**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 |  | Дубровин Д.Н. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2025

## Задание

## Вариант 1. На месте джокера может быть любой символ, за исключением заданного.

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:

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 выполнения заданий был использован алгоритм Ахо-Корасик. Алгоритм реализует поиск подстрок при помощи реализации конечного автомата на боре.

Были реализованы следующие структуры и методы на языке программирования Go:

Структуры:

Node — структура реализующая узел.

Trie — структура реализующая бор.

Методы и функции:

NewNode — Создаёт новую вершину с заданным значением и родителем.

addChild — Добавляет дочернюю вершину с указанным символом.

setEnd — Помечает вершину как конец шаблона.

getPath — Возвращает строку пути от вершины до корня.

String — Форматирует информацию о вершине для вывода.

NewTrie — Создаёт новый бор с корневой вершиной.

addWord — Добавляет слово в бор, создавая новые вершины при необходимости.

genSuffixLinks — Генерирует суффиксные и терминальные ссылки для автомата.

generateTrie — Создаёт бор из списка шаблонов.

findMatchesOnTrie — Ищет совпадения шаблонов в тексте, используя бор.

FindAllEntries — Находит все вхождения шаблонов в текст.

isValidWildcardMatch — Проверяет, допустимо ли совпадение с учётом wildcard и запрещённого символа.

FindEntriesWithWildcard — Ищет вхождения шаблона с wildcard, исключая запрещённый символ.

**Анализ сложности алгоритма**

Временная сложность:

* Добавление паттернов в бор происходит за O(L), L — суммарная длина паттернов.
* Итерация по тексту, на каждом шаге проверяется наличия вхождения патерна. Количество детей у каждой вершины бора не превышает длины алфавита k. Следовательно вычисление ссылок займет O(L\*k).
* Поиск в тексте займет O(N + t), N — длина текста, t — количество всех возможных вхождений паттернов в текст.
* Итоговая сложность O(L\*k + N + t).

Сложность по памяти:

* O(L\*k)

**Тестирование:**

|  |  |
| --- | --- |
| Input | Output |
| banana  2  an  ana | 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 |
| abobaboobab  ba#  # | 4  9 |

Таблица 2. Тестирование решения задания 2

|  |  |
| --- | --- |
| Input | Output |
| engineering  in&  &  g | 4 |
| abcvbcfv  bc\*  \*  f | 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 ограниченным джокером в тексте. Алгоритм использует автомат Ахо-Корасик для эффективного поиска подстрок. Тестирование показало, что реализация алгоритма верна.