МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп’ютерних систем та інфокомунікацій

Кафедра комп’ютерних систем, мереж і кібербезпеки (503)

Лабораторна робота № *6*

|  |  |
| --- | --- |
|  | *Дослідження алгоритмів пошуку підрядка в рядку* |
|  | (назва лабораторної роботи) |
| з дисципліни | *Моделі та структури даних* |
|  | (шифр)  ХАІ**.**503**.**525a**.**03О**.**123-Комп'ютерна інженерія**,** ПЗ №9629619 |

|  |  |  |
| --- | --- | --- |
| Виконав студент гр. | 525а | *Литвиненко А.В.* |
| 29.11.2022 | (№ групи) | (П.І.Б.) |
| (підпис, дата) |  |  |
| Перевірив | канд. техн. наук, доцент | |
|  |  | *А. В. Шостак* |
| (підпис, дата) |  | (П.І.Б.) |

Харків – 2022

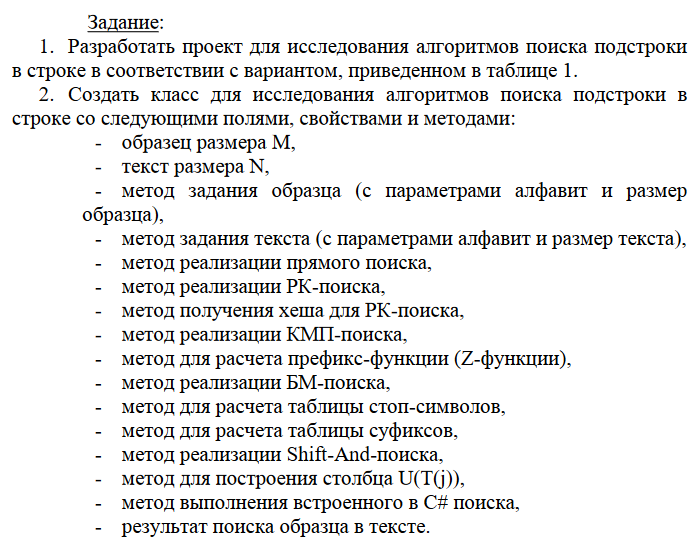
**Тема роботи:** дослідження алгоритмів пошуку підрядка в рядку.

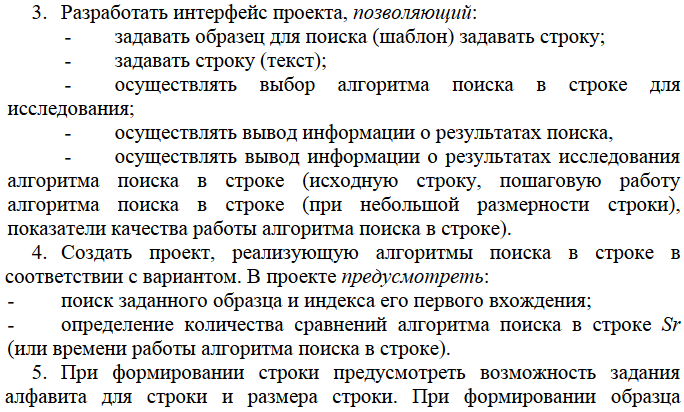
**Варіант 5**

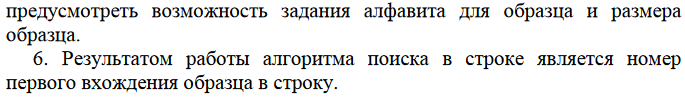
**Задача 1**

**Частина 1**. Постановка завдання

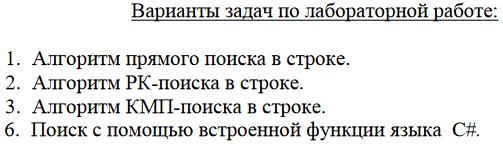
**Умова:**







**Умова з додатка:**



**Варіанти:**



**Основні вбудовані методи пошуку:**



**Частина 2**. Схема класів

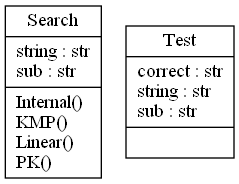


Рисунок 1 - схема класів

**Частина 3**. Текст програми

Відповідно до розробленого алгоритму в середовищі Microsoft Visual Studio була написана програма, яка наведена нижче.

