

Государственное образовательное учреждение высшего профессионального образования
“Московский государственный технический университет имени Н.Э.Баумана”

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Дисциплина: АНАЛИЗ АЛГОРИТМОВ

ЛАБОРАТОРНАЯ РАБОТА № 7

Алгоритмы сравнения с образцом

Студент:

Ильясов Идрис Магомет-Салиевич

Группа: ИУ7-53Б

Преподаватели:

Волкова Лилия Леонидовна

Строганов Юрий Владимирович

Москва, 2019 г.

Содержание

Введение	2
1 Аналитическая часть	3
1.1 Алгоритм Кнута-Морриса-Пратта (КМП)	3
1.2 Алгоритм Бойера-Мура	3
1.3 Выводы	4
2 Конструкторская часть	5
2.1 Функциональная модель	5
2.2 Разработка алгоритмов	5
2.3 Выводы	6
3 Технологическая часть	7
3.1 Требования к программному обеспечению	7
3.2 Средства реализации	7
3.3 Листинг кода	7
3.4 Выводы	8
4 Экспериментальная часть	9
4.1 Примеры работы	9
4.2 Выводы	9
Заключение	10
Список литературы	11

Введение

В задачах поиска информации одной из важнейших задач является поиск точно заданной подстроки в строке. Целью данной работы является получение навыка реализации алгоритмов сравнения с образцом на основании автоматов. Для достижения поставленной цели необходимо решить следующие задачи:

1. описать алгоритмы Кнута-Морриса-Пратта и Бойера-Мура;
2. реализовать указанные алгоритмы;
3. привести примеры работы на данных разных классов эквивалентности.

1 Аналитическая часть

В данном разделе будут описаны алгоритмы Кнута-Морриса-Пратта и Бойера-Мура.

1.1 Алгоритм Кнута-Морриса-Пратта (КМП)

Идея алгоритма Кнута-Морриса-Пратта основана на использовании конечных автоматов. Конечные автоматы используются для решения задачи о том, принадлежит ли данное слово данному языку. Конечный автомат представляет собой простое устройство, описываемое текущим состоянием и функцией перехода. Функция перехода формирует новое состояние автомата по текущему состоянию и значению очередного входного символа. Некоторые состояния считаются принимающими, если после окончания ввода автомат оказывается в принимающем состоянии, то входное слово считается принятым. Конечный автомат, настроенный на данный образец, можно использовать для сравнения с образцом; если автомат переходит в принимающее состояние, то это означает, что образец найден в тексте. Алгоритм состоит из двух этапов: первый заключается в формировании массива значений префикс-функции, используемого при сдвиге образца вдоль текста; второй - непосредственно в поиске образца в тексте.

Префиксом будем считать символы, которые стоят в начале образца, суффиксом - которые стоят в конце. Префикс-функция от строки S - массив, каждый элемент которого $\pi[i]$ является значением, равным максимальной длине совпадающих префикса и суффикса подстроки в образце, которая заканчивается i -ым символом (префикс и суффикс могут перекрываться). Значение префикс-функции в нулевой позиции считается равным 0. Часто значение префикс-функции записывают в виде вектора длиной $|S|$. Например, для строки 'abcabca' префикс-функция будет такой: $\pi(abcabca) = [0001234]$. То есть массив содержит значения, которые позволяют определить, на сколько можно сдвинуть массив вдоль строки. Не рассматриваются префиксы, равные длине слова, так как в таком случае алгоритм теряет смысл.

1.2 Алгоритм Бойера-Мура

Алгоритм Бойера-Мура осуществляет сравнение справа налево, а не слева направо. Таким образом, исследуя образец, можно осуществлять более эффективные сдвиги в тексте при обнаружении несовпадения.

На первом этапе алгоритма осуществляется формирование таблицы смещений, используемой при сдвиге образца по тексту, второй этап состоит из непосредственно поиска образца в тексте. Значениями элементов таблицы смещений является удаленность определенного символа от конца образца. Если встречаются повторяющиеся символы, записывается значение того символа, который более близок к концу. Последнему символу образца сопоставляется значение, равное длине образца, если символ ни разу больше не встречался в образце. Например, для строки 'данные' таблица смещений будет иметь следующие значения: [542216], а для строки 'денные' - [542214]. Символам, не вошедшим в образец, сопоставляется значение, равное длине образца. Стоит отметить также, что неоднократное вхождение символов в образец никак не влияет на вычисление удаленности других символов от конца образца (для строки 'данные' получим [542216] а не [432216]).

На втором этапе проходим по образцу справа налево и сравниваем посимвольно символы образа с символами текста. В случае несовпадения ищем смещение по символу, находящемуся в тексте, а не образце. В случае, когда имел место ряд последовательных совпадений, и после него встретились несовпадающие символы, смещение ищется по последнему символу образца.

