**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи № 12

**З дисципліни:** *“Алгоритми та структури даних”*

**На тему:** *“Алгоритм пошуку Бойєра Мура”*

**Лектор:**

доц. каф. ПЗ

Коротєєва Т.О.

**Виконав:**

ст. гр. ПЗ – 22

Солтисюк Д.А.

**Прийняв:**

асист. каф. ПЗ

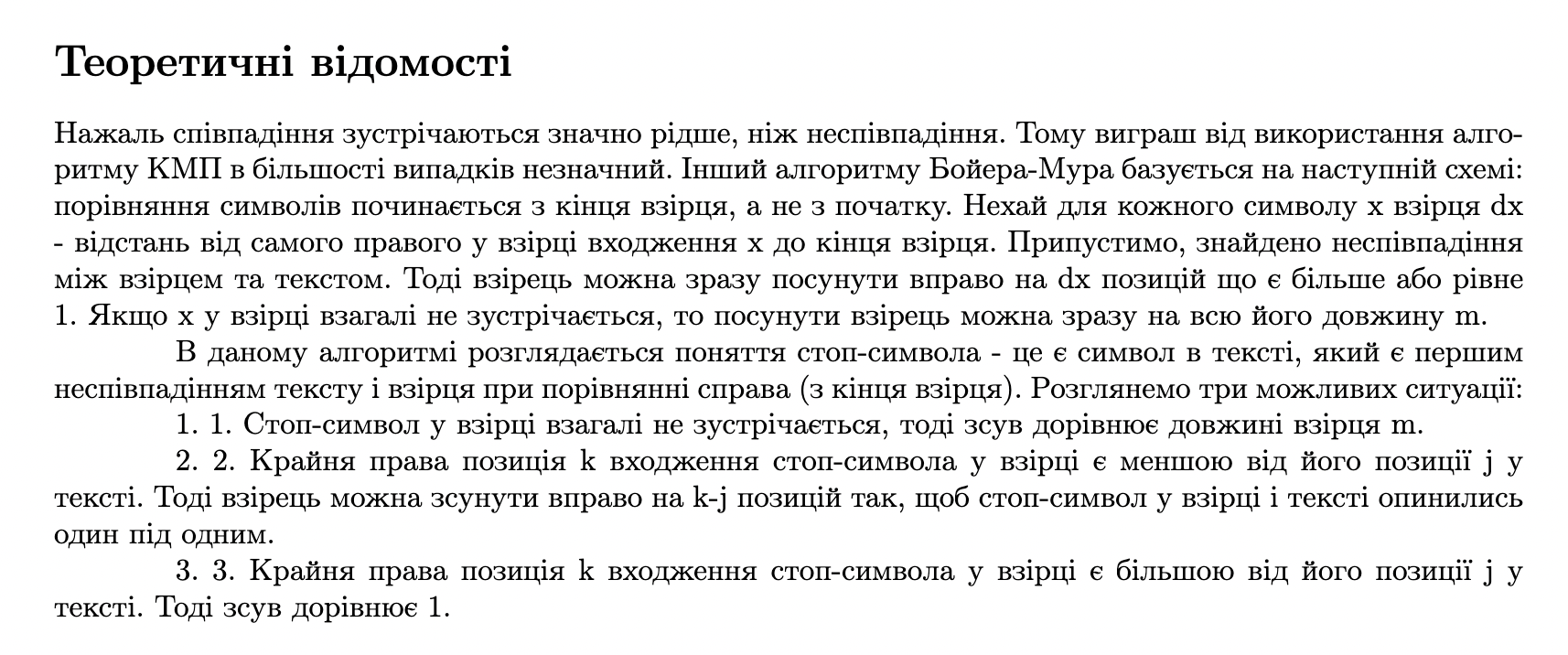
Франко А.В.

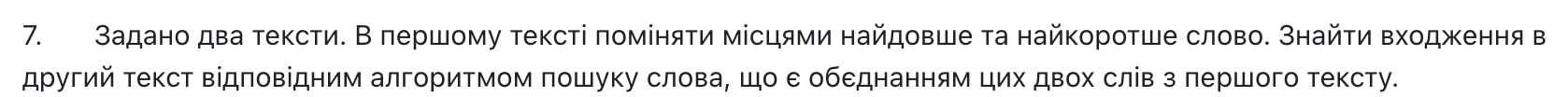
« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_\_ .

