Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
*Факультет комп'ютерно-інформаційних систем і програмної інженерії*
(назва факультету )
*Кафедра кібербезпеки*
(повна назва кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: *"Криптоаналіз історичних шифрів заміни"*

Виконав: студент (ка)
Спеціальності:
*125 «Кібербезпека»*
(шифр і назва напряму підготовки, спеціальності)
*Цубера В.В.*
підпис (прізвище та ініціали)

Керівник *Стадник М.А.*
підпис (прізвище та ініціали)
Нормоконтроль *Кареліна О.В.*
підпис (прізвище та ініціали)

Завідувач кафедри *Загородна Н.В.*
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2021

# АНОТАЦІЯ

Криптоаналіз історичних шифрів заміни // Кваліфікаційна робота ОР «Бакалавр» //Цубера Василь Васильович// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБсз-41 // Тернопіль, 2021 // С.51 , рис. – , табл. – , кресл. – , додат. – 5 .

Ключові слова: КРИПТОГРАФІЯ, АТАКА, ШИФР, КРИПТОАНАЛІЗ, ЧАСТОТНИЙ АНАЛІЗ, МОНОАЛФАВІТНИЙ ШИФРБ ПОЛІАЛФАВІТНИЙ ШИФР.

Кваліфікаційна робота присвячена розробці програмного забезпечення для криптоаналізу історичних шифрів та дослідженню впливу частотного аналізу на них. В роботі обґрунтовано вибір програмного середовища розробки та вибір методів криптоаналізу моно- та поліалфавітних шифрів.. Розроблено програмне забезпечення, в якому реалізовано найпростіші методи криптоаналізу шифрів заміни, яке може бути використане в навчальних цілях для злому шифрованих текстів.

В роботі використано відомі методи криптоаналізу для моноалфавітних, біграмних та поліалфавітних шифрів та проведено оцінку складності злому текстів, зашифрованих різними шифрами. Встановлено, що шифр Віженера є найбільш складним для взлому, особливо якщо період ключа достатньо великий, або текст достатньо короткий. Встановлено також, що всі ці методи використовують частотний аналіз, який можливий лише при достатній довжині шифротексту.

# ANNOTATION

Cryptoanalysis of historical substitution ciphers// Thesis of educational level "Bachelor" // Tsubera Vasyl Vasylovych // Ternopil National Technical University named after Ivan Pulyuy, Faculty of Computer Information Systems and software engineering, Department of Cybersecurity, СБсз-41 group // Ternopil, 2021 // P. 51 , fig. -, table. - , chair. - , added. -5.

Keywords: CRYPTOGRAPHY, ATTACK, CIPHER, CRYPTANALYSIS, FREQUENCY ANALYSIS, MONOALPHABETIC CIPHER, POLYALPHABETIC CIPHER.

The qualification thesis is devoted to the development of software for cryptanalysis of historical ciphers and research of the influence of frequency analysis on them. The paper substantiates the choice of software development environment and the choice of methods of cryptanalysis of mono- and polyalphabetic ciphers.

The known methods of cryptanalysis for monoalphabetic, bigram and polyalphabetic ciphers are used in the thesis and the complexity of hacking of texts encrypted with different ciphers is estimated. It is established that the Vigener cipher is the most difficult to crack, especially if the key period is long enough or the plaintext is short enough. It is also established that all these methods use frequency analysis, which is possible only with a sufficient length of ciphertext.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

| | |
|---|---|
| AI | Artificial Intelligence |
| XSS | Cross-Site Scripting attack |
| ВТ | Відкритий текст |
| НСД | Найбільший спільний дільник |
| НСК | Найменше спільне кратне |
| ШТ | Шифротекст |

# ВСТУП

Шифрування - це процес перетворення даних таким чином, що вони залишаються прихованими, незрозумілими або недоступними для неавторизованих користувачів. Після того, як дані зашифровані, лише уповноважені сторони, які мають “ключ”, можуть читати їх або використовувати. По суті, коли дані зашифровані, навіть якщо неавторизована особа або організація отримає доступ до них, вони не зможуть їх прочитати. Тобто, якщо метод шифрування ефективний, він повинен повністю захищати дані від несанкціонованого доступу. Це допомагає захистити приватну інформацію, конфіденційні дані та може підвищити безпеку спілкування між клієнтськими програмами та серверами.

За даними The Software Alliance, кіберзлочинці вкрали 423 мільйони особи в 2015 році. У 2018 році через порушення даних було виявлено витік п'яти мільярдів записів, що було зменшенням з 7,9 мільярдів записів, які були скомпрометовані в 2017 році, але все ще не мало. Загрози безпеці даних продовжують зростати: у 2019 році серед основних проблем кібербезпеки є відносно нові типи загроз, такі як викрадення форм, XSS-атаки між сайтами та бот-мережі зі штучним інтелектом (AI). Споживачі не єдині жертви, чиї дані скомпрометовані, компанії часто також втрачають дані про працівників.

Компанії повинні добре знати методи шифрування та комунікації, щоб захистити власні та конфіденційні дані своїх клієнтів. Обсяг даних кіберпростору за різними оцінками у 2020 році склав  близько 40 зеттабайта (ZB) , що, за даними Світового економічного форуму, "у 40 разів більше байт, ніж зірок у спостережуваному Всесвіті". Завдяки такому обсягу даних шифрування є абсолютно необхідною умовою спілкування в Інтернеті з використанням конфіденційності та безпеки.

Криптоаналіз - це вивчення методів отримання вмісту зашифрованої інформації без доступу до секретної інформації (ключа), яка зазвичай потрібна для цього. Як правило, це передбачає знання того, як працює система, та пошук секретного ключа. Криптоаналіз також називають розбиванням шифру або

зломом шифру. Зашифрований текст, як правило, є найпростішою частиною криптосистеми для отримання, а отже, є важливою частиною криптоаналізу. Залежно від того, яка інформація доступна і який тип шифру аналізується, криптоаналітики можуть слідувати одній або декільком моделям атак, щоб зламати шифр.

Криптоаналіз використовує формули для пошуку вразливостей алгоритмів та проникнення в криптографічний захист або системи захисту інформації. Одними з найбільш загроз на сьогоднішній день є ransomware, програмне забезпечення, яке шифрує персональні дані і використовується з метою збагачення шахраїв. Розуміння базових основ криптографії та криптоаналізу є передумовою того, що дані залишатимуться захищеними. Зрозуміло, що сучасні криптосистеми володіють достатньою стійкістю до атак криптоаналізу, саме це і сприяє успіху програм-шифрувальників (ransomware), тому в даній роботі ми розглянемо можливості взлому історичних шифрів заміни, що дасть можливість , по-перше, зрозуміти, що ці шифри неварто використовувати, а, по друге, блоки заміни існують в кожному сучасному симетричному блочному шифрі

*Метою* даної роботи є розробка програмного забезпечення для злому історичних шифрів заміни, яке може бути використане для навчальних завдань.

Для досягнення поставленої мети потрібно розв'язати наступні завдання:

- Провести огляд літературних джерел в  предметній області.
- Проаналізувати основні історичні шифри заміни.
- Проаналізувати відомі криптоаналізу шифрів.
- Обрати середовище розробки.
- Розробити програмне забезпечення.
- Дослідити дієвість методів злому.
- Протестувати розроблене програмне рішення.

# 1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

## 1.1 Криптографія та її застосування

Криптографія пов’язана з процесом перетворення звичайного звичайного тексту в незрозумілий текст (шифротекст) і навпаки. Шифрування є дуже важливим по суті, оскільки воно захищає дані та інформацію від несанкціонованого доступу і, таким чином, зберігає конфіденційність. Проте до задач криптографії належить не лише захист даних від крадіжки, сфери застосування криптографії є значно ширшими. Розглянемо їх детальніше.

*Безпека передачі даних*

До 70-х років минулого століття чи не єдиним завданням криптографії було забезпечення безпеки комунікації. Проте в сферу інтенсивного розвитку інформаційних технологій, спілкування може відбуватися багатьма шляхами, тому ця функція криптографії передбачає широкий спектр захисту каналів комунікації, в тому числі і:

- Web трафік: HTTP/HTTPS (SSL/TLS).
- Wireless трафік: протоколи WEP (IEEE 802.11, 1997), WPA (2003), WPA2 (IEEE 802.11i , 2004).
- Мобільний трафік: GSM.
- Bluetooth.

*Секретність зберігання даних*

Зберігання даних у секреті зазвичай підтримується системою з одним ключем, де користувач надає ключ до комп’ютера на початку сеансу, а потім система дбає про шифрування та дешифрування протягом звичайного використання. Наприклад, багато апаратних пристроїв доступні для персональних комп’ютерів для автоматичного шифрування всієї інформації, що зберігається на диску. Інформацію неможливо прочитати змістовно без цього ключа, тому навіть якщо диск викрадено, інформація на ньому не буде використана. Проте, якщо

користувач забув ключ, вся зашифрована ним інформація стає назавжди втраченою.

