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

Программа 1:

```cpp
#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;
}
```

Программа 2:

```cpp
#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;
}
```

Задача поиска списывания. Основные модификации

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

Существующие решения и аналоги

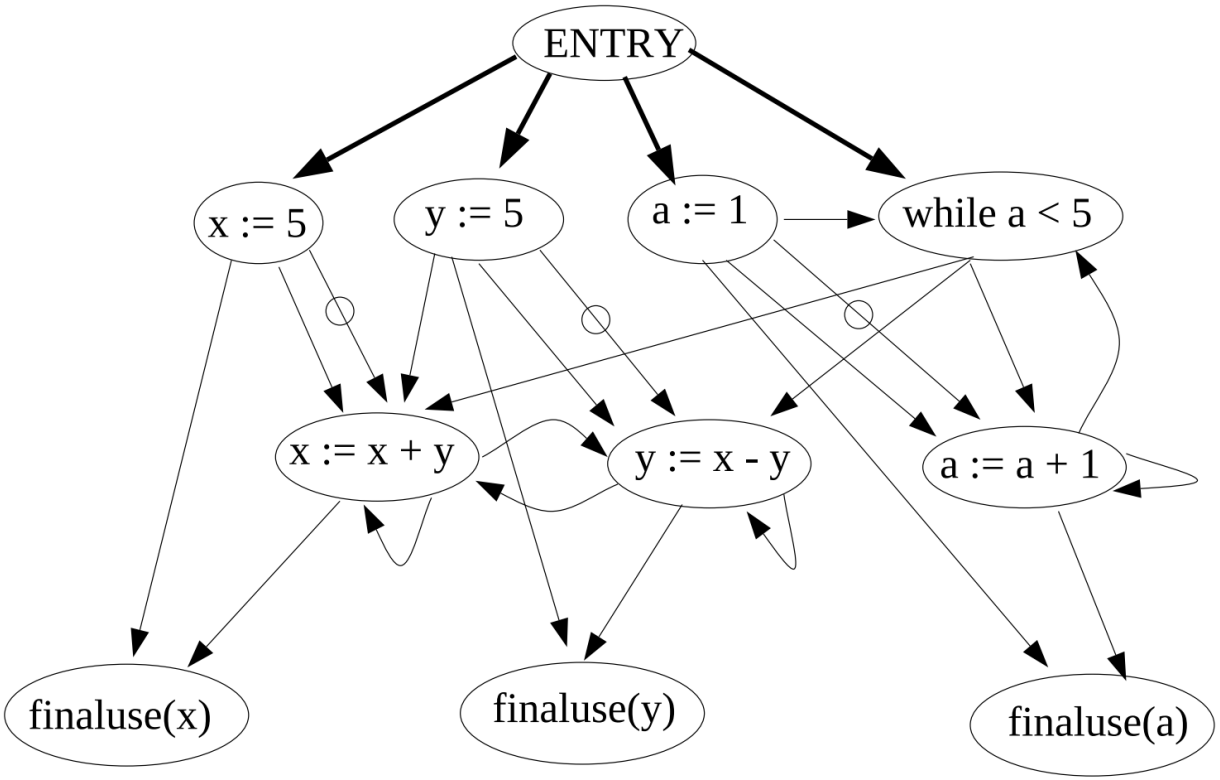
- Антиплагиат
  - проверяет код как обычный текст
- COPYLEAKS
  - токенизация
  - сравнивает программу с программами из базы
- GPLAG
  - Program Dependency Graph
  - Только для Java
  - Код утерян

PDG

Definition

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.

PDG




```
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)
```

Постановка цели и задачи

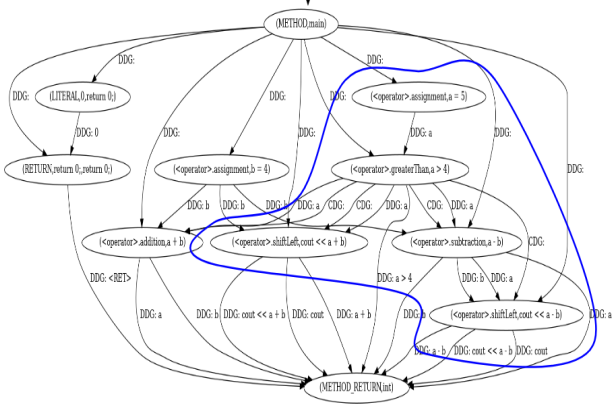
Цель: Оценить применимость подхода статьи GPLAG к решению задачи поиска контестного плагиата

Задачи:

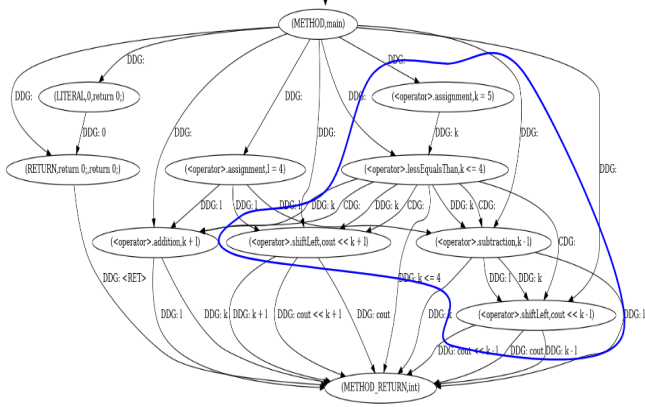
- Реализовать алгоритм из статьи GPLAG
- Собрать датасет
- Провести тестирование и проанализировать работу полученного решения

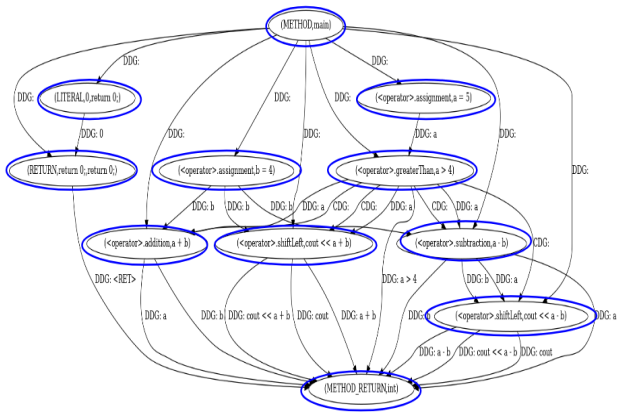
Алгоритм

```cpp
#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;
}
```




```cpp
#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;
}
```





Построение PDG

- 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

Сравнение графов. Постановка проблемы

Для сравнения графов нужно смотреть на изоморфизмы между парами подграфов, на этом этапе возникают следующие проблемы:

- Известные алгоритмы не подходят, так как рассматривают изоморфизмы типа граф - подграф
- Подграфов в графах слишком много, поэтому находить все проблематично, даже если найдем то перебор всех пар будет работать слишком долго
- Нужно учитывать типы вершин и ребер

Сравнение графов. Решение

- Сужаем типы вершин до 60 основных
- Рассматриваем подграфы фиксированного размера: 9 вершин
- Выбираем меньший граф и строим его подграфы оптимальным перебором
- Используем алгоритмы для поиска изоморфизмов вида граф - подграф для всех пар <подграф меньшего графа - больший граф>
- Так как подграфов все равно очень много, а наша задача протестировать работоспособность подхода, предподсчитаем подграфы для тестовых данных заранее

Построение датасета

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

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

Построение датасета

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

Тестирование. Полученные результаты

Итоги

Дальнейшие планы

- 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