Система анализа поведения студентов при обучении программированию

Люлина Елена Сергеевна
научный руководитель: доц. кафедры СП,
к.т.н. Т.А. Брыксин
рецензент: генеральный директор ООО “Цифровые образовательные решения” Н. И. Вяххи

СПБГУ, Математическое обеспечение и администрирование информационных систем, Системное программирование

6 июня 2020 г.

Введение

Зачем это надо?

- Самостоятельное решение задач вызывает трудности у студентов[1]
- Понимание поведения и ошибок студентов поможет экономить ресурсы преподавателей
- Данные о решениях студентов помогут автоматизировать процесс подсказок
- Для эффективности онлайн-курсов и самостоятельного обучения

[1] Lahtinen Essi, Ala-Mutka Kirsti, Järvinen Hannu-Matti. A Study of the Difficulties of Novice Programmers // ITiCSE'05. –– 2005.

Цель и задачи

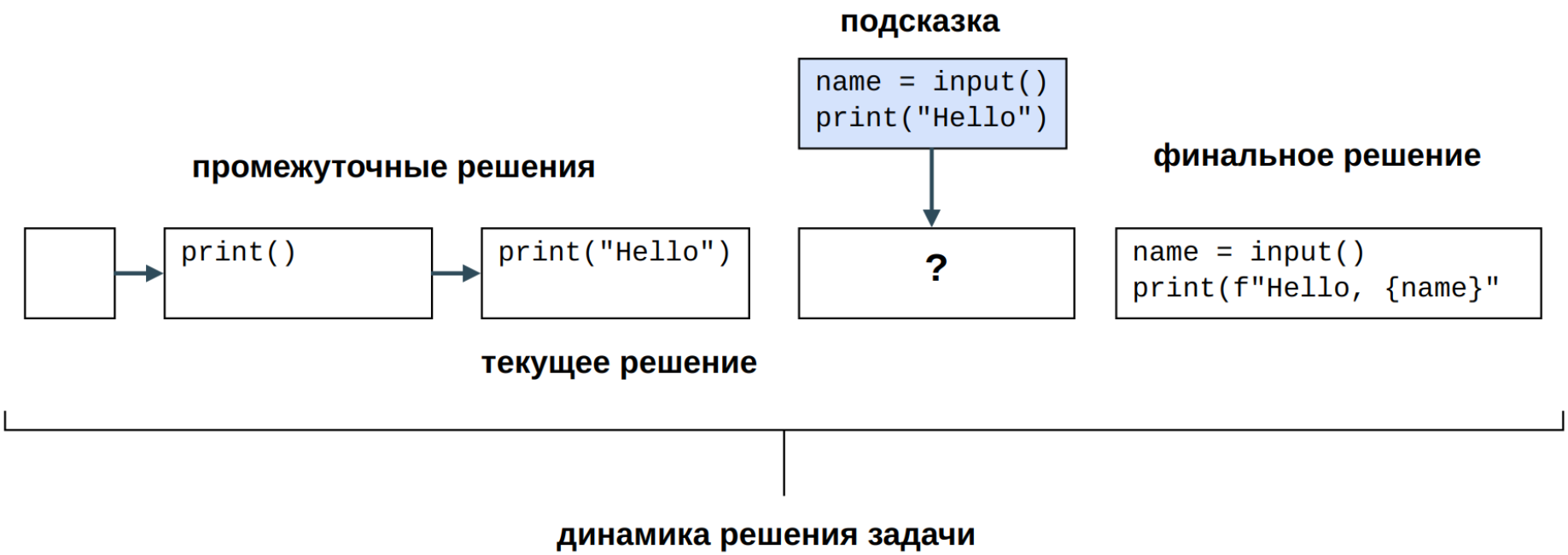
Цель:

Разработка алгоритма генерации подсказок для решения учебных задач на языке Python

Были поставлены следующие задачи:

- сделать обзор существующих решений
- собрать датасет с динамиками решений задач
- разработать алгоритм генерации подсказок
- протестировать алгоритм и оценить релевантность подсказок

