Задачи по Программированию с Трекингом Решений

Владимир Бондарев
кафедра программной инженерии
Харьковский национальный университет радиоэлектроники
Харьков, Украина
volodymyr.bondariev@nure.ua

Юлия Черепанова
кафедра программной инженерии
Харьковский национальный университет радиоэлектроники
Харьков, Украина
yulia.cherepanova@nure.ua

Programming Problems with Solution Tracking

Volodymyr Bondariev
Department of Software Engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
volodymyr.bondariev@nure.ua

Yulia Cherepanova
Department of Software Engineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
yulia.cherepanova@nure.ua

***Аннотация*—Предлагается дополнить систему автоматической проверки задач по программированию детальным протоколированием хода решения. Протокол позволяет анализировать действия обучаемого и корректировать процесс обучения на основании такого анализа. Кроме того, протокол может показать, сам ли обучаемый нашел решение или воспользовался чужим, что весьма актуально при удаленном обучении.**

***Abstract*—It is proposed to supplement the system of automatic verification of programming problems with detailed logging of the solution progress. The protocol allows you to analyze the student's actions and adjust the learning process based on such an analysis. In addition, the protocol can show whether the student himself found a solution or used someone else's, which is very important in remote learning.**

Ключевые слова—обучение программированию; задача; трекинг решения; автоматическая проверка решений; контроль знаний

Keywords—teaching of programming; a task; tracking of solution; automatic testing; knowledge control

I. Введение

Нет нужды доказывать, что залогом успешного обучения программированию является практическая работа самого обучаемого, а именно решение задач. Условием задачи является словесная спецификация некоторой программы или ее фрагмента. Обучаемый должен написать программный код, который полностью удовлетворяет заданной спецификации.

Проверять соответствие кода спецификации можно по-разному. Чаще других применяется проверка программы на наборе тестов. Если все тесты пройдены успешно, задача считается решенной, если хоть один тест провален, задача не решена. Подход, в котором оценка решения зависит от количества пройденных тестов, вносит ненужный субъективизм в оценивание и идет вразрез со сложившейся практикой [1].

Проверка решения при помощи тестирования может быть автоматизирована. Автоматизация проверки имеет несколько следствий. Во-первых, автоматизация кардинально уменьшает участие преподавателя в проверке. Это важно, т.к. для успеха обучения студент должен решить много задач, и преподаватель физически не может проверить все решения.

Во-вторых, наряду с собственно проверкой, открывается возможность сохранения результатов работы обучаемого, как конечных, так и промежуточных. Это даст дополнительную информацию о решении и о том, каким образом оно было получено, что позволит преподавателю точнее оценить решение.

II. Контроль знаний

Вопрос самостоятельности работы при сдаче экзаменов стоял всегда, но при дистанционном контроле знаний он стал особенно острым. Не будет преувеличением сказать, что от решения этого вопроса зависит судьба всего высшего образования, поскольку оценки являются главным мотивирующим фактором для

большинства студентов. Вряд ли здесь возможна серебряная пуля, которая гарантирует самостоятельность выполнения заданий, но можно и нужно затруднить заимствование настолько, чтобы оно стало не целесообразным для основной массы студентов.

В этом отношении будет полезным детальное протоколирование действий студента по решению задачи. Представим, что содержимое окна редактора периодически сохраняется, скажем, каждые 5 сек. Тогда итогом работы будет не только программный код, но и то, каким образом он создавался. По протоколу решения можно судить, работал студент над задачей самостоятельно или же воспользовался чужим решением.

Разумеется, непосредственное восприятие длинной последовательности вариантов кода мало что даст, но после обработки сырых данных ситуация может измениться. Например, просто подсчитав количество добавленных и удаленных символов на каждом шаге наблюдения, можно составить диаграмму, которая имеет характерный вид для определенных вариантов поведения испытуемых (рис.1-4).

