МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **ІКНІ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 11
З дисципліни: *“Алгоритми та структури даних”*
На тему: *“Алгоритм пошуку КМП”*

Лектор:
доц. каф. ПЗ
Коротєєва Т.О.

Виконав:
ст. гр. ПЗ – 22
Ясногородський Н.В.

Прийняв:
асист. каф. ПЗ
Франко А.В.

« ____ » _____ 2022 р.
∑= _____ _____.

Львів – 2022

Тема роботи: Алгоритм пошуку КМП

Мета роботи: Навчитися застосовувати алгоритм пошуку КМП при розв'язуванні задач та перевірити його ефективність на різних масивах даних. Експериментально визначити складність алгоритму

Теоретичні відомості

Маємо масив символів S з n елементів (текст) та масив P з m - взірець. Необхідно знайти перше входження взірця в масив. Схема алгоритму полягає у поступовому порівнянні взірця з текстом та зсуву по тексту па кількість співпавших символів у разі знайденого неспівпадіння. Алгоритм використовує просте спостереження, що коли відбувається неспівпадіння тексту і взірця, то взірець містить у собі достатньо інформації для того, щоб визначити де наступне входження може початися, таким чином пропускаючи кількаразову перевірку попередньо порівняних символів. Попередньо проводиться дослідження взірця та для кожного його підрядка визначається префікс-суфікс-функція. Для цього вираховується найдовший початок підрядка, який співпадає з його кінцем.

Алгоритм КМП

КМП 1. Встаповити i=0.

КМП 2. j=0, d=1.

КМП 3. Поки j<m, i<n

Перевірка: якщо S[i]=P[j], то d++, i++.j++ поки d != m.

КМП 4. Інакше встановити зсув взірця на d-D[d] позицій по тексту . Перейти на крок КМП 2.

КМП 5. Кінець.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

15. Дано текст що складається з n стрічок. Визначити чи є серед даного тексту цифри, якщо є вилучити їх з тексту. Знайти задане слово в тексті.

ВИКОНАННЯ РОБОТИ

Код програми:

```
"""
15. Дано текст що складається з n стрічок.
Визначити чи є серед даного тексту цифри, якщо є вилучити їх з тексту.
Знайти задане слово в тексті.

Виміряти час пошуку,
а також час (окремо) додаткового опрацювання тексту та/чи взірця,
якщо таке є.
"""

from lab11.kmp_search import kmp_search

TEXT = """Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.
It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged.
It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages,
and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum."""


def get_input_text():
    text = []
```

```
    for char in TEXT:
        if not char.isnumeric():
            text.append(char)
    text_str = "".join(text)
    print(f"Text: \n{text_str}\n")
    search_pattern = input("Please enter search pattern: ")
    return text_str, search_pattern


if __name__ == "__main__":
    while True:
        text, search_pattern = get_input_text()
        kmp_search(text, search_pattern)
        print()

import time


def kmp_search(text, search_pattern):
    start_time = time.time()
    arr = _init_preprocessed(search_pattern)
    has_found = _kmp_search(arr, search_pattern, text)
    if has_found:
        print(f"Found '{search_pattern}'")
    else:
        print(f"'{search_pattern}' not found")
    print("--- %s seconds ---" % (time.time() - start_time))


def _init_preprocessed(w):
    arr = [0] * len(w)
```

```
    m = len(w)
    i = 0
    j = 1

    # No proper prefix for string of length 1:
    arr[0] = 0

    while j < m:
        if w[i] == w[j]:
            i += 1
            arr[j] = i
            j += 1

        # The first character didn't match:
        elif i == 0:
            arr[j] = 0
            j += 1

        # Mismatch after at least one matching character:
        else:
            i = arr[i - 1]

    return arr


def _kmp_search(arr, w, s):
    # Initialising variables:
    i = 0
    j = 0
    m = len(w)
    n = len(s)
```

```
# Start search:
while i < m and j < n:
    if w[i] == s[j] and i == m - 1:
        # Last character matches → Substring found:
        return True
    elif w[i] == s[j]:
        print(f"Chars matched: w[{i}]=s[{j}]={s[j]}")
        i += 1
        j += 1
    else:
        # Character didn't match → Backtrack:
        if i != 0:
            i = arr[i - 1]
            print(f"shift {i}")
        else:
            j += 1
            print("shift 1")

# Substring not found:
return False
```

ПРОТОКОЛ РОБОТИ

```
Text:
cabcbbb

Preprocessed prefix-suffix array: [0, 0, 0, 1, 2]

text pointer is 0:
        cabcbbb
        +------
matched:
        cabca
        +----

text pointer is 1:
        cabcbbb
        -+-----
matched:
        cabca
        -+---

text pointer is 2:
        cabcbbb
        --+----
matched:
        cabca
        --+--

text pointer is 3:
        cabcbbb
        ---+---
matched:
        cabca
        ---+-

text pointer is 4:
        cabcbbb
        ----+--
not matched, backtrack:
        cabca
        -+---
```

ВИСНОВКИ

Під час виконання лабораторної роботи я навчився застосовувати алгоритм пошуку КМР при розв’язуванні задач та перевірити його ефективність на різних масивах даних. Експериментально визначив складність алгоритму.