МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Построение и анализ алгоритмов»

Тема: Ахо-Корасик

Вариант 3

Студент гр. 3388 _____ Потоцкий С.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2025

Цель работы

Изучить алгоритм поиска множественных шаблонов в заданном тексте с и без использования символов-джокеров. Данным алгоритмом является алгоритм Ахо-Корасик. Написать реализацию алгоритма, найти асимптотики по времени и по памяти.

Задание 1

Условие:

Разработайте программу, решающую задачу точного поиска набора образцов в тексте.

Входные данные:

Первая строка содержит строку текста T ($1 \le |T| \le 100\,000$).

Вторая строка содержит целое число n ($1 \le n \le 3000$) — количество шаблонов.

Далее идут n строк, каждая из которых содержит шаблон из набора $P = \{p_1, p_2, ..., p_n\}$, длина каждого шаблона от 1 до 75 символов.

Все строки содержат только символы из алфавита {A, C, G, T, N}.

Выходные данные:

Для каждого вхождения шаблона в текст выведите строку из двух чисел:

i — позиция в тексте (нумерация начинается с 1), с которой начинается вхождение шаблона,

p — номер шаблона (нумерация шаблонов также начинается с 1).

Вывод должен быть отсортирован сначала по возрастанию позиции i, затем по возрастанию номера шаблона p.

Пример:

Вход:

NTAG

3

TAGT

TAG

T

Выход:

2 2

2 3

Задание 2

Условие:

Используя реализацию точного множественного поиска, решите задачу точного поиска для одного шаблона с джокером.

В шаблоне встречается специальный символ, именуемый джокером (wild card), который совпадает с любым символом. По заданному шаблону P, содержащему джокеры, необходимо найти все его вхождения в текст T.

Каждый джокер соответствует ровно одному символу, а не подстроке произвольной длины. В шаблоне обязательно должен быть хотя бы один символ, не являющийся джокером, т.е. шаблоны вида ??? недопустимы.

Все строки содержат символы из алфавита {A, C, G, T, N}. Джокер не входит в этот алфавит.

Входные данные:

Строка текста T ($1 \le |T| \le 100\,000$)

Строка шаблона P ($1 \le |P| \le 40$)

Один символ — джокер

Выходные данные:

Выведите строки с позициями вхождений шаблона в текст. Каждая строка должна содержать один номер позиции, начиная с 1. Позиции выводятся в порядке возрастания.

Пример:

Вход:

ACTANCA

A$$A$

$

Выход:

1

Описание алгоритма

Первое задание

Реализует классический алгоритм Ахо-Корасика для поиска нескольких шаблонов в одном тексте.

1. Ввод данных

Считывается текст, количество шаблонов и сами шаблоны.

2. Построение Trie (бор)

Для каждого шаблона вызывается add_string, который добавляет шаблон в бор (префиксное дерево). Каждый узел хранит переходы по символам, флаг терминальности, выходные ссылки и информацию о шаблонах, которые заканчиваются в этом узле.

3. Построение суффиксных и выходных ссылок

Функция build_failure_links строит суффиксные (fail) и выходные (output) ссылки для всех узлов бора. Суффиксная ссылка указывает на самый длинный возможный суффикс, который также присутствует в боре. Выходная ссылка указывает на следующий терминальный узел по цепочке суффиксных ссылок.

4. Поиск всех вхождений шаблонов

Функция search_text проходит по тексту, используя автомат, и находит все вхождения всех шаблонов. Для каждого символа текста происходит переход по бору, при необходимости — по суффиксным ссылкам. Если найден терминальный узел, фиксируется совпадение (позиция и номер шаблона).

5. Сортировка и вывод

Найденные совпадения сортируются по позиции и номеру шаблона. Выводятся все найденные совпадения: позиция в тексте (1-based) и номер шаблона.

6. Дополнительно

Выводится максимальная глубина суффиксных и выходных ссылок автомата.

Второе задание

Это модифицированный алгоритм Ахо-Корасика для поиска шаблона с джокером (wildcard) в тексте. Описание работы:

1. Ввод данных

Считывается текст, шаблон и символ джокера.

2. Разбиение шаблона

Функция split_pattern разбивает шаблон на подстроки между джокерами и запоминает смещения этих подстрок относительно начала шаблона.

3. Построение бора (Trie)

Для каждой непустой подстроки строится бор (префиксное дерево) с помощью функции add_string. Каждый узел хранит переходы по символам, флаг терминальности, выходные ссылки и информацию о подстроках.

4. Построение суффиксных и выходных ссылок

Функция build_failure_links строит суффиксные (fail) и выходные (output) ссылки для всех узлов бора по алгоритму Ахо-Корасика.

5. Поиск подстрок в тексте

Функция search_text проходит по тексту, используя построенный автомат, и находит все вхождения подстрок (без учета джокеров).

Для каждого найденного совпадения сохраняется позиция в тексте и индекс подстроки.

6. Сборка полных совпадений

Для каждого найденного совпадения вычисляется потенциальное начало полного шаблона в тексте с учетом смещения подстроки. Если для некоторой позиции найдены все подстроки (все части шаблона между джокерами), то эта позиция считается совпадением полного шаблона.

7. Вывод результатов

Выводятся позиции, где найдено полное совпадение шаблона с учетом джокеров. Также выводится максимальная глубина суффиксных и выходных ссылок автомата.

Оценка сложности алгоритма:

Время:

O(N+t+M*k), где N – длина строки поиска, t – количество паттернов, M – размер бора, k – количество суффиксальных ссылок.

Память:

O(S) – хранение Бора

O(N) – хранение очереди, суффиксных и прямых ссылок

Итого O(S+N)

Вывод

В ходе выполнения лабораторной работы был изучен алгоритм Ахо-Корасик. Была написана реализация для поиска подстрок в заданном тексте с использованием и без использования символов-джокеров. Были рассчитаны асимптотики работы алгоритма.