Main.py

from time import time

class Search:

def \_\_init\_\_(self, STRING: str = "", SUB: str = "") -> None:

self.string = STRING

self.sub = SUB

def Linear(self):

op = 0

tmp = ""

i = 0

while i < len(self.string):

if self.sub == tmp:

return (True, op)

if self.sub[0] == self.string[i]:

j = 0

tmp = ""

while j < len(self.sub):

if self.sub[j] == self.string[i]:

tmp += self.sub[j]

j += 1

i += 1

op += 1

else:

break

i += 1

op += 1

return (False, op)

def PK(self):

op = 0

pattern = self.sub

text = self.string

q = 2147483647

d = 10

m = len(pattern)

n = len(text)

p = 0

t = 0

h = 1

i = 0

j = 0

for i in range(m-1):

h = (h\*d) % q

op += 1

# Calculate hash value for pattern and text

for i in range(m):

p = (d\*p + ord(pattern[i])) % q

t = (d\*t + ord(text[i])) % q

op += 1

# Find the match

for i in range(n-m+1):

op += 1

if p == t:

for j in range(m):

op += 1

if text[i+j] != pattern[j]:

break

j += 1

if j == m:

# print("Pattern is found at position: " + str(i+1))

return (True, op)

if i < n-m:

t = (d\*(t-ord(text[i])\*h) + ord(text[i+m])) % q

if t < 0:

t = t+q

return (False, op)

def KMP(self):

op = 0

pat = self.sub

txt = self.string

M = len(pat)

N = len(txt)

# create lps[] that will hold the longest prefix suffix

# values for pattern

lps = [0]\*M

j = 0 # index for pat[]

# Preprocess the pattern (calculate lps[] array)

computeLPSArray(pat, M, lps)

i = 0 # index for txt[]

while i < N:

op += 1

if pat[j] == txt[i]:

i += 1

j += 1

if j == M:

# print ("Found pattern at index", str(i-j))

return (True, op)

j = lps[j-1]

# mismatch after j matches

elif i < N and pat[j] != txt[i]:

# Do not match lps[0..lps[j-1]] characters,

# they will match anyway

if j != 0:

j = lps[j-1]

else:

i += 1

return (False, op)

def Internal(self):

return (False, 0) if self.string.find(self.sub) == -1 else (True, 0)

def computeLPSArray(pat, M, lps):

len = 0 # length of the previous longest prefix suffix

lps[0] # lps[0] is always 0

i = 1

# the loop calculates lps[i] for i = 1 to M-1

while i < M:

if pat[i]== pat[len]:

len += 1

lps[i] = len

i += 1

else:

# This is tricky. Consider the example.

# AAACAAAA and i = 7. The idea is similar

# to search step.

if len != 0:

len = lps[len-1]

# Also, note that we do not increment i here

else:

lps[i] = 0

i += 1

def Time(self, func):

start = time()

data = func(self)

end = time() - start

print("Runtime: ", round(end, 10))

return data

def main():

s = Search()

strings = open("tests.txt").readlines()

sub = strings[0]

strings = strings[1:]

s.sub = sub

# N = 10, 50, 100, 150, 200, 250, 300, 400, 500

for Method in (Search.Linear, Search.PK, Search.KMP, Search.Internal):

print("Current method:\t", Method.\_\_name\_\_, end="\n\n")

for string in strings:

s.string = string

result = Time(s, Method)

if result[0] == True:

print("[+] Passed!")

print("[+] Operation Count : ", result[1])

else:

print("[-] FAILED!!!")

print("Expected (%s) got %s" % (True, result[0]))

print()

print("\n")

print("<<< SUBS >>>")

print("="\*40)