*Цілісність передачі даних*

Багатьох користувачів комунікаційних систем турбує не стільки секретність, скільки цілісність. При електронному переказі коштів сума, що надходить з одного рахунку на інший, часто є загальнодоступною. Банк дбає про те, щоб могли здійснюватися лише належні перекази. Якби шахрай міг провести фальшивий переказ, кошти були б переміщені незаконно. Помилка в одному біті може буквально призвести до помилкового зарахування або списання мільйонів доларів. Криптографічні методи широко використовуються для того, щоб переконатись, що навмисне або випадкове внесення змін до переданої інформації не порушує цілісності інформації.

*Аутентифікація користувача*

Автентифікація особи чи системи одне одному є дуже давньою проблемою, яка особливо гостро постає у віртуальному світі. Прості паролі використовувались тисячі років для підтвердження особистості. Більш складні протоколи, такі як послідовності ключових слів, якими обмінюються торони, часто демонструються у фільмах або на телебаченні. Криптографія тісно пов’язана з теорією та практикою використання паролів, і сучасні системи часто використовують сильні криптографічні перетворення у поєднанні з фізичними властивостями людей та загальними секретами для забезпечення надійної автентифікації особи.

*Цифровий підпис*

Цифрові підписи, як і їхні фізичні аналоги, є засобом забезпечення юридично обов’язкової операції між двома або більше сторонами. Щоб бути таким самим корисним, як фізичний підпис, електронний підпис повинен бути принаймні настільки ж важким для підробки, принаймні таким самим простим у використанні та прийнятим на суді як обов'язковий для всіх сторін угоди.

Потреба в цих електронних підписах є особливо важливою в ділових відносинах, коли сторони у договорі не мають можливості бути фізично присутніми при підписанні. В даний час в Європі триває робота із заміни фізичних підписів електронними підписами на основі криптосистеми RSA. Якщо ці зусилля

увінчаються успіхом, це дозволить заощадити багато мільйонів доларів і почне еру цифрових підписів у повномасштабному русі.

*Зберігання паролів*

Пароль є невід'ємною частиною криптографії. Дуже ризиковано зберігати паролі доступним способом. Якщо система зберігає пароль у відкритому вигляді, кожен, хто має доступ до системи - законний чи зловмисний, може прочитати пароль. Шифрування - це лише часткове вирішення проблеми збереження паролів. Якщо хтось має доступ до системи, що зберігає зашифровані паролі, він, ймовірно, матиме доступ до ключа шифрування для дешифрування пароля. Хешування (одностороннє відображення рядка до значення фіксованої довжини) має значно більшу практичну цінність. Адже знання хешу зловмисником не дає його жодної ідеї про сам пароль. Система приймає пароль при вході, хешує його і порівнює із записаним хешованим значенням. У жодному разі система - або зловмисник - не матиме доступу до особистої інофрмації

*Штамп часу*

Штампування часу - це техніка, яка може засвідчити, що певний електронний документ або зв’язок існував або був доставлений у визначений час. Штамп часу використовує криптосхему, яка називається схемою сліпого підпису. Схеми сліпого підпису дозволяють відправнику отримувати повідомлення, отримане іншою стороною, не розкриваючи ніякої інформації про повідомлення іншій стороні. Штампування часу дуже схоже на відправлення рекомендованого листа поштою в США, але забезпечує додатковий рівень підтвердження. Це може довести, що одержувач отримав конкретний документ, при цьому не знаю змісту самого листа. Можливі заявки включають заявки на патенти, архіви авторських прав та контракти. Відмітка часу є критично важливим додатком, який допоможе зробити можливим перехід на електронні юридичні документи.

*Електронні гроші*

Визначення електронних грошей (їх також називають електронною або цифровою готівкою) - це термін, який все ще розвивається. Він включає операції, здійснені в електронному вигляді з чистим переказом коштів від однієї сторони до іншої, яка може бути як дебетовою, так і кредитною та може бути анонімною

або ідентифікованою. Існують як апаратні, так і програмні реалізації. Анонімні схеми є електронним аналогом готівки, тоді як ідентифіковані схеми - електронним аналогом дебетової або кредитної картки. Існують також деякі гібридні підходи, коли платежі можуть бути анонімними щодо продавця, але не банку; або анонімні для всіх, але простежувані (послідовність покупок може бути пов'язана, але не пов'язана безпосередньо з ідентифікацією платника).

Існує багато інших, не менш цікавих, криптопротоколів, але в своїй роботі я зосередився на базових методах шифрування, а вірніше на їх витоках. Розглянемо детальніше класифікацію криптографічних методів.

### 1.2 Класифікація криптографічних методів

Загалом методи криптографії можна поділити на історичні та сучасні (рис.1.1). Цілком окремо стоїть квантова криптографія – розділ криптографії, що будується на основних принципах квантової фізики.

До історичних криптографічних методів належать:

- Шифри перестановки (Шифр Скітали, проста перестановка, маршрутні шифри (табличні, решітка Кардано та інші)).
- Шифри заміни (квадрат Полібія, шифр Цезаря, шифр афінної підстановки, шифр Віженера та інші).
- Комбіновані шифри

На світанку свого розвитку історичні шифри реалізовувались мануально, оскільки не було технічних засобів для автоматизації процесу шифрування. В 90-х роках XIX ст. – початку XX ст. з'явились механічні шифрувальні машини. Ближче до середини XX століття отримали свій розвиток електро-механічні шифрувальні машини ("Енігма", наприклад)

Сучасні криптосистеми бувають:

a. Симетричні, що в свою чергу поділяються на блочні та потокові.

b. Асиметричні криптосистеми, відомі також під назвою криптосистем з відкритими ключами.

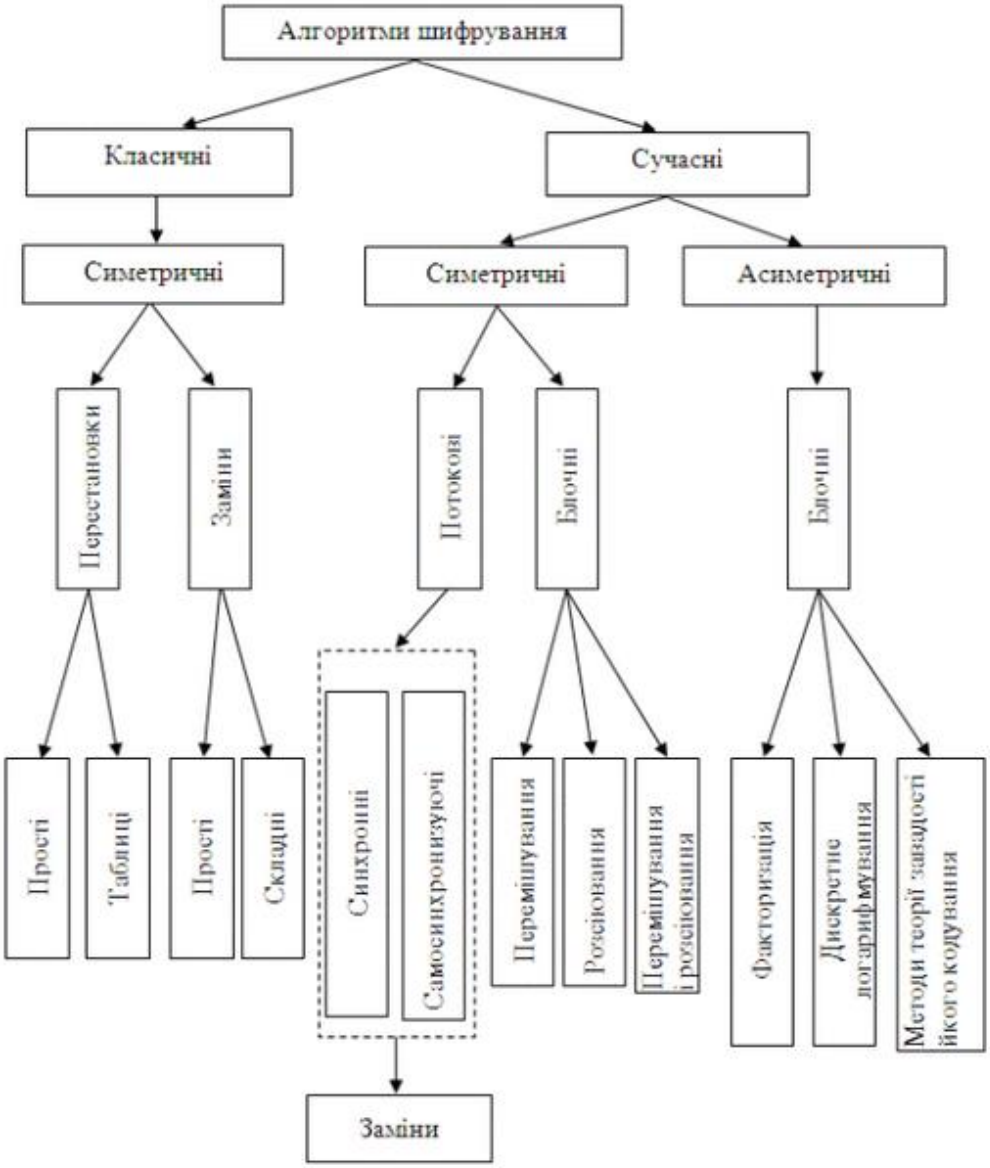


Рисунок 1.1 – Класифікація криптографічних методів

Основною відмінністю між симетричними та асиметричними алгоритмами є те, що симетрична криптографія базується на принципах історичної і вимагає наявності одного секретного ключа, який використовується як для зашифрування інформації, так і для її розшифрування. В свою чергу асиметричні алгоритми використовують два ключа, пов'язаних між собою: відкритий для зашифрування інформації, а закритий – для її розшифрування.

