Федеральное государственное бюджетное образовательное учреждение
высшего образования «Московский государственный технический университет
имени Н. Э. Баумана» (национальный исследовательский университет)

Дисциплина: «Анализ алгоритмов»
Отчет по рубежному контролю №2

Тема работы:
«Конечные автоматы.
Регулярные выражения»

Студент: Левушкин И. К.
Группа: ИУ7-52Б
Преподаватели: Волкова Л. Л.,
Строганов Ю. В.

Москва, 2019 г.

Содержание

Введение

Цель лабораторной работы: При помощи конечных автоматов и регулярных выражений написать программу, находящую группы факультетов ИУ, ИБМ и Э в тексте.

Задачи работы:

1) изучить работу регулярных выражений и конечных автоматов;

2) создать конечный автомат;

3) реализовать поставленную задачу с использованием конечного автомата;

4) реализовать поставленную задачу с использованием регулярные выражения;

5) сравнить время выполнения программы, использующую конечный автомат, и программу, использующую регулярные выражения;

6) описать и обосновать полученные результаты в отчете о рубежном контроле работе, выполненного как расчётно-пояснительная записка.

1 Аналитический раздел

1.1 Конечные автоматы

Конечный автомат (или попросту FSM — Finite-state machine) это модель вычислений, основанная на гипотетической машине состояний. В один момент времени только одно состояние может быть активным. Следовательно, для выполнения каких-либо действий машина должна менять свое состояние. Конечные автоматы обычно используются для организации и представления потока выполнения чего-либо. Конечный автомат можно представить в виде графа, вершины которого являются состояниями, а ребра — переходы между ними. Каждое ребро имеет метку, информирующую о том, когда должен произойти переход.

1.2 Регурярные выражения

Регулярные выражения – это алгебраический способ описания регулярных языков, которые задают конечные автоматы. Алгебраическими регулярными операторами являются: объединение, конкатенация (“точка”) и итерация (“звездочка”).

Регулярные выражения подчиняются многим алгебраическим законам арифметики, хотя есть и различия. Объединение и конкатенация ассоциативны, но только объединение коммутативно. Конкатенация дистрибутивна относительно объединения. Объединение идемпотентно.

2 Конструкторский раздел

В разделе представлен конечный автомат для реализации задачи нахождения всех групп факультетов ИУ, ИБМ и Э в тексте.

На рисунке 1 приведен конечный автомат для поставленной задачи.

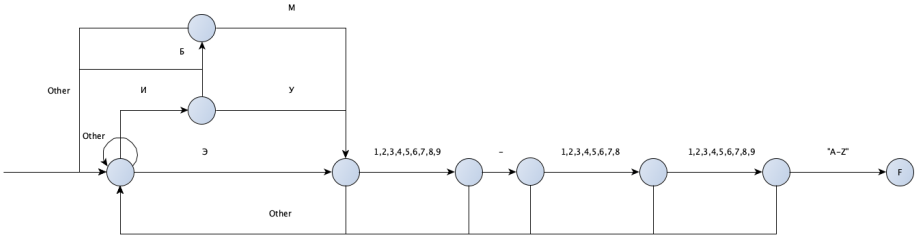

Рис. 1: Конечный автомат для определения группы факультетов ИУ, ИБМ, Э

3 Технологический раздел

Здесь описываются требования к программному обеспечению и средства реализации, приводятся листинги программы и тестовые данные.

3.1 Требования к программному обеспечению

Входные данные:

- строка символов.

Выходные данные: словарь, хранящий в качестве ключа позицию группы в данном тексте, а в качестве значения - саму группу.

3.2 Средства реализации

Программа написана на языке Python [1], который предоставляет программисту мощные инструменты для реализации различных алгоритмов и является достаточно надежным, эффективным и удобным для реализации сложных алгоритмов. Для написания использовался редактор исходного кода *PyCharm* [2].

Замер времени выполнения программы производится с помощью функции *process_time()* из библиотеки *time*, функционал которой позволяет подсчитывать процессорное время в тиках, а затем конвертировать полученный результат в секунды.

3.3 Листинг программы

Реализованная программа представлена в листингах 1-2.

