

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.

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;

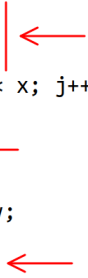
int main()
/* comment */ {
    int x, y, z, t;
    x = 5;
    y = 2;
    z = 0;
    t = 42;
    for (int j = 0; j < x; j++) {
        // comment
        z += j;
        cout << j;
    }
    int out = z + x + y;
    cout << out;
    return 0;
}
```

- 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

Существующие решения и аналоги

- Антиплагиат
 - проверяет код как обычный текст
- SIM ⁴
 - токенизация
 - C, Java, Pascal
- Moss ⁵
 - токенизация
 - C/C++, C, Java, assembly
- GPLAG ⁶
 - Program Dependency Graph
 - Только для Java
 - 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)

Definition

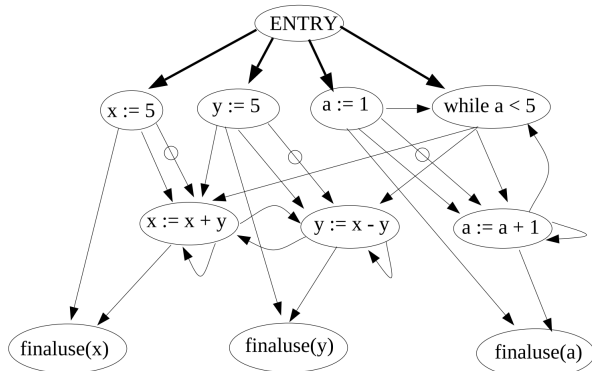
Program Dependency Graph (PDG) – представление программы в виде графа.

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 Dependency Graph



```
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)
```

Цель: Оценить применимость подхода статьи GPLAG к решению задачи поиска контекстного плагиата

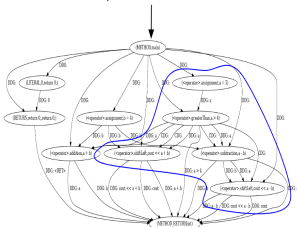
Задачи:

- Реализовать алгоритм из статьи GPLAG
- Собрать датасет
- Провести тестирование и проанализировать работу полученного решения

Алгоритм

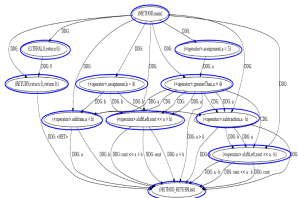
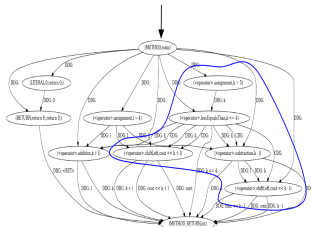
```
#include <iostream>
using namespace std;
```

```
int main() {
    int a, b;
    a = 5;
    b = 4;
    if (a > 4) {
        cout << a - b;
    } else {
        cout << a + b;
    }
    return 0;
}
```



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