Оскільки невід'ємною частиною кожного блочного шифру є блоки заміни, то розглянемо історичні шифри заміни, щоб зрозуміти розвиток та ідею цих методів.

### 1.3 Історичні шифри заміни

Шифр заміни – це схема шифрування даних, в якій одиниці відкритого тексту (як правило, одиничні літери або пари букв звичайного тексту) замінюються іншими символами або групами символів. В залежності від правил заміни, шифри заміни поділяються на моно- та поліалфавітні [1].

#### 1.3.1 Моноалфавітні шифри

Моноалфавітні шифри – це найпростіші шифри заміни, які кожний символ тексту замінюють на деякий фіксований (якщо, для прикладу, згідно правил заміни, буква "а" заміщається буквою "р", то ця заміна відбувається для всіх наявних букв а в тексті). Існує безліч різновидів історичних моноалфовітних шифрів. Розглянемо найбільш відомі з них [2].

*Квадрат Полібія*

Один з найдавніших простих шифрів заміни належить до винаходів відомого грецього історика історика Полібія (III-II ст. до н.е.). В одній із своїх робіт він описав оригінальний алгоритм шифрування, що використовував таблицю у формі квадрату. Це була квадратна таблиця розміром 5x5, заповнена літерами грецького алфавіту (рис. 1.2) , про порядок яких можна було домовлятись заздалегідь.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | α | β | γ | δ | ε |
| 2 | ζ | η | θ | ι | κ |
| 3 | λ | μ | ν | ξ | ο |
| 4 | π | ρ | ς | σ | τ |
| 5 | υ | φ | χ | ψ | ω |

Рисунок 1.2 – Приклад квадрату Полібія, побудованого на грецькому алфавіті

Використовуючи схожий алгоритм, таку таблицю можна сформувати для будь-якої мови, при чому дві схожі букви, які не вплинуть на розуміння тексту,

можна об'єднувати в одну комірку і заміняти однією буквою (наприклад, і/ї в українській мові). Для того, щоб розшифрувати щифротекст необхідно знати таблицю шифрування, її алфавіт і порядок літер. В найпростішому ипадку літери розташовують в алфавітному порядку. Існує модифікація шифру, коли спочатку записують ключове слово, а потім всі інші букви алфавіту в звичному порядку, за винятком тих, що йдуть в ключовому слові.В літературі описано декілька варіантів шифру Полібія.

Шифрування. У найпростішому випадку шифрування літеру відкритого тексту потрібно знайти у цьому квадраті, а у криптограму записати літеру, яка розміщена рядком нижче в тому ж стовпчику. Якщо літера стоїть у нижньому рядку таблиці, то для її відповідника у шифрограмі потрібно обрати саму верхню літеру з того ж стовпчика.

Дешифрування. В симетричних системах отримувач шифротексту повинен був володіти однаковим секретним ключем з відправником (аналогічною таблицею) і при дешифруванні замість літери криптограми записати літеру, яка у таблиці розташована рядком вище.

*Шифр Цезаря*

Шифр Цезаря, описаний у працях римського історика Гая Светонія. Римський імператор Гай Юлій Цезар використовував придуманий самостійно шифр для особистої переписки.

Принцип перетворення інформації, що використовувався в даному шифрі був дуже простим. Кожна буква послання замінювалась на іншу, яка в алфавіті стояла на постійному зміщенні кількості позицій після букви відкритого тексту. Сам Цезар все життя використовував один і той самий ключ. Більшість істориків схильні думати, що Цезар шифрував свої повідомлення зі зміщенням 3 символи.

В загальному випадку шифру Цезаря, кожна літера відкритого тексту замінюється на іншу літеру того ж алфавіту, шляхом циклічного зсуву на k букв вперед. Для деякого алфавіту обсягом *n (0 ≤ j ≤ n-1)* символів, секретним ключем є величина зміщення вправо $k \in \{0,2,\dots,n-1\}$

Більш формально процеси шифрування та дешифрування можна записати якщо кожному символу алфавіту поставити у відповідність його порядковий

номер ( починаючи з 0). Нехай задано деякий ВТ : $m = "m_1 m_2 \dots m_r"$ ШТ, що відповідає даному ВТ позначається: $c = "c_1 c_2 \dots c_r"$.

Тоді шифрування можна подати формулою:

$$J(\mathrm{c_i}) := J(m_i) + k \mod n \tag{1.1}$$

де $m_i$ – символ відкритого повідомлення;

$c_i$ - символ шифротексту;

n – кількість елементів алфавіту;

k – секретний ключ ;

$J(x)$ - функція що обчислює порядковий номер літери $x$ в алфавіті $\mathrm{A_n}$.

Дешифрування можна виконати як

$$J(m_\mathrm{i}) := J(c_i) - k \mod n \tag{1.2}$$

*Шифр афінної підстановки*

Афінна криптосистема є узагальненням Цезаря. Проте секретним ключем тут вже виступає пара цілих чисел k = $\{a, b\}$ така що *0≤a, b≤n-1* (n – потужність алфавіту) причому *НСД (a, n)= 1*. Якщо *a=1,* то шифр афінної підстановки перетворюється на шифр Цезаря

Простір ключів тут вже набагато більший

$|\mathrm{K}| = n * \varphi(n)$ , де $\varphi(n)$ - функція Ейлера.

Шифрування можна записати формулою:

$$\forall i = \overline{1,r} \;\; J(c_i) = (a * J(m_i) + \; b) \mod n\,, \tag{1.3}$$

де $\mathrm{J(m_i)}$ - функція, що ставить у відповідність порядковий номер символу $\mathrm{m_i}$ в алфавіті $\mathrm{A_n}$.

Дешифрування виконується за формулою:

$$\forall i = \overline{1,r} \quad J(m_i) = (J(c_i) - \ b) * a^{-1} \quad mod\, n \tag{1.4}$$

$a^{-1}$ обернений елемент до $a$ в полі $Z_n$ , що існує стовідсотково завдяки тому, що НСД (a, n)= 1. Обернений елемент можна знайти з допомогою розширеного алгоритму Евкліда.

У всіх моноалфавітних шифрів є спільний недолік, а саме те, що зберігається частота використання символів в тексті, що і дає підстави для його криптоаналізу.

### 1.3.2 Поліалфавітні шифри заміни

Щоб усунути недолік, пов'язаний зі збереження розподілу частоти літер у моноалфавітних потрібно одну і ту ж букву початкового повідомлення спробувати заміняти на різні букви в залежності від е=певних факторів (наприклад позиції в тексті). В цьому і полягає основна ідея поліалфавітних шифрів. Отже, в такому випадку, кожна поява символу може мати різну літеру на заміну, тобто в цілому повідомленні та сама буква може бути зашифрована по-різному. Тобто заміни для букви вибираються з багатьох алфавітів залежно від положення в тексті. Під алфавітами мається на увазі не алфавіти різних мов (наприклад української, німецької, англійської), а алфавіти з різним порядком літер для заміни в межах однієї мови. Це є гарним захистом від простого підрахунку частот, тому що не існує єдиної заміни для кожної букви в шифротексті. Найпростішими варіантами поліалфавітних шифрів є біграмні шифри [3].

*Шифр біграмної афінної підстановки*

Афінна підстановка біграм передбачає шифрування тексту по дві букви за раз. Відкритий текст (ВТ) $m = "m_1 m_2 \dots m_r"$ розбивається на біграми, що не перетинаються, $(m_1 m_2)$,$(m_3 m_4)$,… Якщо n – потужність(обсяг) алфавіту, то занумеруємо букви алфавіту числами від 0 до n-1. Тоді кожній біграмі $(m_i m_{i+1})$ можна поставити у відповідність єдине число $X_i$ у межах від 0 до $m^2$-1:

$$J(m_i m_{i+1}) = X_i = J(m_i) \cdot n + J(m_{i+1}) \tag{1.5}$$

Шифрування біграм відбувається за формулою схожою до (1.3). Маючи секретний ключ k = $\{a, b\}$ , причому *0≤a, b≤n-1*, НСД$(a, n^2) = 1$, можна знайти числовий відповідник біграми $(m_i m_{i+1})$ у шифротексті

$$Y_i = (a * X_i + \ b\ ) mod\ n^2 \tag{1.6}$$

При чому, для того, щоб від числа $Y_i$ перейти до пари букв $(\mathrm{c}_1 \mathrm{c}_2)$ потрібно виконати операцію ділення числа $Y_i$ на $n$. Тоді неповна частка від ділення визначатиме $J(\mathrm{c}_i)$, а остача $J(\mathrm{c}_{i+1})$.

