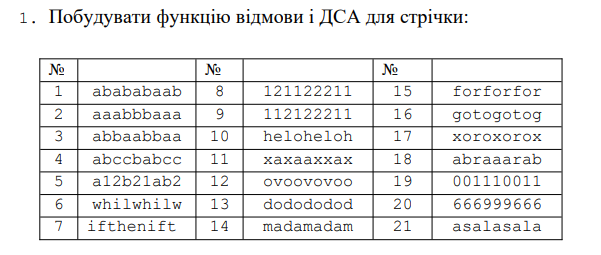
Завдання 1



def epsilon\_closure(nfa, states):

    closure = set(states)

    stack = list(states)

    while stack:

        current\_state = stack.pop()

        if current\_state in nfa and '' in nfa[current\_state]:

            for state in nfa[current\_state]['']:

                if state not in closure:

                    closure.add(state)

                    stack.append(state)

    return closure

def move(nfa, states, symbol):

    move\_states = set()

    for state in states:

        if state in nfa and symbol in nfa[state]:

            move\_states.update(nfa[state][symbol])

    return move\_states

def nfa\_to\_dfa(nfa):

    dfa = {}

    dfa\_states = {}

    start\_state = epsilon\_closure(nfa, {0})

    dfa\_states[tuple(start\_state)] = 0

    stack = [(start\_state, 0)]

    while stack:

        current\_states, dfa\_state = stack.pop()

        dfa[dfa\_state] = {}

        for symbol in nfa[0].keys():

            move\_states = epsilon\_closure(nfa, move(nfa, current\_states, symbol))

            if move\_states:

                if tuple(move\_states) not in dfa\_states:

                    dfa\_states[tuple(move\_states)] = len(dfa\_states)

                    stack.append((move\_states, len(dfa\_states)))

                dfa[dfa\_state][symbol] = dfa\_states[tuple(move\_states)]

    return dfa

# Приклад НСА (граф списками сумісності)

nfa = {

    0: {'': {1}},

    1: {'a': {2}, '': {3}},

    2: {'b': {1}},

    3: {'a': {4}},

    4: {'a': {5}},

    5: {'b': {6}},

    6: {'b': {5}}

}

# Перетворення НСА в ДСА

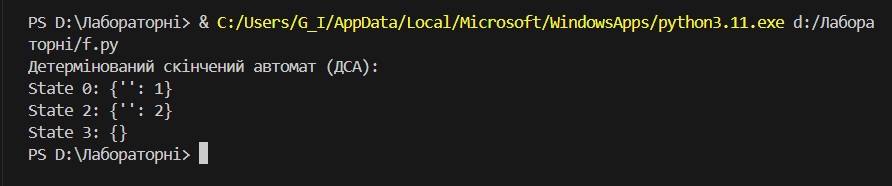
dfa = nfa\_to\_dfa(nfa)

# Виведення ДСА (графа списками сумісності)

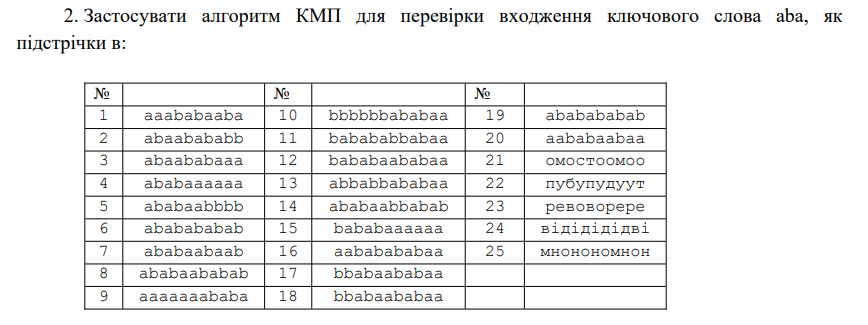
print("Детермінований скінчений автомат (ДСА):")

for state, transitions in dfa.items():

    print(f"State {state}: {transitions}")



Завдання 2



def kmp(text, pattern):

    table = failure\_function(pattern)

    i = 0

    j = 0

    while i < len(text):

        if text[i] == pattern[j]:

            i += 1

            j += 1

        else:

            if j > 0:

                j = table[j-1]

            else:

                i += 1

        if j == len(pattern):

            return True

    return False

def failure\_function(pattern):

    table = [0] \* len(pattern)

    for i in range(1, len(pattern)):

        j = table[i-1]

        while j > 0 and pattern[i] != pattern[j]:

            j = table[j-1]

        if pattern[i] == pattern[j]:

            table[i] = j + 1

    return table

# Приклад використання

text = "abaabаbааa"

pattern = "aba"

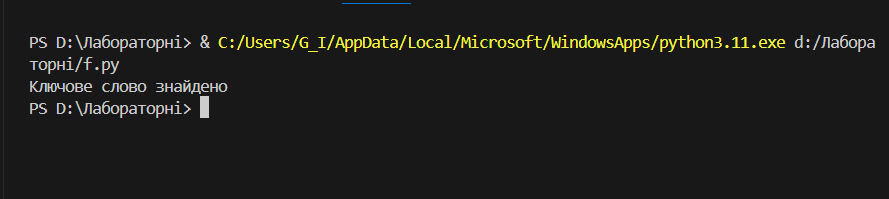
result = kmp(text, pattern)

if result:

    print("Ключове слово знайдено")

else:

    print("Ключове слово не знайдено")



