Утверждение 2. Для области вида $P_{\mathbf{x},\mathbf{n}}$ массовое обновление равносильно единичному обновлению в новом массиве:

$$\text{update}(A, P_{\mathbf{x},\mathbf{n}}, v) \Leftrightarrow \text{update}(C, P_{\mathbf{x},\mathbf{x}}, v \prod_{i=1}^{d} (z_i - x_i)).$$

Ослабленным назовем массовый запрос или обновление, для которого применимо утверждение 1 или 2 соответственно.

Утверждение 3. Произвольные массовые обновления и запросы выражаются не более чем через 2^d ослабленных.

Утверждения 1–3 позволяют свести задачу с массовыми обновлениями к хорошо изученной задаче с единичными обновлениями. Последнюю можно решать с использованием любых существующих структур данных. Это свойство позволяет делать выбор оптимальной структуры данных для каждой задачи отдельно.

При использовании популярных структур данных асимптотическая оценка на выполнение массовых операций составляет $\mathrm{O}(\log^d n)$. При малой размерности пространства эта оценка позволяет говорить об эффективности предлагаемого метода сведения задачи с массовыми обновлениями к задаче с единичными обновлениями. Свобода выбора структуры данных свидетельствует о гибкости метода.

Отметим, что не все задачи выполнения массовых операций на многомерных массивах данных можно сформулировать в терминах, использованных в данной работе. Вопрос об обобщении подхода на более широкие классы задач остаётся открытым.

ЛИТЕРАТУРА

1. *Романовский И. В.* Дискретный анализ. СПб.: Невский Диалект; БХВ-Петербург, 2008. 336 с.
2. *Bentley J. L.* Multidimensional binary search trees used for associative searching // Commun. ACM. 1975. V. 18. No. 9. P. 509–517.
3. *Препарата Ф., Шеймос М.* Вычислительная геометрия. Введение. М.: Мир, 1989. 478 с.
4. *Fenwick P. M.* A New Data Structure for Cumulative Frequency Tables // Software: Practice and Experience. 1994. V. 24. No. 3. P. 327–336.
5. `http://is.ifmo.ru/papers/2011-bachelor-bannykh/` — Банных А. Г. Применение деревьев для реализации массовых операций на многомерных массивах данных [Электронный ресурс]. 2011.

УДК 518.517

АНАЛИЗ АЛГОРИТМОВ ИСПОЛНЯЕМЫХ ФАЙЛОВ ВНУТРИ ПРОГРАММНОЙ ЭВМ С ИЗМЕНЯЕМОЙ СПЕЦИФИКАЦИЕЙ

А. С. Бурлаков

Одним из этапов анализа программного обеспечения является низкоуровневый анализ алгоритмов, скомпилированных в исполняемый файл. Несмотря на то, что данная проблема является довольно популярной, для неё не существует тривиального решения, а имеется набор инструментов и методов [1, с. 3]. Одним из способов решения данной проблемы является использование дизассемблера, чего, однако, часто бывает недостаточно в силу того, что пользователя интересует, каким образом программа взаимодействует с внешними устройствами [2, с. 30–38].

Чтобы решить эту проблему, необходимо запустить исполняемый файл внутри искусственной среды, которая эмулирует работу компьютера вместе с его внешними устройствами и приборами. Таким образом достигается абсолютный контроль над исполняемым приложением. Пользователь может наблюдать за работой не только самой программы, но и среды (внешних устройств), внутри которой она исполняется [3].

В силу того, что программы могут быть скомпилированы для разных типов процессоров и при помощи разных компиляторов, дизассемблеры могут некорректно разбирать скомпилированный файл. Например, выравнивание в файле приложения может быть интерпретировано дизассемблером как операция сложения (в процессорах x86 коду 0x00 соответствует команда `add`). Глядя на дизассемблированный код, однозначно понять, что это действительно сложение или выравнивание, часто бывает проблематично. Чтобы сказать однозначно, необходимо выполнить программу в эмуляторе и проследить за состоянием регистров на проблемном участке [1, с. 4].

Автором разработана система, представляющая собой набор программных средств для динамического анализа кода. В качестве тестового примера выбран компьютер Радио 86РК на базе процессора i8080. Система эмулирует оперативную память и устройства ввода и вывода. Особенностью данной системы является то, что описание машинных команд может меняться в зависимости от типа процессора, это позволяет решить проблему неправильной интерпретации команд при дизассемблировании исполняемого файла программы. Семантика машинных команд описывается на искусственном языке, разработанном автором. При запуске эмулятор читает описание семантики, после чего инициализирует среду для работы с данным типом процессора. Таким образом достигается абстрагирование от аппаратной части, что позволяет эмулятору оперировать более высокоуровневыми структурами.

Язык описания машинных команд напоминает языки ассемблера и C. Чтение семантики происходит в два этапа. На первом этапе участки кода, написанные на языке C, переводятся на язык ассемблера путём разбора выражений. На втором этапе происходит разбор ассемблерного кода в определённые структуры, которые, в свою очередь, используются в ещё более сложных структурах и далее — в эмуляторе.

Кроме функции чтения семантики и эмуляции процессора, система выполняет функцию дизассемблера и анализатора программы. Основываясь на описании машинных команд, система может дизассемблировать исполняемый файл и выполнять пошаговую отладку.

Последняя из них, функция анализа, является первой по значению и самой сложной и интересной из всех представленных (на данный момент находится на этапе разработки). Результатом анализа должно быть представление программы (кода) в наиболее простом для понимания виде. Для этого разработан инструмент, позволяющий строить граф условных переходов.

В докладе излагаются также вопросы, связанные с описанием семантики машинных команд, что включает в себя разбор выражений, и методы динамического анализа приложений, исполняемых внутри искусственной среды — трассировка состояний регистров и построение графа условных переходов.

ЛИТЕРАТУРА

1. *Касперски К.* Образ мышления — дизассемблер IDA. Т. I. М.: СОЛОН-Р, 2001.
2. *Юров В. И.* Ассемблер. Учебник для вузов. 2-е изд. СПб.: Питер, 2010.
3. *Демин А.* Эмулятор Радио 86РК на JavaScript. `http://radio86.googlecode.com/hg/online/radio86.html`, 2009.