Для розшифрування шифротексту $c = "c_1 c_2 \dots c_r"$ потрібно спершу перейти до чисел, що визначають біграми шифротексту

$$J(\mathrm{c}_i \mathrm{c}_{i+1}) = Y_i = J(c_i) \cdot n + J(mc_{i+1}) \tag{1.7}$$

А потім провести розшифрування знаючи ключі $a$ та $b$ за формулою

$$X_i = (Y_i - \ b\ ) \cdot a^{-1} mod\ n^2, \tag{1.8}$$

де $a^{-1}$ обернений елемент до $a$ в полі $Z_{n^2}$, що існує тільки у випадку, якщо НСД$(a, n^2) = 1$.

*Шифр Плейфейера*

У даному шифрі алфавіт розташовується в таблиці в довільному порядку, схожим чином до таблиці Полібія і визначає ключ *k*. Таблицю можна сформувати на основі деякого секретного слова, що записується в Таблицю зліва направо, починаючи з першої комірки. Повторювані літери ключового слова упускаються. Інші комірки заповнюються невикористаними літерами, що залишилися в алфавіті. Щоб уникнути пустих комірок в алфавіт можна додати розділові символи або інші унікальні символи.

Перед шифруванням відкритий текст $m = "m_1 m_2 \dots m_r"$ розбивається на на пари символів, що не перетинаються, $(m_1 m_2)$,$(\ m_3 m_4)$,… $(\mathrm{m_i m_{i+1}})$. Якщо дві

букви пари одинакові то, щоб потрібно вставити між ними фіктивний символ. Після вставки фіктивних символів, якщо загальна кількість букв у відкритому повідомленні непарна, то в кінці додається ще один фіктивний символ.

Кожна пара символів відкритого тексту заміняється на пару символів з таблиці наступним чином:

- якщо символи розташовані в одному рядку, то кожний із символів пари замінюється на символ, що стоїть праворуч;
- якщо символи розташовані в одному стовпці, то кожний символ пари замінюється на символ, що стоїть нижче нього;
- якщо символи пари перебувають у різних рядках і стовпцях, то вони вважаються протилежними кутами прямокутника. Символ, що знаходиться в лівому куті, шифрується символом, що стоїть в іншому лівому куті; заміна символу, що знаходиться в правому куті, проводиться аналогічним способом. Заміна виконується відповідно до порядку символів відкритої біграми. З цього алгоритму зрозуміло, що одна і та ж буква може замінятися на іншу букву шифротексту, в залежності від того, який в біграмі другий символ.

*Шифр Віженера*

Нехай задано деякий ВТ : $m = "m_1 m_2 \dots m_r"$ в деякому алфавіті $A_n$

Секретне слово розміром $l$: $\mathrm{k} = "\mathrm{k}_1 k_2 k_3 \dots k_l"$ $k_i \in A_n$ є ключем в шифрі Віженера. Зрозуміло, що чим довше слова, більша надійність шифру. Отже для того, щоб зашифрувати повідомлення спочатку, так би мовити, розгортають ключ. Ключове слово копіюють необхідну кількість раз на всю довжину повідомлення $k = k_0 k_1 \ \dots \ k_{l-1} k_0 \ \dots \ k_{r\,(mod\ l)}$

Якщо кожному символу алфавіту поставити у відповідність його порядковий номер (починаючи з 0), то шифрування та дешифрування можна записати у вигляді виразів:

$$\begin{aligned} J(c_i) &= \big(J(m_i) + J(k_{i\ mod\ l})\big)(mod\ n), i = \overline{1,r} \\ J(m_i) &= \big(J(c_i) - J(k_{i\ mod\ l})\big)(mod\ n), i = \overline{1,r} \end{aligned} \qquad (1.9)$$

де $m_i$ – символ відкритого повідомлення;

$c_i$ - символ шифротексту;

$k_i$ – номер символу ключа ;

n – потужність алфавіту;

$J(x)$ - функція що обчислює порядковий номер літери $x$ в алфавіті A.

У випадку, якщо довжина ключа K рівна одному символу, d=1, то одержуємо класичний шифр Цезаря.

На рис. 1.3 наведено графіки розподілу значень частоти для текстів, шифрованих за допомогою шифру Плейфеєра, Віженера та для порівняння відкритого тексту.

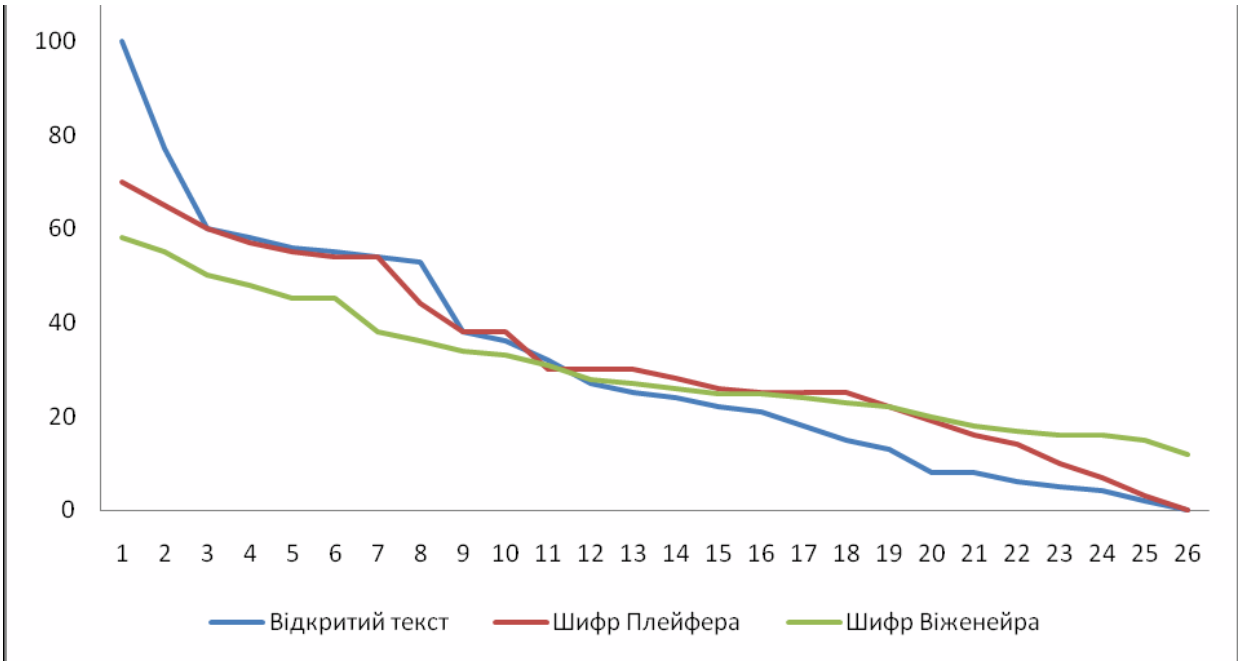


Рисунок 1.3 – Відносні частотні графіки відкритого та зашифрованого тексту

Якщо в процесі шифрування інформація про розподіл літер повністю приховується, та графік для шифротексту, отриманого таким шифром повинен становити горизонтальну пряму лінію, а криптоаналіз такого тексту з використанням лише шифрованого тексту, очевидно, може виявитися практично нереалізованою задачею. Розглянемо методи криптоаналізу у наступному розділі.

# 2 КРИПТОАНАЛІЗ ІСТОРИЧНИХ ШИФРІВ ЗАМІНИ

Криптоаналіз історичних шифрів заміни може бути абсолютно різним і будуватись на математичних властивостях алгоритму. Так, скажімо, для шифру Цезаря достатньо перебрати весь простір ключів і віднайти змістовий текст.

Для прикладу для шифротексту *“uryybjbeyq*” після перебору 26 можливих ключів отримаємо наступні результати

- tqxxaiadxp;
- spwwzhzcwo;
- …
- helloworld;
- …

Як бачимо змістовний текст доволі легко розпізнати. У випадку моноалфавітної афінної підстановки простір ключів є квадратично більший, тобто може становити 600-800 варіантів перебору, але для сучасних комп'ютерів це не проблема. Більшою проблемою є перегляд текстів і відбір того єдиного змістовного тексту. Проте абсолютно всі моноалфавітні шифри можуть бути зламані на підставі того, що вони зберігаю розподіл появи букв в тексті. Метод криптоаналізу, який використовує цю властивість називається частотним криптоаналізом.

