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»

ОТЧЕТ

по лабораторной работе № 7

по курсу «Анализ алгоритмов»

на тему: «Алгоритмы поиска»

Студент ИУ7-54Б _____ Булдаков М.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Волкова Л. Л.
(Подпись, дата) (И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Проблема поиска информации является одной из важнейших задач информатики. Компьютерные методы информационного поиска — активно развивающаяся, актуальная в научном и практическом аспекте тема современных публикаций. Развитие компьютерной техники влечет существенный рост объема информации, представляемой в электронном виде, влияние этого процесса на современные информационные технологии, включая поиск, отмечается в большинстве публикаций в периодических изданиях [1].

Цель данной лабораторной работы — получение навыков использования алгоритмов поиска. Для достижения поставленной цели необходимо выполнить следующие задачи:

— описать стандартный алгоритм и алгоритм Кнута—Морриса—Пратта;

— разработать программное обеспечение, реализующее алгоритмы поиска подстроки в строке;

— выбрать инструменты для реализации алгоритмов и замера процессорного времени выполнения реализаций алгоритмов;

— проанализировать затраты реализаций алгоритмов по времени и количество сравнений в реализованных алгоритмах.

1 Аналитический раздел

В данном разделе будут описаны два алгоритма поиска подстрок: стандартный, Кнута—Морриса—Пратта.

1.1 Стандартный алгоритм

Стандартный алгоритм начинается со сравнения первого символа текста с первым символом подстроки. При совпадении символов, сравниваются следующие символы. Так продолжается до тех пор, пока не окажется, что подстрока целиком совпала с отрезком текста, или пока не встретятся несовпадающие символы. В первом случае задача решена, во втором указатель текущего положения в тексте сдвигается на один символ и заново начинается сравнение с подстрокой [2].

1.2 Алгоритм Кнута—Морриса—Пратта

Алгоритм Кнута—Морриса—Пратта основан на принципе конечного автомата, однако он использует более простой метод обработки неподходящих символов. В этом алгоритме состояния помечаются символами, совпадение с которыми должно в данный момент произойти. Из каждого состояния имеется два перехода: один соответствует успешному сравнению, другой — несовпадению. При успешном сравнении осуществляется переход в следующий узел автомата, а в случае несовпадения осуществляется переход в предыдущий узел, отвечающий образцу. В программной реализации этого алгоритма применяется массив сдвигов, который создается для каждой подстроки, которая ищется в тексте. Для каждого символа из подстроки рассчитывается значение, равное максимальной длине совпадающего префикса и суффикса относительно конкретного элемента подстроки. Создание этого массива позволяет при несовпадении строки сдвигать ее на расстояние большее, чем 1.

Вывод

В данном разделе были описаны два алгоритма поиска подстрок: стандартный, Кнута—Морриса—Пратта.

2 Конструкторский раздел

В данной части работы будут рассмотрены псевдокод стандартного алгоритма и псевдокод алгоритма Кнута—Морриса—Пратта.

2.1 Требования к программному обеспечению

Программа должна поддерживать два режима работы: режим массового замера времени и режим поиска подстроки в заданной строке.

Режим массового замера времени должен обладать следующей функциональностью:

— генерировать строки различного размер для проведения замеров;

— осуществлять массовый замер времени, используя сгенерированные данные;

— результаты замеров должны быть представлены в виде таблицы и графика.

К режиму поиска подстроки выдвигается следующий ряд требований:

— возможность работать со строками разного размера, которые вводит пользователь;

— возможность правильно обрабатывать кириллицу;

— наличие интерфейса для выбора действий;

— на выходе программы, индекс подстроки в строке или -1, если подстрока не найдена.

2.2 Разработка алгоритмов

В листингах 1–2 рассмотрены псевдокоды алгоритмов поиска, входными данными для них являются:

— s — строка, в которой осуществляется поиск;

— $substr$ — подстрока, которую требуется найти.

