КРИПТОГРАФІЯ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ №3

Експериментальна оцінка ентропії на символ джерела відкритого тексту

ФБ-13 Владислав Садохін та Данило Розумовський

Мета роботи

Набуття навичок частотного аналізу на прикладі розкриття моноалфавітної підстановки; опанування прийомами роботи в модулярній арифметиці.

Порядок виконання роботи

0. Уважно прочитати методичні вказівки до виконання комп’ютерного практикуму.

1. Реалізувати підпрограми із необхідними математичними операціями:

обчисленням оберненого елементу за модулем із використанням розширеного алгоритму Евкліда, розв’язуванням лінійних порівнянь. При розв’язуванні порівнянь потрібно коректно обробляти випадок із декількома розв’язками, повертаючи їх усі.

2. За допомогою програми обчислення частот біграм, яка написана в ході виконання комп’ютерного практикуму №1, знайти 5 найчастіших біграм запропонованого шифртексту (за варіантом).

3. Перебрати можливі варіанти співставлення частих біграм мови та частих біграм шифртексту (розглядаючи пари біграм із п’яти найчастіших). Для кожного співставлення знайти можливі кандидати на ключ (a,b) шляхом розв’язання системи (1).

4. Для кожного кандидата на ключ дешифрувати шифртекст. Якщо шифртекст не є змістовним текстом російською мовою, відкинути цього кандидата.

5. Повторювати дії 3-4 доти, доки дешифрований текст не буде змістовним.

Код:

```
import os
from collections import Counter

alphabet = "абвгдежзийклмнопрстуфхцчшщыьэюя"
def ext_gcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, x, y = ext_gcd(b % a, a)
```

```
        return (g, y - (b // a) * x, x)


def mod_inverse(a, m):
    g, x, y = ext_gcd(a, m)
    if g != 1:
        return None  # Оберненого елемента не існує
    else:
        return (x % m + m) % m


def solve_linear_congruences(a, b, m):
    gcd, x0, y0 = ext_gcd(a, m)

    if b % gcd == 0:
        x = x0 * (b // gcd)
        x = x % (m // gcd)
        solutions = []

        # Знаходимо всі розв'язки, додаючи m//gcd
        for i in range(gcd):
            solutions.append(x + i * (m // gcd))
        return solutions
    else:
        return None  # Рівняння немає розв'язку

def is_letter_in_alphabet(letter):
    return letter in alphabet

def is_bigram_in_alphabet(bigram):
    return all(is_letter_in_alphabet(letter) for letter in bigram)

def calculate_non_overlapping_bigram_frequencies(text):
    text = text.lower()
    bigram_count = Counter([text[i:i + 2] for i in range(0, len(text) - 1, 2) if
is_bigram_in_alphabet(text[i:i + 2])])
    total_bigram_count = sum(bigram_count.values())
    bigram_frequency = {bigram: count / total_bigram_count for bigram, count in
bigram_count.items()}
    return bigram_frequency


def find_affine_key_candidates(encrypted_text):
    bigram_frequency =
calculate_non_overlapping_bigram_frequencies(encrypted_text)
    top_bigrams_encrypted = [bigram for bigram, _ in
sorted(bigram_frequency.items(), key=lambda item: item[1], reverse=True)[:5]]
    bigrams = ["ст", "но", "то", "на", "ен"]
    key_candidates = []
    for i in range(len(bigrams)):
        for j in range(len(top_bigrams_encrypted)):
            for k in range(len(bigrams)):
                for n in range(len(top_bigrams_encrypted)):
                    x1 = (alphabet.index(bigrams[i][0]) * 31 +
alphabet.index(bigrams[i][1])) % 961
                    y1 = (alphabet.index(top_bigrams_encrypted[j][0]) * 31 +
alphabet.index(top_bigrams_encrypted[j][1])) % 961

                    x2 = (alphabet.index(bigrams[k][0]) * 31 +
alphabet.index(bigrams[k][1])) % 961
                    y2 = (alphabet.index(top_bigrams_encrypted[n][0]) * 31 +
alphabet.index(top_bigrams_encrypted[n][1])) % 961

                    x1_minus_x2 = x1 - x2
```

```
                    mod_inv = mod_inverse(x1_minus_x2, 961)
                    if mod_inv is not None:
                        a_coefficient = (y1 - y2) * mod_inv % 961
                        b_coefficient = (y1 - x1 * a_coefficient) % 961
                        key_candidates.append((a_coefficient, b_coefficient))

    return key_candidates

def decrypt_affine_cipher(input_text, key, i):

    with open(input_text, 'r', encoding='utf-8') as file:
        encrypted_text = file.read()

    text = [encrypted_text[i:i + 2] for i in range(0, len(encrypted_text) - 1, 2)
if is_bigram_in_alphabet(encrypted_text[i:i + 2])]

    a, b = key
    mod_inverse_a = mod_inverse(a, 961)

    if mod_inverse_a is not None:
        decrypted_text = ""
        for bigram in text:
            y = (alphabet.index(bigram[0]) * 31 + alphabet.index(bigram[1])) % 961
            x = (mod_inverse_a * (y - b)) % 961
            decrypted_index1 = x // 31
            decrypted_index2 = x % 31
            decrypted_char1 = alphabet[decrypted_index1]
            decrypted_char2 = alphabet[decrypted_index2]
            decrypted_text += decrypted_char1
            decrypted_text += decrypted_char2

        output_file = f"decrypted_file_№{i}_key({a},{b}).txt"
        print(f"Розшифрований текст для ключа {key} збережено у файл
decrypted_file_№{i}_key({a},{b}).txt")
        with open(output_file, 'w', encoding='utf-8') as file:
            file.write(decrypted_text)

def is_text_meaningful(input_text):
    # Список заборонених біграм
    forbidden_bigrams = ["аь", "йь", "щз", "жф", "ьь"]
    text = [input_text[i:i + 2] for i in range(0, len(input_text) - 1, 2) if
is_bigram_in_alphabet(input_text[i:i + 2])]
    # Перевірка кожної забороненої біграми в тексті
    for bigram in forbidden_bigrams:
        if bigram in text:
            return False
    return True


file_path = r'C:\Users\alexd\PycharmProjects\pythonProject1\08.txt'
with open(file_path, 'r', encoding='utf-8') as file:
    text = file.read()


print("Task1")
print("---------------------------------")
a = 5
m = 37
res = mod_inverse(a, m)
print(f"Обернене до a = {a} за модулем {m} = {res}")

a = 39
b = 30
m = 111
```

```
res = solve_linear_congruences(a, b, m)
print(f"Розв'язок рівняння {a}x = {b}mod{m} = {res}")
print("---------------------------------\n")

print("Task2")
print("---------------------------------")
bigram_frequency = calculate_non_overlapping_bigram_frequencies(text)
top5_bigrams_encrypted = [bigram for bigram, _ in sorted(bigram_frequency.items(),
key=lambda item: item[1], reverse=True)[:5]]
print(top5_bigrams_encrypted)
print("---------------------------------\n")

print("Task3-4")
print("---------------------------------")


keys = find_affine_key_candidates(text)
print(keys)
i = 0
for key in keys:
    decrypt_affine_cipher(file_path, key, i)
    i += 1

directory = r'C:\Users\alexd\PycharmProjects\pythonProject1'
for filename in os.listdir(directory):
    if filename.startswith("decrypted_file"):
        file_path = os.path.join(directory, filename)
        with open(file_path, 'r', encoding='utf-8') as file:
            text = file.read()
            if is_text_meaningful(text):
                print(f"Файл {filename} є змістовним.")

print("---------------------------------\n")
```