## 2.1 Криптоаналіз моноалфавітних шифрів.

У галузі криптоаналізу частотний аналіз - це методологія "зламу" простих шифрів заміни, не тільки шифру Цезаря, а й усіх моноалфавітних шифрів заміни. Ці шифри замінюють одну букву відкритого тексту іншою, щоб створити зашифрований текст, і будь-яка конкретна буква в відкритому тексті завжди, в найпростіших і найлегших з цих шифрів, перетвориться на ту саму букву в шифрі. Наприклад, всі Е перетворюються на Х.

### 2.1.1 Криптоаналіз шифру Цезаря

Невеликий простір ключів робить brute-force атаку найефективнішим методом криптоаналізу шифру Цезаря.

Для зламу шифру необхідно кожну букву шифротексту на літеру, яка розташована в алфавіті поруч ліворуч. Якщо, в результаті такої заміни не вдалося отримати змістовне повідомлення, то необхідно продовжити криптоаналіз, замінивши букви шифротексту на літери, які стоять на дві літери ліворуч. І знову перевірити текст на змістовність. Отже, в найгіршому випадку нам доведеться перебрати n-1 варіантів, де n – потужність алфавіту.

### 2.1.2 Криптоаналіз шифру афінної підстановки

Для сучасних комп'ютерів, очевидно, найпростішим способом зламу шифру афінної підстановки, описаної в п.1.3.1 кваліфікаційної роботи, як у випадку шифру Цезаря, є перебір всіх можливих варіантів. Зрозуміло, що у випадку двох ключів кількість варіантів суттєво більша. Простір ключів афінної підстановки складає

$$|K| = n \cdot \varphi(n) \tag{2.1}$$

де n – потужність алфавіту, $\varphi(n)$ – функція Ейлера, що обчислюється за формулою:

$$\varphi(n) = \left(p_1^{k_1} - p_1^{k_1-1}\right) \cdot \left(p_2^{k_2} - p_2^{k_2-1}\right) \cdots \left(p_m^{k_m} - p_m^{k_m-1}\right) \tag{2.2}$$

Функція Ейлера $\varphi(n)$ визначає кількість чисел взаємопростих з n і менших за n. Зрозуміло, що максимальне значення функція Ейлера буде мати, якщо n – це просте число. В такому випадку $\varphi(n) = n - 1$. Якщо n=33, то $\varphi(n) = 20$, а $|K| = 660$. Якщо, для прикладу n=31, то $\varphi(n) = 30$, а $|K| = 930$. Як бачимо, навіть якщо n-менше, але просте, то простір ключів зростає на третину. Зрозуміло, що комп'ютеру нескладно перебрати всі 600-900 варіантів. Але в результаті потрібно

буде оцінити на змістовність всі тексти, що є далеко не простою задачею, якщо виконувати її вручну. Щоб автоматизувати цей процес використовують частотний аналіз тексту та мови.

### 2.1.3 Частотний криптоаналіз

Частотний аналіз базується на тому, що певні літери та їх комбінації бувають з характерною частотою практично у всіх текстах певної мови.

На рисунку 2.1 наведено частоти, з яким англійські літери зустрічаються в текстах.

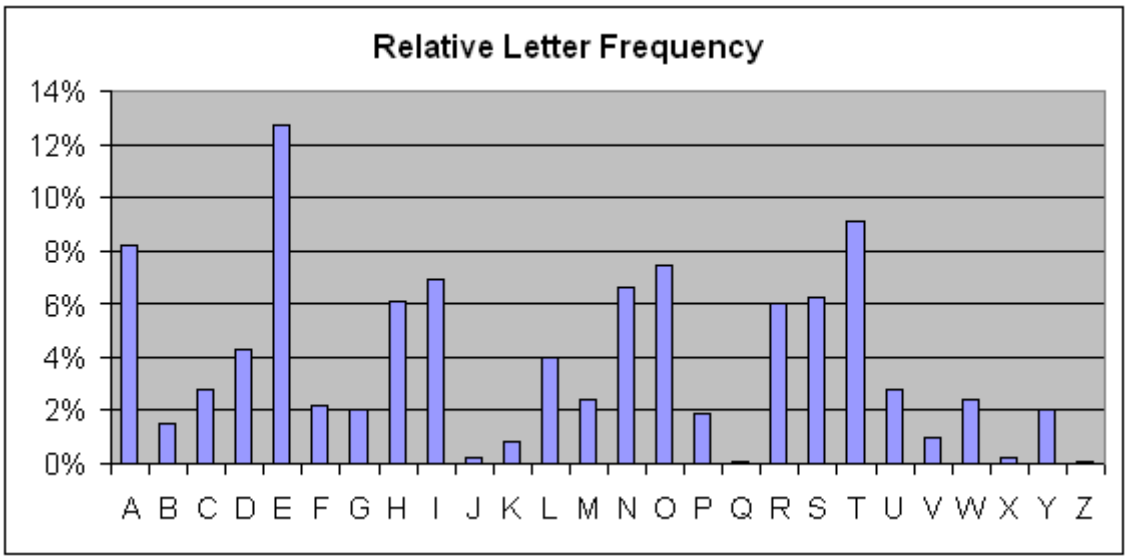


Рисунок 2.1 – Розподіл англійських літер в літературному тексті

Як бачимо, в англійській мові E дуже поширена, тоді як X - ні. З мінімальною імовірністю можна зустріти букви X, J, Q, Z. Подібним чином ST, NG, TH та QU є загальними поєднаннями, тоді як XT, NZ та QJ надзвичайно рідкісні або "неможливі".

На рисунку 2.2 приведено середню частоту букв українського алфавіту

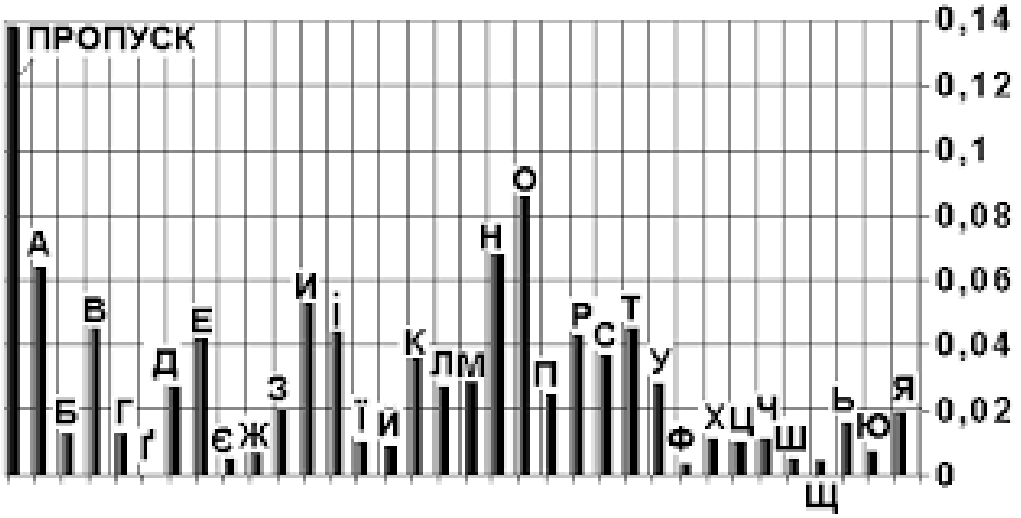


Рисунок 2.2 – Гістограма частот використання букв алфавіту української мови

Як бачимо, з голосних літер на першому місці стоїть "о", а на другому "а". Загалом високу частоту мають літери "в", "н", "т" "и", "і".

Отже, суть методу частотного криптоаналізу полягає в порівнянні розподілі літер у криптограмі з стандартним розподілом літер тієї чи іншої мови. Найбільш часто вживані літери в шифротексті заміняються на їх ймовірні відповідники на основі співставлення частот. Чим більша довжина криптотексту, тим більша імовірність успішного розкриття. Існує багато різних таблиць про розподіл літер у тій або іншій мові, але жодна з них не містить ідеальної інформації для розкриття - навіть порядок літер відповідно до їх частот може різнитися в різних таблицях. Звичайно, що розподіл букв літературного тексту та технічних інструкцій чи програмних кодів може бути абсолютно різним, але маючи апріорну інформацію про тип секретного повідомлення, таку статистику абсолютно нескладно отримати. Тому першою умовою для використання цього методу є отримання частоту букв шифротексту, а потім використання або публічної інформації про розподіл літер в тій чи іншій мові, або встановлення такого розподілу самостійно на основі аналізу значної кількості текстів. В [4] наведено ґрунтовне дослідження про частоти використання одинарних букв та біграм української мови, тому для свого дослідження я брав тексти української мови. При виконання частотного криптоаналізу доцільно також розуміти, які букви можуть зустрічатись в окремих однобуквених словах, які найчастіші двобуквенні слова. Адже, скажімо, буква "н"

самостійно зустрічатись не може, тому доцільно шукати відповідник букви шифротексту серед близьких за частотою букві "н", але з тих, які можуть зустрічатися окремими словами.