В случае отсутствия подстроки в строке происходит возврат -1 — невалидного индекса в строке. Оператор $\leftarrow$ обозначает присваивание значения переменной, оператор $[i]$ обозначает получение буквы из строки с индексом i, оператор $len(str)$ обозначает длину строки str, иные операторы подобны математическим.

Алгоритм 1 Псевдокод стандартного алгоритма.

Цикл от $i = 0$ до $len(s) - len(substr)$ **выполнять**
  $flag \leftarrow 0$
  Цикл от $j = 0$ до $len(substr) - 1$ **выполнять**
    Если $s[i + j] \neq substr[j]$ **тогда**
      $flag \leftarrow 1$
      Выйти из цикла
    Конец условия
  Конец цикла
  Если $flag = 0$ **тогда**
    Возвратить i
  Конец условия
Конец цикла
Возвратить -1

Алгоритм 2 Псевдокод алгоритма Кнута—Морриса—Пратта.

Создать массив целых чисел $next$ длины $len(substr) + 1$
Заполнить массив $next$ нулями
Цикл от $i = 1$ до $len(substr) - 1$ **выполнять**
    $j \leftarrow next[i]$
    До тех пока $j > 0$ и $substr[j] \neq substr[i]$ **выполнять**
        $j \leftarrow next[j]$
    Конец цикла
    Если $j > 0$ или $substr[j] = substr[i]$ **тогда**
        $next[i + 1] \leftarrow j + 1$
    Конец условия
Конец цикла
$i \leftarrow 0$
$j \leftarrow 0$
До тех пока $i < len(s)$ **выполнять**
    Если $j < len(substr)$ и $s[i] = substr[j]$ **тогда**
        $j \leftarrow j + 1$
        Если $j = len(substr)$ **тогда**
            Возвратить $i - j + 1$
        Конец условия
    иначе если $j > 0$
        $j \leftarrow next[j]$
        $i \leftarrow i - 1$
    Конец условия
    $i \leftarrow i + 1$
Конец цикла
Возвратить -1

Вывод

В данной части работы был написан псевдокод для алгоритмов стандартного поиска и алгоритма Кнута—Морриса—Пратта.

3 Технологический раздел

В данном разделе будут приведены средства реализации, листинг кода и функциональные тесты.

3.1 Средства реализации

Для реализации данной работы был выбран язык *Python* [3]. Данный выбор обусловлен следующим:

— язык поддерживает все структуры данных, которые выбраны в результате проектирования;

— язык позволяет реализовать все алгоритмы, выбранные в результате проектирования;

— язык позволяет замерять процессорное время с помощью модуля *time*.

Процессорное время было замерено с помощью функции *process_time()* из модуля *time* [4].

3.2 Сведения о модулях программы

Данная программа разбита на следующие модули:

— *main.py* — файл, содержащий функцию *main*;

— *algorithms.py* — файл, содержащий код реализаций всех алгоритмов поиска;

— *compare_time.py* — файл, в котором содержатся функции для замера и вывода времени выполнения реализаций алгоритмов;

— *count_compares.py* — файл, в котором содержатся функции для подсчета количества сравнений в реализациях алгоритмов.

3.3 Реализации алгоритмов

В листингах 3.1 и 3.2 приведены реализации алгоритмов стандартного поиска и алгоритма Кнута—Морриса—Пратта соответственно.

Листинг 3.1 – Функция стандартного поиска

```
def standard_search(s, substr):
    for i in range(len(s) - len(substr) + 1):
        flag = 0
        for j in range(len(substr)):
            if s[i + j] != substr[j]:
                flag = 1
                break

        if not flag:
            return i

    return -1
```

Листинг 3.2 – Функция алгоритма Кнута—Морриса—Пратта

```
def KMP(s, substr):
    next = [0] * (len(substr) + 1)

    for i in range(1, len(substr)):
        j = next[i]
        while j > 0 and substr[j] != substr[i]:
            j = next[j]
        if j > 0 or substr[j] == substr[i]:
            next[i + 1] = j + 1

    i, j = 0, 0
    while i < len(s):
        if j < len(substr) and s[i] == substr[j]:
            j += 1
            if j == len(substr):
                return i - j + 1
        elif j > 0:
            j = next[j]
            i -= 1
        i += 1

    return -1
```

3.4 Функциональные тесты

В таблице 3.1 приведены функциональные тесты для разработанных алгоритмов поиска. Пустая строка обозначается с помощью символа λ. Все тесты пройдены успешно.

