**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

*Рубежный контроль №2*

*По предмету: «Анализ алгоритмов»*

# Сравнение реализации регулярных выражений через библиотеки и автоматами

Студент: Гасанзаде М.А.,
Группа: ИУ7-56Б

Москва, 2019 г.

# Оглавление

**Введение**

**Регулярные выражения** -– Истоки регулярных выражений лежат в теории автоматов, теории формальных языков и классификации формальных грамматик по Хомскому. Эти области изучают вычислительные модели (автоматы) и способы описания и классификации формальных языков. В 1940-х гг. Уоррен Маккалок и Уолтер Питтс описали нейронную систему, используя простой автомат в качестве модели нейрона.

Математик Стивен Клини позже описал эти модели, используя свою систему математических обозначений, названную «регулярные множества». Кен Томпсон встроил их в редактор QED, а затем в редактор ed под UNIX. С этого времени регулярные выражения стали широко использоваться в UNIX и UNIX-подобных утилитах, например, в expr, awk, Emacs, vi, lex и Perl.

**Автоматы** – Автомат это совокупность (X, S, δ), где X алфавит, S непустое множество, элементы которого называются состояниями автомата, δ функция из S Ч X в S, она называется функцией перехода.

**Целью данной** лабораторной работы является изучение данных алгоритмов и оценка этих алгоритмов по затратам времени и памяти.

Теория автоматов лежит в основе всех цифровых технологий и программного обеспечения, так, например, компьютер является частным случаем практической реализации конечного автомата.

## 1. Аналитическая часть

В данном разделе будут представлены описания регулярных выражений и автоматов.

### 1.1 Описание алгоритмов

**Регулярные выражения** - формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов. Для поиска используется строка-паттерн (шаблон, маска) состоящая из символов и метасимволов и задающая правило поиска. Для манипуляций с текстом дополнительно задаётся строка замены, которая также может содержать в себе специальные символы.

**Автоматы** - Автомат имеет следующие составляющие:

1) входные переменные, которые представляют собой воздействия, генерируемые извне и влияющие на поведение исследуемой системы;

2) выходные переменные, называемые реакцией системы, представляющие собой величины, характеризующие поведение данной системы.

Входные полюсы (входные каналы) соответствуют местам поступления входных переменных, снабжаются стрелками, направленными внутрь «черного ящика». Входные и выходные переменные с точки зрения абстрактной теории автоматов не имеют какого-либо физического смысла.

Предполагается, что любая система, представляемая основной моделью, управляется некоторым синхронизирующим источником. Все переменные системы изменяются в определенные дискретные моменты времени, в которые подается синхронизирующий сигнал. Эти моменты времени называются тактами (тактовыми моментами) и обозначаются буквой ts. Тогда поведение системы в любой момент времени не зависит от интервала времени между и ts_l. Кроме того, независимой величиной, относительно которой определяются все переменные системы, является не время, а порядковый номер, связанный с тактом. Системы, удовлетворяющие вышеизложенным предположениям, называются синхронными. Асинхронные же системы, меняют свои сигналы, не привязываясь к синхронизирующему сигналу.

## 1.2 Вывод:

В данном разделе были рассмотрены реализации поиска слова в строке.

## 2. Технологическая часть

В данном разделе будут приведены требования к программному обеспечению, средства реализации, листинг кода и примеры тестирования.

### 2.1 Требования к программному обеспечению

На вход подаётся код в виде длинной строки, содержащий математические данные(sin,cos…) и разделенная \n. На выходе указать, сколько раз было встретились слова.

Требуется замерить время работы обоих реализаций.

### 2.2 Средства реализации

В качестве языка программирования был выбран Python в связи с его широким функционалом и огромнейшим набором библиотек, а также из-за привычного для меня синтаксиса. Среда разработки - стандартная IDLE Python. Время работы программы замеряется с помощью библиотеки функции perf_counter из библиотеки time[1].

### 2.3 Листинг кода

Листинг кода был представлен на *листингах 1, 2, 3.*

***Листинг 1. Поиск слова с помощью регулярных выражений.***

```
import re
from time import perf_counter

start = perf_counter()
pattern = re.compile(r'[asin|acos|sin|cos|sqrt|abs|tan|sqr]+\(')
matches = pattern.finditer(code_to_search)
count = 0
for match in matches:
    print(matches)
    count+=1

print(count)
stop = perf_counter()
time = stop - start
print("Elapsed time:", time)

f= open("time.txt","a+")
f.write("Time with re %f\n" % time)
f.close()
```

*Листинг 2. Поиск слова с автоматами, без использования каких-либо библиотек.*

```
from time import perf_counter

start = perf_counter()
text = code_to_search.split("\n")

k = 0
for i in range(len(text)):
    if 'acos('  in text[i]:
        k += 1
    elif 'asin(' in text[i]:
        k += 1
    elif 'sin(' in text[i]:
        k += 1
    elif 'cos(' in text[i]:
        k += 1
    elif 'tan(' in text[i]:
        k += 1
    elif 'atan(' in text[i]:
        k += 1
    elif 'abs(' in text[i]:
        k += 1
    elif 'sqrt(' in text[i]:
        k += 1


print('Count', k)
stop = perf_counter()
time = stop - start
print("Elapsed time:", stop-start)

f= open("time.txt","a+")
f.write("Time with at %f\n" % time)
f.close()
```

