Разработать библиотеку для работы с регулярными выражениями. В качестве алфавита могут использовать любые печатные символы, метасимволы экранируются символом ‘\’.

Кроме этого, должен поддерживаться символ, определяющий пустую подстроку ‘#’

Регулярный выражения должны поддерживать следующие операции:

Операция ‘или’: r1|r2 (метасимвол ‘|’)

Операция ‘конкатенация’: r1r2

Операция ‘замыкание Клини’: r\* (метасимвол ‘\*’)

Операция ‘символ из набора’: [a1a2a3…] (метасимвол ‘[]’; допустимо использование пустого набора, аналог #)

●Операция ‘повтор выражения’: r{x} (метасимвол ‘{х}’, где x – количество повторов)

Операция ‘нумерованная группа захвата’: (r) (метасимвол ‘()’), выражения из групп захвата нумеруются в порядке их следования.

Операция ‘выражение из нумерованной группы захвата’: \n (метасимвол ‘\’, n – номер группы захвата)

В регулярных выражениях могут использоваться операторные скобки (r), определяющие приоритет операторов. (метасимвол ‘( )’)

Библиотека должна поддерживать следующие операции:

findall – поиск всех непересекающихся вхождений подстрок в строку соответствующих регулярному выражению ~~с перегрузкой без доступа и с доступом к группам захвата через специальный объект (через итератор и оператор индексации).~~

Регулярные выражения могут быть заранее скомпилированы в ДКА непосредственно без построения НКА (РВ->ДКА->минимальный ДКА), т.е. должен поддерживаться метод compile(), при этом допустим вызов операций с регулярными выражениями, как со скомпилированным объектом, так и со строковым представлением.

Для скомпилированных автоматов библиотека должна поддерживать операцию восстановления регулярного выражения методом построения K-пути.

Библиотека должна поддерживать операцию построения инверсии языка, задаваемым регулярным выражением (скомпилированным ДКА).

Библиотека должна поддерживать операцию построения дополнения к языку, задаваемому регулярным выражением (скомпилированным ДКА).

1. ~~Минимизация ДКА~~
2. ~~Compile+класс DFA~~
3. К-путь
4. ~~Findall~~
5. Инверсия и дополнение
6. Исправить оператор повтора, раскрытие нодов повтора
7. ~~Исправить операцию символ из набора~~
8. Сделать более красивую нумерацию при построении СД