– Применение предложенного подхода к решению практических задач, в частности, применение метода к генетическому выращиванию тестов, описанных в работе [4].

Литература

1. Лотов А.В., Поспелова И.И. Многокритериальные задачи принятия решений: Учебное пособие. – М.: МАКС Пресс, 2008. – 197 с.
2. Буздалов М.В. Генерация тестов для олимпиадных задач по теории графов с использованием эволюционных алгоритмов. Магистерская диссертация. СПбГУ ИТМО, 2011 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/papers/2011-master-buzdalov/, св. Яз. рус. (дата обращения 25.11.2011).
3. Mitchell M. An Introduction to Genetic Algorithms. – MA: MIT Press, 1999. – 221 p.
4. Буздалов М.В. Генерация тестов для олимпиадных задач по программированию с использованием генетических алгоритмов // Научно-технический вестник СПбГУ ИТМО. – 2011. – № 2 (72). – С. 72–77.
5. Sutton R.S., Barto A.G. Reinforcement Learning: An Introduction. – MA: MIT Press, 1998. – 322 p.
6. Mitchell T.M. Machine Learning. – McGraw Hill, 1997. – 432 p.
7. Kaelbling L.P., Littman M.L., Moore A.W. Reinforcement Learning: A Survey // Journal of Artificial Intelligence Research. – 1996. – V. 4. – P. 237–285.
8. От моделей поведения к искусственному интеллекту / Под ред. В.Г. Редько. – 2-е. изд. – М.: КомКнига, 2010. – 456 с.
9. Strehl A.L., Li L., Wiewora E., Langford J., Littman M.L. PAC model-free reinforcement learning // Proceedings of the 23rd international conference on Machine learning. ICML'06. – 2006. – P. 881–888.
10. Da Silva B.C., Basso E.W., Bazzan A.L.C., Engel P.M. Dealing with non-stationary environments using context detection // Proceedings of the 23rd international conference on Machine learning. ICML'06. – 2006. – P. 217–224.

Афанасьева Арина Сергеевна – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, студент, afanasyevarina@gmail.com

Буздалов Максим Викторович – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, аспирант, mbuzdalov@gmail.com

УДК 004.415.53

АВТОМАТИЧЕСКИЕ МЕТОДЫ МОДИФИКАЦИИ РЕШЕНИЙ ДЛЯ ТЕСТИРОВАНИЯ ПРОВЕРЯЮЩИХ ПРОГРАММ

А.А. Ахи, А.С. Станкевич, А.А. Шалыто

Предлагается метод автоматизированного тестирования проверяющих программ для олимпиадных задач по программированию, предназначенный для выявления ошибок в тестируемых программах. Предлагаемый метод основан на использовании модификации решений, проверяемых проверяющей программой. Описывается использование метода для тестирования олимпиадных задач полуфинала чемпионата мира ACM ICPC NEERC 2011 и соревнований Russian Code Cup 2011, при этом в ряде проверяющих программ были обнаружены ошибки.

Ключевые слова: олимпиады по программированию, олимпиадные задачи, проверяющие программы, тестирование.

Введение

Олимпиады по программированию проводятся в большом числе по всему миру. Они способствуют выявлению талантливых программистов среди школьников и студентов. Одной из наиболее значимых является международная студенческая олимпиада по программированию International Collegiate Programming Contest [1], которая проводится Association for Computing Machinery (далее олимпиада будет упоминаться как ACM ICPC). Данная олимпиада обладает обширной сетью отборочных соревнований, в число которых входят полуфинальные соревнования North East European Regional Contest [2] (NEERC), проходящие на территории России. Значимой олимпиадой на территории России также является соревнование Russian Code Cup [3], проводимое корпорацией Mail.ru Group.

На олимпиадах по программированию предлагается решить несколько задач. Формулировка задачи предполагает чтение входных данных, формат которых описан в условии задачи, и получение требуемых выходных данных, формат которых также описан в условии задачи. Решением задачи является программа, написанная на одном из языков программирования (например, в соревновании ACM ICPC используются языки C, C++ и Java [4]) и преобразующая входные данные в выходные.