```
int main() {
    int k = 5;
    int l = 4;
    if (k <= 4) {
        cout << k + l;
    } else {
        cout << k - l;
    }
    return 0;
}
```



- PROGEX

- Парсит только Java, но можно добавлять и свои грамматики
- На практике добавление новой грамматики проблематично

- TinyPDG

- Умеет парсить только Java код
- Добавление новой грамматики не предусмотрено

- Joern

- Умеет парсить только C/C++ и Java код
- Результат в формате *.dot

- Алгоритмы поиска изоморфизмов типа граф - подграф не подходят
- Подграфов в графах слишком много
- Нужно учитывать типы вершин и ребер

- Строим подграфы меньшего графа, ищем изоморфизмы вида граф - подграф для всех пар (подграф меньшего графа, больший граф)
- Фиксируем размер подграфов: 9 вершин. Предподсчитаем подграфы для тестовых данных.
- Сужаем типы вершин до 60 основных

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:

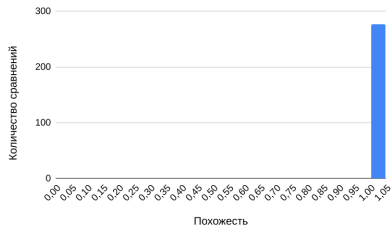
- 23 программы, решающих одну и ту же простую задачу
- 12 программ, решающих одну и ту же сложную задачу

- Смотрим на количество покрытых вершин
- Идеальным результатом считаем покрытие графа при сравнении с собой
- Оцениваем метрику похожести
- Похожесть = $\frac{\text{Полученное покрытие}}{\text{Максимальное покрытие}}$

Тестирование. Результаты

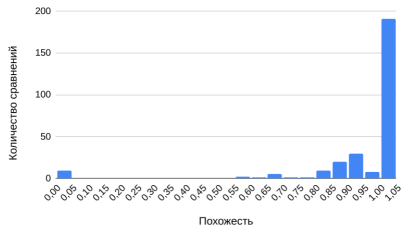
Добавление/удаление комментариев

Среднее время работы: 96 секунд



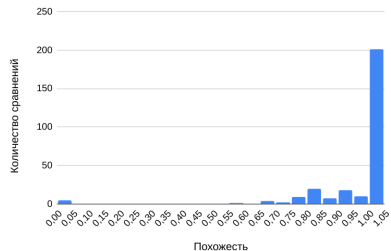
Переименование

Среднее время работы: 141 секунда



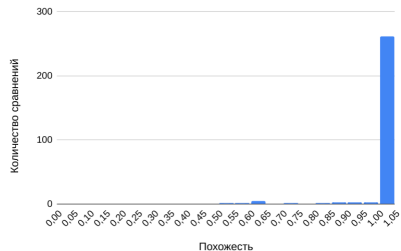
Взаимозаменяемые конструкции

Среднее время работы: 141 секунда



Вставка незначимых строк кода

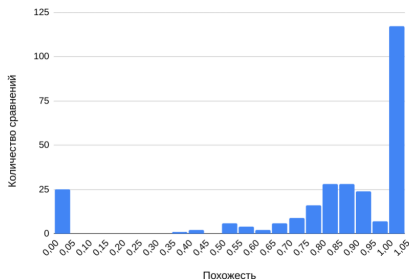
Среднее время работы: 127 секунд



Тестирование. Результаты

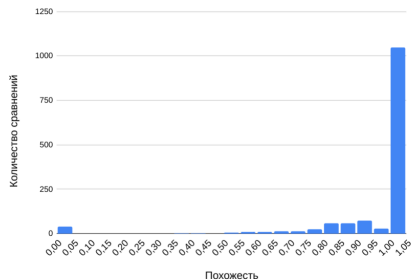
Все модификации

Среднее время работы: 115 секунд



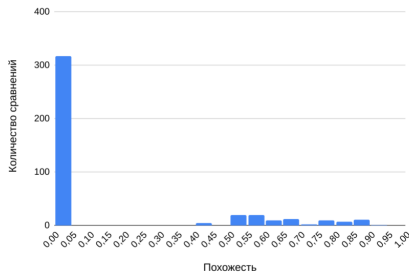
Все сравнения

Среднее время работы: 123 секунды

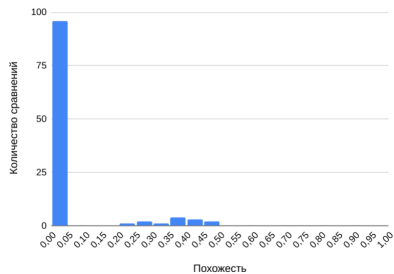


Тестирование. Результаты

Простая задача



Сложная задача



- Реализовала алгоритм из статьи GPLAG
- Поддержала гибкость в работе с разными языками программирования
- Поддержала гибкость в работе с разными подходами к построению PDG
- Собран датасет для оценки возможности применения алгоритма к поиску контекстного плагиата
- Провела исследование и доказала применимость алгоритма

- 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