Як і в попередньому випадку можна було б вручну оцінювати змістовність тексту. Якщо робити заміну поступовою, то, це мабуть і найкращий варіант. Якщо ж автоматизувати процес, то для оцінки змістовності тексту можна використати величину:

$$X = \sum_{i=1}^{n} p_i f_i, \tag{2.3}$$

де $p_i$ – частота і-ої букви в шифротексті, а $f_i$ – частота букви в алфавіті природньої мови. Більші значення X свідчать більшу ймовірність того, що текст написаний природньою мовою. Частотні характеристики англійської мови можна знайти на сайті [5]

### 2.2 Криптоаналіз поліалфавітних шифрів заміни

Зрозуміло, що частотний аналіз символів при застосуванні поліалфавітних шифрів заміни не є можливим в прямому значенні, але залишається дуже корисним інструментом. При аналізу біграмних шифрів важливо аналізувати частоту біграм, а не окремих символів.

#### 2.2.1 Криптоаналіз афінної підстановки біграм

Афінну підстановку біграм за умови пасивної атаки, що базується тільки на знанні шифротексту, можна спробувати ламати brute-force атакою [8]. Проте Всіх біграм в алфавіті обсягу n є $n^2$. Тому простір ключів афінної підстановки біграм становить:

$$|K| = n^2 \cdot \varphi(n^2) \tag{2.4}$$

Якщо n=31, то простір ключів зростає від 930 для звичайної афінної підстановки до 893 730. Тому якщо і перебрати таку кількість варіантів виглядає

можливим, то оцінити змістовність тексту точно буде важко для такої кількості варіантів.

В такому випадку на допомогу знову приходить частотний аналіз. А саме, позначимо через $X^*$ біграму, що найчастіше зустрічається у природній мові, $X^{**}$- наступну за нею щодо частоти і т.д. Позначимо також $Y^*, Y^{**}, \ldots$- 2 найбільш вживані біграми шифротексту. Зробимо припущення, що біграма $X^*$ зашифрована як $Y^*$, а біграма $X^{**}$ - як $Y^{**}$ . Тоді для невідомих параметрів ключа $a, b$ можна записати наступні рівності, що складають систему рівнянь:

$$\begin{aligned} Y^* &= aX^* + b \pmod{n^2} \\ Y^{**} &= aX^{**} + b \pmod{n^2} \end{aligned} \qquad (2.5)$$

Звідси випливає:

$$Y^* - Y^{**} = a(X^* - X^{**}) \pmod{n^2} \qquad (2.6)$$

З даного рівняння нескладно визначити секретний параметр ключа $a$.

$$a = (Y^* - Y^{**})(X^* - X^{**})^{-1} \pmod{n^2} \qquad (2.7)$$

Проте тут необхідно пам'ятати, що $(X^* - X^{**})^{-1}$ – це не просто операція ділення, а обернений елемент до а за модулем $n^2$. Обернений елемент існує, якщо $(X^* - X^{**})$ є взаємнопростим з $n^2$. Отже, якщо $\text{НСД}\big((X^* - X^{**}), n^2\big) = 1$, то рівняння (2.7) має єдиний розв'язок

Якщо $\text{НСД}\big((X^* - X^{**}), n^2\big) = d$ , і якщо $(Y^* - Y^{**})$ не ділиться на $d$, то рівняння (2.7) не має розв'язку. У випадку, якщо $(Y^* - Y^{**})$ ділиться на $d$, то рівняння (2.7) має $d$ розв'язків:

$a_1, a_1+n^2/d, a_1+2n^2/d, \ldots, a_1+(d-1)n^2/d$ , де $a_1$ – розв'язок рівняння

$$a_1^{-1} = \frac{(Y^* - Y^{**})}{d}\left(\frac{X^* - X^{**}}{d}\right)^{-1} \left(\text{mod } \frac{n^2}{d}\right) \qquad (2.8)$$

Якщо з використанням рівняння (2.7) або (2.8) знайдено параметр секретного ключа a, то b з (2.5) можна визначити як

$$b = Y^* - aX^* \pmod{n^2}. \qquad (2.9)$$

Дешифрування при відомих $a$ і $b$ виконується за формулою (1.8).

Припущення про те, що $X^*$ і $X^{**}$ шифруються відповідно в $Y^*$ і $Y^{**}$, може виявитися неправильним. Тоді треба підставити в (2.5) інші пари $X^*$, $X^{**}$ та $Y^*$, $Y^{**}$ з числа найбільш імовірних у природній мові і найчастіших у шифротексті, наприклад, $X^* \leftrightarrow Y^{**}$, $X^{**} \leftrightarrow Y^*$ і т.д. доти, доки при дешифруванні не вийде змістовний текст.

Щоб знайти обернений елемент, можна використати розширений алгоритм Евкліда.

Для того спочатку необхідно виконати звичайний алгоритм Евкліда, який дозволяє знайти НСД двох чисел. Відомо, що для будь яких натуральних чисел a і b (a > b) існує така пара чисел q (неповна частка) і r (остача), така що $a = bq + r$ $(0 \le r < b)$

Тому можна записати

$$a = bq + r1, 0 < r1 < b.$$

Якщо b не ділиться націло на r1, то

$$b = r1q1 + r2, 0 < r2 < r1.$$

Якщо r1 не ділиться r2, то

$$r1 = r2q2 + r3, 0 < r3 < r2$$

В такому випадку можна продовжити систему рівнянь

$$\begin{gathered} a = bq_0 + r_1 \\ b = r_1 q_1 + r_2 \\ \ldots\ldots\ldots\ldots \\ r_{n-2} = r_{n-1} q_{n-1} + r_n \\ r_{n-1} = r_n q_n \end{gathered} \qquad (2.10)$$

Остання ненульова остача $r_n$ є найбільшим спільним дільником чисел $a$ і $b$.

Знайдемо найбільший спільний дільник для чисел 7 і 26. Запишемо алгоритм Евкліда.

26=7·3+5

7=5·1+2

5=2·2+1

2=1·2+0

Розширений алгоритм Евкліда дозволяє знайти коефіцієнти x та y виразу

$$x \cdot a + y \cdot b = \text{НСД}\ (a, b) \tag{2.11}$$

Якщо НСД $(a, b) = 1$, то елемент $y$ є оберненим до b за модулем a. Щоб знайти елемент y можна використати підхідні дроби. Для цього потрібно записати таблицю:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| q | | | $q_0$ | $q_1$ | $q_2$ | … | $q_{n-1}$ |
| p | 0 | 1 | $p_0$ | $p_1$ | $p_2$ | … | $p_{n-1}$ |

і обчислити чисельники підхідних дробів $p$

$$p_i = q_i p_{i-1} + p_{i-2} \tag{2.12}$$

Зробимо це для наших чисел 7 і 26:

| | | | | | |
|---|---|---|---|---|---|
| q | | | 3 | 1 | 2 |
| p | 0 | 1 | 3 | 4 | 11 |

Обернений елемент до b з таблиці обчислюється як

$$y = b^{-1} = (-1)^n p_{n-1} \tag{2.13}$$

Знайдемо обернений елемент до 7 за модулем 26 з таблиці

$$y = (-1)^3 p_3 = -11\ mod\ 26 = 15$$

Виконаємо перевірку, чи правильно знайдено обернений елемент:

$$15 \cdot 7 = 1 \bmod 26$$

Ця рівність задовольняє означення обернених елементів.

### 2.2.2 Криптоаналіз шифру Віженера

Традиційно позначимо – алфавіт ВТ та (ШТ). Вважаємо, що $Z_n$ складається з $n$ букв і $Z_n = \{0, 1, \dots, n-1\}$.

Шифр Віженера є прикладом поліалфавітної підстановки, яка використовувалась ще під час другої світової війни. Ключем цього шифру є послідовність $r$ букв алфавіту $k_1, k_2, \dots, k_r$. Цю послідовність повторюють необхідну кількість разів, щоб отримати довжину відкритого тексту, та підписують під відкритим повідомленням. Число r називається періодом шифру. Як і в попередніх випадках позначимо відкритий текст $M = "m_1 m_2 \dots m_l"$ та через $C = "c_1 c_2 \dots c_l"$ – шифротекст довжини $l$. Шифрування відбувається шляхом додавання букв відкритого повідомлення до підписаних під ними букв ключа за модулем n (обсяг алфавіту), тобто $c_i = m_i + k_{i\ mod r} \bmod n$, $i = 1,\dots,l$. [7].

Криптоаналіз шифру Віженера умовно можна поділити на 2 частини. Перша присвячена визначенню періоду $r$. Для відносно невеликих значень $r$ це можна зробити, обчисливши індекс відповідності:

$$I(C) = \frac{\sum_{i \in Z_n} N_i(C)(N_i(C) - 1)}{n(n-1)} \qquad (2.14)$$

де $N_i(C)$ - кількість появи букви $i$ у ШТ $C$.