Львів – 2022

**Тема роботи:** Алгоритм пошуку Бойєра Мура

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

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

**ВИКОНАННЯ РОБОТИ**

**Код програми:**

Файл main.py:

**from** lab12.boyer\_moore **import** bm\_search

TEXT1 = """Clash Royale is a free-to-play video game (afree-to-play)"""

SEARCH\_PATTERN\_1 = "afree-to-play"

TEXT\_2 = "cabcbbbcabca"

SEARCH\_PATTERN\_2 = "cabca"

**def** get\_minmax\_word(text\_to\_process: str):

text = text\_to\_process.split()

**return** min(text, key=len) + max(text, key=len)

**if** \_\_name\_\_ == "\_\_main\_\_":

text = TEXT1

search\_pattern = get\_minmax\_word(text)

bm\_search(text, search\_pattern)

**print**()

Файл boyer\_moore.py:

"""

1) Boyer-Moore is a smarter naive exact matching algorithm, that skips

unnecessary checks. Has two rules that you need to follow.

2) Bad character rule:

|------------------

A B C D A [B] C D C A (text) |

A B C D C [A] (pattern) |

|----^--------------------------|

| |\_\_\_\_\_|

| |

| searching for b in here

| found here -|

------------------|

|

moving to match this letter

|

A B C D A B C D C [A] (text)

A B C D C [A] (pattern)

|\_\_\_\_\_\_\_\_\_\_\_|

|

exact match

2) Good suffix rule:

|-------------------|

A B C D A [B] C D A B C (text) | the bad charater rule

B C D A B [C] (pattern) | allows to skip 1 alignment

|------------------------|

|

| |------------------|

| \_\_|\_\_ |

| | | |

|--- A B C D A [B C] D A B C (text) |

| B C D A [B C] (pattern) |

| |\_\_\_| |

| | |

| |---------------------------|

|

|---------------|

|

|

A B C D A B C D A B C (text)

B C D A B C (pattern)

"""

**from** collections **import** defaultdict

**def** bad\_char\_map(pattern):

"""

The preprocessing function for

Boyer Moore's bad character heuristic

"""

bad\_chars = defaultdict(**lambda**: -1)

*# Fill the actual value of last occurrence*

**for** i **in** range(len(pattern)):

bad\_chars[pattern[i]] = i

*# return initialized list*

**print**(f"{bad\_chars=}\n")

**return** bad\_chars

**def** bm\_search(txt, pattern):

"""

A pattern searching function that uses Bad Character

Heuristic of Boyer Moore Algorithm

"""

m = len(pattern)

n = len(txt)

bad\_chars = bad\_char\_map(pattern)

*# s is shift of the pattern with respect to text*

**def** \_print\_state(s, j):

t\_display = ["-"] \* n

t\_display[s + j] = "\*"

**print**(f"text and search\_pattern:\n{txt}\n{''.join(t\_display)}")

shift\_display = [" "] \* (s)

**print**(f"{''.join(shift\_display)}{pattern}\n")

s = 0

**while** s <= n - m:

j = m - 1

*# Keep reducing index j of pattern while*

*# characters of pattern and text are matching*

*# at this shift s*

print\_state\_again = **True**

**while** j >= 0 **and** pattern[j] == txt[s + j]:

**print**("char match")

print\_state\_again = **False**

\_print\_state(s, j)

j -= 1

**if** print\_state\_again:

\_print\_state(s, j)

*# If the pattern is present at current shift,*

*# then index j will become -1 after the above loop*

**if** j < 0:

**print**("full match at shift = {}!".format(s))

"""

Shift the pattern so that the next character in text

aligns with the last occurrence of it in pattern.

The condition s+m < n is necessary for the case when

pattern occurs at the end of text

"""

s += m - bad\_chars[txt[s + m]] **if** s + m < n **else** 1

**print**(f"shift={s}")

**else**:

"""

Shift the pattern so that the bad character in text

aligns with the last occurrence of it in pattern. The

max function is used to make sure that we get a positive

shift. We may get a negative shift if the last occurrence

of bad character in pattern is on the right side of the

current character.

