

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ГЕНЕРАЦИИ ТЕСТОВЫХ ЗАДАНИЙ ДЛЯ ОЦЕНКИ ЗНАНИЙ СТУДЕНТОВ ПО ВЫСШЕЙ МАТЕМАТИКЕ

***Черепанов Алексей Александрович,
Сутырина Ирина Алексеевна***

Аспирант, бакалавр

РЭУ имени Г. В. Плеханова, Москва, Россия

E-mail: aleksey.4erepanov@gmail.com, ira.makarena@yandex.ru

Для облегчения труда преподавателей при подготовке тестирования студентов по темам «нахождение предела», «нахождение интеграла» и «нахождение производной функции» разработано программное обеспечение для создания наборов вариантов тестовых заданий с выбором одного правильного ответа. Сложность заданий можно задавать. Задания в наборе вариантов обладают очень близкой сложностью, чтобы давать их студентам из одной группы.

Программное обеспечение использует генерацию случайных математических выражений для получения первого задания в наборе. Это позволяет создавать более разнообразные задачи[1]. Выражение строится, как дерево, состоящее из функций, операций, переменных и констант. Их наборы различаются по темам задач.

Дерево строится рекурсивным методом объекта, описывающего состояние построения — набор счётчиков. Метод получает указатель на узел с незаполненными ветвями, создаёт и прикрепляет новые узлы, изменяет счётчики, вызывает себя для заполнения ветвей новых узлов. Построение начинается с фиктивного корневого узла с одной ветвью. В процессе построения дерева узлы выбираются случайно из заданных наборов с соблюдением следующих ограничений. Подсчитываются незаполненные ветки дерева, чтобы выражение в итоге имело заданный размер. Подсчитывается количество переменных, чтобы была хотя бы одна переменная: если переменных нет и осталось место только для одного узла дерева, то переменная будет там. Если поддерево является аргументом тригонометрической функции, то нигде в нём не будет тригонометрических функций, но обязательно будет либо π , либо переменная.

Пользователь может задать желаемое количество узлов дерева, то есть размер выражения. Сложность задачи примерно соответствует размеру выражения. Чтобы сложность меньше различалась внутри одного набора, набор расширяется до заданного размера задачами, полученными небольшими изменениями из задач, имеющих-

ся в наборе. Способ изменения выбирается случайно. Для генерации задач реализовано два способа изменения: замена тригонометрической функции на другую тригонометрическую функцию и замена константы или переменной на случайную бинарную операцию с ней и случайной константой. Если выбранный способ не дал новой задачи, то процедура повторяется. Замены производятся на уровне строк при помощи регулярных выражений.

В процессе генерации задач программное обеспечение пробует решить созданные задачи, используя библиотеку SymPy[2] для символьных вычислений. Задачи, которые не могут быть решены, отбрасываются, и генерация повторяется. Это позволяет создавать относительно большие задачи, которые можно решить. Однако время создания одной задачи может достигать нескольких минут.

Ответ, полученный в процессе решения, сохраняется как правильный вариант. Неправильные варианты ответа выбираются из списка универсальных правдоподобных ответов (например, 0 , 1 , $-\infty$ и $+\infty$ для задачи нахождения предела) или получаются небольшими изменениями из имеющихся вариантов.

При таком подходе добавление новой темы для генерации задач, строящихся вокруг формул, сводится к определению наборов функций, операций, переменных и констант для использования в выражениях, определению списка правдоподобных ответов (может быть пустым) и написанию кода для решения задачи (использование библиотеки SymPy может сильно упростить это).

Ключевой особенностью разработанного программного обеспечения является сочетание случайной генерации математических выражений со способами их изменения, которые неявно превращают каждое выражение в шаблон для генерации.

Литература

1. Посов И. А. Обзор генераторов и методов генерации учебных заданий // ОТО. 2014. №4. URL: <http://cyberleninka.ru/article/n/obzor-generatorov-i-metodov-generatsii-uchebnyh-zadaniy> (дата обращения: 24.02.2018).
2. Meurer A., Smith C.P., Paprocki M., Čertík O., Kirpichev S.B., Rocklin M., Kumar A., Ivanov S., Moore J.K., Singh S., Vig S., Rathnayake T., Granger B.E., Muller R.P., Bonazzi F., Gupta H., Vats S., Johansson F., Pedregosa F., Curry M.J., Terrel A.R., Roučka Š., Saboo A., Fernando L., Kulal S., et al. SymPy: symbolic computing in Python // PeerJ Computer Science, 2017, 3:e103