# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **ІКНІ** Кафедра **ПЗ** 

# **3BIT**

До лабораторної роботи N 12

**3 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 p. Σ= \_\_\_\_\_ 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

# Теоретичні відомості

Нажаль співпадіння зустрічаються значно рідше, ніж неспівпадіння. Тому виграш від використання алгоритму КМП в більшості випадків незначний. Інший алгоритму Бойера-Мура базується на наступній схемі: порівняння символів починається з кінця взірця, а не з початку. Нехай для кожного символу х взірця dx - відстань від самого правого у взірці входження х до кінця взірця. Припустимо, знайдено неспівпадіння між взірцем та текстом. Тоді взірець можна зразу посунути вправо на dx позицій що є більше або рівне 1. Якщо х у взірці взагалі не зустрічається, то посунути взірець можна зразу на всю його довжину m.

В даному алгоритмі розглядається поняття стоп-символа - це є символ в тексті, який є першим неспівпадінням тексту і взірця при порівнянні справа (з кінця взірця). Розглянемо три можливих ситуації:

- 1. 1. Стоп-символ у взірці взагалі не зустрічається, тоді зсув дорівнює довжині взірця т.
- 2. 2. Крайня права позиція k входження стоп-символа у взірці є меншою від його позиції ј у тексті. Тоді взірець можна зсунути вправо на k-j позицій так, щоб стоп-символ у взірці і тексті опинились один під одним.
- 3. 3. Крайня права позиція k входження стоп-символа у взірці є більшою від його позиції ј у тексті. Тоді зсув дорівнює 1.

### ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

7. Задано два тексти. В першому тексті поміняти місцями найдовше та найкоротше слово. Знайти входження в другий текст відповідним алгоритмом пошуку слова, що є обєднанням цих двох слів з першого тексту.

#### 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)

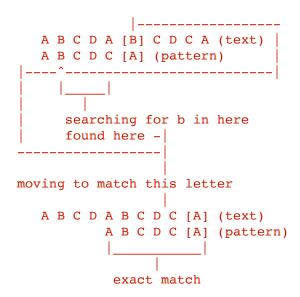
if __name__ == "__main__":
    text = TEXT1
    search_pattern = get_minmax_word(text)
```

```
bm_search(text, search_pattern)
print()
```

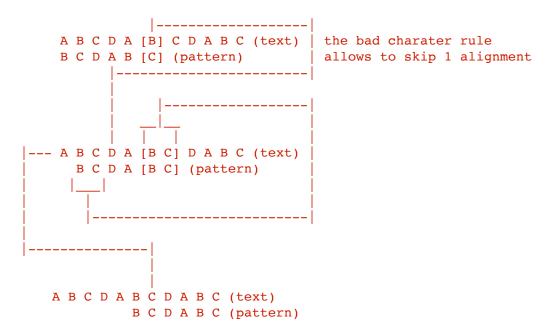
# Файл boyer moore.py:

0.00

- 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:



2) Good suffix rule:



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:</pre>
        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 \ge 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

```
lf
poetry run python -m lab12.main
bad_chars=defaultdict(<function bad_char_map.<locals>.<lambda> at 0x100f3b2e0>, {'a': 13, '(': 1, 'f': 3
text and search_pattern:
Clash Royale is a free-to-play video game (afree-to-play)
a(afree-to-play)
shift=16
text and search_pattern:
Clash Royale is a free-to-play video game (afree-to-play)
              a(afree-to-play)
shift=32
text and search_pattern:
Clash Royale is a free-to-play video game (afree-to-play)
                              a(afree-to-play)
shift=41
char match
text and search_pattern:
Clash Royale is a free-to-play video game (afree-to-play)
                                       a(afree-to-play)
char match
text and search_pattern:
Clash Royale is a free-to-play video game (afree-to-play)
                                        a(afree-to-play)
char match
text and search_pattern:
Clash Royale is a free-to-play video game (afree-to-play)
                                      a(afree-to-play)
char match
text and search_pattern:
Clash Royale is a free-to-play video game (afree-to-play)
                                       a(afree-to-play)
char match
text and search_pattern:
Clash Royale is a free-to-play video game (afree-to-play)
                                       a(afree-to-play)
char match
text and search_pattern:
Clash Royale is a free-to-play video game (afree-to-play)
```

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.