E-learning, информационные технологии в образовании

УДК 510.52:372.851

Рублев В.С., Юсуфов М.Т.

Ярославский государственный университет им. П.Г. Демидова, г. Ярославль, Россия

АВТОМАТИЗИРОВАННАЯ ОБУЧАЮЩАЯ СИСТЕМА «АНАЛИЗ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ АЛГОРИТМОВ» (ИССЛЕДОВАНИЯ ОРГАНИЗАЦИИ 1-ОЙ ЧАСТИ ПРОЕКТА)

Аннотация

В [1-2] было исследовано математическое обеспечение для построения автоматизированной обучающей системы (АОС) «Анализ сложности алгоритмов». В настоящей работе исследуются вопросы организации АОС, связанные с характеристиками процесса обучения и обучением разработке таблицы символьной прокрутки алгоритма (первая часть проекта, дающая основные неравенства для оценки вычислительной сложности алгоритма). Основное внимание при этом уделяется алгоритмам контроля корректности символьных преобразований выражений, которые вводятся учащимся в таблице прокрутки.

Ключевые слова

Автоматизированное обучение; анализ сложности алгоритмов; оценка вычислительной сложности алгоритмов; таблица символьной прокрутки алгоритма; нормальная система символьных преобразований.

Rublev V.S., Yusufov M.T.

P.G. Demidov Yaroslavl State University, Yaroslavl, Russia

DEVELOPMENT OF THE FIRST PART OF THE AUTOMATED SYSTEM FOR TEACHING COMPUTATIONAL COMPLEXITY OF ALGORITHMS COURSE

Abstract

Previous articles [1-2] focus on researching the software for developing the automated teaching system (ATS) "Algorithm complexity analysis". This article addresses issues concerning overall learning process features and learning how to produce symbol scroll tables. The main focus of this article is symbol transformation correctness algorithms.

Keywords

Automated learning; algorithm complexity analysis; estimating algorithm computational complexity; algorithm symbol scroll table; normal symbol transformations system.

1. Введение

В [1] была отмечена важность для образования специалиста по компьютерным наукам обучения анализу вычислительной сложности алгоритмов. Введенные принципы адаптивного обучения необходимо подкрепить возможностью подстраиваться к сложности изучения тех или иных тем, а также к уровню каждого обучаемого (настраиваемые характеристики обучения). Также необходимо разбить материал для обучения на маленькие порции, которые мы назовем секциями обучения, и для каждой такой секции нужно составить контрольные тесты или упражнения, выполнение которых будет свидетельствовать об успешности обучения. Если начальная секция должна быть связана с определениями характеристик сложности алгоритма, то последняя секция первой части проекта АОС должна быть связана с итоговым заданием по построению таблицы символьной прокрутки алгоритма.

2. Процесс обучения и его настраиваемые характеристики

С каждой секцией обучения связан контроль усвоения материала секции, который организован для подготовительных секций в виде тестов и упражнений (тестов с открытыми ответами) и в виде итоговых задач. Только после изучения материала *i*-ой секции учащийся может перейти к тестам. Начальное количество $start_i$ тестов (упражнений, задач), которые учащийся должен успешно выполнить, является настраиваемым параметром и может настраиваться индивидуально для каждой секции.

Под тестом предполагается вопрос с несколькими вариантами ответа (пунктами), из которых необходимо выбрать все правильные. Следует понимать, что правильность пункта зависит от формулировки вопроса. Например, для вопроса «Укажите верные высказывания» пункт «2+2=4» будет правильным, а для вопроса «Укажите неверные высказывания» такой же пункт будет неправильным. Только при выборе всех правильных ответов тест считается успешно выполненным. Ответы теста каждый раз выбираются случайным образом из некоторого предопределенного списка ответов теста. При этом правильными ответами могут быть некоторые, все или ни одного. В последнем случае требуется не выбрать ни одного ответа и ответить на вопрос. Если в вопросе требуется полнота и точность ответа, то правильный ответ должен отвечать этому условию, а любой неточный или неполный ответ не является правильным. На рис. 1 демонстрируется часть текста секции.

