

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

КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №2
Криптоаналіз шифру Віженера

Виконали:
студентки групи ФБ-23
Гуз Вікторія
Шукалович Марія

Мета роботи: засвоєння методів частотного криптоаналізу. Здобуття навичок роботи та аналізу поточкових шифрів гамування адитивного типу на прикладі шифру Віженера.

Постановка задачі:

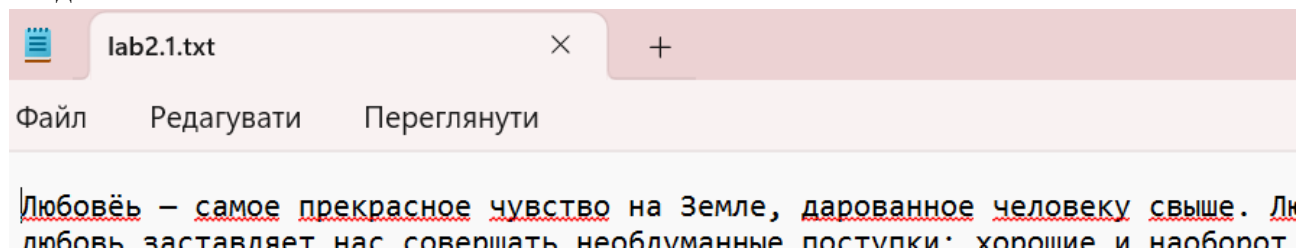
1. Самостійно підібрати текст для шифрування (2-3 кб) та ключі довжини $r = 2, 3, 4, 5$, а також довжини 10-20 знаків. Зашифрувати обраний відкритий текст шифром Віженера з цими ключами.
2. Підрахувати індекси відповідності для відкритого тексту та всіх одержаних шифртекстів і порівняти їх значення.
3. Використовуючи наведені теоретичні відомості, розшифрувати наданий шифртекст (згідно свого номеру варіанта).

Хід роботи(варіант 4):

Для початку ми очистили текст від всіх символи, окрім російського алфавіту (наприклад, цифри, розділові знаки, латиниця тощо), змінили великі літери на малі:

```
def load_and_clean_text(file_path):  
    try:  
        with open(file_path, 'r', encoding='utf-8') as file:  
            text = file.read()  
            text = text.lower()  
            text = re.sub(r'^а-яё', '', text)  
            return text  
    except FileNotFoundError:  
        print(f"Файл '{file_path}' не знайдено")  
        return None
```

Вхідний текст:



Очищений текст:

```
—♥—Меню виведення тексту—♥—  
0. Повернутись  
1. Вивести текст  
2. Вивести текст за варіантом  
  
Виберіть опцію: 1  
  
Оригінальний текст:  
любовьсамоепрекрасноечувствоназемледаров  
ьзаставляетнассовершатьнеобдуманныепоступ  
кихорошиеинаоборот
```

З цим етапом труднощів не виникало.

4 варіант

Після цього ми обрали ключі довжини $r = 2, 3, 4, 5$, а також довжини 10-20 знаків:

```
encryption_keys = {'2': 'хи', '3': 'мяу', '4': 'сова', '5': 'осень', '10':  
'абитуриент', '11': 'бессмертный', '12': 'концентратор', '13': 'глазированный',  
'14': 'мультипликатив', '15': 'ухлёстывавшийся', '16': 'христадельфианин', '17':  
'яфетидологический', '18': 'задокументировавши', '19': 'жизнеобеспечивающий',  
'20': 'евростандартизировав'}
```

Наступним етапом була реалізація шифрування тексту за допомогою шифру Віженера. Функція `vigenere` приймає два параметри: `plaintext` (вхідний текст для шифрування) і `key` (ключ для шифрування). Вона використовує алфавіт, що містить 33 символи і проходить по кожному символу вхідного тексту, якщо символ є маленькою літерою, `vigenere` застосовує зсув згідно з відповідною літерою в ключі. Зсув обчислюється як індекс літери ключа в алфавіті. Потім для кожної букви вхідного тексту відбувається зсув у алфавіті на кількість, що відповідає індексу відповідної літери ключа. Зашифрований текст формується шляхом поєднання зашифрованих літер у новий рядок:

```
def vigenere(plaintext, key):
    cipher_text = []
    key_length = len(key)
    alphabet = 'абвгдеёжзийклмнопрстуфхцчщъыьэюя'
    for i, char in enumerate(plaintext):
        if char.islower():
            key_char = key[i % key_length].lower()
            shift = alphabet.index(key_char) # Отнимаемо зсув за ключем
            encrypted_char = alphabet[(alphabet.index(char) + shift) % 33]
            cipher_text.append(encrypted_char)
    return ''.join(cipher_text)
```

