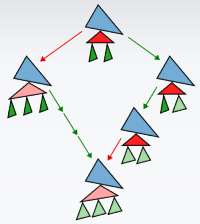
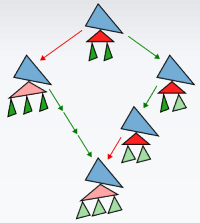


Краткое резюме задачи

Каждая задача состоит из двух частей: конвертация и тестирование.

- (0,1,3,9) Приведение shuffle-регулярных выражений к академическим.
- (2,4,5) Оптимизация неэффективностей в академических регулярных выражениях.
- (6,7,8) Приведение lookahead-регулярных выражений к академическим.

Тестирование основано на принципе фаззинга: порождения случайных выражений и строк, которые должны распознаваться результатом преобразования точно так же, как и исходным выражением. В п.2 фазз-тестирование дополнительно использует принцип «malicious pump» — злонамеренной накачки.

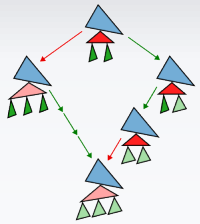


Fuzz-модуль

В fuzz-модуле должны быть следующие составляющие:

- Генератор регулярок
- Генератор строк
- Сравнительный парсинг строки по регуляркам

Генератор строк использует автоматное представление выражения — поэтому даже вариантам 2,4,5 без построения автомата по регулярке не обойтись. У остальных эта опция и так является частью задачи.
В последнем пункте предпочтительно пользоваться библиотеками регулярных выражений. У варианта 0,1,3,9 для регулярок с shuffle-операцией такой возможности не будет, поэтому придётся написать парсер по автомату Брзозовски (он детерминированный, поэтому парсер будет очень простой).

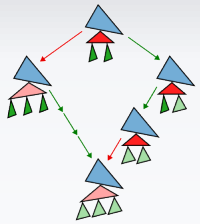


Рандомизация регулярок

Следующие параметры должны быть вынесены в макросы или ключи:

1. размер алфавита (можно предполагать, что он не больше 5 — т.к. автоматы для больших алфавитов превратятся в «ёжики» без факторизации по классам букв, а в **этом году** задача не про неё);
2. звёздная высота (число вложенных квантификаторов итераций);
3. (для вариантов 6,7,8) число lookahead-ов;
4. максимальное число букв в регулярке.

Предполагаем для простоты, что выражения $(a|)$ (т.е. такие, у которых аргументом альтернативы является пустое выражение) невозможны, как невозможны и выражения вида $()^*$ (итерация над пустым выражением). Дальше на каждом шаге генерации можно рандомно выбирать очередной применяемый оператор (если в запасе есть хотя бы 2 буквы) либо константу, либо итерацию, а затем рекурсивно делать генерацию по его аргументам, учитывая, что запас свободных букв при генерации очередной бинарной операции заведомо уменьшается хотя бы на 1.



Генератор строк

Случайные строки с вероятностью почти 1 интереса для фазз-тестирования не представляют. Обычно такие генераторы используют генетические алгоритмы, порождающие «правдоподобные» вводы, но мы ограничимся комбинаторным подходом.

- Строится матрица достижимости в автомате, построенном на базе регулярного выражения.
- В рамках отношения достижимости выбирается случайная последовательность состояний $\{q_i\}$ таких, что q_{i+1} достижимо из q_i, q_0 — стартовое, последнее состояние — финальное.
- Посредством, например, BFS строятся слова на путях из q_i в q_{i+1}. Если автомат недетерминированный, рекомендуется использовать недетерминированные структуры данных.
- К полученным отрезкам применяются случайные мутации: перестановка букв, перестановка фрагментов, повторение букв, повторение фрагментов, удаление букв, удаление фрагментов (либо не применяется никаких).