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: Алгоритм Ахо-Корасик

Студент гр. 3388 _____ Кулач Д.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2025

Задание

Вариант 4. Реализовать режим поиска, при котором все найденные образцы не пересекаются в строке поиска (т.е. некоторые вхождения не будут найдены; решение задачи неоднозначно).

Задача 1:

Вход:
Первая строка содержит текст T (1< |T|< 100000).

Вторая строка содержит число n (1< n< 3000). Каждая следующая из n строк содержит шаблон из набора P = { p_1, ldots, p_n } (1< |p_i|< 75).

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

Выход:
Все вхождения образцов из P в T.

Каждое вхождение образца в текст представить в виде двух чисел - i p.

Где i - позиция в тексте (нумерация начинается с 1), с которой начинается вхождение образца с номером p (нумерация образцов начинается с 1).

Строки выхода должны быть отсортированы по возрастанию, сначала по номеру позиции, затем по номеру шаблона.

Задача 2:

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

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

Например, образец (ab??c?c) с джокером ? встречается дважды в тексте *zabuccbababcax*.

Символ джокер не входит в алфавит, символы которого используются в (T). Каждый джокер соответствует одному символу, а не подстроке неопределённой длины. В шаблон входит хотя бы один символ не джокер, т.е. шаблоны вида ??? недопустимы. Все строки содержат символы из алфавита ({A, C, G, T, N}).

Вход:

- Текст (T) ((1< |T|< 100000))
- Шаблон (P) ((1< |P|< 40))
- Символ джокера

Выход:

- Строки с номерами позиций вхождений шаблона (каждая строка содержит только один номер).
- Номера должны выводиться в порядке возрастания.

Выполнение работы

Для реализации задания использован алгоритм Ахо-Корасик. Алгоритм выполняет поиск подстрок в тексте с использованием конечного автомата, построенного на боре. Реализация поддерживает три режима работы:

• Поиск по нескольким шаблонам.

• Обработка шаблонов с wildcard-символом.

• Фильтрация неперекрывающихся вхождений.

Структуры данных

TrieNode – Узел бора, содержащий:

• children dict — переходы к дочерним узлам по символам.

• suffix_link — суффиксная ссылка для эффективного перехода при несовпадении.

• output_link — терминальная ссылка для быстрого перехода к ближайшему терминальному узлу.

• pattern_ids — индексы шаблонов, завершающихся в этом узле.

• index — уникальный идентификатор узла.

AhoCorasick – Управляет построением и работой автомата:

• root — корневой узел бора.

• patterns — список шаблонов для поиска.

• nodes — список всех узлов автомата.

Методы и функции

add_pattern(pattern, pattern_id, wildcard=False) – Добавляет шаблон в бор. Если включён режим wildcard, символ ? заменяется на все возможные символы алфавита.

build_links() – Строит суффиксные и терминальные ссылки с помощью BFS. Суффиксные ссылки обеспечивают возврат к альтернативным состояниям при несовпадениях, а терминальные ускоряют поиск совпадений.

search(text, non_overlapping=False) – Ищет все вхождения шаблонов в тексте. Если указан параметр non_overlapping, отбираются только непересекающиеся вхождения. Используются суффиксные и терминальные ссылки для переходов по автомату.

wildcard_search(text, pattern) – Отдельная функция для поиска одного шаблона с символами ?, не используя автомат. Перебирает позиции и проверяет посимвольно, игнорируя несовпадения в ?.

print_structure() – Выводит информацию об автомате: индексы узлов, переходы, ссылки, шаблоны, заканчивающиеся в узле.

Интерфейс пользователя (main)

• Позволяет выбрать одну из трёх задач.

• Считывает входные данные: текст и шаблоны.

• Вызывает нужный режим работы и выводит результаты.

Анализ сложности алгоритма

Временная сложность:

• Построение бора: O(L), где L — суммарная длина всех шаблонов.

• Построение суффиксных и терминальных ссылок: $O(L \cdot k)$, где k — размер алфавита (до 26 в верхнем регистре).

• Поиск в тексте: O(N + t), где N — длина текста, t — количество найденных вхождений.

Общая сложность: $O(L \cdot k + N + t)$

Пространственная сложность:

• Хранение узлов и переходов: $O(L \cdot k)$

• Метаданные (списки шаблонов, ссылки): O(L)

Итог: $O(L \cdot k)$

Тестирование:

Input	Output
ababa 2 aba ba	1 1 2 2 3 1 4 2

Таблица 1. Тестирование решения задания 1

Input	Output
ACGTTACA A?G?	1

Таблица 2. Тестирование решения задания 2

Input	Output
AAABBBCCC 3 AAA AB BBBC	1 4

Таблица 3. Тестирование решения задания 3

Вывод

В ходе работы был разработан и протестирован алгоритм для поиска вхождений шаблона с джокером, без джокера, с режимом непересекающихся вхождений. Алгоритм использует автомат Ахо-Корасик для эффективного поиска подстрок. Тестирование показало, что реализация алгоритма верна.