На цьому етапі не виникло значних труднощів. Ініціалізація змінних, обробка кожного символу, визначення зсуву за допомогою ключа та обчислення зашифрованих символів пройшли без проблем. Алгоритм шифрування Віженера виявився простим для реалізації, і всі операції, такі як перевірка на малі літери та обчислення зсуву в алфавіті, були виконані без помилок.

[illegible]

На наведеному скриншоті лише частина для прикладу виконання, решта шифртекстів у прикріплених файлах: `encrypted_{key_choice}.txt`

Перевірили правильність шифрування за допомогою [онлайн-ресурсу](#). Наприклад, слово любовѣ з ключем 11(бессмертный) – мѣтаокм. Як наведено на скриншоті, отримали слово таке ж, як і після виконання скрипта:

любовѣ

Ключ:
бессмертный

мѣтаокм

Далі ми обчислили індекси відповідності для відкритого тексту та шифртекстів за формулою:

$$I = \frac{1}{n(n-1)} \sum_{t=0}^{32} N_t(N_t - 1)$$

де N_t – кількість появи літери t в тексті.

Реалізація функції в коді:

```
def index_of_coincidence(text):  
    letter_counts = Counter(text)  
    n = sum(letter_counts.values())  
  
    total_sum = sum(Nt * (Nt - 1) for Nt in letter_counts.values())  
    index = total_sum / (n * (n - 1))  
  
    return index
```

Приклад виконання:

```
♥Меню♥  
1. Вивести текст  
2. Зашифрувати текст  
3. Обчислити індекси відповідності  
4. Розшифрувати текст  
5. Діаграми  
6. Вийти  
Виберіть опцію: 3  
  
-♥Введіть довжину ключа для знаходження індексу відповідності-♥  
Можливі довжини ключа: 2-5, 10-20  
  
Індекс відповідності для відкритого тексту: 0.05573206248345248  
  
Введіть довжину ключа або '0' для повернення: 5  
  
Індекс відповідності для ключа 5: 0.03626713062773428  
  
Введіть довжину ключа або '0' для повернення: 13  
  
Індекс відповідності для ключа 13: 0.03349471109598195  
  
Введіть довжину ключа або '0' для повернення: |
```