Для проверки программа запускается на наборе заранее подготовленных тестов, не известных участникам. Решение считается прошедшим определенный тест, если оно, при работе с ним, не нарушило ограничений, указанных в условии задачи, завершилось корректно (без ошибок времени выполнения), а

его ответ признан правильным. Для проверки правильности выходных данных служат проверяющие программы (ПП), которым на вход подаются входные данные теста, выходные данные, полученные запуском решения участника, а также выходные данные, полученные запуском решения жюри олимпиады.

Соревнования на высоком уровне требуют высокого качества подготовленных задач. Для этого собираются интересные идеи для будущих задач, пишутся условия, решения и ПП, а также составляются тестовые наборы. В настоящей работе предлагается новый метод автоматизированного тестирования ПП, позволяющий выявлять ошибки и странности поведения тестируемых программ.

Описание предлагаемого подхода

В работе рассматривается автоматическое тестирование ПП, написанных на языке Java. Тестом для ПП является тройка <входные данные, выходные данные участника, выходные данные жюри>. В качестве входных данных в работе используется подготовленный жюри набор тестов. Выходные данные жюри получаются в результате запуска решения жюри на имеющемся наборе тестов. Предлагается метод автоматической генерации выходных данных участника.

Генерация выходных данных участника

Для генерации большого числа выходных данных участника в работе производится модификация имеющихся решений жюри на языке Java и последующий запуск модифицированных решений на подготовленных тестах. Для изменения решений жюри используется метод внесения ошибки, применяющийся при мутационном тестировании программного обеспечения [5]. Ошибками могут являться использование неверной переменной или константы, отсутствие в коде одной из строк, использование неверного оператора и другие ошибки, часто допускаемые при написании программ. Более подробный список возможных изменений приведен в [6]. Модифицированные решения производят обширный и разнообразный набор выходных данных участника. Полученные данные используются для тестирования ПП.

Запуск модифицированных решений

Запуск модифицированных решений представляет собой отдельную проблему. Полученные программы могут работать продолжительное время, не завершаться или же иметь ошибки времени выполнения. В связи с этим измененные решения запускаются в отдельных виртуальных машинах Java Virtual Machine (JVM). При запуске новой машины указывается время, отведенное на исполнение одной программы. Это время немного превышает указанное в условии задачи ограничение на время работы решения на одном тесте. По истечении отведенного времени JVM прекращает выполнение программы. Производится параллельный запуск модифицированного решения на всех тестах, что позволяет сократить временные затраты.

Анализ поведения ПП

Для проверки правильности исполнения ПП на полученном тестовом наборе производится анализ покрытия кода [7] и результата работы тестируемой программы.

Для анализа покрытия кода в настоящей работе используются две метрики: покрытия строк и покрытия решений. Первая проверяет, что каждая строка исходного кода была выполнена хотя бы раз, вторая – каждое условие в операторах ветвления хоть раз приняло как значение true, так и false. Для сбора статистических данных о покрытии в процессе проверки в исходный код ПП вносятся изменения, не меняющие поведения ПП и отвечающие лишь за сбор информации.

В случае отсутствия полного покрытия кода ПП в одной из метрик стоит уделить больше внимания непокрытым частям. Может оказаться, что данные части исходного кода тестируемой программы излишни или же условия в операторах ветвления составлены ошибочно. Исследования показали, что ряд строк ПП не будут покрыты никогда. Например, строки и условия, отвечающие за корректность выходных данных жюри, будут покрыты не полностью, так как предоставляемые ответы жюри всегда верны и корректны. Для покрытия и этих строк производится запуск ПП с переставленными во входной тройке выходными данными жюри и участника.

Результат работы ПП также представляет интерес. Возможны следующие результаты:

- OK – выходные данные участника правильны;
- WA (Wrong Answer) – выходные данные участника неправильны;
- PE (Presentation Error) – выходные данные участника некорректны с точки зрения формата, описанного в условии;
- FAIL – произошла ошибка выполнения ПП или была найдена некорректность во входных данных или выходных данных жюри.

