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 № 3

по курсу «Анализ алгоритмов»

на тему: «Трудоёмкость сортировок»

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 и сбора 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 — описать алгоритмы поиска подстроки в строке. 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;

— проанализировать 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 начинает со сравнения первого символа текста с первым символом подстроки. Если они совпадают, то происходит переход ко второму символу текста и подстроки. При совпадении 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 мы сдвигаем указатель текущего положения в тексте на один символ и заново начинаем 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. Успешное сравнение переводит нас в следующий узел автомата, а в случае несовпадения мы попадаем 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 (в отличие от стандартного алгоритма).

Вывод

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

2 Конструкторский раздел

В данной части работы будет рассмотрен псевдокод стандартного алгоритма поиска подстроки в строке и алгоритма Кнута—Морриса—Пратта.

2.1 Требования к программному обеспечению

Программа должна поддерживать два режима работы: режим массового подсчета количества сравнений и режим поиска подстроки в заданной строке.

Режим массового подсчет количества сравнений должен обладать следующей функциональностью:

— генерировать строки различного размер для проведения замеров;

— осуществлять массовый подсчет количества сравнений, используя сгенерированные данные;

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

К режиму сортировки выдвигается следующий ряд требований:

— возможность работать с массивами разного размера, которые вводит пользователь;

— наличие интерфейса для выбора действий;

— на выходе программы, массив отсортированный тремя алгоритмами по возрастанию.

2.2 Разработка алгоритмов

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

— s — исходная строка;

— $substr$ — искомая подстрока.

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 или массива, 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**
            $flag \leftarrow 1$
            $break$
        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 подстроки в строке 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.

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.

3.3 Реализация 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, а именно: стандартный алгоритм поиска и алгоритм Кнута—Морриса—Пратта.

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
    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
    0 - Выход

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

    Меню:

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

Выберите пункт меню:
```

Рисунок 4.1 – Демонстрация работы программы при поиске подстрок

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: 19.09.2023).