

Поиск списываний в контестах по программированию с помощью построения графов зависимостей программ

Анисимова Карина Витальевна

научный руководитель: А.В. Садовников

НИУ ВШЭ - Санкт-Петербург

19 января 2022 г.

- Число контестов, проводимых онлайн, растет
- Число нечестных участников растет
- Списывания практически не отслеживаются
- Необходимо большое количество сравнений
- Специфические модификации

Задача поиска списывания

Основные модификации¹²³:

```
#include <iostream>
using namespace std;

int main()
{
    int a = 5, b = 2, c = 0;
    for (int i = 0; i < a; i++) {
        c += i;
        cout << i;
    }
    cout << c + a + b;
    return 0;
}
```

- Добавление/удаление комментариев
- Добавление незначимых строк кода
- Переименование
- Перестановка операций
- Взаимозаменяемые конструкции
 - for/while
 - if/else

¹GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis (2006)

²Finding Plagiarisms among a Set of Programs with JPlag(2003)

³Comparison and evaluation of code clone detection techniques and tools: A qualitative approach (2009)

Задача поиска списывания

```
#include <iostream>
using namespace std;

int main()
/* comment */ {
    int a = 5, b = 2, c = 0;
    for (int i = 0; i < a; i++) {
        // comment
        c += i;
        cout << i;
    }
    cout << c + a + b;
    return 0;
}
```

- Добавление/удаление комментариев
- Добавление незначимых строк кода
- Переименование
- Перестановка операций
- Взаимозаменяемые конструкции
 - for/while
 - if/else

Задача поиска списывания

```
#include <iostream>
using namespace std;

int main()
/* comment */ {
    int a, b, c, d;
    a = 5;
    b = 2; ←
    c = 0;
    d = 42; ←
    for (int i = 0; i < a; i++) {
        // comment
        c += i;
        cout << i;
    }
    int ans = c + a + b; ←
    cout << ans;
    return 0;
}
```

- Добавление/удаление комментариев
- Добавление незначимых строк кода
- Переименование
- Перестановка операций
- Взаимозаменяемые конструкции
 - for/while
 - if/else

Задача поиска списывания

```
#include <iostream>
using namespace std;

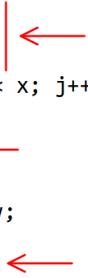
int main()
/* comment */ {
    int x, y, z, t;
    x = 5;
    y = 2;
    z = 0;
    t = 42;
    for (int j = 0; j < x; j++) {
        // comment
        z += j;
        cout << j;
    }
    int out = z + x + y;
    cout << out;
    return 0;
}
```

- Добавление/удаление комментариев
- Добавление незначимых строк кода
- Переименование
- Перестановка операций
- Взаимозаменяемые конструкции
 - for/while
 - if/else

Задача поиска списывания

```
#include <iostream>
using namespace std;

int main()
/* comment */ {
    int x = 5, y, z, t;
    z = 0;
    y = 2;
    for (int j = 0; j < x; j++) {
        // comment
        cout << j;
        z += j;
    }
    int out = z + x + y;
    cout << out;
    t = 42;
    return 0;
}
```



- Добавление/удаление комментариев
- Добавление незначимых строк кода
- Переименование
- Перестановка операций
- Взаимозаменяемые конструкции
 - for/while
 - if/else

Задача поиска списывания

```
#include <iostream>
using namespace std;

int main()
/* comment */ {
    int x = 5, y, z, t, j = 0;
    z = 0;
    y = 2;
    while (j < x) { ←
        // comment
        cout << j;
        z += j;
        j++; ←
    }
    int out = z + x + y;
    cout << out;
    t = 42;
    return 0;
}
```

- Добавление/удаление комментариев
- Добавление незначимых строк кода
- Переименование
- Перестановка операций
- Взаимозаменяемые конструкции
 - for/while
 - if/else

Program Dependency Graph

Definition

Program Dependency Graph (PDG) – представление

программы в виде графа.

Вершинами являются базовые выражения.

Ребра зависимости по данным соединяют вершины, в которых используются одинаковые данные.