1.3 Выводы

В данном разделе были описаны алгоритмы Кнута-Морриса-Пратта и Бойера-Мура.

2 Конструкторская часть

В данном разделе будут приведены функциональная модель и схемы алгоритмов.

2.1 Функциональная модель

На рисунке 1 приведена функциональная модель решаемой задачи.

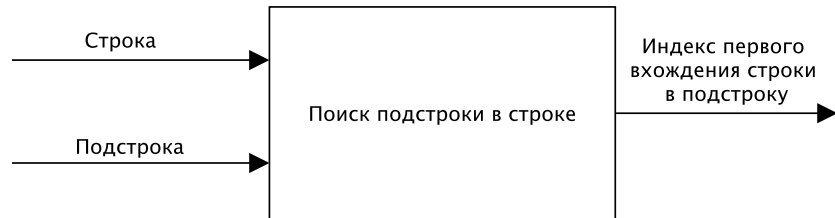


Рис. 1 - Функциональная модель алгоритма нахождения подстроки в строке.

2.2 Разработка алгоритмов

На рисунках 1 и 2 приведены схемы алгоритмов Кнута-Морриса-Пратта и Бойера-Мура.

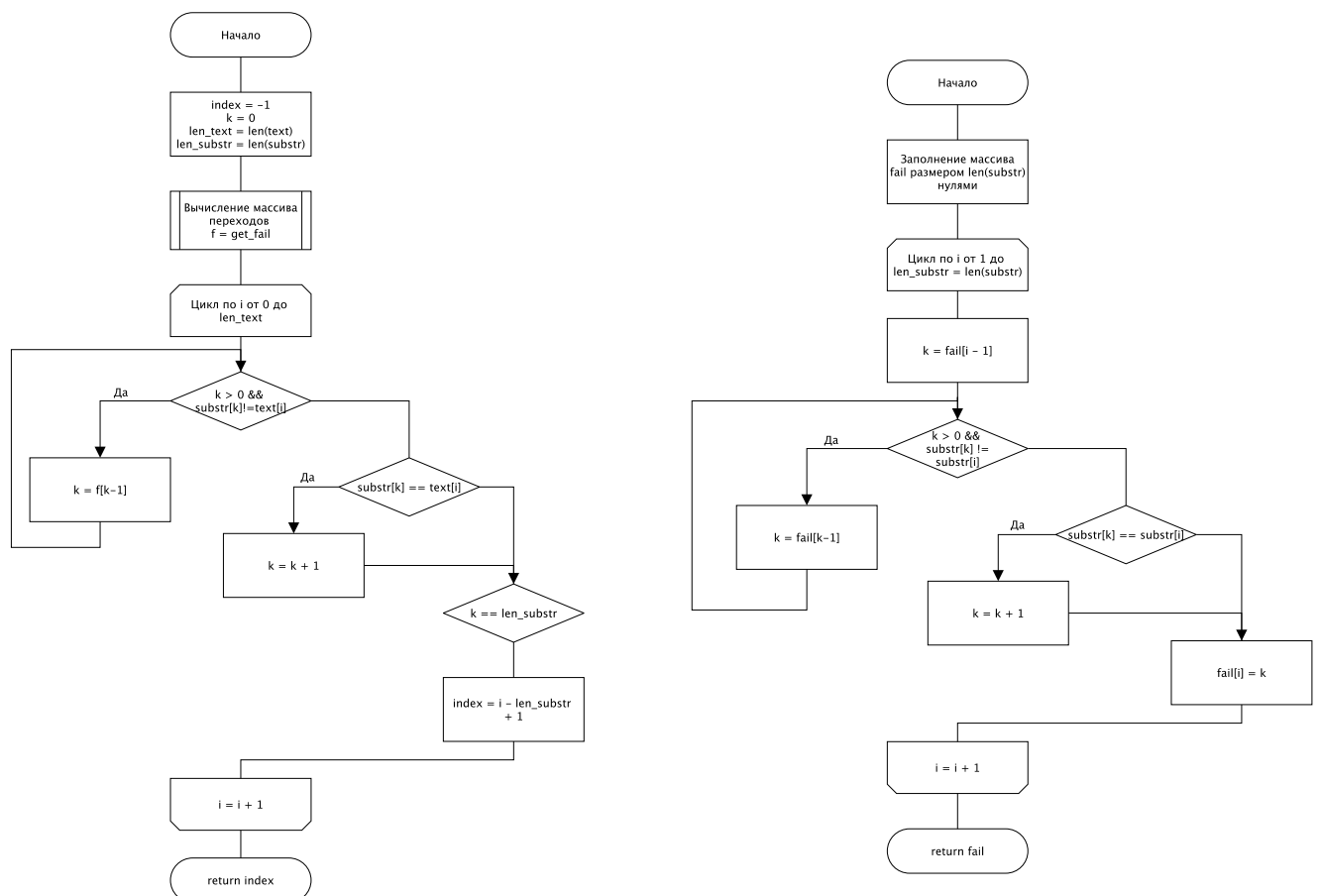


Рис. 2 - Схема алгоритма Кнута-Морриса-Пратта.

