

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім.
ІГОРЯ СІКОРСЬКОГО”**

**КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №3**

Криптоаналіз афінної біграмної підстановки

Виконали роботу:
студент ФБ-23 Гнидюк Данііл
студент ФБ-23 Жушман Ілля

Київ 2024

Мета роботи

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

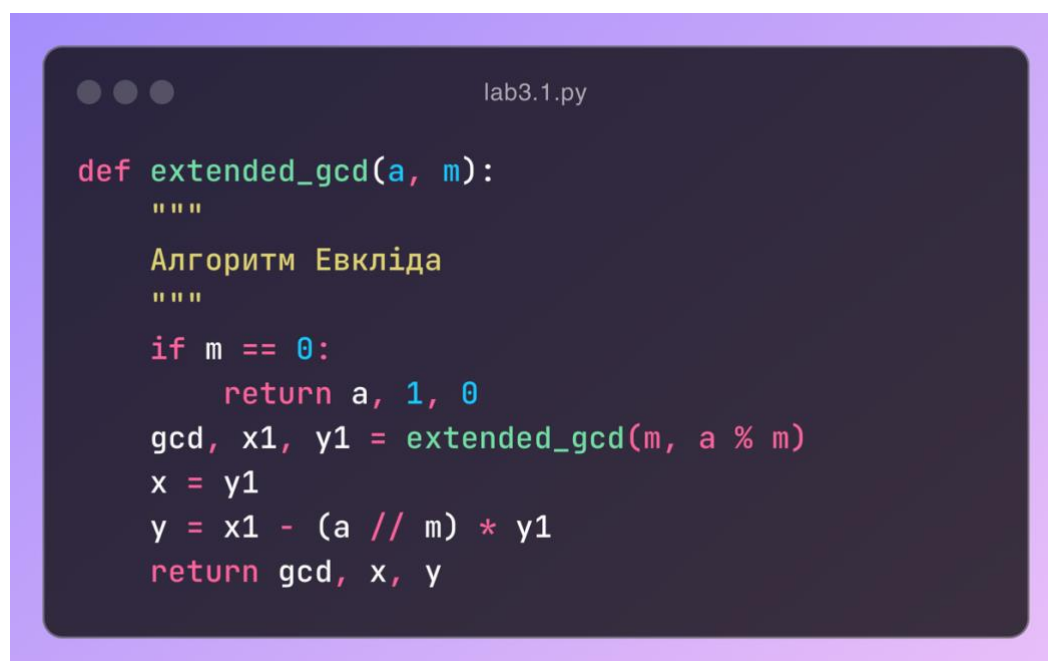
Варіант: 5

Порядок виконання роботи

1. Реалізувати підпрограми із необхідними математичними операціями: обчисленням оберненого елемента 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.

Хід роботи

Обчислення оберненого елемента за модулем передбачає знаходження числа x , яке задовольняє рівняння $a \cdot x \equiv 1 \pmod{m}$, використовуючи розширений алгоритм Евкліда. Розв'язання лінійних порівнянь полягає у знаходженні всіх значень x , які задовольняють рівняння $a \cdot x \equiv b \pmod{m}$, враховуючи випадки з декількома розв'язками, або визначенні, що розв'язків не існує.



```
lab3.1.py

def extended_gcd(a, m):
    """
    Алгоритм Евкліда
    """
    if m == 0:
        return a, 1, 0
    gcd, x1, y1 = extended_gcd(m, a % m)
    x = y1
    y = x1 - (a // m) * y1
    return gcd, x, y
```



lab3.1.py

```
def modular_inverse(a, m):  
    """  
    Знаходить обернений елемент a за модулем m.  
    """  
    gcd, x, _ = extended_gcd(a, m)  
    if gcd != 1:  
        return None # Оберненого елемента не існує  
    return x % m
```



lab3.1.py

```
def solve_linear_congruence(a, b, m):  
    """  
    Розв'язує лінійне порівняння  $ax \equiv b \pmod{m}$ .  
    Повертає список всіх розв'язків або повідомляє, якщо розв'язків немає.  
    """  
    gcd, x, _ = extended_gcd(a, m)  
    if b % gcd != 0:  
        return [] # Немає розв'язків  
    # Знаходимо перший розв'язок  
    x0 = (x * (b // gcd)) % m  
    solutions = [(x0 + i * (m // gcd)) % m for i in range(gcd)]  
    return solutions
```

Результат:

Обернений елемент існує:

```
=== Виберіть операцію ===  
1. Знайти обернений елемент (Евкліда)  
2. Розв'язати лінійне порівняння  $ax \equiv b \pmod{m}$   
3. Вийти  
Ваш вибір: 1  
Введіть a: 3  
Введіть модуль m: 26  
Обернений елемент для 3 за модулем 26: 9
```

Обернений елемент не існує:

```
=== Виберіть операцію ===
1. Знайти обернений елемент (Евкліда)
2. Розв'язати лінійне порівняння  $ax \equiv b \pmod{m}$ 
3. Вийти
Ваш вибір: 1
Введіть a: 6
Введіть модуль m: 15
Обернений елемент для 6 за модулем 15 не існує
```