Результати:

Реалізована підпрограма із необхідними математичними операціями: обчисленням оберненого елементу за модулем із використанням розширеного алгоритму Евкліда, розв’язуванням лінійних порівнянь

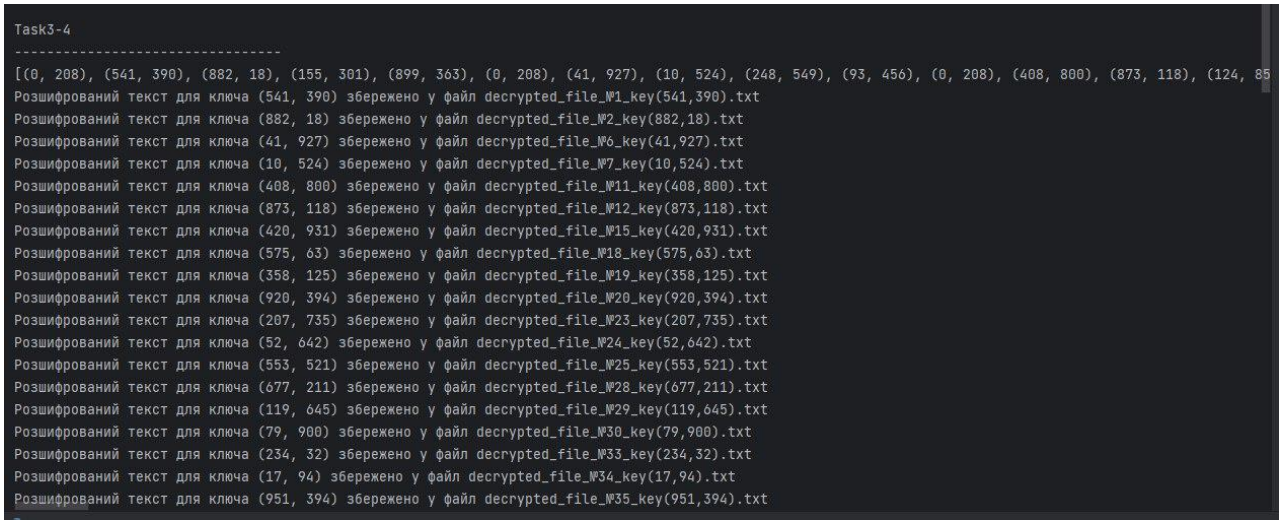
```
Task1
---------------------------------
Обернене до a = 5 за модулем 37 = 15
Розв'язок рівняння 39x = 30mod111 = [15, 52, 89]
---------------------------------
```