Для ряда случаев некоторые результаты являются заведомо ошибочными. Прежде всего, результат проверки никогда не должен быть FAIL. При проверке решения жюри следует ожидать только результатов OK. При запуске ПП с измененными местами выходными данными участника и жюри нужно исклю-

чить возможность результата PE, так как ответ жюри корректен с точки зрения условия. В ряде задач ПП производит проверку корректности ответов участника и жюри, после чего сравнивает их, определяя лучший. Для таких задач при «обратном» запуске ПП результат должен быть OK или FAIL. В случае обнаружения неверного результата тестирование прекращается. При этом пользователю становится доступна та тройка <входные данные, выходные данные участника, выходные данные жюри>, на которой и наблюдается ошибочное поведение ПП.

Применение предлагаемого подхода

Для апробации описываемого подхода в условиях реальных олимпиадных задач были выбраны задачи полуфинальных соревнований ACM ICPC NEERC 2010, задачи соревнования Russian Code Cup 2011 и задача quest, использовавшаяся на петрозаводских сборах [8].

Результаты тестирования ПП полуфинала чемпионата мира по программированию NEERC 2010 (табл. 1) показывают, что ПП выполнены на качественном уровне. Разработанный метод достигает полного покрытия исходного кода ПП и не выявляет ошибок в процессе исполнения ПП, а также анализа результата их работы. Заметим, что для некоторых задач тестирование занимает значительное время. Это связано с большим временем исполнения правильного решения, которое составляет для этих задач порядка секунды, а также большим размером тестового набора, составляющего более 70 тестов.

Задача	Число строк в решении	Число единичных модификаций	Число компилирующихся решений	Число правильных решений	Время работы, мин	Число непокрытых строк проверяющей программы
alignment	79	1994	1143 (57%)	335 (17%)	9	0 (0) из 12 (4)
binary	208	5597	3349 (60%)	629 (11%)	8	0 (0) из 7 (2)
cactus	280	7811	4738 (61%)	1340 (17%)	300	0 (0) из 58 (19)
dome	191	3482	2237 (64%)	513 (15%)	75	0 (0) из 25 (7)
evacuation	133	6446	4009 (62%)	1750 (27%)	400	0 (0) из 36 (12)
factorial	156	5770	3623 (63%)	331 (6%)	400	0 (0) из 14 (4)
hands	308	7919	4200 (53%)	757 (10%)	145	0 (0) из 7 (2)
ideal	201	6502	3919 (60%)	2214 (34%)	700	0 (0) из 10 (3)
jungle	161	2991	1610 (54%)	357 (12%)	100	0 (0) из 7 (2)
kgraph	206	2941	1840 (63%)	556 (19%)	200	0 (0) из 27 (7)

Таблица 1. Результаты тестирования ПП соревнования NEERC 2010. В последнем столбце без скобок приведена метрика покрытия строк, в скобках – метрика покрытия решений

Выявленная проблема может иметь несколько решений. Первое состоит в улучшении используемого решения жюри с целью уменьшения времени его работы. Другим решением может являться исключение некоторых тестов из тестового набора задачи, так как обычно при подготовке задачи тестовый набор дополняется случайно сгенерированными тестами, не проверяющими каких-либо крайних случаев поставленной задачи. От части таких тестов часто можно отказаться.

Задача	Число строк в решении	Число единичных модификаций	Число компилирующихся решений	Число правильных решений	Время работы, мин	Число непокрытых строк проверяющей программы
guess	108	4100	2213 (54%)	926 (23%)	10	0 (0) из 53 (15)
birds	93	3791	2679 (71%)	903 (24%)	3	0 (0) из 40 (7)
queuesort	113	4518	2922 (65%)	1114 (25%)	5	0 (0) из 45 (15)
square	105	2192	1214 (55%)	316 (14%)	3	0 (0) из 37 (13)

Таблица 2. Результаты тестирования ПП соревнования Russian Code Cup 2011. В последнем столбце без скобок приведена метрика покрытия строк, в скобках – метрика покрытия решений

Как видно из табл. 2, при тестировании задач соревнования по программированию Russian Code Cup 2011 проблем со временем работы выявлено не было. Тем не менее, были выявлены ошибки в ПП к задачам guess и birds. Выявленные ошибки были устранены до проведения указанного соревнования. Заметим, что не все ошибки были найдены и устранены с первого раза. Повторное тестирование выявляло новые ошибки в ПП задачи birds, которые также были устранены.