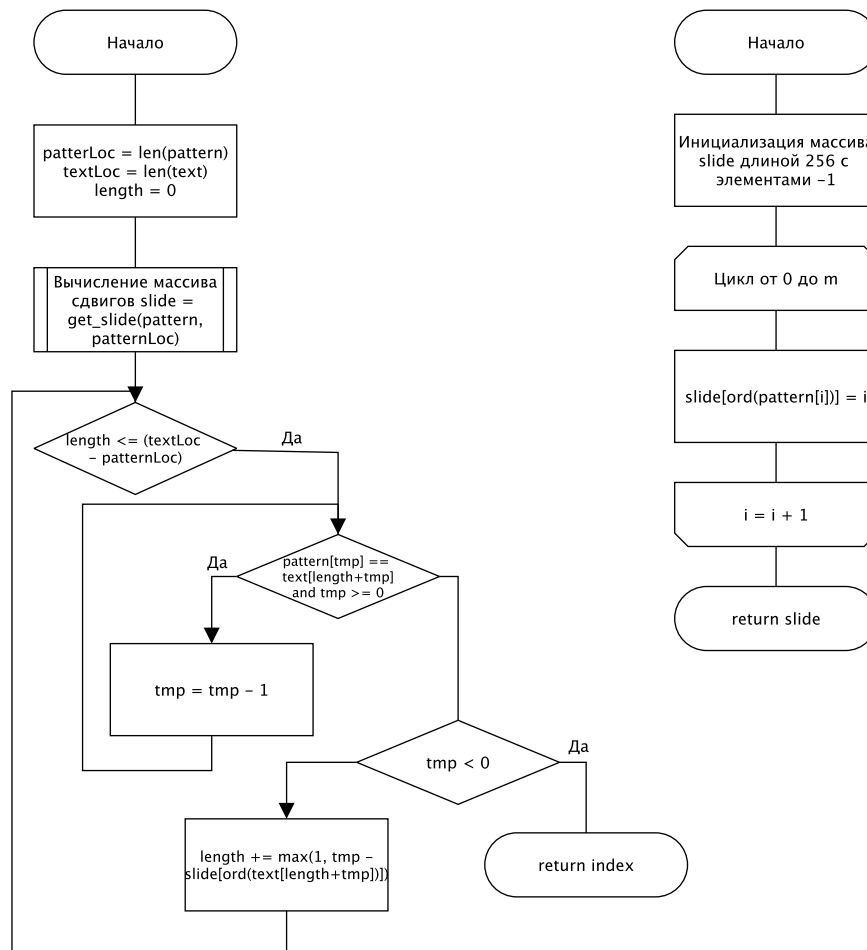


Рис. 23- Схема алгоритма Бойера-Мура.

2.3 Выводы

В данном разделе были рассмотрены схемы алгоритмов Кнута-Морриса-Пратта и Бойера-Мура.

3 Технологическая часть

В данном разделе представлены требования к разрабатываемому программному обеспечению, средства, использованные в процессе разработки для реализации поставленных задач, а также листинг кода программы.

3.1 Требования к программному обеспечению

ПО должно реализовать алгоритмы Кнута-Морриса-Пратта и Бойера-Мура, которые предназначены для нахождения подстроки в строке.

3.2 Средства реализации

Для реализации поставленной задачи был использован язык программирования Python.

3.3 Листинг кода

В приведенных ниже листингах представлены реализации алгоритма Кнута-Морриса-Пратта(листинг 1), алгоритма Бойера-Мура (листинг 2).

Листинг 1: Алгоритм Кнута-Морриса-Пратта

```
1 def get_fail(substr):
2     fail = [0] * len(substr)
3     for i in range(1, len(substr)):
4         k = fail[i-1]
5         while k > 0 and substr[k] != substr[i]:
6             k = fail[k-1]
7         if substr[k] == substr[i]:
8             k = k + 1
9         fail[i] = k
10
11     return fail
12
13 def kmp(substr, text):
14     index = -1
15     f = get_fail(substr)
16     k = 0
17     for i in range(len(text)):
18         while k > 0 and substr[k] != text[i]:
19             k = f[k-1]
20         if substr[k] == text[i]:
21             k = k + 1
22         if k == len(substr):
23             index = i - len(substr) + 1
24             break
```



```
25
26     return index
```