Динамика решения задачи



Абстрактное синтаксическое дерево

Фрагмент кода

```
name = input()
print("Hello, " + name)
```

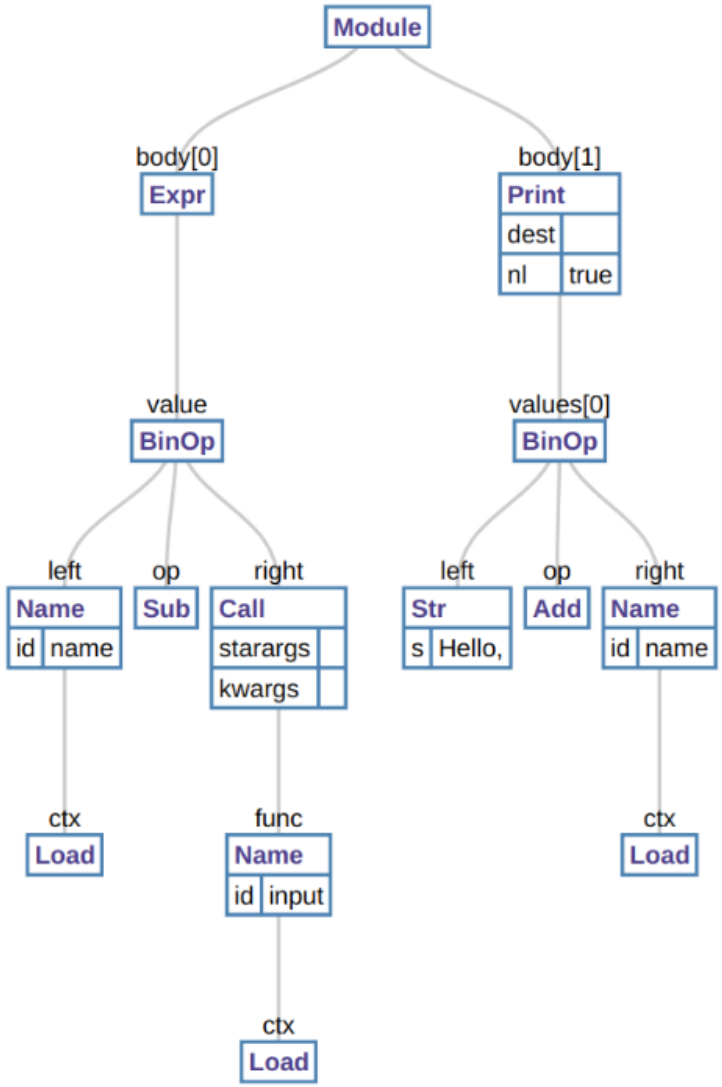
Изменения AST:

1. удаление вершины

2. добавление вершины

3. замена вершины

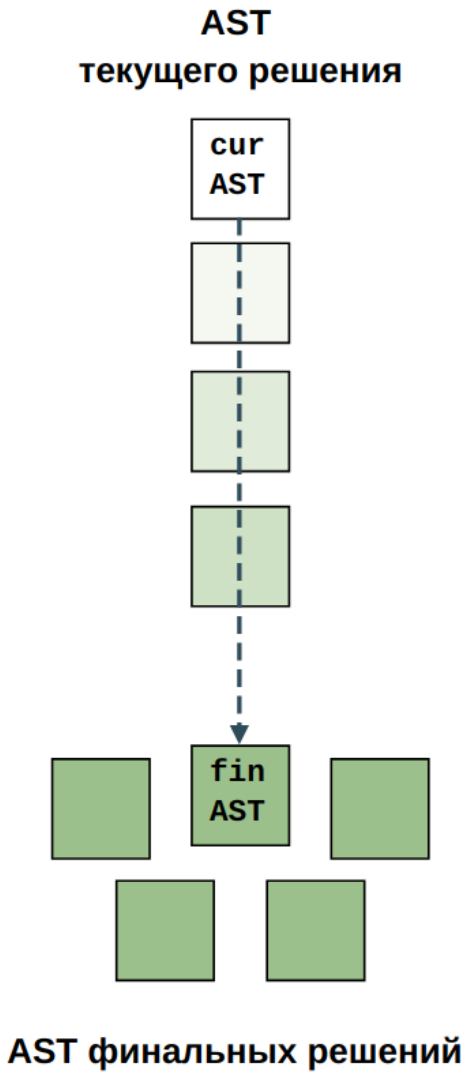
Abstract Syntax Tree (AST)



Обзор существующих решений

Генерация подсказок для языка Python[2]:

- предлагается алгоритм каноникализации AST
- используются только финальные решения
- подсказка ищется перебором булеана изменений
- плохо работает на сложных задачах

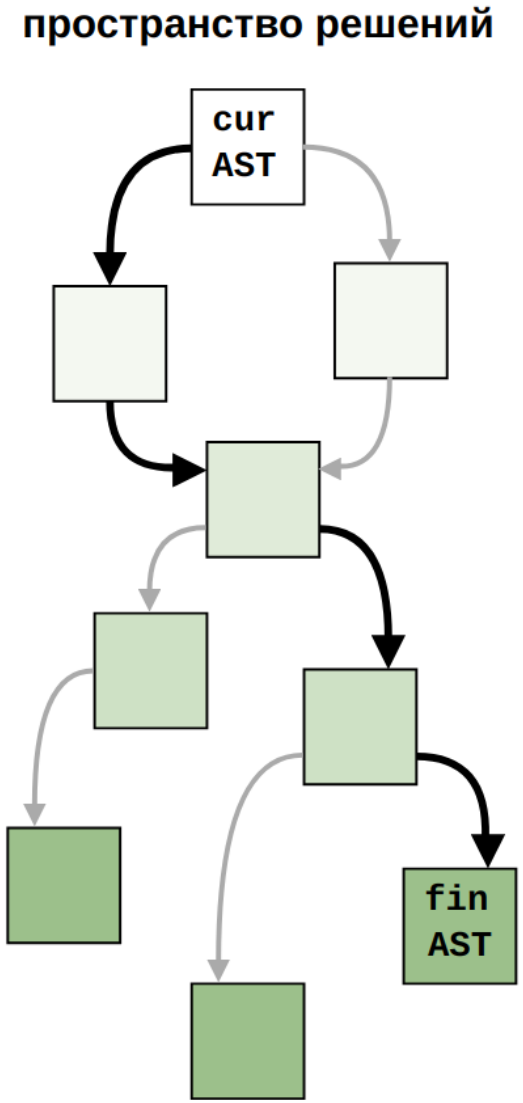


[2] Rivers, K. ”Data-Driven Hint Generation