# N = 10, 50, 100, 150, 200, 250, 300, 400, 500

subs = [x.replace("\n", "") for x in open("subs.txt").readlines()]

string = subs[0]

s.string = string

subs = subs[1:]

for Method in (Search.Linear, Search.PK, Search.KMP, Search.Internal):

print("Current method:\t", Method.\_\_name\_\_, end="\n\n")

for sub in subs:

s.sub = sub

result = Time(s, Method)

if result[0] == True:

print("[+] Passed!")

print("[+] Operation Count : ", result[1])

else:

print("[-] FAILED!!!")

print("Expected (%s) got %s" % (True, result[0]))

print()

print("\n")

main()

tests.txt

hello

ZNvhelloas

lYZsLNbfFjOFqcBhelloAgSQaMWLCrixNBAthoUpPDtdNKdzSu

bOHsDKdpnloMggoqNMomZhelloDEhFNBxGDkqJWZTDqVFtoqMlbaXwBXuftXyYduhlUBKVXevGaDqjhKcVucXEulgEQhSjITLLsh

OogDxyzNkFmHpiodLOkkFCihelloyyEtXooHRcaaPONrODEmcdIlicQcGCESwXpvTrhoihRhdztTfetQIlUpbbRHQAuUKvvxTXyXIHRgySuBgjeHEQJPsoKAtkldsEaCVLqIUYXkmBdByrAHcBSmGc

SSNVSQXcGTFWQCFwokFhVmOWazQMThelloOtQQuvVeBONdLynQRQGlEddXbOWKxQZJSVLfbPFvSZArZqqvSFivWxcDqjJTSTbpujiVexKupRwwOwfyHkjqsJtgBySIFYakxAeJmnBeCrRfzZTvtrLHrXyhqkOaOdNNBblsgPEkGFswCbOzTLsQRfWSkLLUpnYZqLhPMV

FcGBytDaSoRmhrCFVtDeNRZhpjixGblDAWAMDgKnBwEiLehelloVjBeUaJVRPTYqpZPPMIMLfDAHTKQNlpqydkOZPlmdpeubhkxtkIdGrTTPMEMuKXwSamRLciQgGixsPzvdbnACwdbrYSwiStQzoDvbJONGBuBOiUykoNYrXinFpEQVcNZLIWPMKWrUDekBpbdrpdbfxQqnkbPiRmhtqrkYhnJvTKwBsETrSssZYtkImxJxOuItBgKmUa

MhfLkiSCLVChelloyQTAvPeLxlQhOUgXxTfNwRRTKheMXWpfmoxrfVjEwohMSydPcvECRVbOerOAZknGPaFwyQSAvXYwyVPbyUgoyCRjfvEDRxvezteNBJKMOxovdRvwLRcMsyyCxWFLKUhcFzmLhWwPoStABAjItdidExjiPmhRdtXBHITBZlTvlxSMUksMufYFBfWpcdgHWwWDTMDQirTqejBotSBHQaaZzlTcrWPDFOjoEeWqQmpCoXKyExQDEiKyFnYioXUxaMbfaFwyWdHPpGbICAfQftimFSgSEuwT

AUDFwwtZBUIwMFfGVSRNfroEtqXbuEjdvPKBInOplhelloKKqgmiYDJghxtqMXqNTPlTNbCsmINOMfUYDyUPsytNGPqVXBDhUuEAqTiZgKbsuLLeVPfxVKNVIwChjkRHeaeCSYsqstGyFQaqOgwCLGTvzUIOhCMnvfZObJjVxTosvpoFehgnIxLNmXBWjvUuTJnnxYSDRDLVNowAORzZZMNVipyLAMeRatqBrujJoZfFnXIrDrqsRnPMwJSNqAhGvubiBsSpkbPWCMXYmmQyNeJLwVvWkXjNALdnDuaxQNFRMLhOdTGJqLUZIgCPVkSJvtKFLhReGWxgvCnqNFthUbvkyDPVrIvsuaBXIVNFHtXvGiliVQWNHQMmptLDCQzinZZDAdMQwGFfuBmg



subs.txt



AAAAVtCnDp

AAAAVtCnDpwjuaKWdjSTSOhPHTEYoNmgwCgFotEqhRdCwpiPlD

AAAAVtCnDpwjuaKWdjSTSOhPHTEYoNmgwCgFotEqhRdCwpiPlDnPMPRnoqUCNJbXUCUFzUHzGIPGyphMPfBJbiiRIsvsEeaUBbDY