Ребра передачи управления соединяют две вершины, если контролирующая вершина определяет, будет ли выполняться выражение в зависимой вершине.

program

`x := 5; y := 5; a := 1;`

while `a < 5` **do**

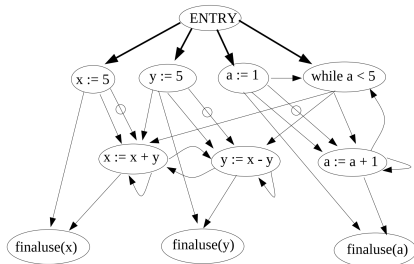
`x := x + y;`

`y := x - y;`

`a := a + 1;`

end

end(x, y, a)



Существующие решения и аналоги

- Антиплагиат
 - проверяет код как обычный текст
- SIM ⁴
 - справляется с форматированием и переименованием
 - C, Java, Pascal
- Moss ⁵
 - справляется с форматированием и переименованием
 - C/C++, C, Java, assembly
- GPLAG ⁶
 - Справляется со всеми основными модификациями
 - Только для Java
 - Оценка качества поиска контекстного плагиата не проводилась
 - Код утерян

⁴Sim: A Utility For Detecting Similarity in Computer Programs (1999)

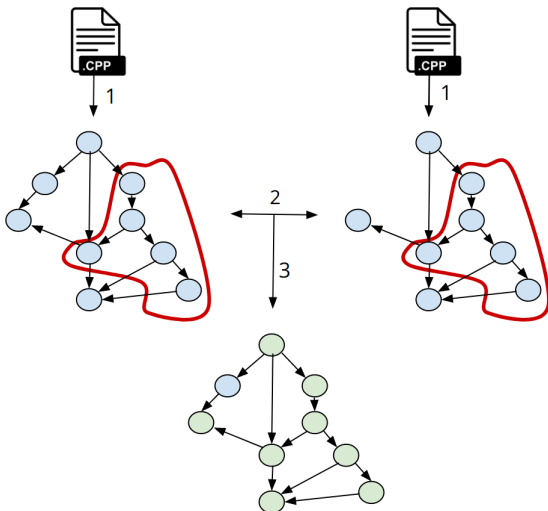
⁵MOSS, A System for Detecting Software Plagiarism(2002)

⁶GPLAG: Detection of Software Plagiarism by Program Dependence Graph Analysis (2006)

Цель: Оценить применимость алгоритма GPLAG к решению задачи поиска контекстного плагиата

Задачи:

- Реализовать алгоритм GPLAG, так как работающей версии не существует
- Собрать датасет для оценки применимости подхода
- Провести тестирование и проанализировать работу полученного решения



- 1** Построение PDG с помощью Joern
- 2** Сравнение графов
- 3** Оценка покрытия графа

Реализация алгоритма. Сравнение графов

Проблемы:

- 1 Алгоритмы поиска изоморфизмов вида граф - подграф не подходят
- 2 Подграфов в графах слишком много
- 3 Нужно учитывать типы вершин и ребер

Решения:

- 1 Сводим задачу к поиску изоморфизмов граф - подграф
- 2 Фиксируем размер подграфов: 9 вершин.
- 3 Сужаем типы вершин до 60 основных

Общедоступного датасета нет, поэтому необходимо его собрать.

Датасет для оценки способности алгоритма находить плагиат и чувствительности к разным видам модификаций:

- 372 программы из 23 контекстов с Codeforces
- С помощью инструмента `gorshochek` построены модификации:
 - Добавление/удаление комментариев
 - Переименование
 - Замена взаимозаменяемых конструкций
- Добавлена возможность построения модификации вставки незначимых строк кода в `gorshochek`
- Для каждой программы построен файл с случайным набором модификаций

Датасет для оценки работы алгоритма в случаях отсутствия плагиата:

- 23 программы, решающих одну и ту же простую задачу
- 12 программ, решающих одну и ту же сложную задачу

- Похожесть = $\frac{\text{Полученное покрытие}}{\text{Максимальное покрытие}}$
- При сравнении графов получаем покрытие
- Максимальное покрытие - покрытие полученное при сравнении программы с собой

Тестирование. Результаты

	Средняя похожесть	Стандартное отклонение	Среднее время работы (сек)
Добавление/удаление комментариев	1	0	95
Вставка незначимых строк кода	0,986	0,067	126
Замена взаимозаменяемых конструкций	0,944	0,151	140
Переименование	0,932	0,187	140
Рандомные модификации	0,816	0,288	115