in Vast Solution Spaces: a Self-Improving Python Programming Tutor”, International Journal of Artificial Intelligence in Education — 2017.

Обзор существующих решений

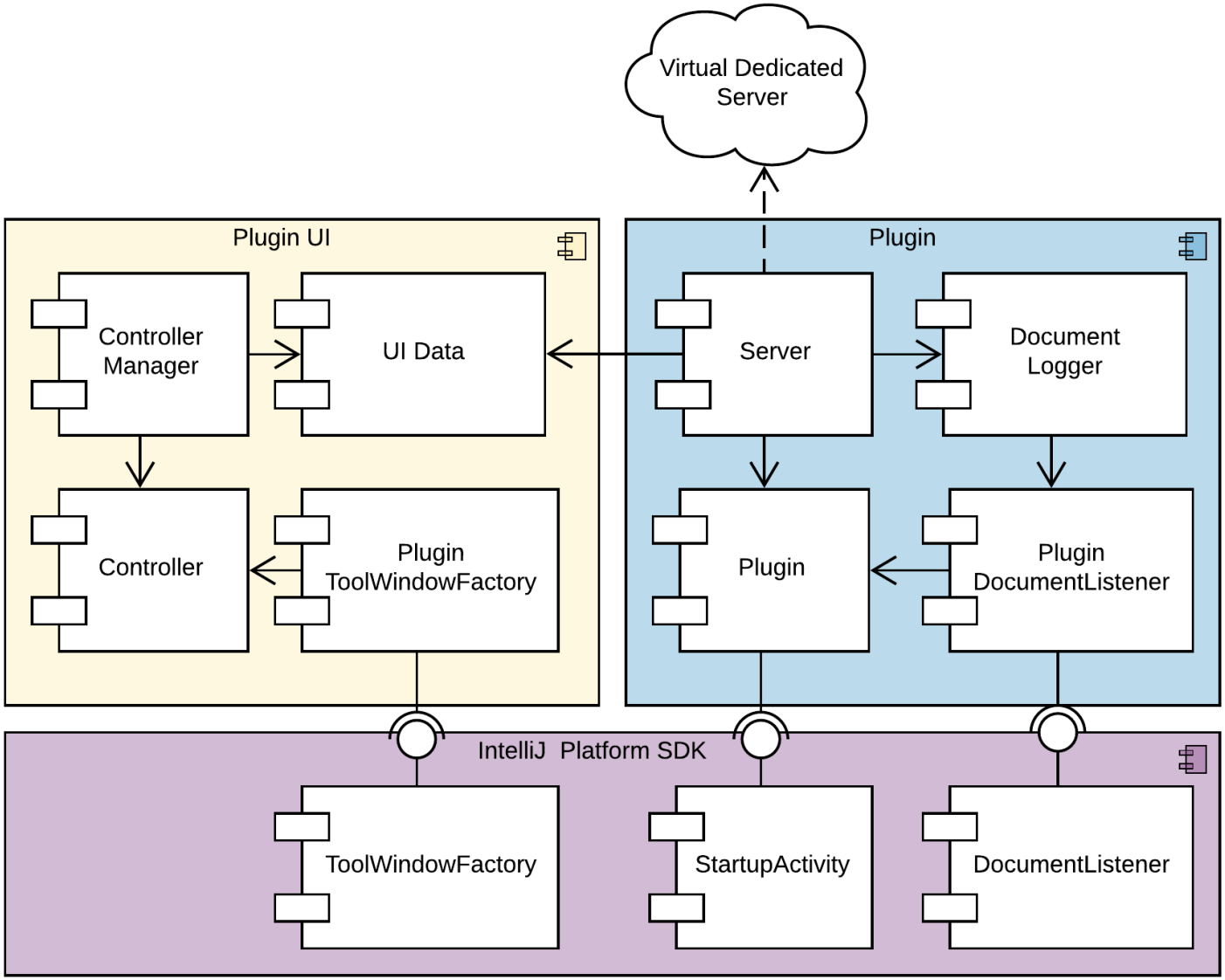
Генерация подсказок для блочных языков[3]:

- используются динамики решений, объединенные в пространство решений
- подсказка ищется как следующее решение на пути к финальному
- требует дополнительных данных

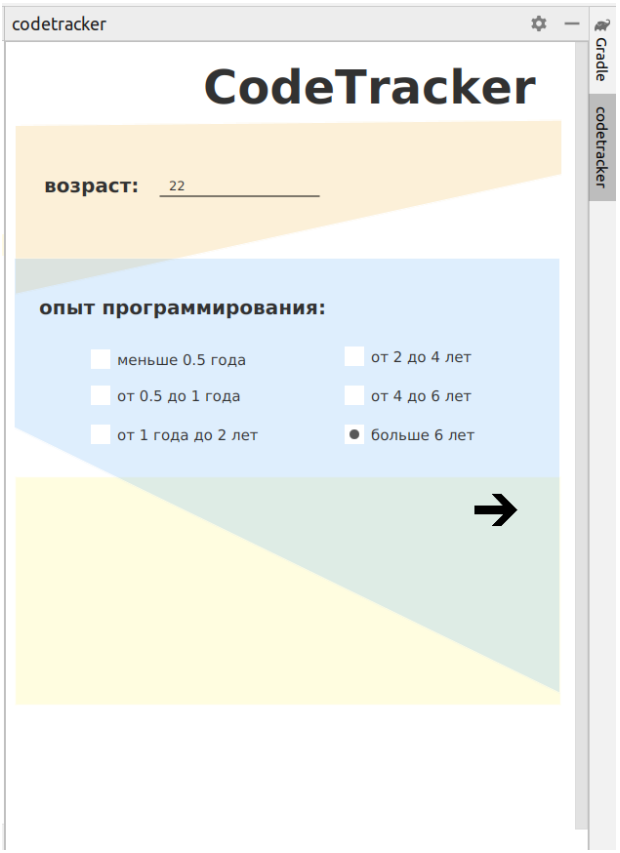


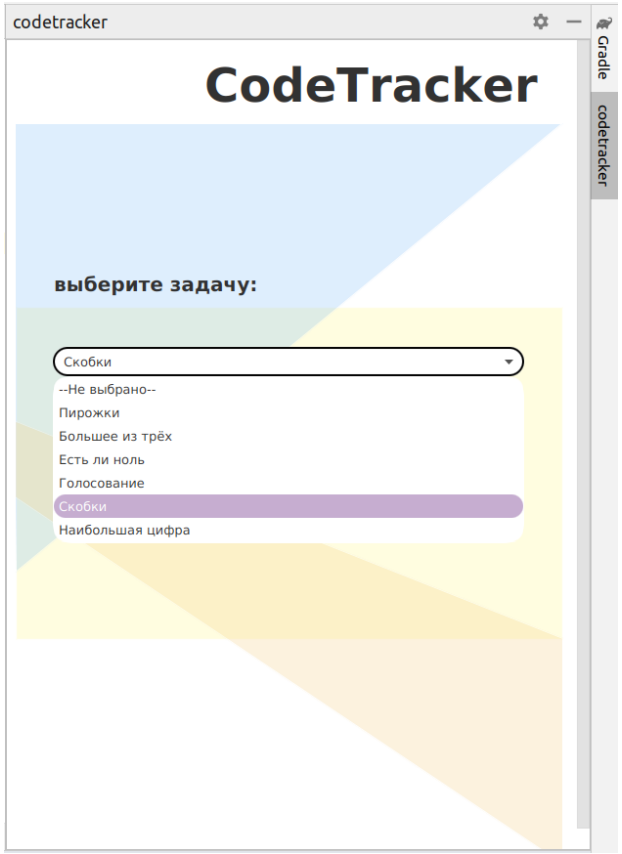
[3] Piech, C.”Autonomously Generating Hints by Inferring Problem Solving Policies”, L@S ’15 — 2015.

