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

Інститут **КНІТ** Кафедра **ПЗ** 

#### **3BIT**

До лабораторної роботи  $\mathbb{N}$  11 **На тему**: "*Алгоритм пошуку КМП*" **З дисципліни**: "Алгоритми та структури даних"

> **Лектор**: доцент кафедри ПЗ Коротеєва Т.О.

> > Виконав:

студент групи ПЗ-22 Коваленко Д.М.

Прийняв:

асистент кафедри  $\Pi 3$  Франко А.В.

Тема. Алгоритм пошуку КМП.

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

### Лабораторне завдання

Розробити програму, яка:

2. Задано два тексти. В першому тексті знайти найкоротше слово і знайти його входження в другий текст відповідним алгоритмом пошуку.

### Теоретичні відомості

Маємо масив символів S з п 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 на 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. Встановити 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. Кінепь.
```

## Хід роботи

```
fn first indexof needle <N: AsRef < [u8] >> (& self, needle: N) -> Option < usize > {
let needle = needle.as ref();
let pattern_table = Self::pattern_table(needle);
let haystack = &self.as ref();
if needle.is empty() {
    return Some(0);
}
let mut s: usize = 0;
let mut i: usize = 0;
loop {
    if i >= needle.len() {
        println!("Found '{} ': ...{}...", from_utf8(&needle).unwrap(),
from utf8(&haystack[s.checked sub(3).unwrap or(0)..usize::min(s+needle.len()
+3, haystack.len())).unwrap());
        return Some(s);
    if s + i >= haystack.len()  {
         println!("Not found");
        return None;
    if needle[i] = haystack[s + i] 
         println!("{} = {}", needle[i] as char, haystack[s + i] as char);
         i += 1;
    } else {
```

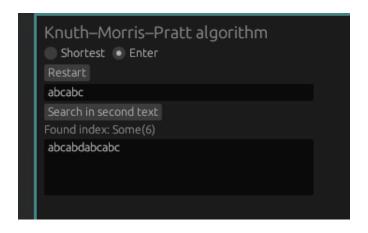


Рис. 1: Вигляд програми

```
D(P,i) = [0, 0, 0, 1, 2, 3]
a == a
b == b
c == c
a == a
b == b
c != d
shift 5
a != d
shift 1
a != d
shift 1
a != d
shift 1
a == a
b == b
c == c
a == a
b == b
c == c
Found 'abcabc': ..abdabcabc..
```

Рис. 2: Покроковий вивід

# Висновоки

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.