Таблица 3.1 – Функциональные тесты

Строка	**Подстрока**	**Ожидаемый результат**	**Фактический результат**
мама	ам	1	1
абоба	оба	2	2
абабабцб	ababцб	2	2
мама	пап	-1	-1
абабаба	аб	0	0
мамы	ы	3	3
ы	ы	0	0
λ	ы	-1	-1
ы	λ	-1	-1
λ	λ	-1	-1
абоба	абобаабоба	-1	-1

Вывод

Были разработаны и протестированы спроектированные алгоритмы поиска.

4 Исследовательский раздел

В данном разделе будут приведены: пример работы программы, постановка эксперимента и сравнительный анализ алгоритмов на основе полученных данных.

4.1 Демонстрация работы программы

На рисунке 4.1 представлена демонстрация работы разработанного программного обеспечения, а именно показаны результаты поиска подстроки «ам» в строке «мама».

```
    Меню:

    1 - Найти подстроку в строке (стандартный алгоритм)
    2 - Найти подстроку в строке (алгоритм КМП)
    3 - Функциональные тесты
    4 - Замеры количества сравнений
    5 - Замеры времени
    0 - Выход

Выберите пункт меню: 1
Введит строку: мама
Введит подстроку: ам
Результат (стандартный алгоритм): 1

    Меню:

    1 - Найти подстроку в строке (стандартный алгоритм)
    2 - Найти подстроку в строке (алгоритм КМП)
    3 - Функциональные тесты
    4 - Замеры количества сравнений
    5 - Замеры времени
    0 - Выход

Выберите пункт меню: 2
Введит строку: мама
Введит подстроку: ам
Результат (алгоритм КМП): 1
```

Рисунок 4.1 – Демонстрация работы программы при поиске подстрок

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялись замеры по времени, следующие:

— процессор: AMD Ryzen 5 4600H 3 ГГц [5];

— оперативная память: 16 ГБайт;

— операционная система: Windows 10 Pro 64-разрядная система версии 22H2 [6].

При замерах времени ноутбук был включен в сеть электропитания и был нагружен только системными приложениями.

4.3 Время выполнения реализаций алгоритмов

Результаты замеров времени выполнения реализаций алгоритмов поиска приведены в таблицах 4.1 – 4.3. Замеры времени проводились на строках одной длины и усреднялись для каждого набора одинаковых экспериментов. Строки состояли из букв «а», искомой подстрокой являлась строка «аааааффффф».

В таблицах 4.1 – 4.3 используются следующие обозначения:

— С — реализация стандартного алгоритма;

— КМП — реализация алгоритма Кнута—Морриса—Пратта.

Таблица 4.1 – Время работы реализаций алгоритмов на строках, где искомая подстрока стоит на первой позиции (в с)

Длина строки	КМП	С
256	$5.656 \cdot 10^{-6}$	$1.750 \cdot 10^{-6}$
512	$5.406 \cdot 10^{-6}$	$1.594 \cdot 10^{-6}$
1024	$5.406 \cdot 10^{-6}$	$1.500 \cdot 10^{-6}$
2048	$5.438 \cdot 10^{-6}$	$1.438 \cdot 10^{-6}$
4096	$5.375 \cdot 10^{-6}$	$1.438 \cdot 10^{-6}$
8192	$5.438 \cdot 10^{-6}$	$1.438 \cdot 10^{-6}$
16384	$5.313 \cdot 10^{-6}$	$1.469 \cdot 10^{-6}$
32768	$5.406 \cdot 10^{-6}$	$1.406 \cdot 10^{-6}$
65536	$5.344 \cdot 10^{-6}$	$1.438 \cdot 10^{-6}$