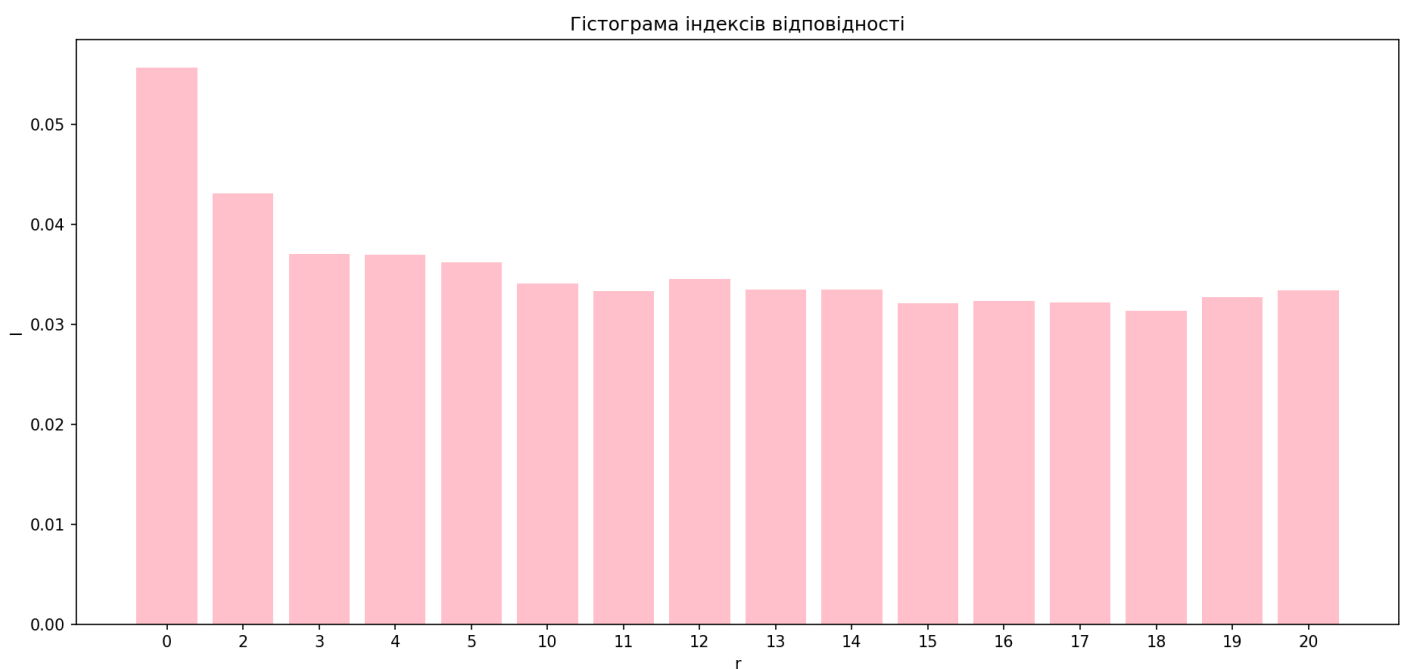
Довжина ключа	Індекс відповідності
0	0.05573206248345248
2	0.04316476921719178
3	0.037057629511958345
4	0.036958029174073655
5	0.03626713062773428
10	0.03410996381607978
11	0.03334215867972818
12	0.034548709608280694
13	0.03349471109598195
14	0.0335186656076251
15	0.0321285140562249
16	0.03237137057629512
17	0.03219612441216889
18	0.03136275955974129
19	0.03276851116406319
20	0.03341023992334556

Таблиця збережена у файлі data.csv.

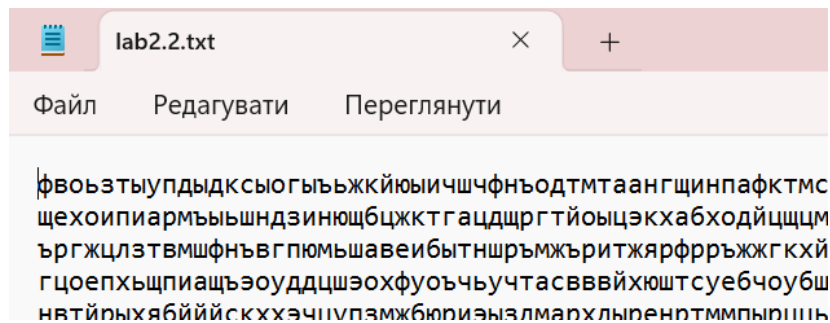
За даними методички, величина індексу відповідності та його математичне очікування має стрімко падати із ростом довжини ключа r . Проте ми помітили, що у деяких випадках це не так. Так як у шифрі Віженера кожен символ тексту зашифровується зсувом, який залежить від конкретної літери ключа, то для різних ключів зсуви символів відрізняються, і частотний розподіл у кожному зашифрованому блоці теж буде різним. Це може призвести до того, що в деяких випадках символи будуть шифруватися так, що їхній розподіл ставатиме ближчим до природного для мови, на якій написаний текст, а в інших випадках він буде менш регулярним.

З цим етапом також труднощів не виникало.

Також додали графічне зображення значень індексів відповідності для вказаних значень r :



Наступним кроком було розшифрування тексту згідно варіанту 4:



Функція `find_key_length` визначає оптимальну довжину ключа для шифру Віженера, аналізуючи індекс відповідності (`ic`) для різних можливих довжин ключа. Вона розбиває зашифрований текст на блоки відповідно до кожної довжини ключа та обчислює середній `ic` для кожного блоку. Потім порівнює середній `ic` для кожної довжини ключа з очікуваним значенням `ic` для російської мови (0.0557), який ми знайшли у попередньому пункті, і вибирає ту довжину ключа, де різниця між обчисленим і теоретичним `ic` найменша. Результат функції — це найкраща довжина ключа та словник з усіма обчисленими `ic` для кожної перевіреної довжини:

```
def find_key_length(text, max_key_length=31):
    best_key_length = 0
    closest_ic_diff = float('inf')
    ic_dict = {}
    ic_russian = 0.0557

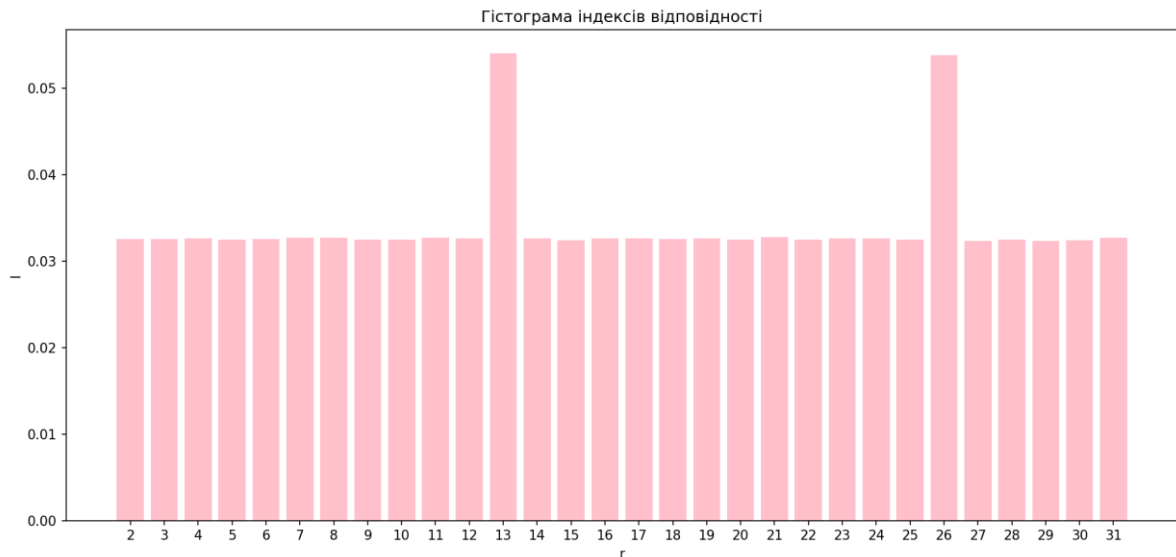
    for key_length in range(2, max_key_length + 1):
        avg_ic = 0

        # Розбиваємо текст на блоки
        for i in range(key_length):
            block = text[i::key_length]
            avg_ic += index_of_coincidence(block)

        avg_ic /= key_length
        ic_diff = abs(avg_ic - ic_russian)
        ic_dict[key_length] = avg_ic

        if ic_diff < closest_ic_diff:
            closest_ic_diff = ic_diff
            best_key_length = key_length

    return best_key_length, ic_dict
```



З наведеної діаграми індексів можна бачити, що найближче до теоретичного значення

Функція `find_key` використовує частотний аналіз для пошуку самого ключа, коли вже відома його довжина. Вона бере кожен блок тексту, отриманий розбиттям за довжиною ключа, і знаходить найпоширенішу літеру в цьому блоці. Потім, використовуючи дані з 1 лабораторної роботи, дізнались, що найчастіша літера в російській мові — це "о":

Літера	Кількість	Частота
о	65358	0.113308
е	48944	0.084852
а	46206	0.080105

Наступна функція обчислює зміщення цієї найчастішої літери відносно "о" для кожного блоку. Так обчислюється кожна літера ключа, і всі складаються в один ключ.

```
def find_key(text, key_length):
    key = []
    most_common_letter = "о"
    var_alphabet = 'абвгдежзийклмнопрстуфхцщъыьэюя'

    for i in range(key_length):
        block = text[i:key_length]
        most_common = Counter(block).most_common(1)[0][0]
        key_letter = (var_alphabet.index(most_common) -
var_alphabet.index(most_common_letter)) % len(var_alphabet)
        key.append(var_alphabet[key_letter])

    return ''.join(key)
```

