базе данных, на которой находится информация о названиях файлов, в которых сохранены решения. Выборка названий происходит по следующим критериям: файл должен представлять решение той же задачи, решение должно быть верным (то есть предварительно прошедшим все тесты и, если проверялось вручную, принятым преподавателем). Последовательно происходит считывание решений из файлов с последующим форматированием (такое же преобразование в единую строку). Затем для каждой пары строк (проверяемое на плагиат и выбранное из списка ранее загруженных решений) вычисляется степень похожести на основе алгоритма редакционного расстояния (а именно расстояния Левенштейна). Степень похожести представлена в процентном значении: чем меньше процент, тем меньше вероятность того, что новое решение было скопировано у другого пользователя.

Стоит подробнее остановиться на эффективности алгоритма редакционного расстояния. Эффективность алгоритмов принято измерять временем выполнения, которое зависит от количество необходимых операций для достижения конца алгоритма, а также памяти, которая задействуется при выполнении алгоритма.

Выбранный алгоритм можно сравнить с предложенными альтернативами и построить таблицу эффективности алгоритмов.

Алгоритм редакционного расстояния работает за O(n*m), где n – количество символов в первом файле, а m – количество символов во втором файле. При этом на хранение требуется также зарезервировать памяти O(n*m), что делает данный алгоритм одним из наиболее эффективных среди предложенных решений.

Таблица 1. Таблица сравнений эффективности предложенных алгоритмов

| **Название алгоритма** | **Время** | **Память** |
|---|---|---|
| Редакционное расстояние | O(n*m) | O(n*m) |

| | | |
|---|---|---|
| Алгоритм Кнута-Морриса-Пратта | O(n+m) | O(n) |
| Алгоритм нахождения подстрок на основе суффиксных массивов | O(n*logn) | O(n) |
| Алгоритм Рабина-Карпа | O(n*m) | O(1) |

Алгоритм на основе разбиения текста на шинглы не рассмотрен, так как его применение в контексте поставленной задачи не актуально и может привести к дополнительным нагрузкам на систему проверки. Данный алгоритм также не целесообразен из-за небольших объёмов обрабатываемой информации (проверяемый код на данный момент не превышает больше 100 строк).

Стоит также учитывать, что асимптотика, приведённая выше, представлена только в худшем случае работы алгоритма антиплагиата (например, алгоритм Рабина-Карпа может работать за O(n+m) или O(n), если была выбрана эффективная хеш-функция).

Для отображения всех возможных заимствований была создана отдельная страница, на которой преподаватели могут выбрать интересующую дисциплину, раздел и задачу (см. рис. 2). Затем на стороне сервера происходит формирование списка возможных заимствований и этот список на стороне пользователя формирует таблицу, в которой представлены ID совпадающих решений, ФИО студентов и процент похожести. Данные о решениях и пользователях выводятся в порядке возрастания ID, то есть на первой позиции всегда будет расположено решение, которое было отправлено раньше. Также для удобства отклонения решений был добавлен столбец с кнопками, отметив которые, преподаватель может отклонить отправленное решение (отклоняется то решение, которое было отправлено позже). Решения, процент похожести которых составляет 100, отмечены автоматически системой.

Стоит отметить, что вердикт об отклонении возможно скопированных решений остаётся на стороне преподавателя и он может проигнорировать похожие решения, если это позволяет контекст (например, задание слишком тривиальное и большинство студентов из группы могли решить задачу одинаковым способом. Пример такой задачи – это нахождение суммы двух целых чисел).