*Листинг 3. Текст на котором выполнялось тестирование и замеры.*

```
code_to_search = '''
from math import sin
sin(m)+
sin
asin(db)+
acos(rnrf)+
tan()+
prosto stroka tang()
just string but in english sqrt(1)+
sadece bir xett cos()+
cos
sqrt()+
sqrt
abs!
abs()+
abs)
abs[()]
'''
```

## 2.4 Описание тестирования

Было сделано 13 замеров времени для каждой из реализаций.

Таблица 1 — Данные тестов при 8 совпадениях

| **Регулярными выражениями** | **С помощью автоматов** |
|---|---|
| **0.042347** | **0.009796** |
| **0.045695** | **0.018142** |
| **0.048732** | **0.020606** |
| **0.100947** | **0.018051** |
| **0.088357** | **0.015749** |
| **0.092935** | **0.018869** |
| **0.090647** | **0.017277** |
| **0.091942** | **0.018694** |
| **0.096962** | **0.015488** |
| **0.093488** | **0.027529** |
| **0.091514** | **0.026849** |
| **0.093907** | **0.030135** |
| **0.145669** | **0.011706** |

Все тесты пройдены успешно.

## 2.1 Вывод

В данном разделе мы рассмотрели листинг кода, а также убедились в безошибочной работе программы.

## 3. Экспериментальная часть.

В данном разделе будут рассмотрены примеры работ программы.

### 3.1 Пример работы программы

На рисунках 1, 2 приведены изображения внешнего вида интерфейса программы во время его работы

```
Count 8
Elapsed time: 0.014600400000000001
```

*Рисунок 1 - пример работы программы с автоматами.*

```
Count 8
Elapsed time: 0.0103154
```

*Рисунок 2 - пример работы программы с регулярными выражениями.*

### 3.2 Сравнительный анализ алгоритмов

Как мы видим обе реализации достаточно быстры, но реализация автоматами немного выигрывает. Более подробно можно рассмотреть по графикам.

### 3.3 Сравнительный анализ на материале экспериментальных данных

Алгоритмы были протестированы по скорости работы, график представлен на рисунке 3.

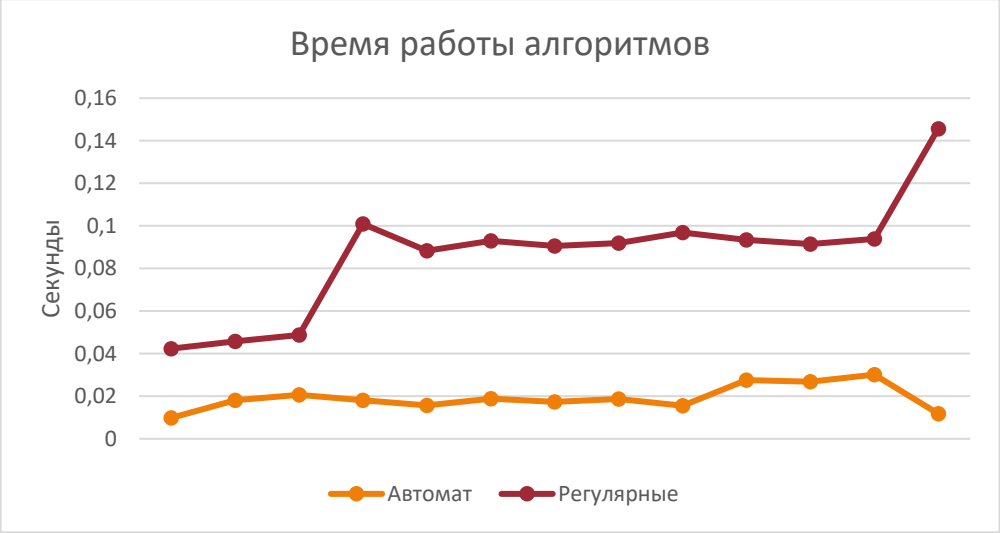


*Рисунок 3 – Замер времени*

### 3.3 Вывод

В данном разделе был представлен эксперимент по замеру времени. Не производилась оценка памяти, т.к. в Python3 без при использования функций – `def` выделяется 31.4 мб памяти, и она каждый раз инкрементируется. Так как наша реализация достаточно проста и не использует `def` память будет выделена только 1 раз.

## Заключение

В данной работе были реализованы и протестированы два современных методов поиска слов в строке. Проведён сравнительный анализ обоих реализаций.

Реализация с автоматами показала себя стабильнее и быстрее чем через регулярные выражения.

**Список литературы**


1. time — Time access and conversions // Python URL: https://docs.python.org/3/library/time.html
2. Regular expression operations URL: https://docs.python.org/3/library/re.html
3. Дж. Фридл Регулярные выражения
4. Белоусов А.И. Математика в техническом университете