Описание задания

Необходимо реализовать 2 из 3 алгоритмов:

- Алгоритм Кока-Янгера-Касами (3 балла из 10)
- Алгоритм Эрли (5 баллов из 10; за реализацию, работающую в худшем случае дольше, чем за кубическое время (в предположении, что размер грамматики - константа) 2 балла из 10)
- Одна из версий алгоритма LR: LR(0) (2 балла из 10), LR(1) (5 баллов из 10) или LR(k) (6 баллов из 10).

В зачёт суммируются только два максимальных алгоритма!

Необходимо реализовать алгоритм в виде класса, в котором есть следующие методы:

- `fit(G: Grammar) -> Algo`: - препроцессинг (в алгоритме Эрли можно просто запоминать грамматику). В случае $LR(1)$ ($LR(k)$, $LR(0)$) если грамматика не является $LR(1)$ ($LR(k)$, $LR(0)$) грамматикой, должно бросаться исключение.
- `predict(word: String) -> Boolean` - проверка принадлежности слова языку.

Нужно реализовать алгоритмы в виде классов и написать программу, которая считывает грамматику, строит по ней анализатор, считывает слова, проверяет, лежат ли они в языке, и выводит для каждого слова, лежит ли оно в языке. Если входные данные не соответствуют формату входных данных или не задают КС грамматику из того класса, с которым умеет работать алгоритм, программа должна бросать исключение.

Подумайте, каким образом выводить все стадии алгоритма. Для этого необходимо использовать паттерны программирования. При этом необходимо отделить код ввода и вывода грамматики от алгоритма парсинга

Очень рекомендуется реализовать тестирование алгоритма. Если вы пишите тесты, добавляйте их в репозиторий. Тестирование не будет оцениваться, но будет проверяться, что алгоритм работает корректно. Если проверяющий сможет придумать корректные входные данные, на которых ваша программа выдаёт неправильный ответ, то даже после исправления ошибки вы получите за задание мало баллов (насколько мало зависит от того, насколько существенна ошибка, но точно не больше половины). Необходимо продумать, как можно задавать грамматику посредством исходного кода: это необходимо для Unit-тестирования. Код, в котором входные данные принимаются данные для тестов через stdin, будет отправляться на доработку.

Формат сдачи заданий

В коде репозитория необходимо создать файл README.md, в котором должны быть инструкции:

- по сборке проекта;
- по запуску кода в режиме тестирования кода преподавателем (см. ниже)
- по запуску автоматизированных тестов, написанных Вами.

Формат входных данных для тестирования кода

В первой строке содержатся 3 целых числа $|N|$, $|\Sigma|$ и $|P|$ — количество нетерминальных символов, терминальных символов и правил в порождающей грамматике. Все числа неотрицательные и не превосходят 100.

Во второй строке содержатся $|N|$ нетерминальных символов. Нетерминальные символы являются заглавными латинскими буквами.

В третьей строке содержатся $|\Sigma|$ символов алфавита. Символы являются строчными латинскими буквами, цифрами, скобками или знаками арифметических операций.

В каждой из следующих P строк записано одно правило грамматики в формате левая часть правила -> правая части правила. ε в правой части правила обозначается отсутствием правой части (концом строки после `->`).

Следующая строка состоит из одного нетерминального символа — стартового символа грамматики.

Следующая строка состоит из одного целого числа $m, 1 \leq m \leq 100000$ — количества слов, принадлежности которых языку надо проверить.

В каждой из следующих m строк содержится одно слово, состоящее из символов алфавита грамматики, принадлежность которого языку надо проверить. Суммарная длина всех слов не превосходит 10000000.

Формат выходных данных для тестирования кода

Выведите m строк. i-ая строка должна содержать слово "Yes" (без кавычек), если i-ое слово из входных данных лежит в языке, пораждаемом грамматикой, и "No" (без кавычек) иначе.

- Если сомневаетесь в архитектуре - нарисуйте UML-диаграммы и поймите, что в ней может быть не так. Можно будет задавать вопросы ассистентам и семинаристам по архитектуре - только подготовьте UML-диаграммы