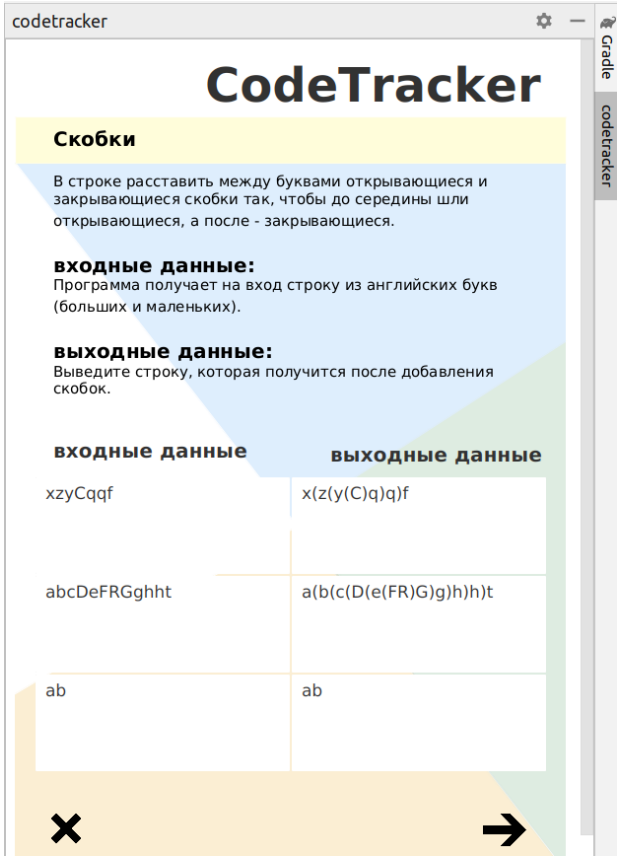
Архитектура плагина для сбора данных



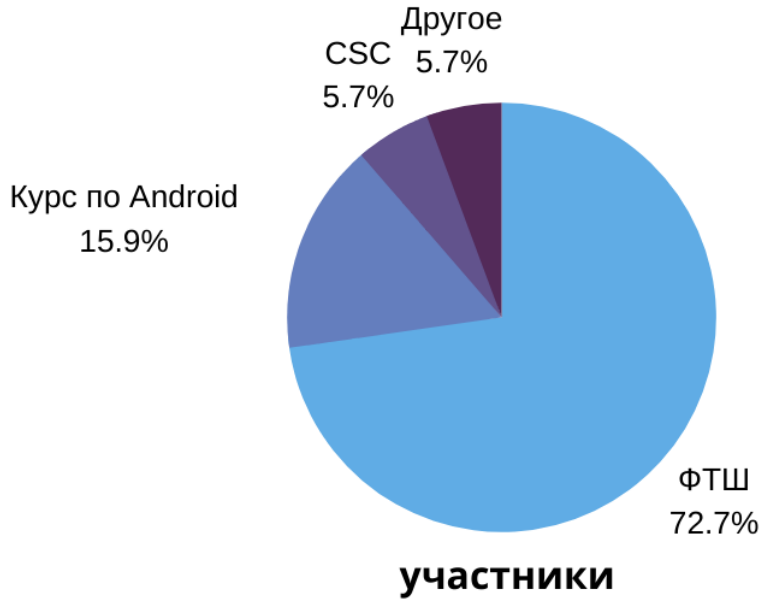
Функциональность плагина

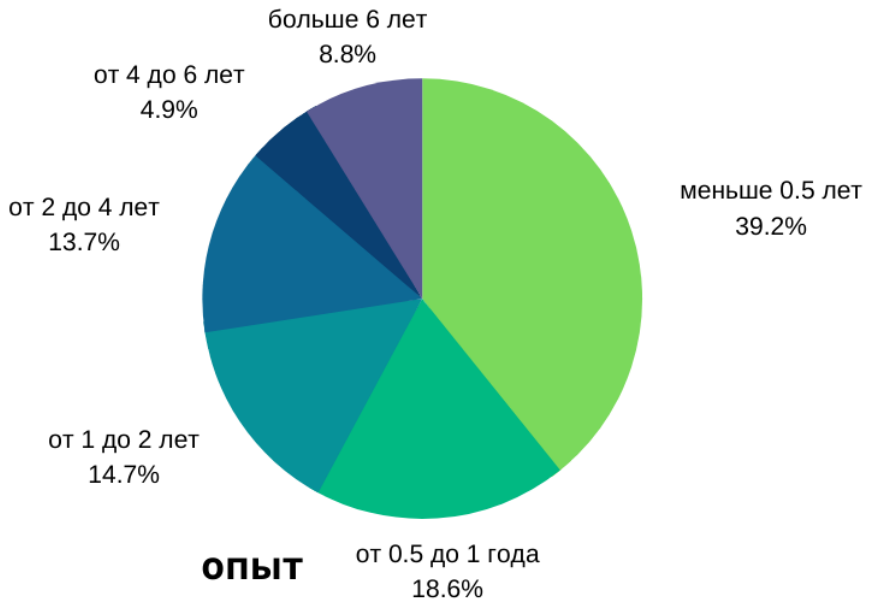


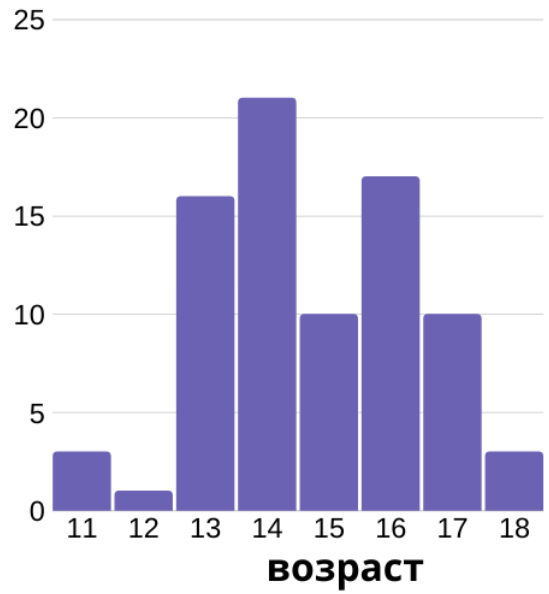




Собранные данные

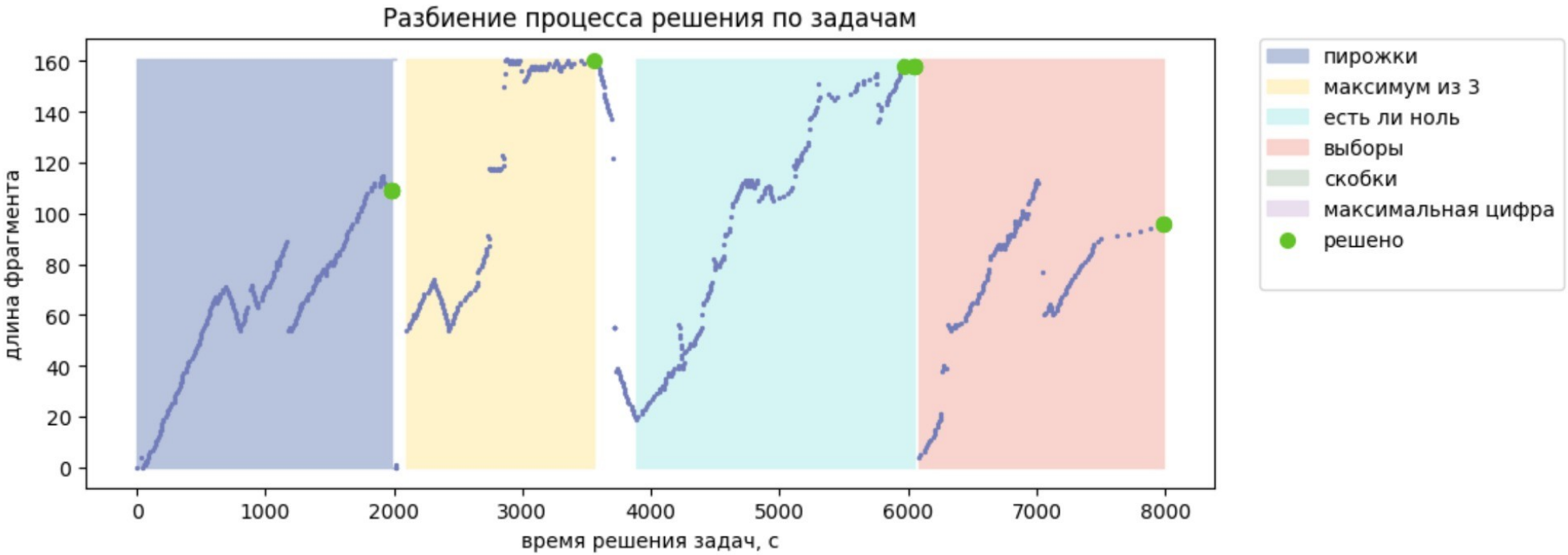






Обработка данных

- фильтрация некорректного кода
- оценка решений задач при помощи тестов
- очистка кода от лишних фрагментов



Алгоритм генерации подсказок

Основные особенности алгоритма:

- использует пространство решений для текстовых программ
- позволяет применять алгоритмы генерации подсказок для блочных языков
- использует алгоритм каноникализации для сжатия пространства решения
- использует более точный подсчёт изменений между фрагментами кода
- учитывает опыт и возраст
- учитывает структуру фрагмента

Примеры генерации подсказок

Задача: сколько будут стоить *n* пирожков, если один пирожок стоит *a* рублей и *b* копеек?

1.

aAST текущего фрагмента:

```
g4 = int(input())
g3 = int(input())
g2 = int(input())
g0 = (g4 * g2)
g1 = (g3 * g2)
if (g1 >= 100):
    g5 = (g1 // 100)
g1 = (g1 - (g5 * 100))
g0 = (g0 + g5)
print(g0, g1)
```

aAST, предложенное в качестве подсказки:

```
g4 = int(input())
g3 = int(input())
g2 = int(input())
g0 = (g4 * g2)
g1 = (g3 * g2)
if (g1 >= 100):
    g5 = (g1 // 100)
    g0 = (g0 + g5)
    g6 = (g1 % 100)
    g1 = g6
print(g0, g1)
```

2.

aAST текущего фрагмента:

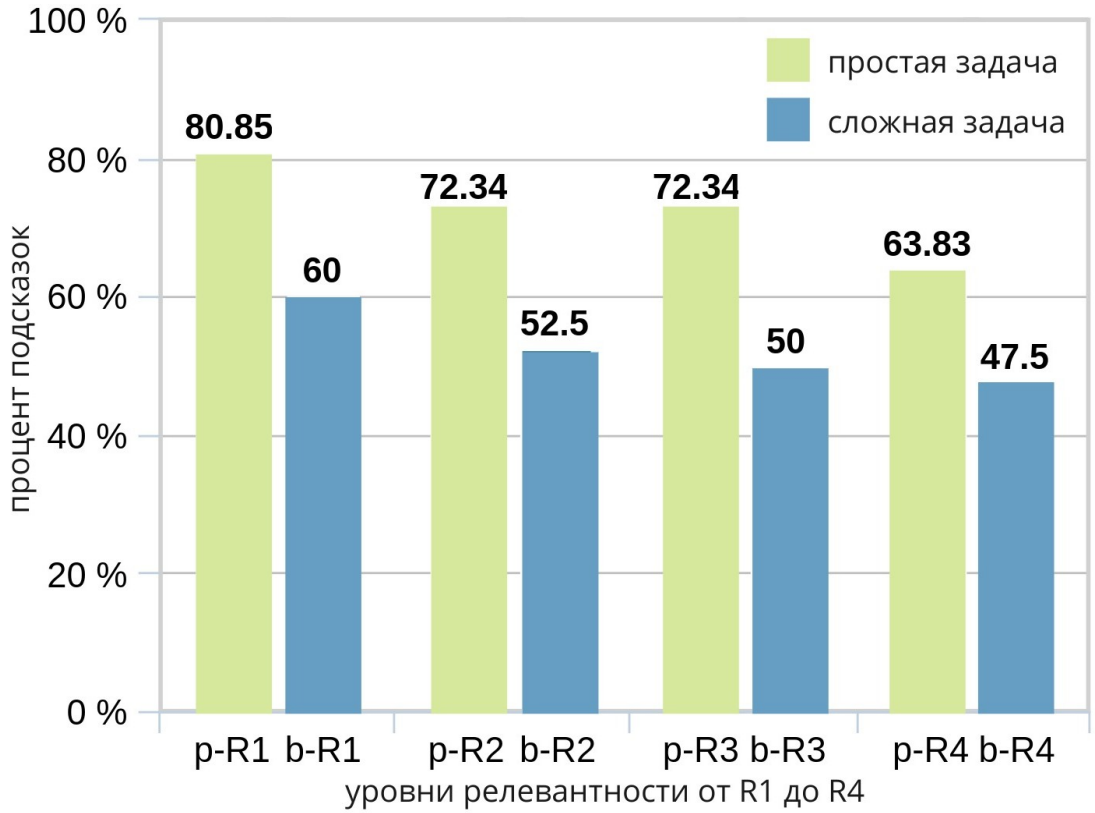
```
g4 = int(10)
g3 = int(15)
g2 = int(2)
g1 = (g2 * g4)
g0 = (g2 * g3)
print()
```

aAST, предложенное в качестве подсказки :

```
g4 = int(input())
g3 = int(input())
g2 = int(input())
g1 = (g2 * g4)
g0 = (g2 * g3)
```

Тестирование генерации подсказок

- критерии подсказки:
  - логичность
  - шаг
  - структура
  - приближение к решению
- оценены 50 подсказок для двух задач различной сложности
- определены 4 уровня релевантности



Результаты

- сделан обзор существующих решений
- создан инструмент для сбора данных
- собран набор данных, включающий динамики решений
- реализованы алгоритмы обработки данных, построения пространства решений
- разработан алгоритм генерации подсказок с использованием пространства решения
- протестирован алгоритм генерации подсказок для задач разного уровня сложности, достигнуты 80.85% и 60% релевантных подсказок

Задачи для тестирования

Название: Пирожки
Описание: Пирожок в столовой стоит *a* рублей и *b* копеек. Определите, сколько рублей и копеек надо заплатить за *n* пирожков.

Пример решения:

```
a = int(input())
b = int(input())
n = int(input())
print(a * n + (b * n) // 100, b * n % 100)
```

Пример решения:

```
a = int(input())
b = int(input())
n = int(input())
r = a * n
k = b * n
while k >= 100:
    k -= 100
    r += 1
print(r, k)
```

Название: Скобки
Описание: В строке расставить между буквами открывающиеся и закрывающиеся скобки так, чтобы до середины шли открывающиеся, а после - закрывающиеся.
В случае нечётной длины: example → e(x(a(m)p)l)e
В случае чётной длины: card → c(ar)d, но не c(a()r)d

Пример решения:

```
s = input()
l = len(s)
print('('.join(s[:(l + 1) // 2])[:l - l % 2] +
      ')'.join(s[l // 2:]))
```

Пример решения:

```
data = input()
length = len(data)
if length % 2 == 1:
    for i in range(length):
        if i < length // 2:
            print(data[i] + '(', end='')
        elif i != length-1:
            print(data[i] + ')', end='')
        else:
            print(data[i], end = '')
else:
    for i in range(length):
        if i < length / 2 - 1:
            print(data[i] + '(', end='')
        elif (i != length / 2 - 1) and (i != length-1):
            print(data[i] + ')', end='')
        else:
            print(data[i], end = '')
```