

Комп'ютерний практикум №3

Криптоаналіз афінної біграмної підстановки

ФБ-12 Шестопалов Олександр

8 варіант

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:

```
# coding=utf-8

alphabet = "абвгдежзийклмнопрстуфхцчщъыэюя"

def gcd(a, b):
    if a==0:
        return b, 0, 1
    else:
        g, x, y = gcd(b % a, a)
        return g, y-x*(b//a), x

def mod_minus1(a, m):
    g, x, y = gcd(a, m)
    if g != 1:
```

```

        return None
    else:
        return (x % m + m) % m

def count_bigram_frequency(text):
    text = ''.join([char.lower() for char in text if char.isalpha() or
char.isspace()])
    bigram_frequency = {}
    for i in range(0, len(text) - 1, 2):
        bigram = text[i:i + 2]
        if bigram in bigram_frequency:
            bigram_frequency[bigram] += 1
        else:
            bigram_frequency[bigram] = 1
    return bigram_frequency

def find_possible_keys(encrypted_text):
    bigram_frequency = count_bigram_frequency(encrypted_text)
    top5_bigrams = [item[0] for item in sorted(bigram_frequency.items(), key=lambda
item: item[1], reverse=True)[:5]]
    bigrams = ["то", "на", "ст", "но", "по"]
    possible_keys = []
    for i in range(len(bigrams)):
        for j in range(len(top5_bigrams)):
            for k in range(len(bigrams)):
                for n in range(len(top5_bigrams)):
                    x1 = (alphabet.index(bigrams[i][0])*31 +
alphabet.index(bigrams[i][1]))%961
                    y1 = (alphabet.index(top5_bigrams[j][0])*31 +
alphabet.index(top5_bigrams[j][1]))%961

                    x2 = (alphabet.index(bigrams[k][0])*31 +
alphabet.index(bigrams[k][1]))%961
                    y2 = (alphabet.index(top5_bigrams[n][0])*31 +
alphabet.index(top5_bigrams[n][1]))%961

                    mod_i = mod_minus1(x1 - x2, 961)
                    if mod_i is not None:
                        a_i = (y1 - y2)*mod_i % 961
                        b_i = (y1 - x1*a_i)%961
                        possible_keys.append((a_i, b_i))
    return possible_keys

def text_maye_sens(input_text):
    bigrams_not_exist = ["аь", "йь", "щэ", "жф", "ьь"]
    text = [input_text[i:i + 2] for i in range(0, len(input_text) - 1, 2)]
    for bigram in bigrams_not_exist:
        if bigram in text:
            return False
    return True

def decrypt_afinne(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)]
    a, b = key
    mod_minus1_a = mod_minus1(a, 961)
    if mod_minus1_a is not None:
        decrypted_text = ""
        for bigram in text:
            y = (alphabet.index(bigram[0])*31 + alphabet.index(bigram[1]))%961
            x = (mod_minus1_a *(y - b))%961
            d_index1 = x//31
            d_index2 = x%31
            d_char1 = alphabet[d_index1]
            d_char2 = alphabet[d_index2]
            decrypted_text += d_char1
            decrypted_text += d_char2
        if text_maye_sens(decrypted_text):
            output_file = f"decrypted{key}.txt"
            with open(output_file, 'w', encoding='utf-8') as file:

```

```

        file.write(decrypted_text)

with open(r'081.txt', 'r', encoding='utf-8') as file:
    text = file.read()
print("Найчастіші біграми в тексті:")
bigram_frequency = count_bigram_frequency(text)
top5_bigrams_encrypted = [bigram for bigram, i in sorted(bigram_frequency.items(),
key=lambda item: item[1], reverse=True)[:5]]
print(top5_bigrams_encrypted)
keys = find_possible_keys(text)
print("Можливі ключі:")
print(keys)
i = 0
for key in keys:
    decrypt_afinne(r'081.txt', key, i)
    i += 1
print("Можливі варіанти розшифрованого тексту збережені у файли decrypted[ключ].txt")






```

Отримуємо найчастіші біграми для зашифрованого тексту, список можливих ключів та найвірогідніші варіанти розшифрованого тексту, збережені в текстові файли

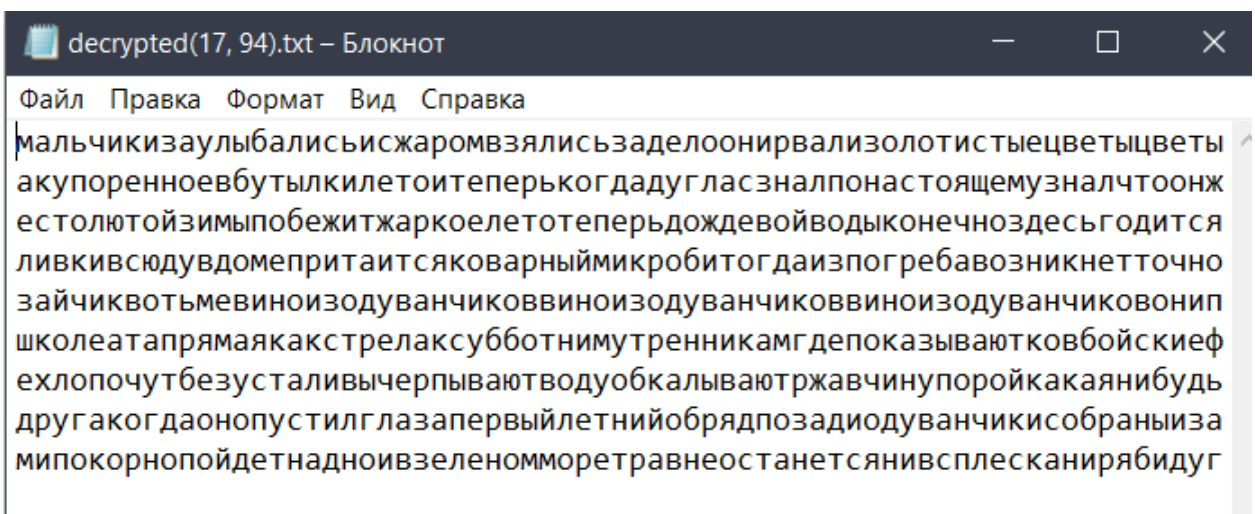
```

Найчастіші біграми в тексті:
['жц', 'дэ', 'цэ', 'сц', 'оц']
Можливі ключі:
[(0, 208), (97, 462), (469, 59), (868, 549), (806, 456), (0, 208), (358, 125), (17, 94), (806, 456), (62,
Можливі варіанти розшифрованого тексту збережені у файли decrypted[ключ].txt

```

 081.txt	15.11.2023 19:36	Текстовый докум...	18 КБ
 decrypted(17, 94).txt	15.11.2023 23:10	Текстовый докум...	18 КБ
 decrypted(141, 187).txt	15.11.2023 23:10	Текстовый докум...	18 КБ
 decrypted(234, 32).txt	15.11.2023 23:10	Текстовый докум...	18 КБ
 lab3.py	15.11.2023 23:10	Файл "PY"	4 КБ

Розшифрований текст:



Висновки: під час виконання роботи було зроблено криптоаналіз афінної біграмної підстановки та розшифровано текст за допомогою частотного аналізу