AAAAVtCnDpwjuaKWdjSTSOhPHTEYoNmgwCgFotEqhRdCwpiPlDnPMPRnoqUCNJbXUCUFzUHzGIPGyphMPfBJbiiRIsvsEeaUBbDYCFZvZQPHuPvUntQzenJWvdmUsEdwjjUceiMGgHOnxnQEUgyiBj

AAAAVtCnDpwjuaKWdjSTSOhPHTEYoNmgwCgFotEqhRdCwpiPlDnPMPRnoqUCNJbXUCUFzUHzGIPGyphMPfBJbiiRIsvsEeaUBbDYCFZvZQPHuPvUntQzenJWvdmUsEdwjjUceiMGgHOnxnQEUgyiBjRkqWYZeJSJjBLWkTLzqDIFGKoAJTrMxFtaRmBgKttVUMqGYFEL

AAAAVtCnDpwjuaKWdjSTSOhPHTEYoNmgwCgFotEqhRdCwpiPlDnPMPRnoqUCNJbXUCUFzUHzGIPGyphMPfBJbiiRIsvsEeaUBbDYCFZvZQPHuPvUntQzenJWvdmUsEdwjjUceiMGgHOnxnQEUgyiBjRkqWYZeJSJjBLWkTLzqDIFGKoAJTrMxFtaRmBgKttVUMqGYFELSxahppYGoSEBlnLdrHoacATAXfoneJzXHSiGzUqfodipFvLIRG

AAAAVtCnDpwjuaKWdjSTSOhPHTEYoNmgwCgFotEqhRdCwpiPlDnPMPRnoqUCNJbXUCUFzUHzGIPGyphMPfBJbiiRIsvsEeaUBbDYCFZvZQPHuPvUntQzenJWvdmUsEdwjjUceiMGgHOnxnQEUgyiBjRkqWYZeJSJjBLWkTLzqDIFGKoAJTrMxFtaRmBgKttVUMqGYFELSxahppYGoSEBlnLdrHoacATAXfoneJzXHSiGzUqfodipFvLIRGhqNElhTkuPzRFSENtQVtxBhXFbgolPaFhuhENkWFAtSSIPLlJK

AAAAVtCnDpwjuaKWdjSTSOhPHTEYoNmgwCgFotEqhRdCwpiPlDnPMPRnoqUCNJbXUCUFzUHzGIPGyphMPfBJbiiRIsvsEeaUBbDYCFZvZQPHuPvUntQzenJWvdmUsEdwjjUceiMGgHOnxnQEUgyiBjRkqWYZeJSJjBLWkTLzqDIFGKoAJTrMxFtaRmBgKttVUMqGYFELSxahppYGoSEBlnLdrHoacATAXfoneJzXHSiGzUqfodipFvLIRGhqNElhTkuPzRFSENtQVtxBhXFbgolPaFhuhENkWFAtSSIPLlJKfhDJibmsMGOUWVkARfjftqmSwSsbNkIYOijVgbjewQshqpOWyjSyoZtphelloBbtCyeSsJIuoFhpbfcMTOWzPjbYimzrZIYVYIdk



**Частина 4.** Порівняння

0 – означає, що прямує до нуля

N – кількість елементів

T – час виконання (в секундах)

S – кількість операцій

Таблиця 1 – Алгоритм прямого пошуку в рядку, довжина прикладу М=5

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10 | 50 | 100 | 150 | 200 | 250 | 300 | 400 | 500 |
| T | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.001 |
| S | 9 | 21 | 27 | 29 | 35 | 52 | 17 | 47 | 154 |