Листинг 2: Алгоритм Бойера-Мура

```
1 def get_slide(pattern, m):
2     slide = 256 * [-1]
3     for i in range(m):
4         slide[ord(pattern[i])] = i
5
6     return slide
7
8 def bm(pattern, text):
9     patternLoc = len(pattern)
10    textLoc = len(text)
11
12    slide = get_slide(pattern, patternLoc)
13    length = 0
14
15    while (length <= (textLoc - patternLoc)):
16        tmp = patternLoc - 1
17        while (tmp >= 0 and pattern[tmp] == text[length + tmp]):
18            tmp -= 1
19        if (tmp < 0):
20            return length
21        else:
22            length += max(1, tmp - slide[ord(text[length + tmp])])
```

3.4 Выводы

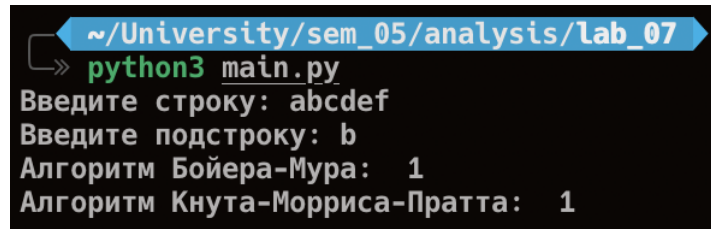
На основе схем алгоритмов, представленных в конструкторском разделе, в соответствии с указанными требованиями к реализации с использованием средств языка Python было разработано программное обеспечение, содержащее реализации выбранных алгоритмов.

4 Экспериментальная часть

В данном разделе будут приведены примеры работы разработанного программного обеспечения.

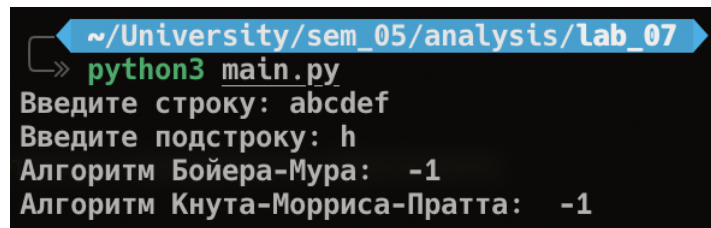
4.1 Примеры работы

На приведенных ниже рисунках продемонстрирована работа выбранных алгоритмов.



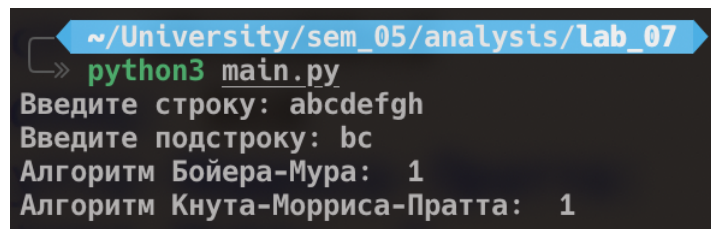
```
~/University/sem_05/analysis/lab_07
» python3 main.py
Введите строку: abcdef
Введите подстроку: b
Алгоритм Бойера-Мура: 1
Алгоритм Кнута-Морриса-Пратта: 1
```

Рис. 4 - Пример работы



```
~/University/sem_05/analysis/lab_07
» python3 main.py
Введите строку: abcdef
Введите подстроку: h
Алгоритм Бойера-Мура: -1
Алгоритм Кнута-Морриса-Пратта: -1
```

Рис. 5 - Пример работы



```
~/University/sem_05/analysis/lab_07
» python3 main.py
Введите строку: abcdefgh
Введите подстроку: bc
Алгоритм Бойера-Мура: 1
Алгоритм Кнута-Морриса-Пратта: 1
```

Рис. 6 - Пример работы

4.2 Выводы

В этом разделе были приведены примеры работы разработанного ПО.

Заключение

В ходе данной работы были изучены алгоритмы Кнута-Морриса-Пратта и Бойера-Мура для сравнения с образцом на основании автоматов. Подход алгоритма Бойера-Мура состоит в том, чтобы попытаться сопоставить последний символ образца вместо первого с предположением, что если последние символы не совпадают, не нужно пытаться искать совпадение в начале. Это позволяет осуществлять переходы за раз на несколько позиций, поэтому алгоритм Бойера-Мура работает лучше, когда образец и текст напоминают "естественный текст". Алгоритм КМП выполняет поиск вхождения образца в текст, используя наблюдение, что, когда происходит несоответствие, сам образец содержит достаточную информацию для определения возможного начала следующего совпадения, минуя повторное рассмотрение ранее совпадающих символов. Это означает, что алгоритм КМП лучше подходит для случаев, когда образец и текст содержат повторяющиеся последовательности.

Список литературы

1. <https://habr.com/ru/post/111449/>
2. Семинары по анализу алгоритмов.