Таблица 4.2 – Время работы реализаций алгоритмов на строках, где искомая подстрока стоит в конце (в с)

Длина строки	КМП	С
256	$1.250 \cdot 10^{-4}$	$3.125 \cdot 10^{-4}$
512	$3.125 \cdot 10^{-4}$	$6.875 \cdot 10^{-4}$
1024	$5.625 \cdot 10^{-4}$	$1.437 \cdot 10^{-3}$
2048	$1.250 \cdot 10^{-3}$	$2.875 \cdot 10^{-3}$
4096	$2.375 \cdot 10^{-3}$	$5.875 \cdot 10^{-3}$
8192	$4.562 \cdot 10^{-3}$	$9.938 \cdot 10^{-3}$
16384	$8.250 \cdot 10^{-3}$	$2.013 \cdot 10^{-2}$
32768	$1.619 \cdot 10^{-2}$	$4.069 \cdot 10^{-2}$
65536	$3.331 \cdot 10^{-2}$	$8.125 \cdot 10^{-2}$

Таблица 4.3 – Время работы реализаций алгоритмов на строках, где искомая подстрока отсутствует (в с)

Длина строки	КМП	С
256	$1.875 \cdot 10^{-4}$	$3.125 \cdot 10^{-4}$
512	$3.125 \cdot 10^{-4}$	$7.500 \cdot 10^{-4}$
1024	$5.625 \cdot 10^{-4}$	$1.313 \cdot 10^{-3}$
2048	$1.250 \cdot 10^{-3}$	$2.938 \cdot 10^{-3}$
4096	$2.188 \cdot 10^{-3}$	$5.000 \cdot 10^{-3}$
8192	$4.125 \cdot 10^{-3}$	$1.000 \cdot 10^{-2}$
16384	$8.500 \cdot 10^{-3}$	$2.025 \cdot 10^{-2}$
32768	$1.625 \cdot 10^{-2}$	$4.088 \cdot 10^{-2}$
65536	$3.275 \cdot 10^{-2}$	$7.994 \cdot 10^{-2}$

На рисунках 4.2 – 4.4 изображены графики зависимостей времени выполнения реализаций алгоритмов поиска от размеров строки.