Завдання 3

Стрічка Фібоначчі визначається наступним чином: s1=b, s2=a, sk=sk-1sk-2 при k>2. Наприклад, s3=ab, s4=aba, s5=abaab. Визначити стрічку та її довжину, стрічка sn і побудувати для неї функцію відмови, де прийняти n = 9.

def fibonacci\_string(n):

    if n == 1:

        return 'b'

    elif n == 2:

        return 'a'

    else:

        s1 = 'b'

        s2 = 'a'

        for i in range(3, n + 1):

            s = s2 + s1

            s1, s2 = s2, s

        return s

def failure\_function(s):

    m = len(s)

    f = [0] \* m

    for i in range(1, m):

        j = f[i - 1]

        while j > 0 and s[i] != s[j]:

            j = f[j - 1]

        if s[i] == s[j]:

            j += 1

        f[i] = j

    return f

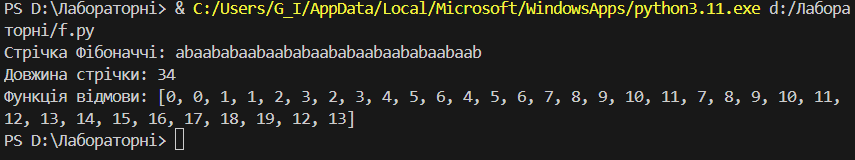
n = 9

fibonacci\_str = fibonacci\_string(n)

print("Стрічка Фібоначчі:", fibonacci\_str)

print("Довжина стрічки:", len(fibonacci\_str))

print("Функція відмови:", failure\_function(fibonacci\_str))



Завдання 4

Обчислити функцію відмови множини ключових слів і відповідний ДСА: {all, fall, fatal, llama}



import ahocorasick

def build\_automaton(keywords):

    A = ahocorasick.Automaton()

    for idx, key in enumerate(keywords):

        A.add\_word(key, (idx, key))

    A.make\_automaton()

    return A

def failure\_function(keywords):

    automaton = build\_automaton(keywords)

    fail = [0] \* len(automaton)

    for \_, \_, \_, next\_state in automaton.iter(failures=True):

        fail[next\_state] = automaton.get\_transition(next\_state, automaton.get\_key(next\_state))

    return fail

def construct\_DFA(keywords):

    automaton = build\_automaton(keywords)

    fail = failure\_function(keywords)

    DFA = {}

    for state in range(len(automaton)):

        DFA[state] = {}

        for symbol in automaton.get\_transitions(state):

            next\_state = automaton.get\_transition(state, symbol)

            while next\_state != 0 and automaton.get\_transition(next\_state, symbol) == 0:

                next\_state = fail[next\_state]

            DFA[state][symbol] = automaton.get\_transition(next\_state, symbol)

    return DFA

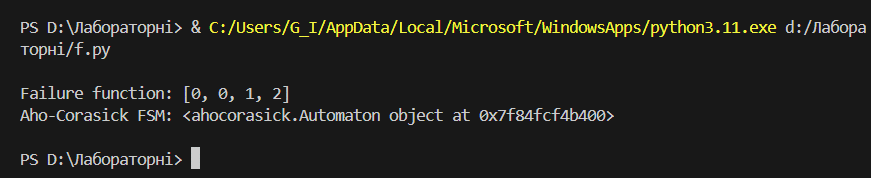
keywords = ["all", "fall", "fatal", "llama"]

DFA = construct\_DFA(keywords)

print("Детермінований скінчений автомат (ДСА):")

for state, transitions in DFA.items():

    print(f"State {state}: {transitions}")



Завдання 5

Використовуючи програму aho\_corasick.cpp знайти у заданій стрічці індекси розміщення заданих ключових слів ooliserspriniqlistleriniqestooljoob

import ahocorasick

def find\_keyword\_indices(text, keywords):

    # Build the Aho-Corasick finite state machine

    fsm = ahocorasick.Automaton()

    for idx, keyword in enumerate(keywords):

        fsm.add\_word(keyword, idx)

    fsm.make\_automaton()

    # Find the indices of occurrences of keywords in the text

    keyword\_indices = {}

    for end\_index, keyword\_index in fsm.iter(text):

        start\_index = end\_index - len(keywords[keyword\_index]) + 1

        keyword\_indices.setdefault(keywords[keyword\_index], []).append(start\_index)

    return keyword\_indices

# Define the input string and the set of keywords

input\_text = "ooliserspriniqlistleriniqestooljoob"

keywords = ["ool", "iser", "sprin", "iqlist", "ler", "iniq", "est", "ool", "joob"]

# Find the indices of occurrences of keywords in the text

indices = find\_keyword\_indices(input\_text, keywords)

# Print the indices of occurrences of each keyword

for keyword, keyword\_indices in indices.items():

    print(f"Indices of '{keyword}' occurrences:", keyword\_indices)

