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

Студентка гр. 3388 _____ Басик В.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 wildcard-шаблонов с исключением символа, фильтрация неперекрывающихся вхождений.

Структуры:

- Node — узел бора, содержащий:

1. Переходы по символам (children).
2. Суффиксные (suffix_link) и терминальные ссылки (terminal_link).
3. Индексы шаблонов, завершающихся в узле (pattern_indices).
4. Длину шаблона (pattern_length).

- AhoCorasick — класс, управляющий автоматом (построение бора, генерация ссылок, поиск).

Методы и функции:

- Инициализация:

1. AhoCorasick() — конструктор, создаёт бор и генерирует ссылки.

2. build_trie() — строит бор из списка шаблонов.

3. build_links() — вычисляет суффиксные и терминальные ссылки.

- Поиск:

1. search() — ищет все вхождения шаблонов в текст, возвращает позиции и индексы.

2. filter_non_overlapping() — фильтрует неперекрывающиеся вхождения.

3. split_pattern() — разбивает wildcard-шаблон на части для поиска.

- Вспомогательные функции:

1. print_trie(), print_automaton() — отладочный вывод структуры бора и автомата.

2. handle_multiple_patterns(), handle_wildcard_pattern(), handle_non_overlapping() — обработка режимов работы.

Режимы работы:

1. Множественные шаблоны — поиск всех вхождений нескольких шаблонов.

2. Wildcard — поиск шаблона с wildcard-символом, исключая запрещённый символ.

3. Неперекрывающиеся вхождения — фильтрация результатов для исключения перекрытий.

Анализ сложности алгоритма

Временная сложность:

1. Построение бора (Trie):

Сложность: O(L), где L — суммарная длина всех шаблонов.

Обоснование: Каждый символ каждого шаблона обрабатывается ровно один раз.

2. Построение суффиксных и терминальных ссылок:

Сложность: O(L·Σ), где Σ — размер алфавита.

Обоснование: Для каждого узла выполняется поиск суффиксной ссылки через переходы, что в худшем случае требует O(Σ) операций.

3. Поиск вхождений в текст:

Сложность: O(M+Z), где M — длина текста, Z — общее количество вхождений.

Обоснование: Каждый символ текста обрабатывается один раз, а терминальные ссылки проверяются за константное время для каждого вхождения.

4. Дополнительные операции:

Фильтрация непересекающихся вхождений: O(ZlogZ).

Разделение шаблона с джокерами: O(K), где K — длина шаблона.

Итоговая сложность:

Основной алгоритм (Ахо-Корасик): O(L·Σ+M+Z).

Режим с джокерами: O(L·Σ+M+Z+K).

Режим непересекающихся вхождений: O(L·Σ+M+ZlogZ).

Сложность по памяти:

O(L + Z + M), где:

L — суммарная длина всех шаблонов,

Z — общее количество найденных вхождений,

M — длина текста.

Тестирование:

Input	Output
makaka 2 ak aka	2 1 2 2 4 1 4 2
NTAG 3 TAGT TAG T	2 2 2 3

Таблица 1. Тестирование решения задания 1

Input	Output
ACTANCA A$$A$ $	1
baobab ao# #	2

Таблица 2. Тестирование решения задания 2

Input	Output
NTAG 3 TAGT TAG T	2

Таблица 3. Тестирование решения задания 3

Выводы:

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.