Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий

Кафедра ИС

# ОТЧЁТ

по лабораторной работе №9

по дисциплине «Технологии обработки информации»

на тему «Реализация алгоритмов поиска по тексту»

Выполнил:

ст. гр. ИС/б-21-2-о

Мовенко К. М.

Проверил:

Карлусов В.Ю.

Севастополь

2024

# ЦЕЛЬ РАБОТЫ

Научиться реализовывать на выбранном языке программирования алгоритмы поиска по тексту: прямой поиск; алгоритм Кнута, Морриса и Пратта; алгоритм Бойера-Мура.

# ВАРИАНТ ЗАДАНИЯ

Написать программу на выбранном языке программирования, реализующую алгоритмы для поиска подстроки в строке. Программа должна запрашивать имя входного файла. Оценить трудоёмкость рассматриваемых алгоритмов.

# ХОД РАБОТЫ

Прежде всего был создан файл text.txt, по которому будет производится поиск.

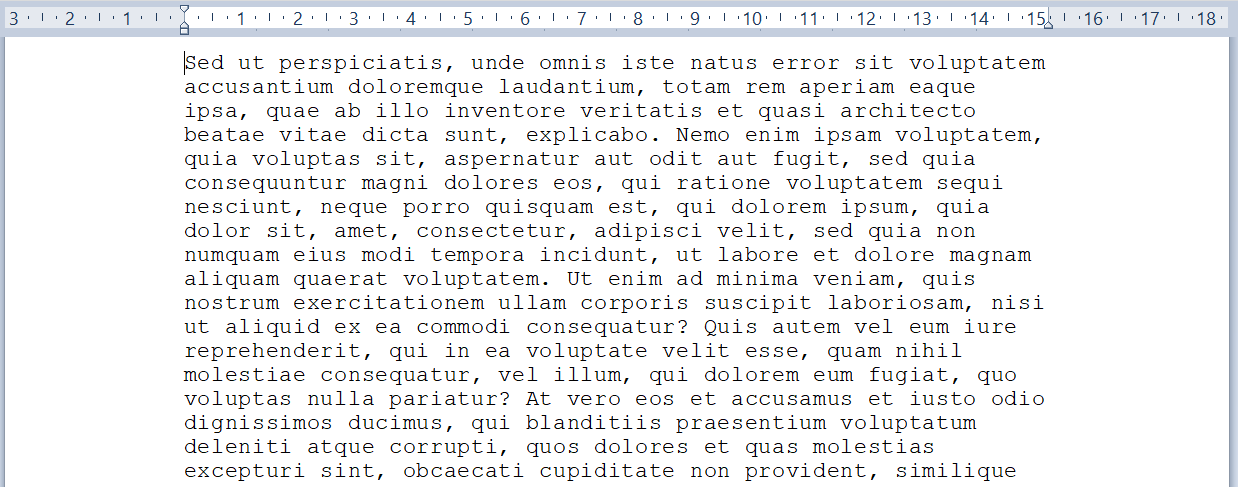


Рисунок 1 – Входные данные

Затем был написан алгоритм прямого поиска (листинг 1).

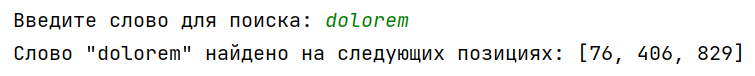


Рисунок 2 – Прямой поиск

Затем был написан алгоритм поиска Кнута, Морриса и Пратта (листинг 2).

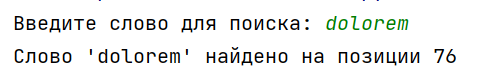


Рисунок 3 – Алгоритм Кнута, Морриса и Пратта

Наконец, был написан алгоритм Бойера-Мура (листинг 3).

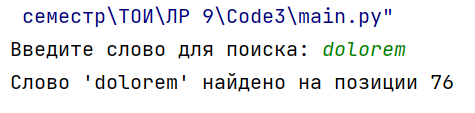


Рисунок 4 – Алгоритм Бойера-Мура

# программный код

Листинг 1 – Алгоритм прямого поиск

def direct\_search(text, pattern):  
 len\_text = len(text)  
 len\_pattern = len(pattern)  
 positions = []  
  
 for i in range(len\_text - len\_pattern + 1):  
 match = True  
 for j in range(len\_pattern):  
 if text[i + j] != pattern[j]:  
 match = False  
 break  
 if match:  
 positions.append(i)  
  
 return positions  
  
with open('text.txt', 'r') as file:  
 text = file.read()  
  
pattern = input("Введите слово для поиска: ")  
positions = direct\_search(text, pattern)  
  
if positions:  
 print(f'Слово "{pattern}" найдено на следующих позициях: {positions}')  
else:  
 print(f'Слово "{pattern}" не найдено в тексте.')