Таблиця 2 – Алгоритм PK пошуку в рядку, довжина прикладу М=5

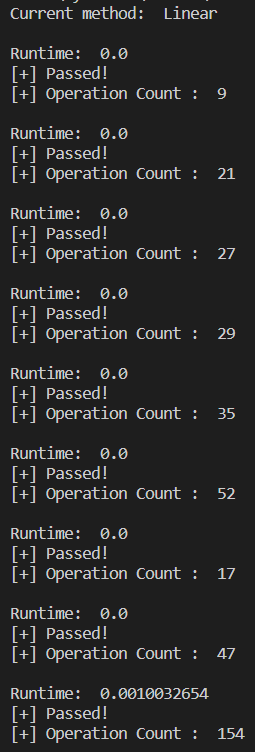
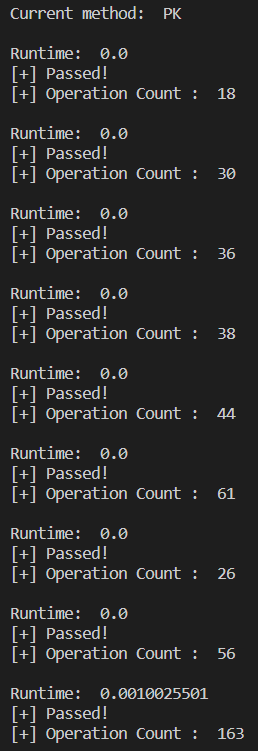
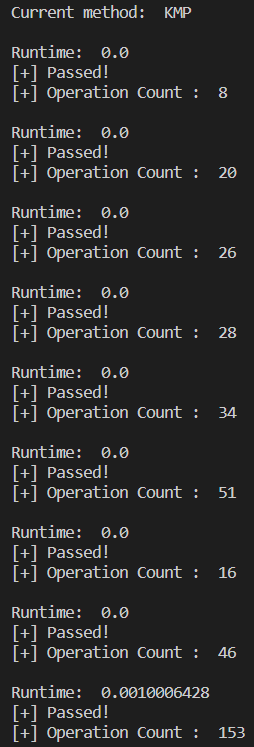
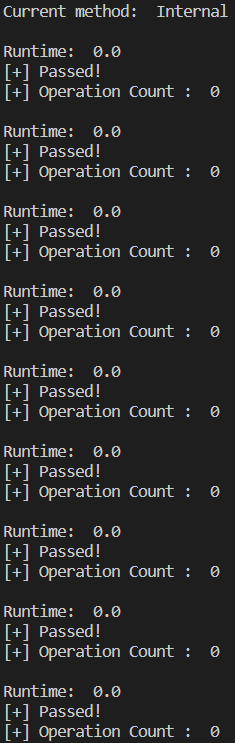
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10 | 50 | 100 | 150 | 200 | 250 | 300 | 400 | 500 |
| T | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.001 |
| S | 18 | 30 | 36 | 38 | 44 | 61 | 26 | 56 | 163 |

Таблиця 3 – Алгоритм KMP пошуку в рядку, довжина прикладу М=5

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10 | 50 | 100 | 150 | 200 | 250 | 300 | 400 | 500 |
| T | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.001 |
| S | 8 | 20 | 26 | 28 | 34 | 51 | 16 | 46 | 153 |

Таблиця 4 – Алгоритм internal пошуку в рядку, довжина прикладу М=5

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 10 | 50 | 100 | 150 | 200 | 250 | 300 | 400 | 500 |
| T | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| S | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Таблиця 5 – Алгоритм прямого пошуку в рядку, довжина рядку N=5000

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M | 10 | 50 | 100 | 150 | 200 | 250 | 300 | 400 | 500 |
| T | 0.003 | 0.002 | 0.002 | 0.0018 | 0.005 | 0.003 | 0.005 | 0.005 | 0.005 |
| S | 3992 | 4082 | 4032 | 4082 | 4132 | 4182 | 4232 | 4382 | 4482 |

