

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №3
Криптоаналіз афінної біграмної підстановки

Виконали:
студентки групи ФБ-23
Гуз Вікторія
Шукалович Марія

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

Постановка задачі:

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.

Хід роботи(варіант 4):

Першим етапом була реалізація підпрограми з математичними операціями:

```
def extended_euclidean(a, n):  
    r0, r1 = a, n  
  
    while True:  
        gcd, _, _ = extended_euclidean_core(r0, r1)  
        if gcd == 1:  
            break  
        r0 //= gcd  
        r1 //= gcd  
  
    gcd, u, v = extended_euclidean_core(r0, r1)  
  
    if gcd != 1:  
        raise ValueError("Обернений елемент не існує, оскільки gcd(a, n) ≠ 1.")  
  
    b = u % n  
  
    if ((a * b) - 1) % n == 0:  
        print(f"{a} * {b} - 1 ≡ 0 (mod {n}) - Перевірка успішна:D")  
        return b  
    else:  
        print(f"({a} * {b} - 1) не дорівнює 0 за модулем {n} - Перевірка не  
пройдена: (")  
        return
```

Функція `extended_euclidean` знаходить обернений елемент для числа a за модулем n за допомогою розширеного алгоритму Евкліда. Спочатку задаються $r_0 = a$ і $r_1 = n$ — два початкових числа для обчислення НСД. За допомогою `extended_euclidean_core(r0, r1)` знаходиться НСД gcd , а також коефіцієнти u і v , які представляють НСД як лінійну комбінацію $u * r_0 + v * r_1$. Цикл повторюється, поки НСД gcd не стане рівним 1, при цьому

щоразу r_0 і r_1 зменшуються, ділячись на НСД. Після виходу з циклу, якщо НСД дорівнює 1, відбувається повторний виклик `extended_euclidean_core` для отримання остаточної значень u і v .

```
def extended_euclidean_core(a, b):
    r0, r1 = a, b
    u0, u1 = 1, 0
    v0, v1 = 0, 1

    while r1 != 0:
        q = r0 // r1
        r0, r1 = r1, r0 - q * r1
        u0, u1 = u1, u0 - q * u1
        v0, v1 = v1, v0 - q * v1

    return r0, u0, v0
```

Функція `extended_euclidean_core` реалізує основний алгоритм розширеного Евкліда для знаходження НСД двох чисел a і b , а також коефіцієнтів u і v , які задовольняють рівняння:

Обернений елемент існує:

```
—♥—Меню математичних операцій—♥—
0. Повернутись
1. Обчислити обернений елемент(розш. алг. Евкліда)
2. Розв'язування лінійних порівнянь
Виберіть опцію: 1

Введіть натуральне значення для знаходження оберненого  $b \equiv a \pmod{n}$ :
Введіть число a: 32
Введіть модуль n: 99
 $32 * 65 - 1 \equiv 0 \pmod{99}$  – Перевірка успішна:D
Обернений елемент числа 32 за модулем 99 дорівнює 65
```

Обернений елемент не існує:

```
—♥—Меню математичних операцій—♥—
0. Повернутись
1. Обчислити обернений елемент(розш. алг. Евкліда)
2. Розв'язування лінійних порівнянь
Виберіть опцію: 1

Введіть натуральне значення для знаходження оберненого  $b \equiv a \pmod{n}$ :
Введіть число a: 3
Введіть модуль n: 99
 $(3 * 1 - 1)$  не дорівнює 0 за модулем 99 – Перевірка не пройдена:(
Оберненого елементу числа 3 за модулем 99 не існує.
```

```
def solve_linear_congruence(a, b, n):
    gcd, u, _ = extended_euclidean_core(a, n)

    if gcd == 1:
        x = (u * b) % n
```

4 варіант

```

        return [x]

    elif gcd > 1:
        if b % gcd != 0:
            return []

    a1, b1, n1 = a // gcd, b // gcd, n // gcd

    _, u1, _ = extended_euclidean_core(a1, n1)
    x0 = (u1 * b1) % n1

    solutions = [(x0 + i * n1) % n for i in range(gcd)]
    return solutions

```

Функція solve_linear_congruence розв'язує лінійне рівняння $ax \equiv b \pmod{n}$. Вона використовує розширений алгоритм Евкліда для знаходження НСД і оберненого елемента

Кілька розв'язків:

```

-♥-Меню математичних операцій-♥-
0. Повернутись
1. Обчислити обернений елемент(розш. алг. Евкліда)
2. Розв'язування лінійних порівнянь
Виберіть опцію: 2

Введіть значення для лінійного порівняння  $ax \equiv b \pmod{n}$ :
Введіть a: 45
Введіть b: 5
Введіть модуль n: 55
Розв'язки лінійного порівняння: [5, 16, 27, 38, 49]