Для англійської мови індекс відповідності має орієнтовне значення 0.067, в той час як для випадкового набору букв цей показник дорівнює 0.038.

Причому, для тексту зашифрованого за допомогою моноалфавітною підстановкою, індекс відповідності також дорівнює 0.0667. Це можна пояснити

тим, що частота різних букв в тексті залишається незмінним в результаті моноалфавітної підстановки.

Індекс відповідності використовується для знаходження довжини ключа шифру Віженера. Спочатку робиться припущення про те, що r=2. З шифротексту по черзі вибираються кожна друга літери і для отриманої послідовності обчислюється індекс відповідності. Якщо обчислене значення індексу приблизно відповідає індексу відповідності природної мови, значить довжина ключа дорівнює двом. Якщо ж ні, то з шифротексту вибирається кожна третя буква і знову розраховується індекс збігів. Процес повторюється до тих пір, поки не отримаємо високе значення індексу у відповідності до певної довжини ключа.

Успішність методу криптоаналізу пояснюється тим, що якщо довжина ключа вгадана вірно, то вибрані букви утворюють шифртекст, зашифрований простим шифром Цезаря. І індекс збігів повинен приблизно відповідати індексу збігів природної мови.

Після того як довжина ключа буде знайдена злом зводиться до злому кількох шифрів Цезаря, для яких можна використати частотний. Для цього можна використовувати спосіб, описаний в першому розділі даного топіка.

При більших $r$ можна застосувати наступний метод: потрібно обчислити:

$$D_j = \sum_{i=1}^{n} \delta(c_i, c_{i+j \bmod n}) \quad \text{для } j = 2,\ldots, n \tag{2.15}$$

де $\delta(\mathrm{a}, b)$ – символ Кронекера, що обчислюється за формулою:

$$\delta(\mathrm{a}, b) = \begin{cases} 1, \textit{якщо } a = b \\ 0, \textit{ якщо } a \neq b \end{cases} \tag{2.16}$$

Для зсувів $j$, кратних дійсному значенню періоду $r$, значення $D_j$ буде істотно більшим, ніж для інших зсувів.

При відомому $r$ ШТ поділяють на $r$ фрагментів:

$C_1 = c_1,\ c_{r+1},\ c_{2r+1},\ \ldots$

$C_2 = c_2,\ c_{r+2},\ c_{2r+2},\ \ldots$

……

$C_r = c_r, \quad c_{2r}, \quad c_{3r}, \ldots$

Кожен фрагмент $C_i$ зашифрований шифром Цезаря з ключем $k_i$ , $i = 1,\ldots,r$. Знайти ключ шифру Цезаря можна, вивівши формулу з формули (1.1) $k = c^* - m^* \ mod \ n$, де $c^*$ - буква, що частіше за всіх зустрічається у фрагменті $C_i$, $m^*$ - найімовірніша буква у природній мові, якою написане відкрите повідомлення. При цьому можливі помилки, які можна відносно легко виправити при аналізі розшифрованого тексту. Хоча потрібно визнати, що частотний аналіз в цьому шифрі значно складніший ніж, скажімо, у моноалфавітній підстановці.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ КРИПТОАНАЛІЗУ

## 3.1 Вибір програмного середовища

Для програмної реалізації описаних в розділі 2 методів криптоаналізу, необхідно було насамперед вибрати середовище розробки. За даними Stack Overflow Developer Survey 2020 [9] Rust отримала перше місце як найулюбленіша мова програмування серед опитаних розробників. Тим не менш, більшість розробників, які взяли участь в опитуванні, не знайомі з цією мовою. На рисунку 3.1 наведено відсоток розробників, які програмують на даній мові та виявили зацікавленість продовжувати розвиватися в цій мові.

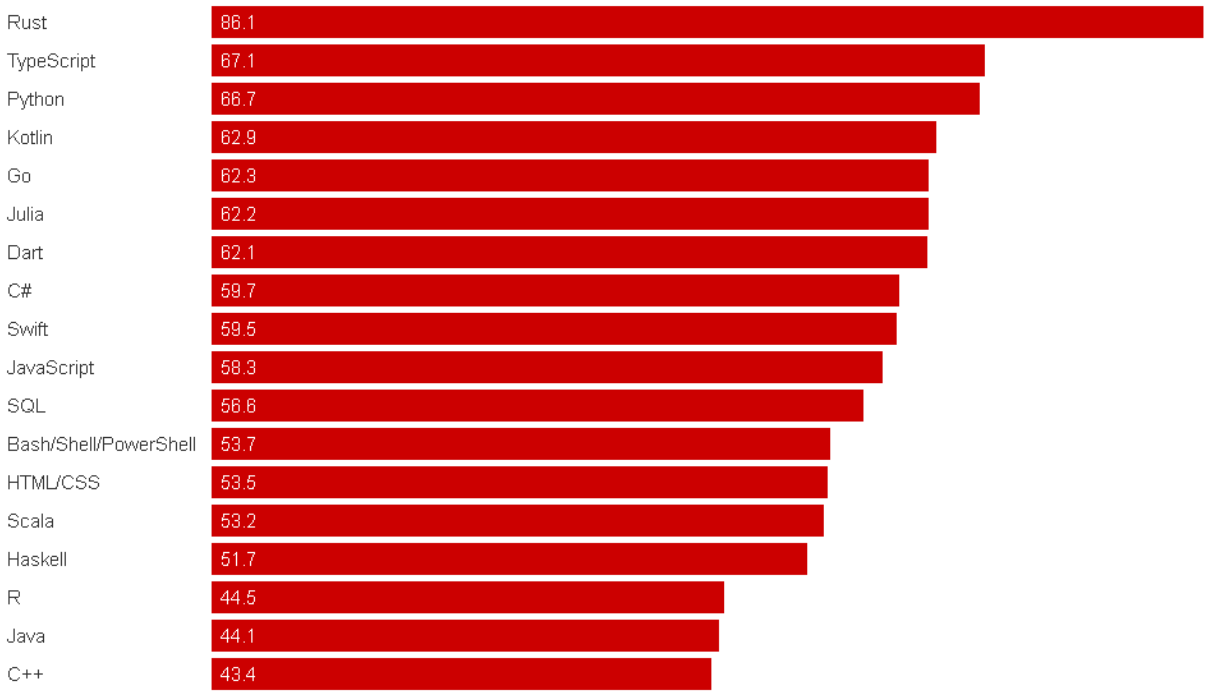


Рисунок 3.1 – Відсоток розробників, які люблять мову програмування, з якою працюють

На рисунку 3.2 приведено список мов програмування, якими розробники найбільше прагнуть користуватися, але ще не розпочали. Цей список кардинально відрізняється від списку найулюбленіших (рис.3.1), які складаються з мов, які

розробники вже використовують під час розробки програмного забезпечення. Як бачимо, Python очолює цей список, а потім йдуть JavaScript і Go

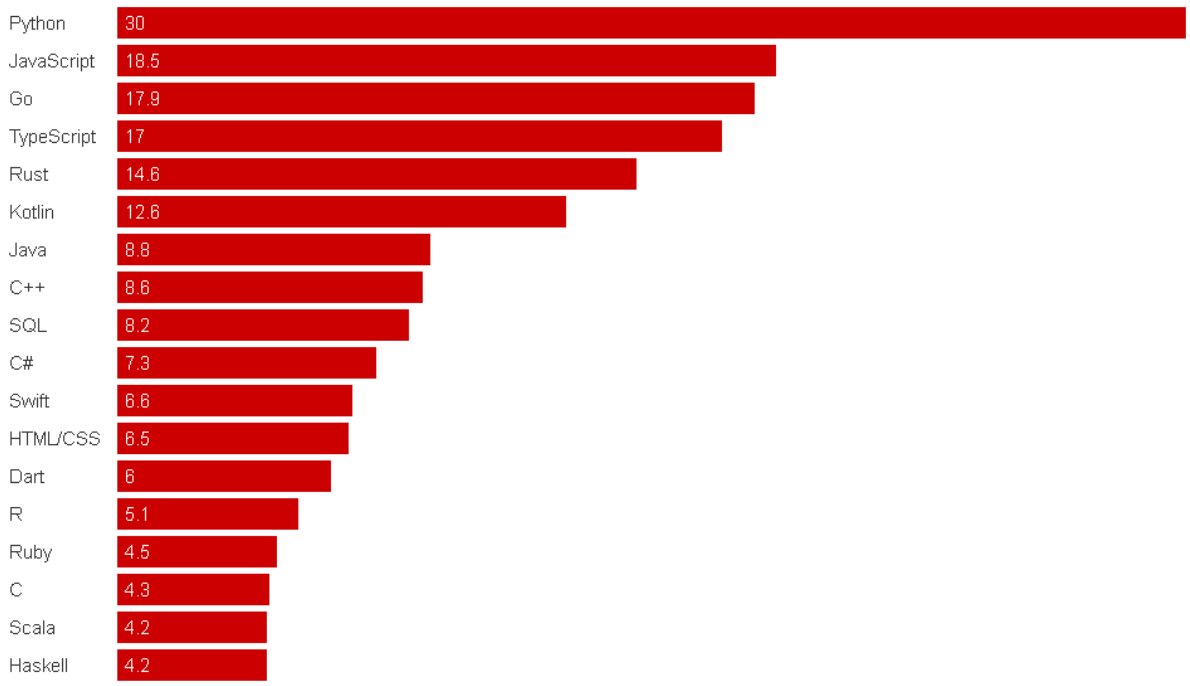


Рисунок 3.2 - Відсоток розробників, які прагнуть вивчити мову програмування і працювати з нею

Тож спробуємо розібратися, чому Python такий популярний і порівняти його з іншими середовищами розробки. Завдяки популярності Python, мабуть завжди, можна знайти готове рішення будь-якої задачі, яка може виникнути. Спільнота ентузіастів Python сильна і вони невтомно працюють над вдосконаленням мови щодня. Python також має ряд корпоративних спонсорів, які намагаються продовжити популяризацію мови. Серед них є такі технічні гіганти, як Google, який сам використовує Python.