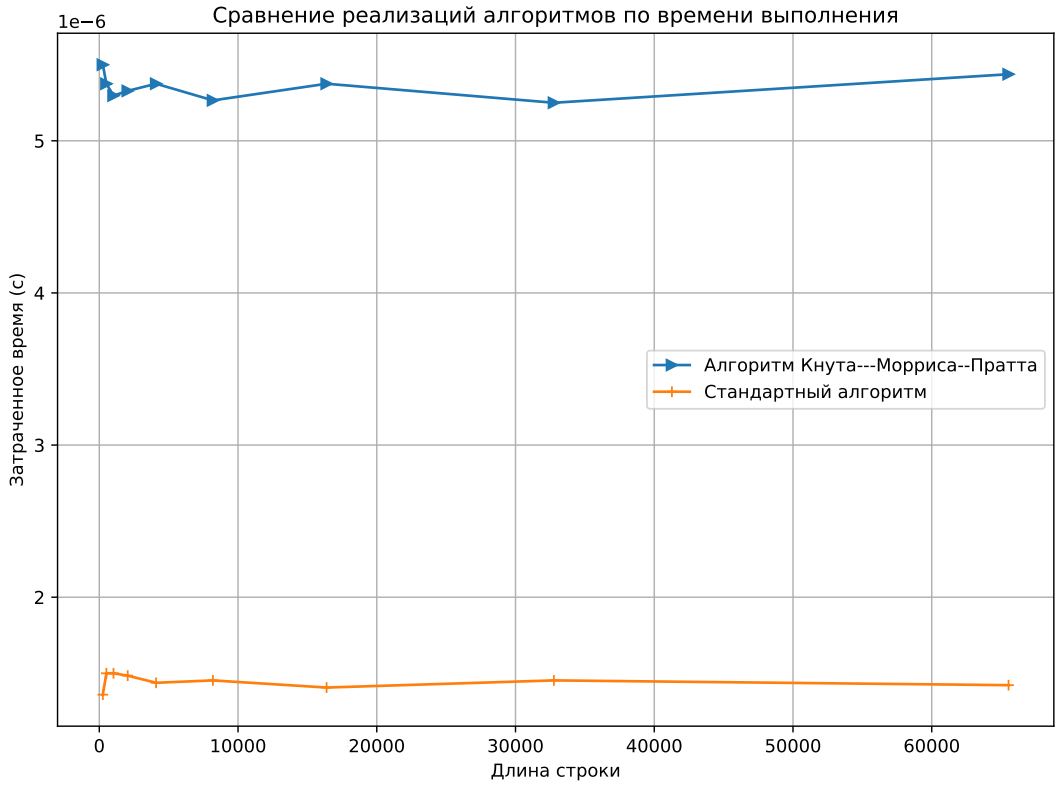

Рисунок 4.2 – Сравнение реализаций алгоритмов по времени выполнения, в случае когда подстрока находится в начале

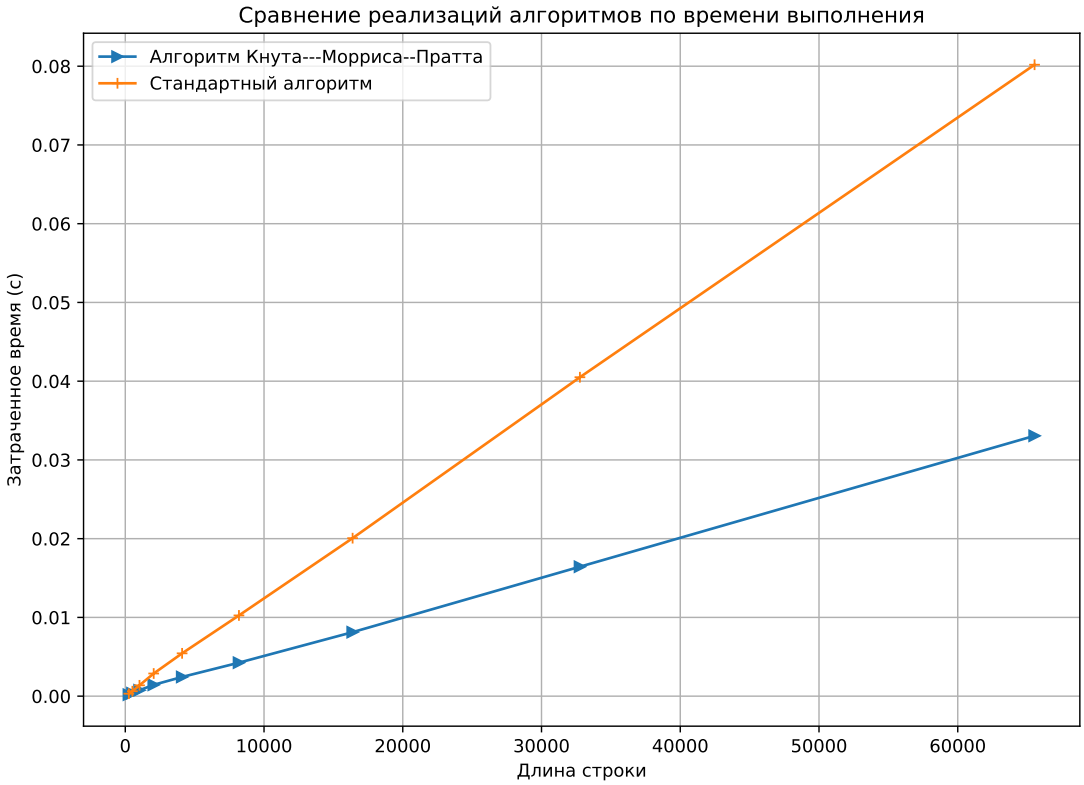

Рисунок 4.3 – Сравнение реализаций алгоритмов по времени выполнения, в случае когда подстрока находится в конце