Вывод: в случае наличия плагиата алгоритм показывает достаточно большое значение похожести

	Средняя похожесть	Стандартное отклонение	Среднее время работы (сек)
Простая задача	0,154	0,286	121
Сложная задача	0,054	0,153	1217

Вывод: в случае отсутствия плагиата алгоритм показывает достаточно маленькое значение похожести, но в случаях маленьких программ результат неоднозначный

- Реализован алгоритм из статьи GPLAG, поддержана гибкость в работе с разными языками программирования и разными подходами к построению PDG
- Собран датасет из 1895 программ для оценки возможности применения алгоритма к поиску контекстного плагиата
- Проведено исследование и доказана применимость алгоритма GPLAG

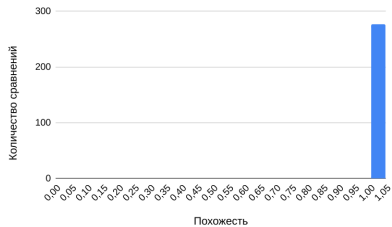
Репозиторий: github.com/Karina5005/Plagiarism

- Оценить точность других алгоритмов поиска плагата в применении к задаче поиска списываний в контекстах
- Заменить в системе построение PDG по абстрактному синтаксису на построение PDG по assembler и сравнить эти два подхода
- Придумать и применить эвристики для ускорения поиска подграфов и сокращения их количества без сильной потери точности работы алгоритма
- Решить проблему нахождения плагиата в случаях популярных паттернов

Тестирование. Результаты

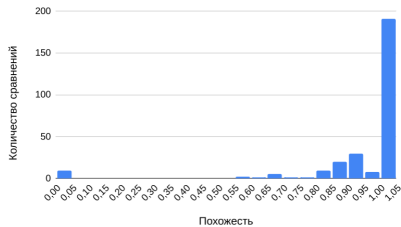
Добавление/удаление комментариев

Среднее время работы: 96 секунд



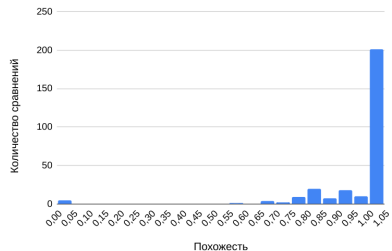
Переименование

Среднее время работы: 141 секунда



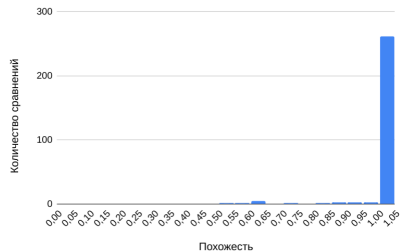
Взаимозаменяемые конструкции

Среднее время работы: 141 секунда



Вставка незначимых строк кода

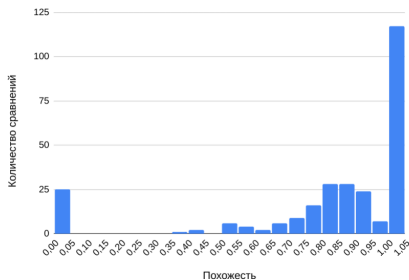
Среднее время работы: 127 секунд



Тестирование. Результаты

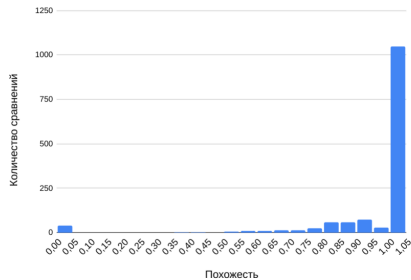
Все модификации

Среднее время работы: 115 секунд



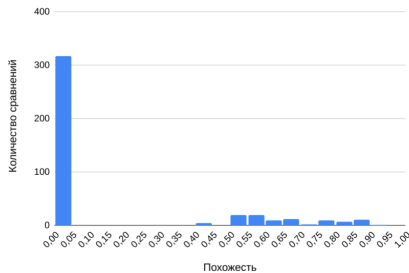
Все сравнения

Среднее время работы: 123 секунды



Тестирование. Результаты

Простая задача



Сложная задача

