**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе № 5**

**по дисциплине «Построение и Анализ Алгоритмов»**

Тема: «**Ахо-Корасик**»

|  |  |  |
| --- | --- | --- |
| Студент гр. 3343 |  | Коршков А.А. |
| Преподаватель |  | Жангиров Т. Р. |

Санкт-Петербург

2025

# Цель работы

Написать программы на основе алгоритма Ахо-Корасик для нахождения вхождения всех образцов в строке, а также найти индексы вхождения образцов с джокерами.

# Задания

**№1**

Первая строка содержит текст (T, 1 ≤ ∣T∣ ≤ 100000).

Вторая - число n (1 ≤ n ≤ 3000), каждая следующая из n строк содержит шаблон из набора P={p1, … ,pn}1 ≤ ∣pi∣ ≤ 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 номера позиции, затем номера шаблона.

**Sample Input:**

NTAG

3

TAGT

TAG

T

**Sample Output:**

2 2

2 3

**№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 Р в текст Т.

Например, образец аb??с? с джокером ?? встречается дважды в тексте xabvccbababcax.

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.

**Sample Input:**

ACTANCA

A$$A$

$

**Sample Output:**

1

**Задание варианта:**

**№7.** Вывод графического представления автомата.

**Примечания для варианта:**

1) В автомате должны быть и использоваться не только суффиксные ссылки, но и конечные ссылки

2) Для обоих заданий на программирование должны быть версии кода с выводом промежуточных данных. В них, в частности, должны выводиться построение бора и автомата, построенный автомат (в виде, например, описания каждой вершины автомата), процесс его использования.

## Основные теоретические положения

**Описание алгоритмов:**

КМП очень быстро может найти вхождение строки, но если строк слишком много, то он не очень эффективен. Алгоритм Ахо-Корасик позволяет находить.

Алгоритм создает префиксное дерево из букв искомых подстрок. Вершины, в которых искомая подстрока заканчивается называет терминальной и выделяется специальным цветом в графическом представлении. Суффиксная ссылка вершины u – это вершина v, такая что строка v является максимальным суффиксом строки u. Для корня и вершин, исходящих из корня, суффиксной ссылкой является корень. Для остальных вершин осуществляется переход по суффиксной ссылке родителя и, если оттуда есть ребро с заданным символом, суффиксная ссылка назначается в вершину, куда это ребро ведет. Cуффиксные ссылки находятся не автоматически для каждой вершины, а вычисляются во время работы программы при обращении к специальному методу.

Алгоритм Ахо-Корасик иногда называют «расширенной версией КМП». Схожесть этих алгоритмов заключается в нахождении .... Только если в КМП была специальная префикс функция для вычисления результата в векторной форме, то здесь мы используем бор (дерево строк) для нахождения нужной строки.

**Оценка сложности по памяти и операциям**

## Выполнение работы

**Описание работы**

Для решения заданий были написаны два класса Vertex и Trie, которые представляют вершину автомата и сам бор.

В классе Vertex описаны следующие методы:

\_\_init\_\_(self, id\_: int, alpha: int, parent: "Vertex" or None = None, pchar: str or None = None) -> None – констурктор класса вершины Vertex. В качестве аргументов принимает номер вершины, размер алфавита, родительскую вершину (если есть), символ родительской вершины (если есть).

is\_terminal(self) -> bool – возвращает True, если вершина является терминальной

@is\_terminal.setter

is\_terminal(self, value: bool) -> None – позволяет установить флаг для терминальной вершины

id(self) -> int – возвращает присвоенный идентификатор вершины

sufflink(self) -> "Vertex" or None – возвращает суффиксную ссылку на вершину, если суффиксная ссылка была вычислена для данной вершины.

@sufflink.setter

sufflink(self, value) -> None - позволяет установить значение для суффиксной ссылки

parent(self) -> "Vertex" or None – возвращает родительскую вершину, если это не корень бора.

pchar(self) -> str or None – возвращает символ родительской вершины, если это не корень бора.