Листинг 1: Реализация программы с использованием регулярных выражений

```
import re

def regular(line):
  result = re.findall(r'(?:IU|IBM|E)[1-9]-[1-8][1-9][A-Z]', line)
  return result
```

Листинг 2: Реализация программы с применением конечного автомата

```
def machine(line):
  result = {}
  faculty = ''
  state = 0
  state_3_choice = [1,2,3,4,5,6,7,8,9]
  state_5_choice = [1,2,3,4,5,6,7,8]
  state_6_choice = [1,2,3,4,5,6,7,8,9]
  state_7_choice = [A-Z]
```

```
  for i in range(len(line)):
    if (state == 0):
      if (line[i] == 'I'):
        state = 1
        faculty += line[i]
      elif (line[i] == 'E'):
        state = 3
        faculty += line[i]
      else:
        continue
    elif (state == 1):
      if (line[i] == 'B'):
        state = 2
        faculty += line[i]
      elif (line[i] == 'U'):
        state = 3
        faculty += line[i]
      else:
        faculty = ''
        state = 0
        i -= 1
    elif (state == 2):
      if (line[i] == 'M'):
        state = 3
        faculty += line[i]
      else:
        faculty = ''
        state = 0
        i -= 1
    elif (state == 3):
      if (int(line[i]) in state_3_choice):
        state = 4
        faculty += line[i]
      else:
        faculty = ''
        state = 0
        i -= 1
    elif (state == 4):
      if (line[i] == '-'):
        faculty += line[i]
        state = 5
      else:
        faculty = ''
        state = 0
        i -= 1
    elif (state == 5):
      if (int(line[i]) in state_5_choice):
        state = 6
        faculty += line[i]
      else:
        faculty = ''
        state = 0
        i -= 1
    elif (state == 6):
      if (int(line[i]) in state_6_choice):
```

```
64          state = 7
65          faculty += line[i]
66        else:
67          faculty = ''
68          state = 0
69          i -= 1
70      elif (state == 7):
71      if (line[i] in state_7_choice):
72        faculty += line[i]
73        result.update({i - len(faculty) + 1: faculty})
74        state = 0
75        faculty = ''
76      else:
77        faculty = ''
78        state = 0
79        i -= 1
80    return result
```

4 Исследовательский раздел

В разделе представлены примеры выполнения программы, результаты сравнения времени выполнения программ, реализованных на конечных автоматах и с использованием регулярных выражений.

4.1 Примеры работы

На рис. 2-6 приведены примеры работы программы.

```
Введите строку: ФИУ7-52Б
Выберете способ нахождения факультетов:
    1)Регулярные выражения
    2)Конечные автоматы
1
['ИУ7-52Б']
Время выполнения в секундах: 0.00079499999999971 seconds
```

Рис. 2: Пример работы регулярных выражений

```
Введите строку: ИУ7-52БКАРБЕЛЭНЭЗ-83Е
Выберете способ нахождения факультетов:
    1)Регулярные выражения
    2)Конечные автоматы
1
['ИУ7-52Б', 'ЭЗ-83Е']
Время выполнения в секундах: 0.00094400000000004 seconds
```

Рис. 3: Пример работы регулярных выражений

```
Введите строку: 723ИБМ5-32ЦФЭ7-93БЭ9-65У
Выберете способ нахождения факультетов:
    1)Регулярные выражения
    2)Конечные автоматы
1
['ИБМ5-32Ц', 'Э9-65У']
Время выполнения в секундах: 0.00051099999999976 seconds
```

Рис. 4: Пример работы регулярных выражений

```
Введите строку: 723ИБМ5-32ЦФЭ7-93БЭ9-65У
Выберете способ нахождения факультетов:
    1)Регулярные выражения
    2)Конечные автоматы
2
{3: 'ИБМ5-32Ц', 18: 'Э9-65У'}
Время выполнения в секундах: 5.2999999999997494e-05 second
```

Рис. 5: Пример работы конечного автомата

```
Введите строку: ФУИУ7-52Б
Выберете способ нахождения факультетов:
    1)Регулярные выражения
    2)Конечные автоматы
2
{2: 'ИУ7-52Б'}
Время выполнения в секундах: 3.899999999999737e-05 seconds
```

Рис. 6: Пример работы конечного автомата

4.2 Постановка эксперимента

1. Сравнить время работы алгоритма, разработанного на основе конечного автомата и алгоритма с использованием регулярных выражений на строке длинной 10-100 символов с шагом в 10 символов.