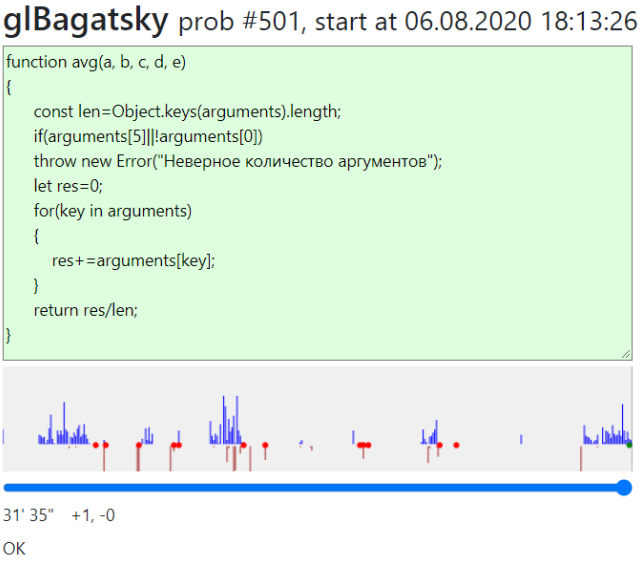

Рис. 1. Окно анализа решения.

Рис. 2. Диаграмма быстро написанного решения простой задачи.

Рис. 3. Диаграмма решения, которое последовательно редактировалось и тестировалось на протяжении сеанса тестирования.

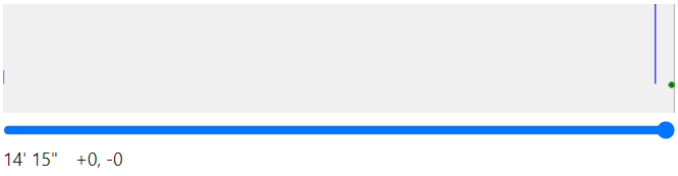

Рис. 4. Диаграмма решения, внезапно полученного в конце сеанса тестирования, – подозрение на плагиат.

Другим применением протокола является выявление трудностей, которые испытал студент в ходе решения задачи. В черновом виде такое выявление также можно автоматизировать, например, указывая на отрезки времени, в течение которых студент не предпринимал никаких действий. Уверены, что это позволит преподавателю скорректировать свои действия при изложении материала, например, остановиться на тех моментах, которые хуже усвоены аудиторией.

III. Реализация трекинга

Решение задач с автоматической проверкой производится в рамках портала по обучению программированию [2]. Трекинг представляет собой последовательность состояний решения задачи. Под состоянием понимается уже написанный программный код, вообще говоря, строка. Состояния фиксируются через равные промежутки времени. Чтобы уменьшить количество сохраняемых данных, сохраняются не сами состояния, а разности между двумя соседними состояниями.

Определение разности основано на алгоритме нахождения максимальной общей подстроки двух строк. Разность состояний A и B, B-A, определяется следующим образом. Находится максимальная подстрока M, которая разбивает строки A и B на три области: (A1, M, A2) и (B1, M, B2) соответственно. Разность B-A определяется рекурсивно как (pos, len, B1-A1, B2-A2), где pos - место вхождения M в A, а len - длина подстроки M.

Базу рекурсии составляет случай, когда строка M пуста (либо меньше минимально допустимого размера d). В этом случае разность B-A = B.

Для поиска максимальной общей подстроки применяется динамический вариант алгоритма LCS (Lagest Common String) [3].

Література References

[1] Официальный сайт компании TopCoder. [Online]. Available: https://www.topcoder.com/

[2] Бондарев В.М., Черепанова Ю.Ю. Сетевая среда для подготовки и чтения лекций. / Збірник наукових праць Харківського національного університету Повітряних Сил. 4(53). 2017. С. 171-177. http://www.hups.mil.gov.ua/periodic-app/article/17810

[3] Т.Х. Кормен, Ч.И. Лейзерсон, Р.Л. Ривест, К. Штайн. Алгоритмы: построение и анализ, 3-е издание = Introduction to Algorithms, Third Edition. — М.: «Вильямс», 2013. — 1328 с. — ISBN 978-5-8459-1794-2.