Рис. 1. Пример текста секции

При успешном выполнении теста количество оставшихся для выполнения тестов уменьшается на 1, и, когда оно становится равным 0, секция считается пройденной, а учащемуся предоставляется материал следующей секции. При неуспешном выполнении теста сообщается об этом одним из предложений системы: «*Не все правильные ответы даны*», «*Среди ответов есть неправильные*», «*Не все ответы правильные и есть неправильные*». После этого предоставляется вновь материал секции с выделением фрагмента (фрагментов), на который сделана ошибка (ошибки) для повторного исправления и в некоторых случаях подсказки. Далее учащемуся даётся ещё одна попытка. Если учащийся допустил ошибку, но исправил ее при повторном тестировании, то этот тест заменяется новым тестом, и, таким образом, количество тестов, которые он должен пройти успешно, остается прежним. Если же повторное исправление оказалось неудачным, то помимо замены теста новым добавляется еще 1 тест. Таким образом, количество тестов, необходимых для успешного усвоения материала секции, может расти. Если при этом это количество достигнет некоторого предельного значения lim_i (еще один настраиваемый параметр), то сеанс работы системы с этим учащимся прекращается. При следующем сеансе работы этого учащегося с системой используемый параметр предельного значения увеличивается на значение add_i (также настраиваемый параметр), но при его достижении учащийся вызывается к преподавателю для сдачи устного зачета по материалу секции.

Если количество оставшихся тестов $numb_j$ учащегося *j* больше начального количества $start_i$, то при каждом успешном выполнении подряд двух тестов количество оставшихся тестов уменьшается еще на 1.

Ниже на рис. 2-5 приведены примеры начальной секции определений сложности алгоритма и тестов к начальной секции определений сложности алгоритма с различными наборами ответов и реакцией системы на них.

Секции, которые обучают элементарным символьным преобразованиям, контролируют усвоение приемов при помощи упражнений. Возможности по заданию начального количества упражнений, предельного количества упражнений при ошибках и других параметров подобны секциям с контролем тестами.

Секции, которые контролируют символьное преобразование, состоящее из совокупности элементарных преобразований, содержат меню элементарных преобразований и кнопку контроля преобразования для выполнения задачи поэтапно. Контрольная задача задается исходным алгебраическим выражением и требованием его преобразования к определенному задачей виду. Учащийся разлагает требуемое преобразование на последовательность элементарных преобразований. Затем для каждого элементарного преобразования в последовательности выделяет сначала в исходном (а далее в текущем) выражении его часть, подлежащую очередному элементарному преобразованию, и указывает это преобразование в меню. После этого он производит элементарное преобразование, редактируя выражение, и нажимает кнопку контроля преобразования. Система сообщает о правильности или ошибочности преобразования, и учащийся может

поправить ошибку. Если ошибок не было, то задача считается успешно выполненной и количество задач для выполнения уменьшается на 1. Если была допущена только одна ошибка, то задача засчитывается, но количество задач для выполнения не изменяется. Если ошибок было несколько, то задача заменяется и добавляется еще одна задача для выполнения.

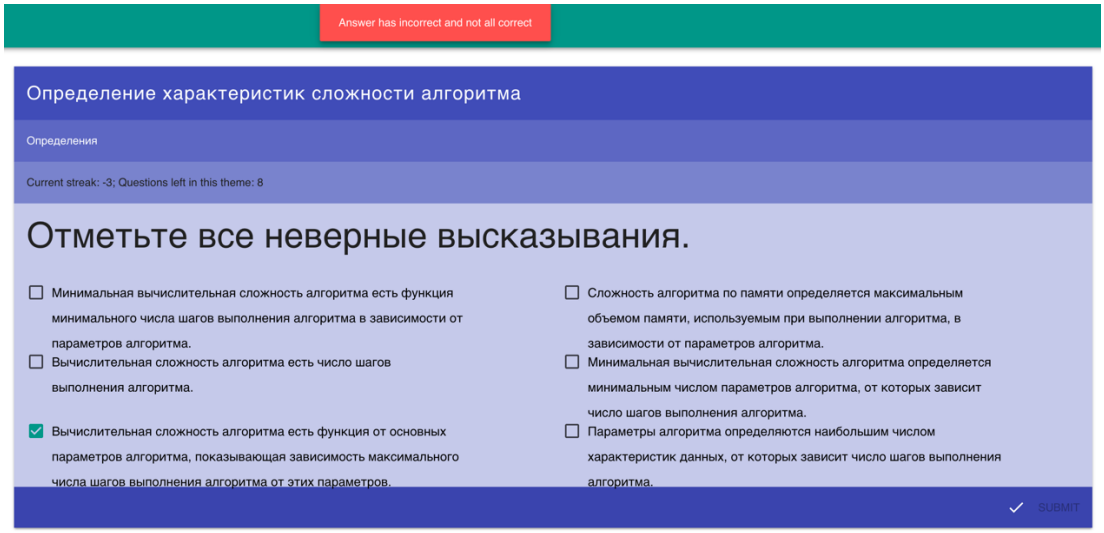

Рис. 2. Реакция системы на ответ не со всеми правильными и некоторыми неправильными пунктами