4.3 Тестовые данные

Строки будут состоять из следующих символов:

1. $str10$: *уюмЭ7-52Бв*;
2. $str20$: *уюмЭ7-52БИУ3-21Ее4К6*;
3. $str30$: *уюмЭ7-52БИУ3-21Ее4К6Э5-32Кптры*;
4. $str40$: *уюмЭ7-52БИУ3-21Ее4К6Э5-32КптрыИБМ4-19Цеу*;

5. $str50$: *уюмЭ7-52БИУ3-21Ее4К6Э5-32КптрыИБМ4-19ЦепуЭ5-32К*;
6. $str60$: *уюмЭ7-52БИУ3-21Ее4К6Э5-32КптрыИБМ4-19ЦепуЭ5-32КИУ9-61У*;
7. $str70$: *уюмЭ7-52БИУ3-21Ее4К6Э5-32КптрыИБМ4-19ЦепуЭ5-32КИУ9-61УИБМ4-19Ц*;
8. $str80$: *уюмЭ7-52БИУ3-21Ее4К6Э5-32КптрыИБМ4-19ЦепуЭ5-32КИУ9-61УИБМ4-19ЦИБМ4-19Цапывы*;
9. $str90$: *уюмЭ7-52БИУ3-21Ее4К6Э5-32КптрыИБМ4-19ЦепуЭ5-32КИУ9-61УИБМ4-19ЦИБМ4-19ЦИУ9-61УЭ5-32КИУ9-61Уиа*;
10. $str100$: *уюмЭ7-52БИУ3-21Ее4К6Э5-32КптрыИБМ4-19ЦепуЭ5-32КИУ9-61УИБМ4-19ЦИБМ4-19ЦИУ9-61УЭ5-32КЭ5-32КИУ9-61УИУ9-61Уйц*.

4.4 Сравнительный анализ на основе эксперимента

4.4.1 Сравнение времени работы

Замеры произведены на 4-ядерном процессоре *Intel Core i7* с тактовой частотой 2,4 ГГц, оперативная память — 8 ГБ.

Экспериментально получена таблица сравнения времени (табл. 1, время в секундах (с)):

Таблица 1: Сравнение времени выполнения алгоритмов на основе конечных автоматов и с использованием регулярных выражений

Длина слова	Конечный автомат, с	Регулярные выражения, с
10	0.000038	0.000588
20	0.000056	0.000516
30	0.000064	0.000642
40	0.000072	0.000564
50	0.000081	0.000641
60	0.000157	0.000665
70	0.000104	0.000650
hline 80	0.000092	0.000632
90	0.000148	0.000833
100	0.000156	0.000624

Ниже полученные данные представлены в виде графика:

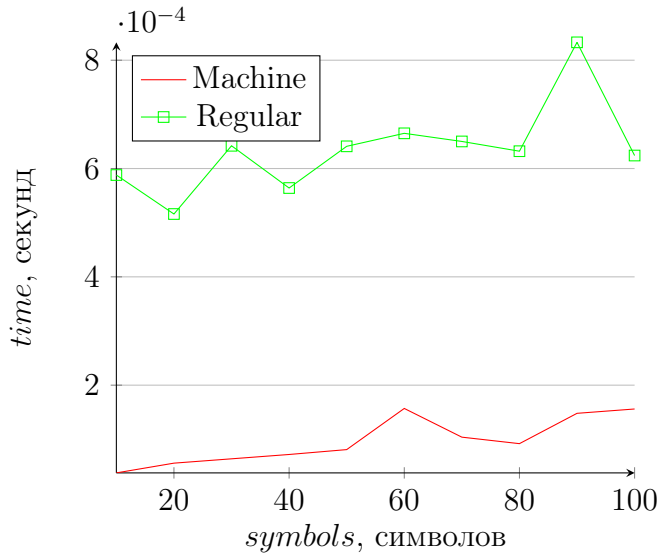

Рис. 7: График зависимости времени работы алгоритмов от количества символов в строке

Видно, что алгоритм, основанный на конечных автоматах, в среднем затрачивает в 10 раз меньше времени на поиск групп факультетов нежели алгоритм использующий регулярные выражения.

5 Выводы по экспериментальному разделу

В данном разделе было проведено исследование временных затрат разработанного программного обеспечения, вместе с сравнительным анализом реализованных алгоритмов на основе экспериментальных данных. В результате проведенного исследования выяснилось, что алгоритм, основанный на конечных автоматах, в среднем затрачивает в 10 раз меньше времени на поиск групп факультетов нежели алгоритм использующий регулярные выражения, что напрямую вытекает из теоретического материала приведенного в аналитическом разделе.

Заключение

В ходе выполнения данной лабораторной работы были изучены конечные автоматы и регулярные выражения. Алгоритмы были разработаны и реализацованы, было проведено исследование временных затрат алгоритмов, а также дано описание и обоснование полученных результатов.

Список литературы

[1] Python 3.8.2rc1 documentation [Электронный ресурс]. - Режим доступа: https://docs.python.org/3/, свободный - (28.11.2019)

[2] PyCharm documentation [Электронный ресурс]. - Режим доступа: https://www.jetbrains.com/pycharm/documentation/ - (28.11.2019)