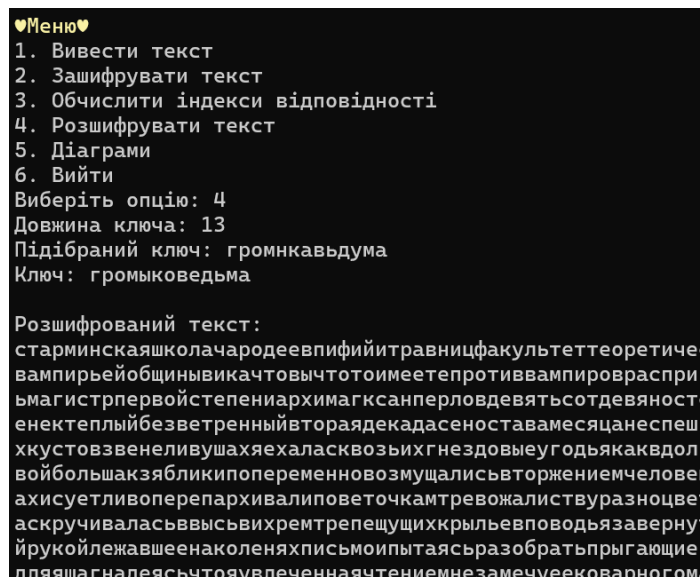
```
♥Меню♥
1. Вивести текст
2. Зашифрувати текст
3. Обчислити індекси відповідності
4. Розшифрувати текст
5. Діаграми
6. Вийти
Виберіть опцію: 4
Довжина ключа: 13
Підібраний ключ: громнкавьдума
Ключ: громьковедьма
```

Алгоритм підібрав правильно 9 літер з 13(70%), а інші ми знайшли самостійно.

Тобто, ключ: громыковедьма

Функція `vigenere_decrypt` розшифровує зашифрований текст, використовуючи знайдений ключ. Вона проходить по кожній літері зашифрованого тексту і зсуває її назад на кількість позицій, відповідних літері ключа

```
def vigenere_decrypt(cipher_text, key):  
    decrypted_text = []  
    key_length = len(key)  
    alphabet = 'абвгдежзийклмнопрстуфхцчщщъьэя'  
  
    for i, char in enumerate(cipher_text):  
        if char in alphabet:  
            key_char = key[i % key_length].lower()  
            shift = alphabet.index(key_char)  
            decrypted_char = alphabet[(alphabet.index(char) - shift) % 32]  
            decrypted_text.append(decrypted_char)  
        else:  
            decrypted_text.append(char)  
  
    return ''.join(decrypted_text)
```



```
♥Меню♥  
1. Вивести текст  
2. Зашифрувати текст  
3. Обчислити індекси відповідності  
4. Розшифрувати текст  
5. Діаграми  
6. Вийти  
Виберіть опцію: 4  
Довжина ключа: 13  
Підібраний ключ: громнкавьдума  
Ключ: громыковедьма  
  
Розшифрований текст:  
старминскаяшколачародеевпифийитравницфакультеттеоретиче  
вампирейобщинывикачтовычтооимеетепротиввампировраспри  
ьмагистрпервойстепениархимагксанперловдевятьсотдевяност  
енектепыйбезветренныйвтораядекадасеноставамесяцанеспеш  
хкустовзвенеливушахяхаласквозьихгнездовьеугодякаквдол  
войбольшакзябликипопеременновозмущалисьтворжениемчелове  
ахисуетливоперепархивалиповеточкамтревожалиствуразноце  
аскручиваласьвысвихремтрепещущихкрыльевповодьязаверну  
йрукойлежавшеенаколеняхписьмоипытаясьразобратьпрыгающие  
пляшгагнадезсчотояувлеченнаячтениемнезамечуековарногом
```

Розшифрований текст збережено у `decrypted_text.txt`.

Процес розшифрування також не викликав складнощів, оскільки для кожної літери зашифрованого тексту обчислювалося зміщення відповідно до літери ключа, що давало точний і передбачуваний результат. Оскільки в тексті не було нестандартних або незвичних символів, процес розшифрування був безпомилковим.

Висновки: робота з шифром Віженера показала, що стійкість шифр залежить від довжини та складності ключа. Водночас, цей шифр є відмінним навчальним прикладом для розуміння основних принципів шифрування та методів криптоаналізу, таких як частотний аналіз та використання індексу відповідності.