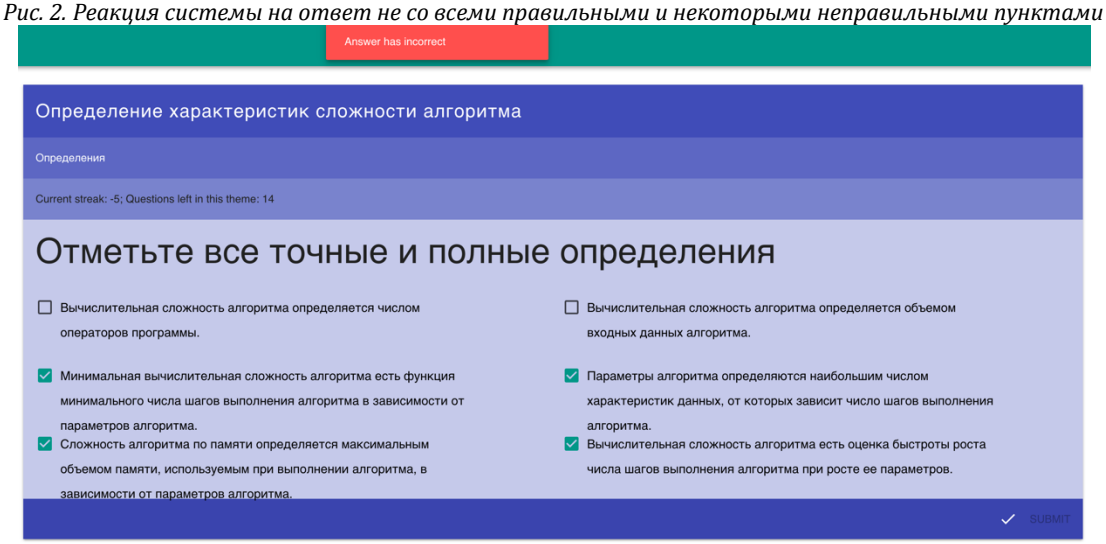

Рис. 3. Реакция системы на ответ с некоторыми неправильными пунктами

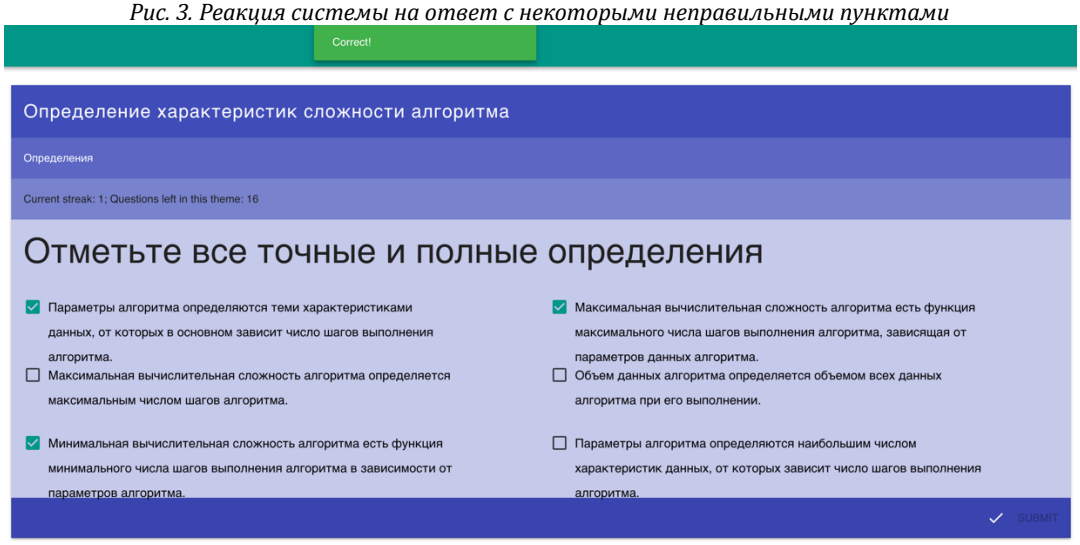

Рис. 4. Реакция системы на правильный ответ

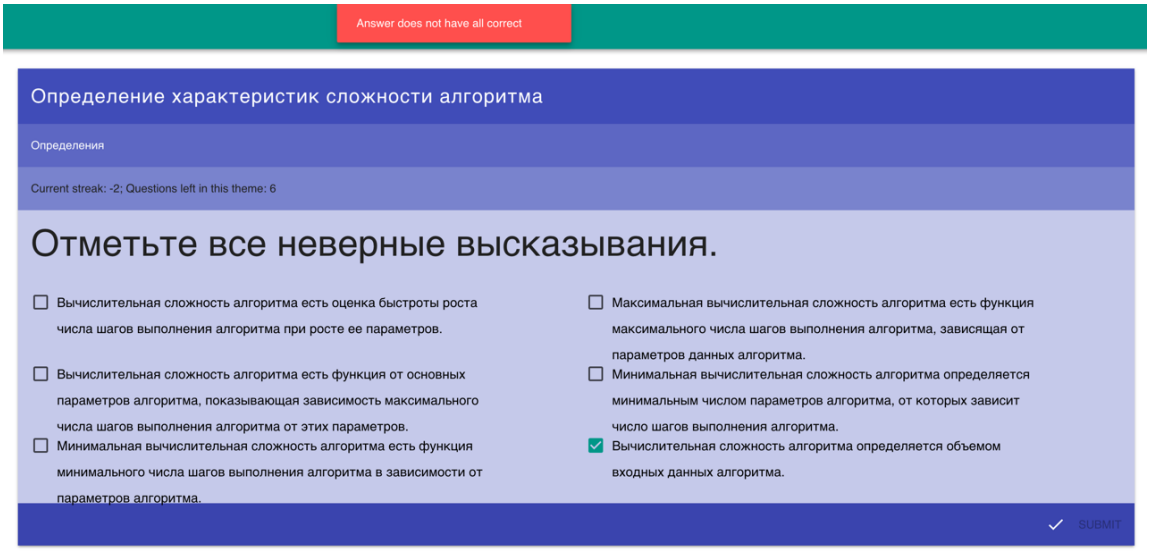

Рис. 5. Реакция системы на ответ не со всеми правильными пунктами

Таким образом, при помощи настраиваемых параметров стратегия обучения осуществляется подобным образом. Таким же образом происходит выполнение итоговой секции по разработке таблицы символьной прокрутки алгоритма.

Пример упражнения к секции по символьным преобразованиям приведен ниже на рис. 6.

3. Разбиение материала на группы секций и первая группа

Весь материал, подлежащий обучению, и контрольные тесты, упражнения, задачи разобьём на секции так, что каждая секция будет представлять совокупность материала (текста для обучения) и контроля (обучения этому материалу).

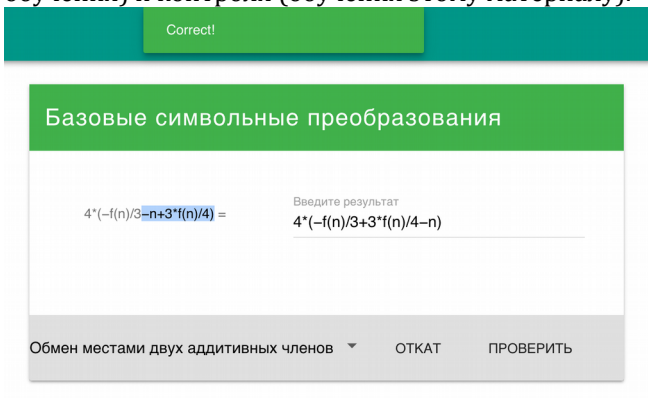

Рис. 6. Пример упражнения для символьного преобразования

Весь материал для обучения 1-ой части проекта АОС можно разделить на 3 группы. В первую группу войдут секции, дающие основные теоретические знания и понятия по определению сложности алгоритма и асимптотическим оценкам вычислительной сложности. Основная цель этих секций – развитие и тренировка памяти обучаемого с использованием навыков запоминания. Контроль этих секций чаще всего производится с помощью тестирования.

Во вторую группу мы включим секции, связанные с обучением символьным преобразованиям, которые потребуются в заключительной третьей группе секций, связанных с разработкой таблицы символьной прокрутки.

Первую группу определим как следующую последовательность секций:

1. Определения характеристик сложности алгоритма (вычислительная сложность и сложность по объему данных) и характеристик данных алгоритма (объем данных, параметры данных).;
2. Определение оценок вычислительной сложности алгоритма (определения асимптотических оценок сверху и снизу для вычислительной сложности, двусторонние оценки скорости роста функции вычислительной сложности, ряд функций для оценивания скорости роста функций вычислительной сложности);
3. Анализ взаимного расположения циклов и оценивание вычислительной сложности алгоритма через вычислительные сложности циклов (невложенные независимые циклы, невложенные зависимые циклы, вложенные независимые циклы, вложенные зависимые циклы, слабая зависимость вложенных циклов).