```

Один розв'язок:

```

-♥-Меню математичних операцій-♥-
0. Повернутись
1. Обчислити обернений елемент(розш. алг. Евкліда)
2. Розв'язування лінійних порівнянь
Виберіть опцію: 2

Введіть значення для лінійного порівняння  $ax \equiv b \pmod{n}$ :
Введіть a: 12
Введіть b: 6
Введіть модуль n: 37
Розв'язки лінійного порівняння: [19]

```

Немає розв'язків:

```

-♥-Меню математичних операцій-♥-
0. Повернутись
1. Обчислити обернений елемент(розш. алг. Евкліда)
2. Розв'язування лінійних порівнянь
Виберіть опцію: 2

Введіть значення для лінійного порівняння  $ax \equiv b \pmod{n}$ :
Введіть a: 14
Введіть b: 21
Введіть модуль n: 98
Порівняння не має розв'язків.

```

Під час реалізації коду для математичних обчислень труднощів не виникало. Весь процес виявився зрозумілим і логічним, завдяки чітким алгоритмам, які дозволили швидко реалізувати потрібні обчислення.

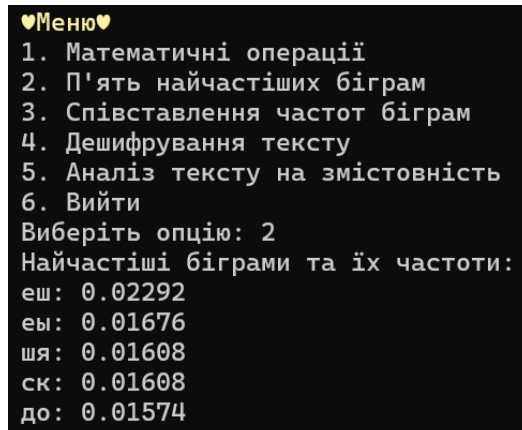
Наступним кроком був пошук 5 найчастіших біграм шифртексту:

```
# Частота біграм
def bigram_frequencies(bigram_counts, total_bigrams):
    frequencies = {bigram: count / total_bigrams for bigram, count in
bigram_counts.items()}
    return frequencies
```

Функція створює словник `frequencies`, у якому для кожної біграми обчислюється частота. Частота визначається як відношення кількості появи біграми до загальної кількості біграм у тексті.

```
# Виведення топ-5 біграм
def print_top_5_bigrams(bigram_frequencies, i=False):
    top_5_bigrams = dict(sorted(bigram_frequencies.items(), key=lambda x: x[1],
reverse=True)[:5])
    if i:
        print("Найчастіші біграми та їх частоти:")
        for bigram, frequency in top_5_bigrams.items():
            print(f"{bigram}: {frequency:.5f}")
    else:
        return top_5_bigrams
```

Функція `print_top_5_bigrams` спочатку створює словник `top_5_bigrams`, відсортований за частотою біграм у порядку спадання, зберігаючи лише перші п'ять. Якщо параметр `i` дорівнює `True`, функція виводить назви біграм та їх частоти на екран



```
♥Меню♥
1. Математичні операції
2. П'ять найчастіших біграм
3. Співставлення частот біграм
4. Дешифрування тексту
5. Аналіз тексту на змістовність
6. Вийти
Виберіть опцію: 2
Найчастіші біграми та їх частоти:
еш: 0.02292
еы: 0.01676
шя: 0.01608
ск: 0.01608
до: 0.01574
```

З цим етапом також труднощів не виникало.

Після знаходження найчастіших біграм слід було підібрати ключ і розшифрувати текст:

```
# Знаходження кандидатів для ключів
def keys_find(cipher_bigrams, plain_bigrams):
    m = 31
    keys = []
    for i in range(len(cipher_bigrams) - 1):
        for j in range(i + 1, len(cipher_bigrams)):
```