\_\_str\_\_(self) -> str – возвращает информацию в строковом виде для вершины

В классе Trie описаны следующие методы:

\_\_init\_\_(self, alpha: int = 5) -> None – конструктор для класса автомата Ахо-Корасик. На вход принимает размер алфавита (по умолчанию 5, для заданного алфавита {A: 0, C: 1, G: 2, T: 3, N: 4})

size(self) -> int – возвращает кол-во вершин в дереве

last(self) -> Vertex – ввозвращает последнюю вершину в дереве

alpha(self) -> int – возвращает размер алфавита

vertices(self) -> list[Vertex] – возвращает список вершин

root(self) -> возвращает корень дерева

add(self, s: str, pattern\_num: int) -> None – добавляет образец в дерево

search(self, s: str) -> list[tuple[int, int]] - проверяет, есть ли строка в дереве и возвращает

get\_link(self, v: Vertex) -> Vertex – находит и возвращает суффиксную ссылку для вершины.

go(self, v: Vertex, char: str) -> Vertex - Возвращает вершину, в которую ведет переход по символу char из вершины v.

precompute\_sufflinks(self) -> None – предварительно вычисляет все суффиксные ссылки (нужно для визуализации автомата)

visualize(self, file\_name: str = "aho\_corasick") -> None - Создает графическое представление автомата Ахо-Корасик и сохраняет его в png файл. Также создаётся легенда для графа. На вход принимает имя файла, в который нужно сохранить визуализацию.

Также для класса Trie есть вспомогательные внутренние методы:

\_num(c: str) -> int – возвращает номер буквы в заданном алфавите из 5 букв: {A: 0, C: 1, G: 2, T: 3, N: 4}

\_char(idx: int) -> int – возвращает символ буквы, соответствующий номеру в алфавите.

В файла main.py есть несколько функций для решения заданий:

main() -> None – главная функция, которая запускает функции для решения заданий

aho\_corasick\_search() -> None – функция, которая ищет позиции вхождения всех заданных образцов в тексте.

search\_with\_wildcard() -> None – функция, которая ищет индексы вхождения образца с джокером.

# Тестирование

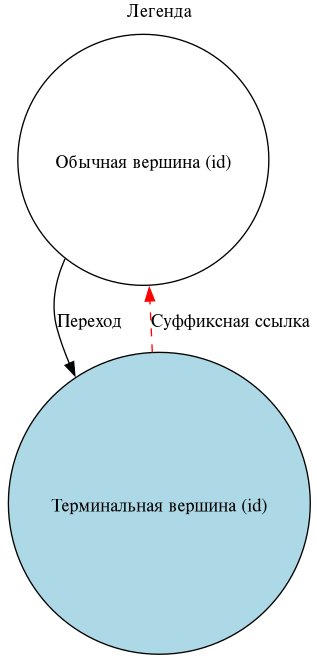


Рисунок – Легенда графа

Таблица 1 – Тестирование алгоритмов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Комментарии |
| 1 | NTAG  3  TAGT  TAG  T | 2 2  2 3 | Алгоритм Ахо-Корасик. Терминальные вершины отмечены корректно, суффиксные ссылки также корректны. Идентификаторы вершин отмечаются корректно. |
| 2 | ACGT  3  ACGT  CG  GT | 1 1  2 2  3 3 | Алгоритм Ахо-Корасик. Два последних образца содержатся в одном длинном образце. |
| 3 | ACGTNGGTCCG  5  ACG  CGT  TNG  CC  A | 1 1  1 5  2 2  4 3  9 4 | Алгоритм Ахо-Корасик. Алгоритм хорошо справляется с построением автомата для большого количества образцов. |
| 4 | ACTANCA  A$$A$  $ | 1 | Ахо-Корасик с джокером. Символ в маске только A. |
| 5 | ACGTNNTGCA  TNNXG  X | 4 | Ахо-Корасик с джокером. Корректно распознаёт маску в виде другой буквы. Дерево содержит в себе части TNN и G из маски. |
| 6 | ACGTNNTGCA  T&  & | 4  7 | Ахо-Корасик с джокером. Находит несколько вхождений в строке. |

# Выводы

Изучен принцип работы алгоритма Ахо-Корасик. Написаны программы, корректно решающие задачу поиска набора подстрок в строке, в также программа поиска подстроки с джокером. Также была написана визуализация для автомата Ахо-Корасик в графическом представлении.

ПРИЛОЖЕНИЕ А  
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

"""

Главный файл программы.

Название файла: vertex.py

Название файла: trie.py

Название файла: requirements.txt

pylint

graphviz