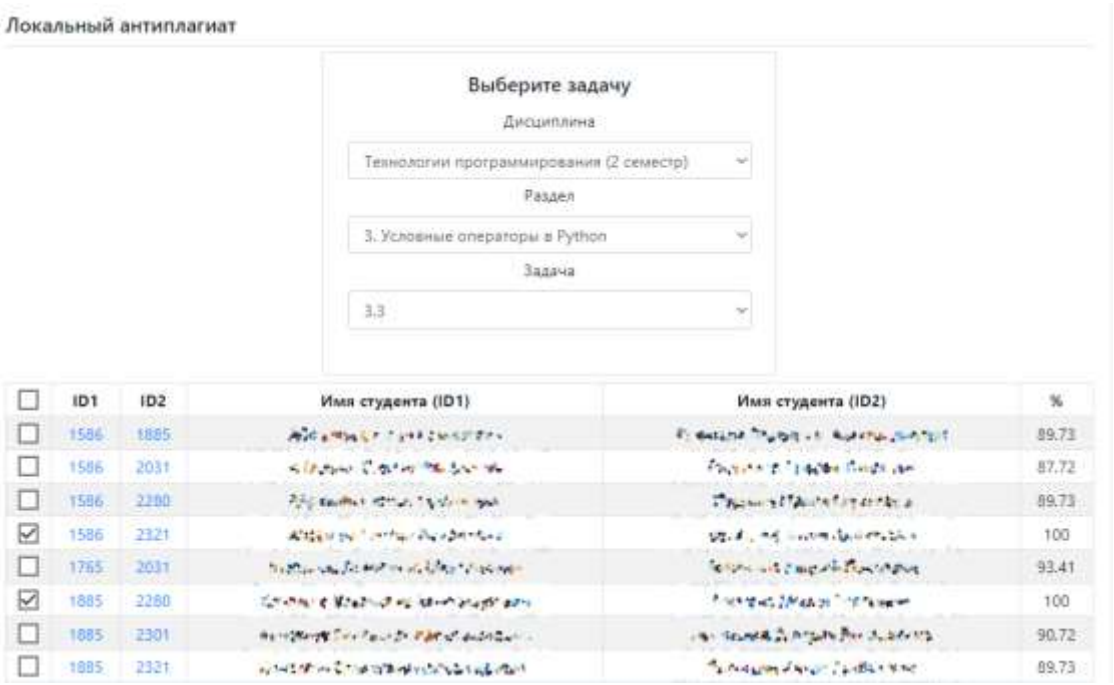


Рисунок 2. Страница локального антиплагиата

Тестирование в реальных условиях работы антиплагиата (то есть на этапе внедрения в информационную систему) показало высокую эффективность и позволило обеспечить объективную оценку знаний студентов группы по направлению “Информационные системы и технологии”. Система локального антиплагиата проектировалась с учётом высоких нагрузок, при которых проверка решений могла занять продолжительное количество времени и на этапе внедрения проблем замедления времени работы выявлено не было (кроме проблем с доступом к решениям, так как некоторые решения чистились на стороне сервера администратором или преподавателем, то они становились недоступными при проверке, что приводило к нежелательным результатам во время работы алгоритма поиска совпадений).

**Выводы**

Внедрение системы локального антиплагиата позволило найти заимствования среди студентов и объективно оценить их знания, предварительно отклонив все скопированные решения для задач, где предполагалось несколько возможных решений (то есть в учёт не брались тривиальные задачи, например, сравнение чисел или нахождение суммы двух чисел). Также система локального антиплагиата спроектирована таким образом, что преподаватель может идентифицировать самую раннюю отправку в систему, которую потом копировали другие студенты (полезно для нахождения связей между студентами и отправленными решениями, в перспективе предполагается реализация построения графа отношений для более наглядного определения хронологии копирования решений).

**Библиографический список**

1. Хованская Т. В., Сандирова М. Н. Использование системы "Антиплагиат" в высшей школе // Проблемы современного образования. 2019. № 3. С. 51-58.
2. Сафин К. Ф., Чехович Ю. В. О комбинированном алгоритме обнаружения заимствований в текстовых документах // Труды Института системного программирования РАН. 2022. Т. 34. № 1. С. 151-160.
3. Кузьминых Е. С., Ильина С. П., Маслова М. А. Анализ сходства кода и поиска его заимствований // Научный результат. Информационные технологии. 2023. Т. 8. № 3. С. 3-10.
4. Половикова О. Н., Иванова В. Е. Разработка детектора автоматической проверки на плагиат блоков программного кода для образовательной среды // Высокопроизводительные вычислительные системы и технологии. 2020. Т. 4. № 1. С. 173-178.
5. Часов Е. А. и др. Разработка методов поиска заимствований в исходном коде на языках программирования // Форум молодых ученых. 2019. № 2(30). С. 1601-1608.
6. Яковлев П. А. Метод быстрого множественного попарного выравнивания на основе префиксных деревьев // Доклады Академии наук. 2019. Т. 484. № 4. С. 401-404.