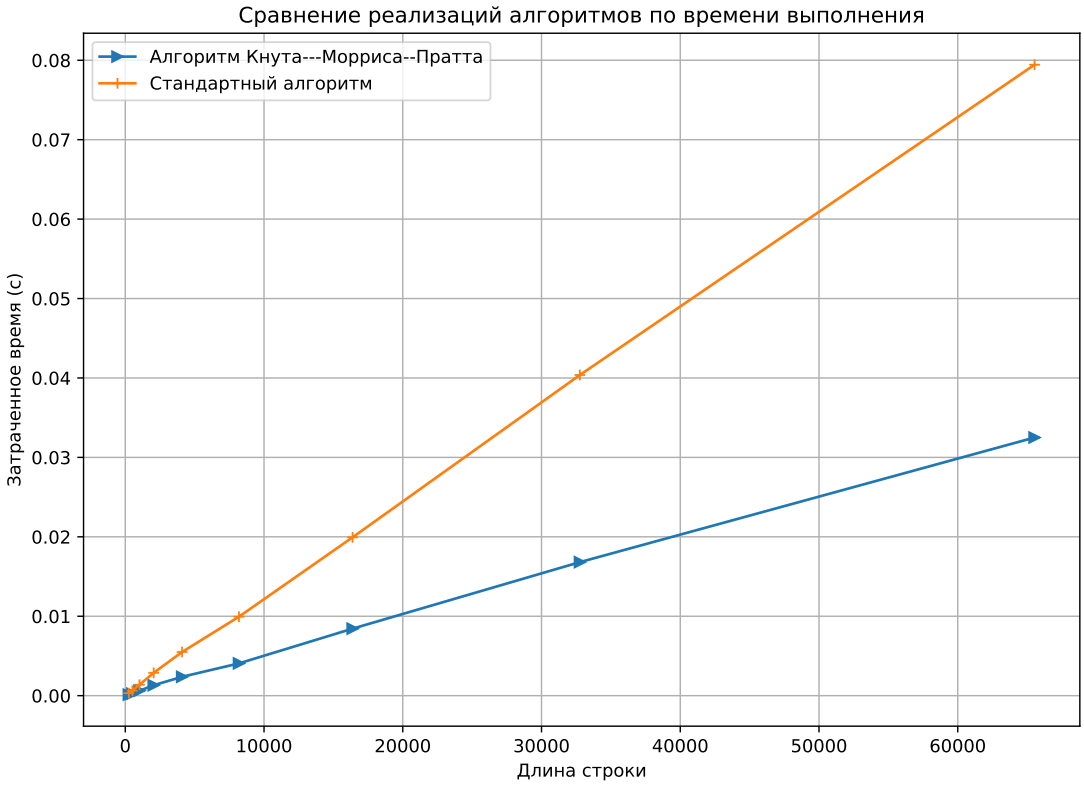

Рисунок 4.4 – Сравнение реализаций алгоритмов по времени выполнения, в случае когда подстрока отсутствует в строке

4.4 Количество сравнений

Подсчет количества сравнений производился для строки длинной 256 символов без учета длины подстроки, состоящей из символов «a». Подстрокой являлась строка «аааааффффф».

Результаты подсчетов числа сравнений представлены на диаграммах 4.5 и 4.6. На диаграммах 4.5 и 4.6 индекс -1 обозначает отсутствие подстроки в строке.