Рассмотрим более подробно ошибку, выявленную в ПП задачи birds. Ошибка проявилась при подаче пустого выходного файла на вход ПП. Это приводило к ошибке исполнения из-за попытки преобразовать пустую строку в число. Данная ошибка была устранена с помощью окружения этой строки бло-

ком try-catch, сообщающим о некорректном формате выходного файла в случае возникновения ошибки. Исправленный код ПП приведен на сайте соревнования.

Интересными представляются результаты запуска разработанной программы для автоматизированного тестирования ПП задачи quest (табл. 3). В результате тестирования ПП в ней был выявлен ряд ошибок. Одна из ошибок возникала при генерации сообщения о неправильности ответа участника. Данная ошибка не была ранее обнаружена при использовании задачи на соревнованиях. Ошибка была успешно устранена, и ПП была перетестирована.

Задача	Число строк в решении	Число единичных модификаций	Число компилирующихся решений	Число правильных решений	Время работы, мин	Число непокрытых строк проверяющей программы
quest	337	9652	5911 (61%)	1410 (15%)	50	9 (15) из 280 (92)

Таблица 3. Результаты тестирования ПП задачи quest. В последнем столбце без скобок приведена метрика покрытия строк, в скобках – метрика покрытия решений

Не все строки ПП были покрыты. Интересно отметить, что первые три непокрытых строки соответствуют созданию сообщения об ошибке в случае некорректного входного файла. Так как разработанный метод не производит каких-либо изменений входных файлов, то данные строки не будут покрыты никогда. Проверка корректности входного файла и частей кода, ее проверяющих, остается на совести программиста, занимающегося подготовкой задачи к соревнованию. Остальные непокрытые строки соответствую нарушению формата выходного файла. Такие строки наиболее сложны для покрытия, так как иногда требуют от программы вывода чрезвычайно странных выходных данных.

Также в ПП задачи quest наблюдается большое число непокрытых решений. Девять из них являются операторами if, находящимися непосредственно перед непокрытыми строками. Именно тот факт, что в процессе работы условия операторов всегда не выполнялись, и привел к неполноте покрытия строк исходного кода ПП. Оставшиеся непокрытые решения соответствуют имеющимся в коде конструкциям while-true, условие которых очевидным образом всегда выполняется.

Помимо перечисленных задач, на этапе разработки тестированию были подвергнуты ПП задач из цикла интернет-олимпиад сезона 2010–2011 г.г. [9]. Стоит отметить, что в одной из задач была выявлена часть кода, которая никогда не могла быть исполнена. Данная часть кода была устранена, тем самым был уменьшен объем ПП.

Выводы

В процессе тестирования было обнаружено, что наиболее сложными для покрытия строками исходного кода ПП являются строки, проверяющие некоторые специфические ограничения по формату или содержанию выходного файла решения участника, а также строки, соответствующие генерации сообщения в случае невыполнения этих ограничений. Например, блок кода, проверяющий, что выходной файл не содержит более никакой информации, чаще всего будет являться последней покрытой частью ПП. Стоит отметить, что необходимость именно в таком блоке чаще всего отсутствует, так как при запуске ПП данное условие проверяется системой автоматически.

Результаты показывают, что описанный в работе метод позволяет находить ошибки, не найденные даже во время использования задачи при проведении соревнований. Это свидетельствует об эффективности разработанного метода. Отметим, что тестирование ПП может требовать длительного времени, в связи с этим рекомендуется резервировать значительное время после подготовки олимпиады и до проведения соревнования для тестирования ПП и исправления выявленных ошибок.

Заключение

В работе описан метод, позволяющий автоматически тестировать проверяющие программы олимпиадных задач по программированию с помощью модификации правильных решений. Данный метод был применен для тестирования проверяющих программ реальных олимпиадных задач, где успешно выявил ряд ошибок, которые могли бы существенно повлиять на результаты соревнования. Полученные результаты позволяют утверждать, что описанный метод является достаточно перспективным в плане его применения при подготовке задач по олимпиадному программированию в целях повышения качества олимпиад.