4. Система нормальных преобразований и вторая группа секций

В процессе разработки таблицы символьной прокрутки алгоритма возникает необходимость в преобразовании символьных алгебраических выражений. Правильность этих преобразований должна контролироваться системой. Но при этом возникает непростая задача в установлении эквивалентности двух выражений: исходного и преобразованного. Для того чтобы сделать этот

контроль более простым, мы ограничиваем систему возможных преобразований, назвав ее системой нормальных преобразований, а сами преобразования элементарными. При этом любое алгебраическое преобразование должно быть достигнуто при помощи некоторой последовательности таких нормальных преобразований. Если каждое нормальное преобразование может быть проконтролировано сравнительно простым алгоритмом контроля, то проблема контроля сложных алгебраических преобразований будет решена.

Любое допустимое выражение в системе состоит из символов данных, символов алгебраических операций сложения (+), вычитания (−), умножения (*), деления (/) и возведения в степень (^), а также скобок, определяющих порядок выполнения операций при помощи общепринятых правил приоритета операций и скобок. Слагаемые сумм выражения, перед которыми нет знака (интерпретируется как знак +), есть знак +, или есть знак –, будем называть аддитивными членами сумм, а операнды операций умножения и деления будем называть мультипликативными членами.

Правильность записи выражений контролируется нажатием кнопки контроля после любого нормального преобразования. В случае некорректности введенного выражения система сообщает об ошибке, и пользователь может вернуться к прежнему выражению, нажав кнопку отката. Данные выражения могут быть символьными переменными (возможно с индексами в квадратных скобках), целочисленными или рациональными константами в виде несокращаемых дробей из числителя и знаменателя, а также именами функций (с аргументами-выражениями в круглых скобках), выбираемыми из меню функций системы. Приведем пример выражения, которое, хотя и не является достаточно общим, сможет нам продемонстрировать некоторые из элементарных нормальных преобразований:

4*(−f(n)/3−n+3*f(n)/4).

Во-первых, любое нормальное преобразование имеет строку аргументов, которая является правильной частью выражения (после ее удаления оставшаяся часть исходного выражения тоже будет правильной). Во-вторых, эта строка аргументов для большинства преобразований содержит ровно 2 аргумента, разделенных знаком операции, который характеризует нормальное преобразование. Первый аргумент может начинаться со знака операции и без него также является правильным выражением, а второй аргумент начинается со знака операции, разделяющей строку аргументов, и без него также является правильным выражением. Так, например, если для вышеуказанного примера применяется нормальное преобразование *обмен местами двух аддитивных членов* и выделена строка

4*(−f(n)/3−n+3*f(n)/4),

то первый аргумент –n, второй аргумент +3*f(n)/4 , и выполнение преобразования приведет к выражению

4*(−f(n)/3+3*f(n)/4−n).

В другом примере применяется нормальное преобразование *обмен местами мультипликативных членов* и выделена строка аргументов

4*(−f(n)/3+3*f(n)/4−n),

где первый аргумент *f(n), а второй - /4, и применение преобразование приведет к выражению:

4*(−f(n)/3+3/4*f(n)−n).

В первом аддитивном члене в скобках знаменатель 3 так не перенести. Поэтому воспользуемся нормальным преобразованием *умножения слева на 1*. Это преобразование имеет 1 аргумент, выделенный в примере:

4*(−f(n)/3+3/4*f(n)−n).

При выполнении этого преобразования получится выражение:

4*(−1*f(n)/3+3/4*f(n)−n).

Тогда при применении нормального преобразования *обмен местами мультипликативных членов* с выделенной строкой аргументов получится выражение:

4*(−1/3*f(n)+3/4*f(n)−n).

При применении нормального преобразования *приведение подобных членов* с выделенной строкой аргументов с выделенной строкой аргументов получится выражение:

4*((−1/3+3/4)*f(n)−n).

При применении нормального преобразования *сложение дробей* с выделенной строкой аргументов с выделенной строкой аргументов получится выражение:

4*((5/12)*f(n)−n).

При применении нормального преобразования *раскрытие скобок* с выделенной строкой аргументов с выделенной строкой аргументов получится выражение:

4*(5/12)*f(n)−4*n,

и после еще одного нормального преобразования *раскрытие скобок* с выделенной строкой аргументов с выделенной строкой аргументов получится выражение:

4*5/12*f(n)−4*n,

которое после нормального преобразования *умножение дробей* с выделенной строкой аргументов с выделенной строкой аргументов даст окончательное выражение:

5/3*f(n)−4*n.