Лінійне рівняння

Кілька розв'язків:

```
=== Виберіть операцію ===
1. Знайти обернений елемент (Евкліда)
2. Розв'язати лінійне порівняння  $ax \equiv b \pmod{m}$ 
3. Вийти
Ваш вибір: 2
Введіть a: 4
Введіть b: 8
Введіть модуль m: 12
Кілька розв'язків: 2, 5, 8, 11
```

Один розв'язок:

```
=== Виберіть операцію ===
1. Знайти обернений елемент (Евкліда)
2. Розв'язати лінійне порівняння  $ax \equiv b \pmod{m}$ 
3. Вийти
Ваш вибір: 2
Введіть a: 7
Введіть b: 5
Введіть модуль m: 13
Один розв'язок:  $x = 10$ 
```

Немає розв'язків:

```
=== Виберіть операцію ===
1. Знайти обернений елемент (Евкліда)
2. Розв'язати лінійне порівняння  $ax \equiv b \pmod{m}$ 
3. Вийти
Ваш вибір: 2
Введіть a: 6
Введіть b: 14
Введіть модуль m: 15
Немає розв'язків
```

Далі завдання полягає в аналізі шифртексту для визначення 5 найчастіших біграм, які використовуються для подальшого дешифрування. Ми зчитали текст із файлу, видалили зайві символи, розбили його на біграми (пари символів), підраховували частоти появи кожної біграми за допомогою Counter, а потім відсортували їх у порядку спадання частоти. У підсумку ми вивели 5 біграм із найвищими частотами та їх відсоткову частку в тексті.

```
lab3.2.py

def calculate_bigram_frequencies(text):
    """
    Обчислює частоти біграм у тексті.
    """
    bigrams = [text[i:i+2] for i in range(len(text) - 1)] # Створюємо всі біграми
    total_bigrams = len(bigrams) # Загальна кількість біграм
    bigram_counts = Counter(bigrams) # Підраховуємо кількість кожної біграми
    bigram_frequencies = {bigram: count / total_bigrams for bigram, count in
        bigram_counts.items()} # Частота
    return bigram_frequencies
```

Ця функція розбиває текст на біграми (пари сусідніх символів), підраховує кількість кожної біграми за допомогою Counter і обчислює їх частоти (частка кожної біграми в загальній кількості біграм).

```
lab3.2.py

def find_top_bigrams(file_path, top_n=5):
    """
    Знаходить топ N біграм із тексту у файлі.
    """
    text = read_text_from_file(file_path)
    frequencies = calculate_bigram_frequencies(text)
    top_bigrams = sorted(frequencies.items(), key=lambda x: x[1], reverse=True)[:top_n] # Топ
    N біграм за частотою
    return top_bigrams
```

Ця функція знаходить топ-5 біграм із тексту у файлі. Вона використовує функції для зчитування тексту та обчислення частот біграм, сортує їх за частотою в спадному порядку і повертає список біграм із найвищими частотами.

Результат:

```
Топ-5 біграм із файлу 05.txt:
вн: 0.0096
тн: 0.0087
дк: 0.0086
ун: 0.0086
хщ: 0.0082
```


Пошук кандидатів на ключі: Створюються всі можливі пари співставлень частих біграм мови та шифртексту. Для кожної пари обчислюються числові значення біграм. Використовуючи систему рівнянь, обчислюються можливі значення a та b , які додаються до списку кандидатів.

Можливі ключі (a , b):

```
(837, 385)
(923, 604)
(589, 44)
(183, 284)
(837, 385)
(892, 201)
(589, 44)
(307, 935)
(186, 571)
(832, 227)
(558, 602)
(950, 304)
(93, 323)
(943, 275)
(279, 819)
(289, 174)
(124, 261)
```

```
lab3.3.py

def decrypt(nums, a, b, m):
    """Дешифрує текст за допомогою ключа (a, b)."""
    decrypted_text = []
    a_inv = modular_inverse(a, m ** 2)
    if a_inv is None:
        return None
    for i in range(0, len(nums), 2):
        y = nums[i] * m + (nums[i + 1] if i + 1 < len(nums) else 0)
        x = (a_inv * (y - b)) % (m ** 2)
        decrypted_text.append(alphabet[x // m])
        decrypted_text.append(alphabet[x % m])
    return ''.join(decrypted_text)
```

Дешифрування тексту: Для кожного згенерованого ключа ми дешифрували текст, перетворюючи його з числового представлення назад у символи. Дешифрування базувалося на формулі:

$$X_i = a^{-1}(Y_i - b) \bmod m^2$$

```
lab3.3.py

def count_rare_bigrams(text, rare_bigrams):
    """Підрахунок кількості рідкісних біграм у тексті."""
    return sum(1 for i in range(0, len(text) - 1, 2) if text[i:i + 2] in rare_bigrams)
```