4 варіант

```

        y1 = cipher_bigrams[i]
        y2 = cipher_bigrams[j]
        x1 = plain_bigrams[i]
        x2 = plain_bigrams[j]
        diff_y = (y1 - y2) % (m ** 2)
        diff_x = (x1 - x2) % (m ** 2)
        a_candidates = solve_linear_congruence(diff_x, diff_y, m ** 2)
        for a in a_candidates:
            b = (y1 - a * x1) % (m ** 2)
            keys.append((a, b))

    return keys

# Функція для дешифрування тексту
def decrypt_affine(ciphertext, a, b, m):

    a_inverse = extended_euclidean(a, m ** 2)
    if a_inverse is None:
        print(f"Обернений елемент для a={a} не існує, пропускаємо ключ (a, b) = ({a}, {b})")
        return None

    decrypted_text = []
    for i in range(0, len(ciphertext) - 2, 2):
        Y_i = alphabet.index(ciphertext[i]) * m + alphabet.index(ciphertext[i + 1])

        X_i = (a_inverse * (Y_i - b)) % (m ** 2)
        p2 = X_i % m
        p1 = (X_i - p2) // m
        decrypted_text.append(alphabet[p1])
        decrypted_text.append(alphabet[p2])

    return ''.join(decrypted_text)

```

Функція `keys_find` знаходить кандидати для ключів шифру, аналізуючи біграми з зашифрованого та відкритого тексту (`plain_top_bigrams = ['ст', 'но', 'то', 'на', 'ен']`). Вона використовує різницю між біграмами, щоб розв'язати лінійні конгруенції та знайти можливі значення для параметрів шифру. Функція `decrypt_affine` здійснює дешифрування тексту за допомогою афінного шифру. Вона обчислює обернений елемент для параметра `a`, а потім, використовуючи формули, перетворює зашифровані біграми назад у відкритий текст, формуючи результат із відповідних символів. Для дешифрування була використана наступна формула:

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

```

Виберіть опцію: 3
Можливі кандидати на ключі (a, b):
(503, 890)
(200, 733)
(390, 10)
(436, 887)
(867, 940)
(664, 62)
(462, 785)
(915, 178)
(885, 413)

```


Зашифрований текст:

Щжужащпккфшчфбждоцпюдйсвжбэдуэййэдцмодпмурзфбряцкмдйдосштцмижбчфипмугфбзчшохдодвзбряцкмдбэдцхэнощк
яоэоэйтцюзныертзилгфоцбчполфмэдцщкйкшйэйсйрэйкчозычфждьмйшотдотзьюйсцзоюдууюзсшштзрэюсыяфоешыенывд
ьмиыыящрбгнянмзюдшскдмаыйяаоешезвжпнорэкжцчжшбчдофщцфбяоязфщжвонцеырайхмучмсшывчфвэрфешмяояйвщ
еййсбжощлзшярфбждоцпюдлвюпщкмзешжмоуяхямзюдлвзбкзешдбшящксавотзйбкжзщцопсйкоефтцрзюэдцсшямсканзоми
жуэыыцсшмычмэжлрзщыезскщквкшятоьэйштибшяшкочщкфмыйеййывдьмиыщчвккцощеызонорйвкхпшсзунрмоншзоаяшяэдхп
езхлсoppiипейзохлншплбйщждоыкфоскщквкшягоефоцэзччскщкванквкзешюшлцромглтдоккжшскзыадншууезжурфешщпнз
шятоужертцлвхящжпофожущпккшяэывдьмиыйсжусжощкшйжррэсзешьоктдоскыкфотфлцжшвдзылвхзпмжушжелыяцдюппкгф
кшскщквкшяозноуяйэвзхягжжзшрфяоэщпсчкжйэцшвдрйрэйкчфолжыймывдьмиыщчдорддокыбзлжвочыезыяйейтъяочмск
мзшядяешмюяхщжбгяжрйашайюпмогйжшфшайрмлэннтзхаокшйбчаошяанбччйтжмкжучбуфпошфбждоцпюдлвюпэзкбтцзопз
аоешйшохзодонофшайсцзожурфмовоцяанфшляйбмьюосклкюнсккжезьоешшоешоцэжлыдяйейзопыщжфоочсквжаббжнзбля

Розшифрований текст:

♥Меню♥
1. Математичні операції
2. П'ять найчастіших біграм
3. Співставлення частот біграм
4. Дешифрування тексту
5. Аналіз тексту на змістовність
6. Вийти
Виберіть опцію: 5
Знайдено не змістовний текст (ключ: a=503, b=890).
Знайдено не змістовний текст (ключ: a=200, b=733).
Змістовний текст (ключ: a=390, b=10): если правда что достоевский в сибире был подвержен припадкам то это лишь подтверждает то что его припадки были
его карой он боле евных не нуждался его дабыл кара еминым образом оно доказать это не возможно скорее этой необходимости в наказание для психической экономики
достоевского объясняет ся то что он прошел несломленным через эти годы бедствий и унижений осуждение достоевского о качестве политического преступника был
он несправедливым и он должен был это знать но он принял это не заслуженно не наказание от башки царя как замена наказания заслуженного им за свой грех по отношению
ни к своему собственному отцу вмести с амона наказания он дал себе наказание заместителю отца это дает нам некоторое представление о психологическом оправдан
ии наказания и присуждаемых обществом это насамо делает акногие из преступников жаждут наказания его требуют с верха извбавляя себя таким образом от само
аказания и от того что знает сложное и изменчивое значение истерических симптомов и мет что мы здесь не пытаемся добиться смысла припадков достоевского во все
полное достаточного что можно предположить что их первоначальная сущность осталась неизменной несмотря на все последующие наслоения можно сказать что
достоевский такникогда не освободился от угрызений совести в связи с намерением убить отца то лежащее на совести бремя определило так же его отношение к дв
ум другим исферам коим смяно отношения к отцу к государству к авторитету к веревбогавпервой он пришел к полному подчинению башке царю однажды раз
гравшему с ним комедию убийства в действительности находит ую столько раз отражение его припадках здесь верхвзяло пакаяние больше свободы ставало сун
его во власти религии он не допускал сомнений в сведениям он до последней минуты своей жизни в себе все колебался между верой и безбожием его высокий ум не позв
олял ему не замечать трудностей осмысливания которых приводит в равновесии индивидуальное повторение исторического оразвития он надеялся в идеале х
и станайти выход из освобождения от грехов и использовать свои собственные страдания чтобы притязать на роль христа если он в конечном счете не пришел к свобо
е и стал реакционером то это объясняется тем что общечеловеческая сыновья вина на которой строится религиозное чувство достигла у него сверхиндивидуально
й силы и не могла быть преодолена даже его высокий интеллектuality здесь насказалось бы можно упрекнуть в том что мы от казываем ся от беспристрастности п
сихолога и за и подверга ем достоевского оценкам ишей прав на существование лишь с пристрастий точки зрения определенного мировоззрения консерваторс
у ет ся лишь тем что его в своем сердце желал кто по его совершению и его приветствовали поэтом уплотью до контрастной фигуры лешивсе братья равновинны движ
имый первичными позывами искатель на слаждений полный скепсис ациники илпелитический преступник в братьях карамазовых есть сцена в высшей степени харак
терная для достоевского из разговора с дмитрием старец постигает что дмитрий носит все бего готовность к цуебийству и бросает ся перед ним на колени и то не мо
ж е т я в л я т с я в ы р а ж е н и е м о с х и щ е н и я д o л ж н o о з н а ч а т ь ч т o с в я т o й o т c т p a н я е т c e б я к у ш e н и ю и c п o л н и т ь c я п р e з р e н и е м к у б и й ц e и л и и п o г н у ш а т ь c я и п o т o м у п e
р д н и м и р я е т c я c и м п а т и я д o c т o в c k o г o к п р e c т y п н и к у д e й c t в и т e л ь н o б e з г p a n и ч н a o н a д a л e k o в ы x o д и т з a п p e д e л ь c т p a д a н и я н a k o т o p o e н e c ч a c т ь н ы й и м e e т п p a
в o o n a n a п o м и н a e т б л a г o г o в e н и e c k o r o m в д p e в н o c т и o т н o c и л и c ь к э п и л e п т и к у и д у ш e в н o б o л ь n o м п p e c т y п н и к д л я н e г o п o ч т и c п a c и т e л ь в з я в ш и й н a c e б я в и н y k o т
o p y в d p y г o м c л y ч a e н e c л и б д p y г и e
Знайдено не змістовний текст (ключ: a=436, b=887).
Знайдено не змістовний текст (ключ: a=867, b=940).
Знайдено не змістовний текст (ключ: a=664, b=62).
Знайдено не змістовний текст (ключ: a=462, b=785).
Знайдено не змістовний текст (ключ: a=915, b=178).
Знайдено не змістовний текст (ключ: a=885, b=413).
Змістовні тексти збережено у файл 'decrypted_var4.txt'.

Отже, правильний ключ [\(390.10\)](#)

Розшифрований змістовний текст збережено у decrypted_var4.txt

Висновки: у процесі виконання лабораторної роботи з афінного шифру вдалося з'ясувати, як шифрування та дешифрування текстів реалізується за допомогою математичних формул. Також, під час роботи над частотним аналізом афінного шифру було продемонстровано, як можна використовувати частоти біграм для виявлення можливих ключів. Це стало важливим етапом у розкритті зашифрованих повідомлень, оскільки знання про частотність букв у мові допомогло спростити задачу шифрування. Крім того, вдалося набути практичних навичок у роботі з оберненими елементами, що є критично важливим для реалізації алгоритмів дешифрування. Застосування методів, таких як розширений алгоритм Евкліда, показало, як математичні принципи можуть бути інтегровані в криптографічні процеси.