"""

s += max(1, j - bad\_chars[txt[s + j]])

**print**(f"shift={s}")

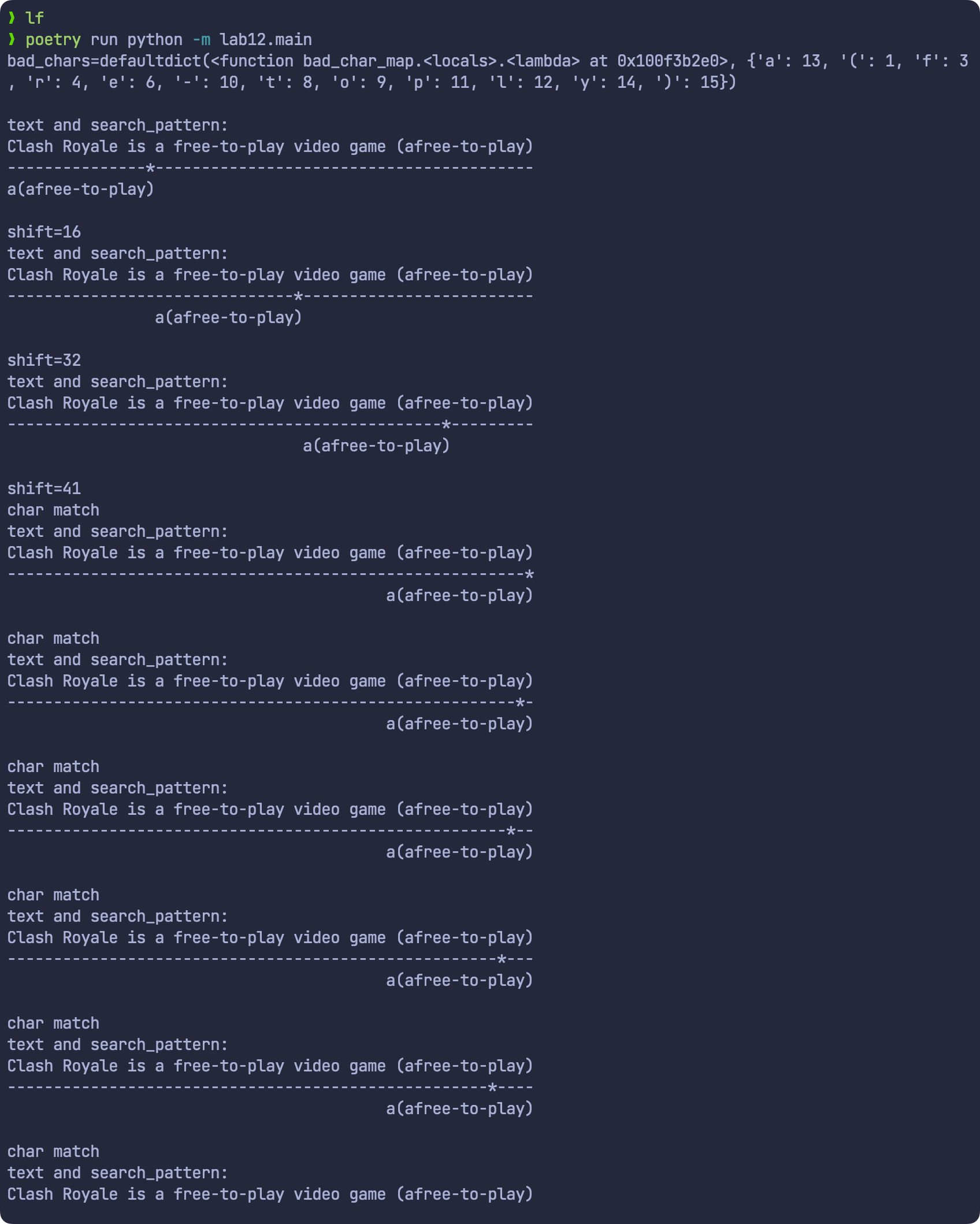
**ПРОТОКОЛ РОБОТИ**

Рис. 1. Огляд роботи програми

**ВИСНОВКИ**

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