Під час дешифрування виникла потреба відокремити змістовний текст російською мовою від випадкового набору символів, що з'являється при використанні неправильних ключів. Через значну кількість можливих ключів було розроблено автоматичний метод перевірки тексту на змістовність. Цей метод базувався на підрахунку рідкісних біграм, які рідко зустрічаються в російській мові, наприклад, «бб», «эо», «щд» тощо. Найкращими результатами вважалися тексти, у яких кількість таких біграм була мінімальною, що дозволяло обрати найбільш ймовірний змістовний текст.

```
Ключі з найменшою кількістю рідкісних біграм:
1. Ключ (a=654, b=777), Рідкісні біграми: 5
2. Ключ (a=627, b=535), Рідкісні біграми: 9
3. Ключ (a=315, b=478), Рідкісні біграми: 14
4. Ключ (a=765, b=825), Рідкісні біграми: 18
5. Ключ (a=11, b=447), Рідкісні біграми: 21
6. Ключ (a=887, b=615), Рідкісні біграми: 22
7. Ключ (a=188, b=230), Рідкісні біграми: 22
8. Ключ (a=918, b=212), Рідкісні біграми: 22
9. Ключ (a=69, b=8), Рідкісні біграми: 23
10. Ключ (a=508, b=260), Рідкісні біграми: 23
```

У кінці програми ми вивели список ключів із найменшою кількістю рідкісних біграм, а також дешифрований текст для найкращого ключа. Цей підхід дозволив автоматизувати аналіз шифртексту та забезпечити відбір змістовного тексту російською мовою.

Зашифрований текст:

```
05.txt

кеюибщяефдмдкдрлрцисвнуншвйняэшскевдтнюдаобсюсызхзмдълюхунхмъввнсдуэмндтихкеюибщяцзкзхшвнос
ютнйщтцншуссянхшлвжвпкшвнмщзфтсхщпддкясввцтнавпгнунъввнлхиверддцрихэкзэцижщъехшмэкжлрибуждэм
химъпъавсттнэцосфспъуипдкнхрххульацкчашьяншибжаксэкццзтчиюцншумщошьящкщнфрхуюижсгцыззфрчихзтчщрихн
эпозтгфккщкдмкльоьеынунйилцяэрхнмкпмдкйпоиэуныэнсмнмсхэцъедктництндущоэивупхюфчсьивйэютнрцшэбв
щншуоздкдктнунянккфкяящисббинкурдцбщдскрщянщкдкяяшжшсвьербщяшндуйнкщнвнгоьэииспътуумщшдекхнду
аошдвдеигебуаявюсшьйдрощцвнфибжлакццвбвбываккклстхъщзйъцъжьбрьецфтспъбишиовдъезбтнмсэкжлрчсхщърпъшв
шнйьяншибжлтъчсьърэчтнундулфтснсшбйнбжцнмющккюиеуяэзтъяяреурндуьцозгкмбобмщкскехюксдцтсывзтмсун
йкксшссшнщщзйьцнпнрщкфкяслркеййнавпъхсуншнузеумкжлаклцисуьдьбкфипьйнмсуншнхтуйнцнцмсьямынкцркч
уюклзфкчпъвныуозрбжлжвцнхщсссцъьбипсрзфкаьихмнщэчсавозулбутнзцнулзткоццвнфибхюпвиэислбиювинхыршьив
```

Розшифрований текст:

```
Текст для ключа (a=654, b=777): убивать больше ненадо по слогом а кону же убил носледуете мубыть благодарны мина чепришлось б
убиватьса момуэто неодишшь доброесостраданиеэто оххдствлениенаоснованиииодинаковых импульсовкубийствусобственногов
орялишьв минимальнойстепенисмещенныйнарциссимэтическаяценностьэтой добротыэтимнеоспариваетса можетбытьэто вообщемех
низмашегодоброгоучастияпоотношениюдругомучеловекуособеннояснопроступающийивчрезвычайномслучаеобремененногосозна
иясвоейвиныписателянетсомнениячтоэтасимпатияпопричинеоххдствлениарешительноопределилавьорматериаладостоевского
носначалаонизэгоистическихпобужденийвыводилобыкновеногопреступникаполитическогои религиозногопреждечемкконцусвоей
жизнивернутьсякакпервопреступникуотцеубийцеисделатъеголицеисвоепоэтическоепризнаниеопубликованиеегопосмертногонасл
едияидневникогегоженяркоосветилооднэпизодегожизнитовремякогдадостоевскийвгерманииибылобуреваемигорнойстрастьюдос
тоевскийзарулеткойявныйприпадкопатологическойстрастикоторыйнеподдаетсяяинойоценкенисакойсторонынебыло недостаткво
```


Висновки:

У ході роботи ми реалізували алгоритми для знаходження оберненого елемента за модулем, розв'язання систем лінійних порівнянь та дешифрування шифртексту за допомогою афінного шифру. Ми згенерували можливі ключі (a, b), зіставляючи часті біграми мови та шифртексту, і створили автоматичний розпізнавач російської мови для оцінки змістовності дешифрованих текстів. Оцінка проводилася шляхом підрахунку кількості рідкісних біграм, які зазвичай не зустрічаються в російській мові. Завдяки сортуванню результатів за кількістю рідкісних біграм ми обрали найкращий ключ і успішно дешифрували текст. Цей підхід дозволив автоматизувати процес розшифрування та підвищити точність відбору змістовних текстів.