Нормальной формой выражения называется выражение, отвечающее следующим условиям:

1) в выражении отсутствуют скобки (кроме, возможно, функциональных скобок);

2) выражение представляет собой сумму (возможно, из одного слагаемого);
3) числовой коэффициент каждого аддитивного члена, если он есть, стоит первым;
4) все остальные (кроме коэффициента) мультипликативные члены аддитивного слагаемого упорядочены лексикографическим способом;
5) все аддитивные члены также идут в лексикографическом порядке, если не считать их числового коэффициента;
6) нет двух аддитивных членов, которые были бы одинаковы, если не считать коэффициенты.

Так, в примерах, описанных выше, только последнее выражение имеет нормальную форму. Заметим, что нормальная форма выражения однозначна.

Приведем список нормальных преобразований.

1. Обмен местами двух аддитивных членов (a1+a2 -> a2+a1; a1−a2 -> −a2+a1; −a1+a2 ->a2−a1; −a1−a2 -> −a2−a1)
2. Обмен местами двух мультипликативных членов (a1*a2 -> a2*a1; *a1*a2 -> *a2*a1; /a1/a2-> /a2/a1; /a1*a2 -> *a2/a1; *a1/a2 -> /a2*a1)
3. Раскрытие скобок правой суммы у множителя (a0*(a1±…±ak) -> a0*a1±…±a0*ak)
4. Вынесение за скобки общего множителя слева (a0*a1±…±a0*ak -> a0*(a1±…±ak))
5. Раскрытие скобок левой суммы у множителя ((a1±…±ak)*a0 -> a1* a0±…±ak* a0)
6. Вынесение за скобки общего множителя справа (a1*a0±…±ak*a0 -> (a1±…±ak)*a0)
7. Раскрытие скобок произведения у делителя (a0/(a1*…*ak) -> a0/a1/…/ak)
8. Группирование делителей (a0/a1/…/ak -> a0/ (a1*…*ak))
9. Раскрытие скобок произведения у делимого ((a1*…*ak)/a0 -> a1/a0*…*ak/a0)
10. Группирование делимого (a1*/a0…*ak/a0 -> (a1*…*ak)/a0)
11. Умножение двух степеней с одинаковым основанием (a^p1*a^p2 -> a^(p1+p2))
12. Разложение степени на произведение двух степеней с тем же основанием при задании показателя одной из степеней (a^(p) -> a^p1*a^(p-p1))
13. Умножение мультипликативного члена слева на 1 ([±]a -> [±]1*a)
14. Удаление в мультипликативном члене множителя 1 ([±]1*a -> [±]a)
15. Деление двух степеней с одинаковым основанием (a^p1/a^p2 -> a^(p1−p2))
16. Разложение степени на деление двух степеней с тем же основанием при задании показателя одной из степеней (a^(p) -> a^p1/a^(p1-p))
17. Сложение рациональных дробей (результирующая дробь сокращается) (m1/n1+m2/n2 -> m/n; m1+m2/n2 -> m/n; m1/n1+m2 -> m/n)
18. Разложение рациональной дроби на сумму двух рациональных дробей при заданной одной из них (m/n-> m1/n1+m2/n2)
19. Умножение рациональных дробей (результирующая дробь сокращается) (m1/n1*m2/n2 -> m/n)
20. Разложение рациональной дроби на произведение двух рациональных дробей при задании одной из них (m/n –> m1/n1*m2/n2)
21. Деление рациональных дробей (результирующая дробь сокращается) (m1/n1/(m2/n2) -> m/n)
22. Разложение рациональной дроби на деление двух рациональных дробей при задании одной из них (m/n -> m1/n1/(m2/n2))
23. Использование основных тождеств для функций, заранее определённых в интерфейсе системы (например, для функции логарифм: $\log_a b^c \equiv c*\log_a b$; $a^{\log_a b} \equiv b$; $\log_a b \equiv 1/\log_b a$; $\log_c b \equiv \log_a b*\log_c a$)

Для каждого нормального преобразования существует довольно простой алгоритм контроля правильности преобразования. Но нужно убедиться, что система нормальных преобразований полна, то есть можно получить любое преобразование исходного выражения, ограничиваясь только системой нормальных преобразований.