Таблиця 6 – Алгоритм PK пошуку в рядку, довжина рядку N=5000

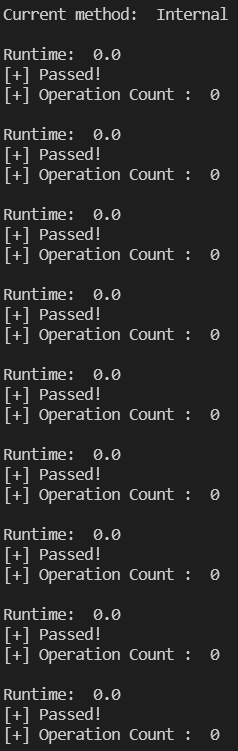
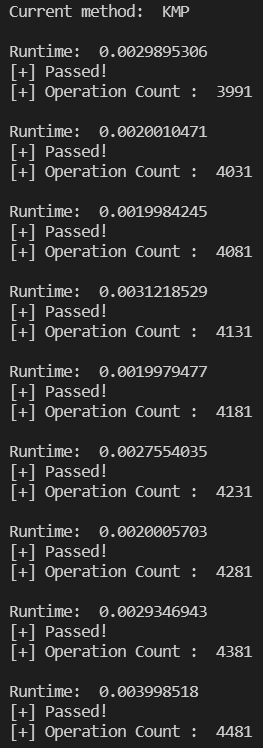
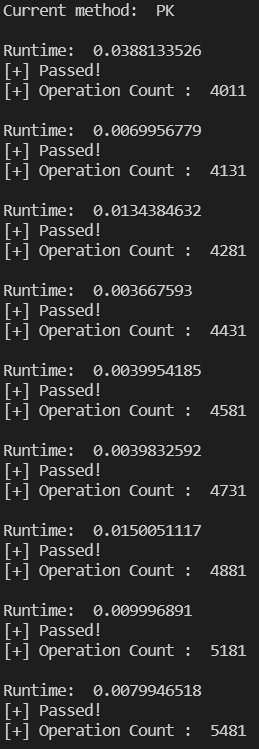
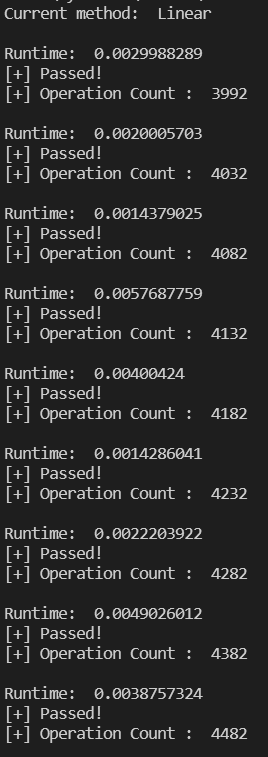
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M | 10 | 50 | 100 | 150 | 200 | 250 | 300 | 400 | 500 |
| T | 0.004 | 0.007 | 0.134 | 0.003 | 0.003 | 0.003 | 0.015 | 0.009 | 0.008 |
| S | 4011 | 4131 | 4281 | 4431 | 4581 | 4731 | 4881 | 5181 | 5481 |

Таблиця 7 – Алгоритм KMP пошуку в рядку, довжина рядку N=5000

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M | 10 | 50 | 100 | 150 | 200 | 250 | 300 | 400 | 500 |
| T | 0.003 | 0.002 | 0.002 | 0.003 | 0.002 | 0.002 | 0.002 | 0.003 | 0.004 |
| S | 3991 | 4031 | 4081 | 4131 | 4181 | 4231 | 4281 | 4381 | 4481 |

Таблиця 8 – Алгоритм internal пошуку в рядку, довжина рядку N=5000

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M | 10 | 50 | 100 | 150 | 200 | 250 | 300 | 400 | 500 |
| T | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| S | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



**Висновки**

Під час цієї лабораторної роботи я досліджував алгоритмів пошуку підрядка у рядку та реалізував проект для досліджування алгоритмів пошуку підрядка у рядку у зв’язку з варіантом.