Литература

1. ACM International Collegiate Programming Contest [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/ACM_ICPC, св. Яз. англ. (дата обращения 13.10.2011).

2. North East European Regional Contest [Электронный ресурс]. – Режим доступа: http://neerc.ifmo.ru, св. Яз. англ. (дата обращения 13.10.2011).
3. Russian Code Cup [Электронный ресурс]. – Режим доступа: http://russiancodecup.ru, св. Яз. англ. (дата обращения 13.10.2011).
4. Правила проведения полуфинала NEERC [Электронный ресурс]. – Режим доступа: http://neerc.ifmo.ru/information/contest-rules.html, св. Яз. рус., англ. (дата обращения 13.10.2011).
5. Мутационное тестирование [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/Mutation_testing, св. Яз. англ. (дата обращения 13.10.2011).
6. Ахи А.А. Автоматические методы модификации решений для тестирования проверяющих программ [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/papers/2011-bachelor-akhi/, св. Яз. рус. (дата обращения 13.10.2011).
7. Покрытие кода [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/Code_coverage, св. Яз. англ. (дата обращения 13.10.2011).
8. Сборы команд вузов-участников чемпионата мира по программированию в Петрозаводске [Электронный ресурс]. – Режим доступа: http://karelia.snarknews.info/, св. Яз. рус. (дата обращения 13.10.2011).
9. Интернет-олимпиады по информатике [Электронный ресурс]. – Режим доступа: http://neerc.ifmo.ru/school/io/, св. Яз. рус. (дата обращения 13.10.2011).

Ахи Антон Андреевич – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, студент, anton.akhi@gmail.com

Станкевич Андрей Сергеевич – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, доцент, stankev@rain.ifmo.ru

Шалыто Анатолий Абрамович – Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, доктор технических наук, профессор, зав. кафедрой, shalyto@mail.ifmo.ru

УДК 004.021

ПРИМЕНЕНИЕ ДЕРЕВЬЕВ ДЛЯ РЕАЛИЗАЦИИ МАССОВЫХ ОПЕРАЦИЙ НА МНОГОМЕРНЫХ МАССИВАХ ДАННЫХ

А.Г. Банных

Предлагается метод построения структур данных для выполнения массовых операций на многомерных структурах данных для узкого класса задач. Предлагаемый метод применим в том случае, если элементы многомерной структуры данных принадлежат абелевой группе, и позволяет эффективно выполнять вычисление суммы и прибавление значения к многомерной области.

Ключевые слова: структуры данных, массовые операции, многомерные массивы данных, дерево отрезков, дерево Фенвика.

Введение

Для эффективной работы с информацией были разработаны разнообразные структуры данных. Одна из самых распространенных из них – дерево. Существует огромное множество различных типов деревьев. Эти структуры позволяют собирать статистику на отрезках и изменять отдельные элементы. Представление об этих структурах данных можно получить в классических трудах [1–4].

Практически все эти деревья предназначены для данных, на которых можно ввести линейный порядок. В данной работе рассматриваются многомерные массивы данных. Введение нескольких измерений может быть как естественным (растровые изображения и видео), так и искусственным (базы данных). В последнем случае дополнительные измерения позволяют точнее указывать интересующую выборку информации. Широко известны квадродеревья [5], кд-деревья [6], обобщения дерева отрезков [7] и дерева Фенвика [8]. Существующие структуры данных для работы с многомерными структурами данных позволяют эффективно получать статистическую информацию и изменять отдельные элементы.

Цель настоящей работы заключается в том, чтобы исследовать возможность выполнения массовых обновлений – единообразного изменения целых областей данных. Ни одна вышеперечисленная структура данных в чистом виде не предоставляет подобную функциональность. Приведем простой пример. Пусть дан двумерный массив целых чисел размера $n \times n$. Требуется эффективно выполнять две операции – вычислять сумму на прямоугольнике и прибавлять число к прямоугольнику. Оказывается, что добиться асимптотики $O(\log^2 n)$ для выполнения этих операций – вовсе не тривиальная задача, как может показаться на первый взгляд. Подробнее о возникающих проблемах можно узнать в работе [9].