Теорема о полноте системы нормальных преобразований. *Для любого исходного выражения любое эквивалентное ему выражение может быть получено некоторой последовательностью нормальных преобразований.*

Доказательство. Пусть A и B эквивалентные выражения (при любых одинаковых наборах значений исходных данных выражений оба выражения имеют одно и тоже значение). При помощи нормальных преобразований приведем оба выражения к нормальной форме, но так, что при приведении выражения B арифметические операции с числовыми коэффициентами будем делать в последнюю очередь (это возможно, если в отложить такие операции с числовыми коэффициентами и проводить остальные операции). Арифметические выражения с числовыми коэффициентами мы запомним вместе с порядком их выполнения. Тогда, так как выражения A и B эквивалентны, то в результате проведенных преобразований получатся совершенно идентичные выражения. Пусть P_A и P_B соответственно последовательности нормальных преобразований при приведении соответствующих выражений к нормальной

форме. Так как для каждого нормального преобразования существует обратное нормальное преобразование, восстанавливающее предыдущий вид (каждое из нормальных преобразований с четным номером обратно нормальному преобразованию с предыдущим номером, а последнее в списке нормальное преобразование использует тождество, которое также можно обратить), то можно построить последовательность $P^{-1}{}_B$ нормальных преобразований, которая нормальное выражение, полученное из *B,* возвращает в *B.* Поэтому конкатенация списков нормальных преобразований $P_A + P^{-1}{}_B = P_{AB}$ преобразует выражение *A* в выражение *B*, что и требовалось доказать.

Определим теперь список секций второй группы – обучения нормальным преобразованиям.

- Определения выражения и его нормальных преобразований.
- Нормальные преобразования с подсекциями, каждое из которых связано с обучением одному преобразованию.
- Преобразование выражений.

Контроль усвоения материала секции 4 проводится при помощи тестов, а в секции 5 идет при помощи упражнений. Каждая из 11 первых подсекций секции 4 связана с освоением двух нормальных преобразований: прямого и обратного. В подсекции 12 при помощи упражнений осваиваются предопределенные функции преобразований (нормальное преобразование 23). Для освоения материала секции 6 используются задачи по выполнению последовательности преобразований (приведение исходного выражения к нормальной форме).

5. Таблица прокрутки и секции третьей группы

Разработка таблицы прокрутки начинается с анализа текста процедуры программы, который задает алгоритм. При этом заводится таблица «Анализ алгоритма», в которую заносятся:

1) параметры процедуры, которые являются параметрами данных (например, *n*);
2) параметры тела процедуры и их инициализация до выполнения циклов программы (например, *k=n, m=0*);
3) количество циклов программы (например, 2);

   и далее для каждого цикла:
4) параметр цикла (например, *i*);
5) параметр номера выполнения цикла, если он не совпадает с параметром цикла или параметр цикла не задан (например, *i1*);
6) параметр номера последнего выполнения цикла (например, *p1*);
7) условие выполнения цикла (например, *i<n*);
8) функция итерации параметра цикла (например, *i+1* или $i^{1/2}$);
9) изменение других параметров процедуры в теле цикла, если это имеет место (например, *m*(*p1+1*)=(*m*(*p1*)+*k*)/2);

   …

   и далее, если циклов несколько, взаимодействие циклов:
10) вложенность циклов (например, цикл 2 вложен в цикл 1 или циклы не вложены);
11) зависимость циклов (например, цикл 2 зависим от цикла 1 по такому-то параметру или циклы независимы).

Правильность заполнения этой таблицы контролируется системой, для чего соответствующие данные хранятся в системе вместе с процедурой алгоритма.

После составления таблицы «Анализ алгоритма» следует перейти к разработке таблицы символьной прокрутки алгоритма. Сначала заводятся столбцы таблицы:

1. Если количество циклов более одного, то следует завести столбец с номером цикла №ц;
2. Помимо столбцов с именами параметров циклов следует завести столбцы с параметрами номера выполнения цикла, если они не совпадают с именами параметров цикла;
3. Для каждого параметра алгоритма, не являющегося параметром данных процедуры алгоритма, следует завести отдельный столбец, если значения такого параметра влияет на количество выполнения того или иного цикла;
4. Следует завести обязательный столбец *Условие цикла*, в котором будет записываться выражение условия продолжения цикла, или условие последнего выполнения цикла, или условие выхода из цикла.

Теперь можно перейти к пошаговому выполнению алгоритма процедуры и одновременному заполнению таблицы символьной прокрутки. При этом надо учесть следующие особенности, связанные с шагом выполнения цикла:

1. Выполнение начинается с определения начального или следующего значения параметра цикла, при котором условие цикла будет проверяться. В случае цикла **for** это значение определяется выражением шага параметра, а в случае цикла **while** это значение может быть изменено префиксной операцией выражения условия (например, преинкрементная или предекрементная операции). При проверке условия выполнения цикла берется именно это значение;

2. После проверки условия выполнения цикла, если оно выполнено, то параметры цикла могут измениться в теле цикла или в постфиксной операции выражения условия (например, постинкрементная или постдекрементная операции). Это изменение отмечается в следующей строке таблицы символьной прокрутки.