Рисунок 4.5 – Диаграмма зависимости числа сравнений от индекса подстроки при использовании стандартного алгоритма

Рисунок 4.6 – Диаграмма зависимости числа сравнений от индекса подстроки при использовании алгоритма Кнута—Морриса—Пратта

Вывод

В результате замеров времени выполнения реализаций алгоритмов поиска было выявлено, что реализация алгоритма Кнута—Морисса—Пратта медленнее реализации стандартного алгоритма в 3 раза, в случае когда подстрока стоит в начале, при длине подстроки 10 и размеров строки 256. Такой результат обусловлен необходимостью вычислять массив сдвигов для подстроки в реализации алгоритма Кнута—Морисса—Пратта. В то же время, реализация алгоритма Кнута—Морисса—Пратта выигрывает по скорости реализацию стандартного алгоритма в 2.4 раза при длине строк 65536, когда подстрока отсутствует или стоит в конце строки. Такой результат обусловлен тем, что в реализации алгоритма Кнута—Морисса—Пратта используются особенности искомой подстроки, что позволяет осуществлять смещения сразу на несколько шагов.

В результате подсчетов количества сравнений было выявлено, что для реализации алгоритма Кнута—Морриса—Пратта худшим является случай, когда подстрока отсутствует в строке. Для реализации стандартного алгоритма, худшим является случай, когда подстрока стоит на последних позициях в строке. При этом в реализации алгоритма Кнута—Морриса—Пратта осуществляется примерно в 3 раза меньше сравнений при индексе подстроки 256, чем в реализации стандартного алгоритма.

ЗАКЛЮЧЕНИЕ

Цель данной лабораторной работы была достигнута, а именно были описаны алгоритмы поиска подстроки в строке.

Для достижения поставленной цели были выполнены следующие задачи:

— описаны стандартный алгоритм и алгоритм Кнута—Морриса—Пратта;

— разработано программное обеспечение, реализующее алгоритмы поиска подстроки в строке;

— выбраны инструменты для реализации алгоритмов и замера процессорного времени выполнения реализаций алгоритмов;

— проанализированы затраты реализаций алгоритмов по времени и количество сравнений в реализованных алгоритмах.

В результате замеров времени выполнения реализаций алгоритмов поиска было выявлено, что реализация алгоритма Кнута—Морисса—Пратта медленнее реализации стандартного алгоритма в 3 раза, в случае когда подстрока стоит в начале, при длине подстроки 10 и размеров строки 256. Такой результат обусловлен необходимостью вычислять массив сдвигов для подстроки в реализации алгоритма Кнута—Морисса—Пратта. В то же время, реализация алгоритма Кнута—Морисса—Пратта выигрывает по скорости реализацию стандартного алгоритма в 2.4 раза при длине строк 65536, когда подстрока отсутствует или стоит в конце строки. Такой результат обусловлен тем, что в реализации алгоритма Кнута—Морисса—Пратта используются особенности искомой подстроки, что позволяет осуществлять смещения сразу на несколько шагов.

В результате подсчетов количества сравнений было выявлено, что для реализации алгоритма Кнута—Морриса—Пратта худшим является случай, когда подстрока отсутствует в строке. Для реализации стандартного алгоритма, худшим является случай, когда подстрока стоит на последних позициях в строке. При этом в реализации алгоритма Кнута—Морриса—Пратта осуществляется примерно в 3 раза меньше сравнений при индексе подстроки 256, чем в реализации стандартного алгоритма.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Урвачева В.* Обзор методов информационного поиска // Вестник Таганрогского института имени А. П. Чехова. — 2016. — № 1.
2. *Макконнелл Д.* Анализ алгоритмов. Активный обучающий подход. — Техносфера, 2009.
3. The official home of the Python Programming Language [Электронный ресурс]. — Режим доступа: `https://www.python.org/` (дата обращения: 23.12.2023).
4. time — Time access and conversions [Электронный ресурс]. — Режим доступа: `https://docs.python.org/3/library/time.html` (дата обращения: 23.12.2023).
5. Amd [Электронный ресурс]. — Режим доступа: `https://www.amd.com/en.html` (дата обращения: 23.12.2023).
6. Windows 10 Pro 22h2 64-bit [Электронный ресурс]. — Режим доступа: `https://www.microsoft.com/ru-ru/software-download/windows10` (дата обращения: 23.12.2023).