Листинг 2 – Алгоритм поиска Кнута, Морриса и Пратта

def kmp\_search(pattern, text):  
 def compute\_prefix\_function(pattern):  
 pi = [0]  
 j = 0  
 for i in range(1, len(pattern)):  
 while j > 0 and pattern[j] != pattern[i]:  
 j = pi[j-1]  
 if pattern[j] == pattern[i]:  
 j += 1  
 pi.append(j)  
 return pi  
  
 pi = compute\_prefix\_function(pattern)  
 q = 0  
 for i in range(len(text)):  
 while q > 0 and pattern[q] != text[i]:  
 q = pi[q-1]  
 if pattern[q] == text[i]:  
 q += 1  
 if q == len(pattern):  
 return i - (q - 1)  
 return -1  
  
word\_to\_search = input("Введите слово для поиска: ")  
  
with open('text.txt', 'r', encoding='utf-8') as file:  
 contents = file.read()  
 index = kmp\_search(word\_to\_search, contents)  
 if index != -1:  
 print(f"Слово '{word\_to\_search}' найдено на позиции {index}")  
 else:  
 print(f"Слово '{word\_to\_search}' не найдено")

Листинг 3 – Алгоритм поиска Бойера-Мура

def boyer\_moore\_search(pattern, text):  
 m = len(pattern)  
 n = len(text)  
  
 shift\_table = [m] \* 256  
 for i in range(m - 1):  
 shift\_table[ord(pattern[i])] = m - 1 - i  
  
 i = m - 1  
 j = m - 1  
  
 while i < n:  
 if text[i] == pattern[j]:  
 if j == 0:  
 return i  
 else:  
 i -= 1  
 j -= 1  
 else:  
 i = i + shift\_table[ord(text[i])]  
 j = m - 1  
  
 return -1

# контрольные вопросы

## Что такое поиск подстроки в строке?

Поиск подстроки в строке − это процесс нахождения одной строки (подстроки) внутри другой строки (основной строки). Это распространённая задача в области обработки текста и анализа данных.

## Принципы работы алгоритма прямого поиска

Алгоритм прямого поиска начинает сравнивать подстроку с основной строкой слева направо. Если символы совпадают, он переходит к следующему символу. Если все символы подстроки совпадают, то подстрока найдена. В противном случае подстрока сдвигается на одну позицию вправо и процесс повторяется.

## Принципы работы алгоритма Кнута, Морриса и Пратта

Алгоритм Кнута-Морриса-Пратта (КМП) использует префикс-функцию для определения, на сколько символов следует сдвинуть подстроку при несовпадении. Это позволяет алгоритму пропустить уже проверенные символы и ускорить процесс поиска.

## Принципы работы алгоритма Бойера и Мура.

Алгоритм Бойера-Мура начинает сравнение подстроки с основной строкой справа налево. Если символы не совпадают, подстрока сдвигается на величину, определенную таблицей смещений. Если все символы подстроки совпадают, то подстрока найдена.

## Как оценить трудоемкость алгоритма поиска по тексту?

Трудоёмкость алгоритма поиска по тексту обычно оценивается в терминах “О-большое” (Big O notation), которое описывает наихудший случай времени выполнения алгоритма в зависимости от размера входных данных. Например, алгоритм прямого поиска имеет трудоемкость O((n-m+1)\*m), а алгоритмы КМП и Бойера-Мура - O(n+m).

## Каким образом задается величина сдвига в алгоритме Кнута, Морриса и Пратта?

В алгоритме КМП величина сдвига определяется с помощью префикс-функции. Если символы не совпадают, подстрока сдвигается так, чтобы префикс подстроки совпал с суффиксом уже проверенной части текста.

## Каким образом задается величина сдвига в алгоритме Бойера-Мура?

В алгоритме Бойера-Мура величина сдвига определяется с помощью таблицы смещений. Если символы не совпадают, подстрока сдвигается на величину, указанную в таблице смещений для данного символа. Если символа нет в подстроке, подстрока сдвигается на свою длину.

# ВЫВОД

В ходе выполнения лабораторной работы была рассмотрена оценка трудоёмкости алгоритмов поиска подстрок в строках, а также реализованы все возможные алгоритмы на практике.