Процесс занесения в ячейку символьного выражения начинается с пометки ячейки. Затем в отдельной строке преобразования выражения оно заносится и при необходимости преобразуется с помощью одного или нескольких нормальных преобразований. После нажатия кнопки *Конец преобразования* оно переносится в ячейку. Условие преобразования должно заноситься в ячейку строки последним, когда все другие выражения ячеек этой строки определены. Разработка таблицы символьной прокрутки заканчивается, когда для всех циклов таблицы заполнены ячейки *Условие выхода*.

На рис. 7 изображена часть упражнения на составление таблицы символьной прокрутки для следующего алгоритма:

```
void f2 (unsigned long n) {
  float x = n, z = n;
  while (x > 2) {
    x = sqrt(x);
    z = z * z;
  }
  while (z /= 2 > 1);
}
```

Определим теперь список секций третьей группы – обучения разработке таблицы символьной прокрутки.

1. Определения анализа алгоритма и таблицы символьной прокрутки.
2. Анализ алгоритма процедуры.
3. Разработка таблицы символьной прокрутки алгоритма процедуры.

Секция 7 требует контроля тестами правильности усвоения материала по анализу алгоритма и разработке таблицы символьной прокрутки. Секция 8 контролируется поэтапным выполнением задачи анализа алгоритма. Секция 9 является итоговой для первой части разработки АОС и сначала в подсекции идет с помощью упражнений обучение разработке отдельных частей таблицы, а затем основная подсекция организует контроль выполнения итоговой задачи по разработке символьной таблицы прокрутки алгоритма.

6. Заключение

Описанный подход обучения анализу сложности алгоритмов позволил разработать

- методическое обеспечение первой части АОС, выделяющей для каждой секции и подсекции материал обучения и контрольные тесты, упражнения, задачи, необходимые для взаимодействия учащегося с материалом в процессе обучения и контроля усвоения материала;
- программное обеспечение, позволяющее через интерфейс выполнять все этапы обучения.

Выразим надежду, что последующая опытная эксплуатация АОС «Анализ вычислительной сложности алгоритмов» будет успешна и позволит перейти к разработке второй части АОС.

Nц	i1	i2	x	z	условие цикла	
	p1				n^(1/2)^(p1-1) > 2	посл. вып.
			n^(1/2)^p1	n^2^p1		
	p1 + 1				n^(1/2)^p1 > 2	выход
2		1		n^2^p1 / 2	n^2^p1 / 2 > 1	
		2		n^2^p1 / 2 / 2 = n^2^p1 / 2^2		

Рис. 7. Заполнение таблицы символьной прокрутки

Литература

1. Рублев В.С., Юсуфов М.Т. Автоматизированная система для обучения анализу вычислительной сложности алгоритмов / В.С. Рублев, М.Т. Юсуфов // Современные информационные технологии и ИТ-образование. - 2016. - Т.12 (№ 1). - С.135-145.
2. Rublev V.S., Yusufov M.T. Automated system for teaching computational complexity of algorithms course // Selected Papers of the First International Scientific Conference Convergent Cognitive. - Moscow, Russia, November 25-26, 2016. URL: http://ceur-ws.org/Vol-1763/ (ISSN 1613-0073 VOL-1763 urn.nbn.de: 0074-1763-4 indexed Scopus).

References

1. Rublev V.S., Jusufov M.T. Avtomatizirovannaja sistema dlja obuchenija analizu vychislitel'noj slozhnosti algoritmov / V.S. Rublev, M.T. Jusufov // Sovremennye informacionnye tehnologii i IT-obrazovanie. - 2016. - T.12 (№ 1). - S.135-145.
2. Rublev V.S., Yusufov M.T. Automated system for teaching computational complexity of algorithms course // Selected Papers of the First International Scientific Conference Convergent Cognitive. - Moscow, Russia, November 25-26, 2016. URL: http://ceur-ws.org/Vol-1763/ (ISSN 1613-0073 VOL-1763 urn.nbn.de: 0074-1763-4 indexed Scopus).

Поступила: 15.06.2017

Об авторах:

Рублев Вадим Сергеевич, кандидат физико-математических наук, профессор кафедры теоретической информатики, Ярославский государственный университет им. П.Г. Демидова, roublev@mail.ru;

Юсуфов Мурад Теймурович, аспирант кафедры теоретической информатики, Ярославский государственный университет им. П.Г. Демидова, flood4life@gmail.com.

Note on the authors:

Rublev Vadim S., Candidate of Physical and Mathematical Sciences, Professor of the Department of Theoretical Informatics, P.G. Demidov Yaroslavl State University, roublev@mail.ru;

Yusufov Murad T., Post-graduate student of the Department of Theoretical Informatics, P.G. Demidov Yaroslavl State University, flood4life@gmail.com.