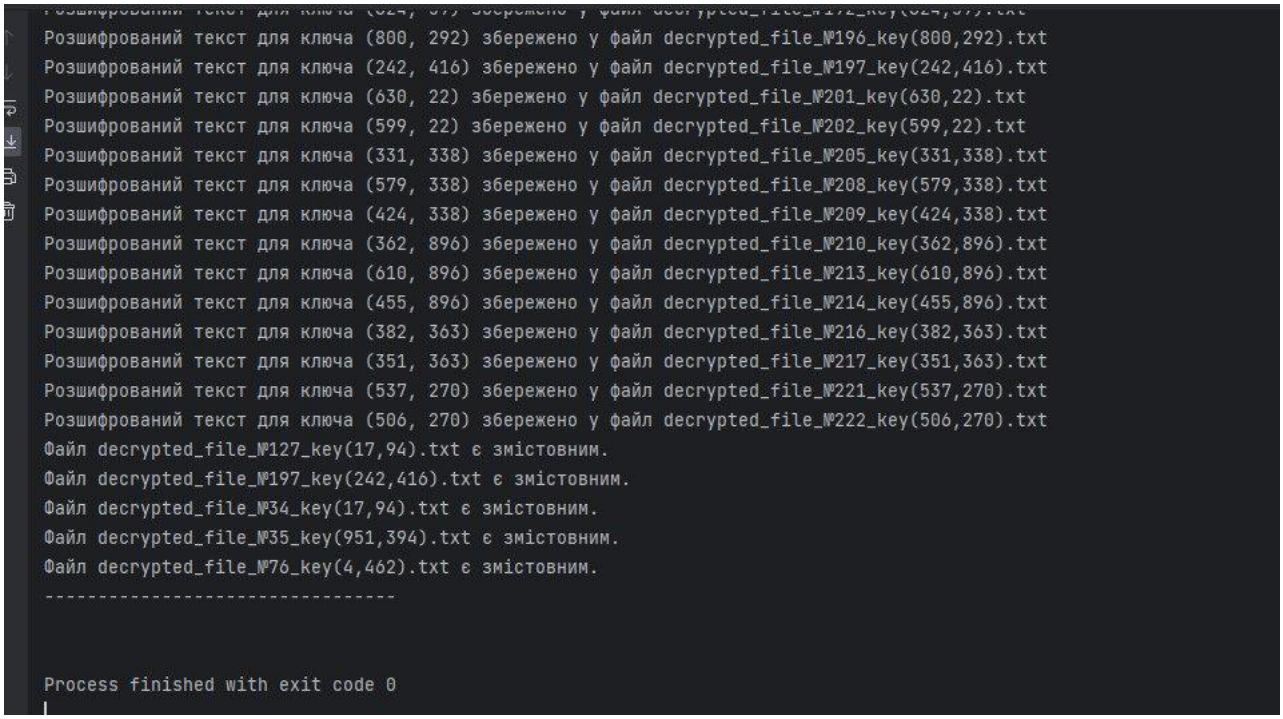
5 найчастіших біграм запропонованого шифртексту (за варіантом №8).

```
Task2
---------------------------------
['жц', 'дэ', 'цэ', 'сц', 'оц']
---------------------------------
```

Перебір можливих варіантів співставлення частих біграм мови та частих біграм шифртексту , для кожного співставлення знайдено можливі кандидати на ключ

(a,b) шляхом розв’язання системи (1). Для кожного кандидата на ключ дешифрувано шифртекст. Якщо шифртекст не є змістовним текстом російською мовою, його не враховано.





На цьому скріншоті можемо побачити результат розшифровки файлу.

Висновки:

У ході виконання лабораторної роботи №3 ми вивчили та засвоїли навички роботи з інструментом частотного аналізу на прикладі розкриття моноалфавітної підстановки.Повторення роботи з модулярною арифметикою .Також варто зазначити ,що у ході виконання комп’ютерного практикуму ми повтроювали та отсаточно закріплювали навички ,отримані під час вионання попередніх комп’ютерних практикумів ,зокрема робота з частотами біграм та інше.