Основними перевагами Python є:

- Висока швидкість розробки. Python розроблений, щоб бути доступним. Це робить написання коду Python дуже простим, а розробку програмного забезпечення на Python дуже швидким.
- Численні бібліотеки і фреймворки. Величезною перевагою Python є широкий вибір бібліотек та фреймворків, які він пропонує. Це зменшує час

програмування, адже багато речей можна використати готових, а не кодувати функції вручну.

- Можливість інтеграції з іншими мовами. Python повільніший в порівнянні з деякими іншими мовами програмування. Коли продуктивність набуває пріоритету, Python дає вам можливість інтегрувати інші, ефективніші мови у код.

- Простота обслуговування. Python читається інтуїтивно, оскільки він нагадує справжню англійську. Це робить мову легкою, без потреби докладати зусиль для розшифровки та обслуговування.

Порівняємо Python і JavaScript, яка займає друге місце серед бажаних мов. Хоча, потрібно визнати, що порівняти Python та JavaScript непросто. Вони були розроблені для досягнення різних результатів, тому важко вказати, який із них є найкращим. Часто програмісти використовують декілька мов для досягнення бажаного результату.

Python - чудова мова загального призначення. Завдяки простому синтаксису, він набув популярності не лише серед інженерів програмного забезпечення, а й серед науковців даних та наукових дослідників. JavaScript: чудово підходить для веб-програм та розробки інтерфейсу Все, що асоціюється з JavaScript, - це веб-програми та розробка інтерфейсу. Разом з HTML і CSS ця мова дозволяє розробникам створювати інтерактивні елементи на веб-сайтах.

Основні відмінності між Python та JavaScript:

JavaScript краще працює для побудови інтерфейсу, тоді як Python краще для frontend, в той час як Python використовується для backend та серверної роботи зі сценаріями. JavaScript працює швидше порівняно з Python, але для створення коду Python потрібно менше часу. Python краще використовувати для аналізу даних, машинного навчання або штучного інтелекту, оскільки його легше зрозуміти та підтримувати, ніж JavaScript.

Оскільки дана робота пов'язана з аналізом текстів, то мені зручнішою видалась мова програмування Python.

### 3.2 Програмна реалізація та тестування частотного криптоаналізу

Для проведення частотного криптоаналізу необхідно спочатку створити модуль для підрахунку частот в тексті, який потім можна використати для частотного криптоаналізу

В лістингу 3.1 приведено фрагмент програми, де я для початку імпортував бібліотеку math, оскільки в програмі будуть математичні дії, яких немає в стандартній бібліотеці idle. Після чого створив змінну text, це текст, який ми будемо аналізувати. Після цього створив кілька списків, в яких будуть записуватися елементи, їх частоти та індекси

Лістинг 3.1 – Фрагмент програми для підрахунку частот

```
import math
text = input('Введіть текст: ')
print('№ літера частота')
double = []
text = text
chast = []
chast1 = []
chast2 = []

for i in range(len(text)):

    if text[i] not in double: #and text[i]!= ' ':
        a = float(text.count(text[i])/len(text))
        print(i+1,' ',text[i],' ', a)
        chast.append(a)
```

В список chast записується частота літер в тексті. Для частотного криптоаналізу їх необхідно впорядкувати. В лістингу 3.2 наведено сортування списку.

Лістинг 3.2 – Сортування списку частот букв в тексті

```
print('10 найбільших частот для букв: ')
for max in range(len(a)):
    chast_max.append(a[max])
chast_max.reverse()
[ print(p+1,'-',chast_max[p]) for p in range(len(chast_max)) ]
print()
```

Для злому я взяв зашифрований текст, про який було відомо лише одне, що він зашифрований шифром моноалфавітної підстановки. На рисунку 3.1 наведено скрін програми з введеним текстом

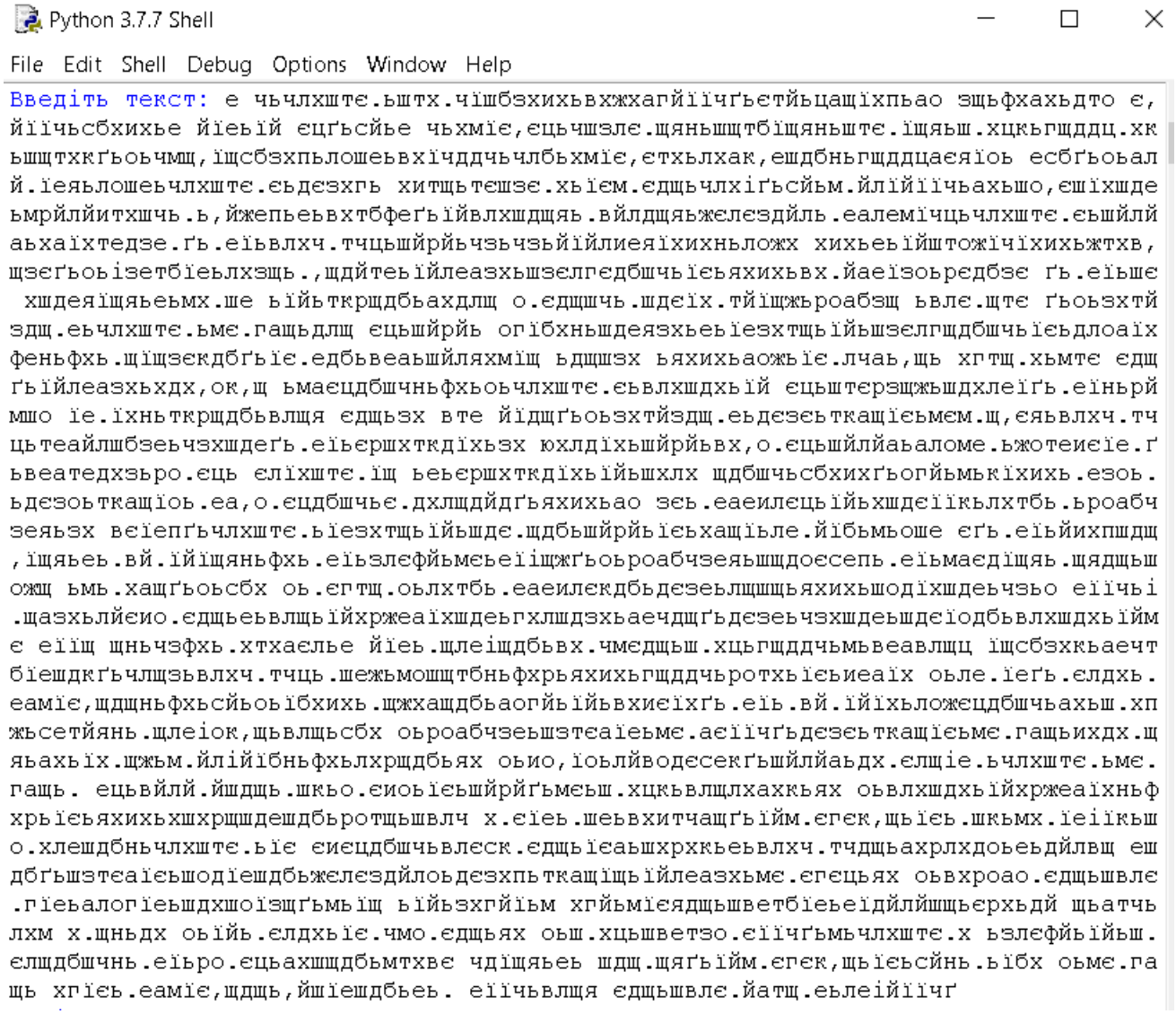


Рисунок 3.1 – Скрін тексту для розшифрування

На рисунку 3.2 наведено результати виконання обрахунку частот цього тексту

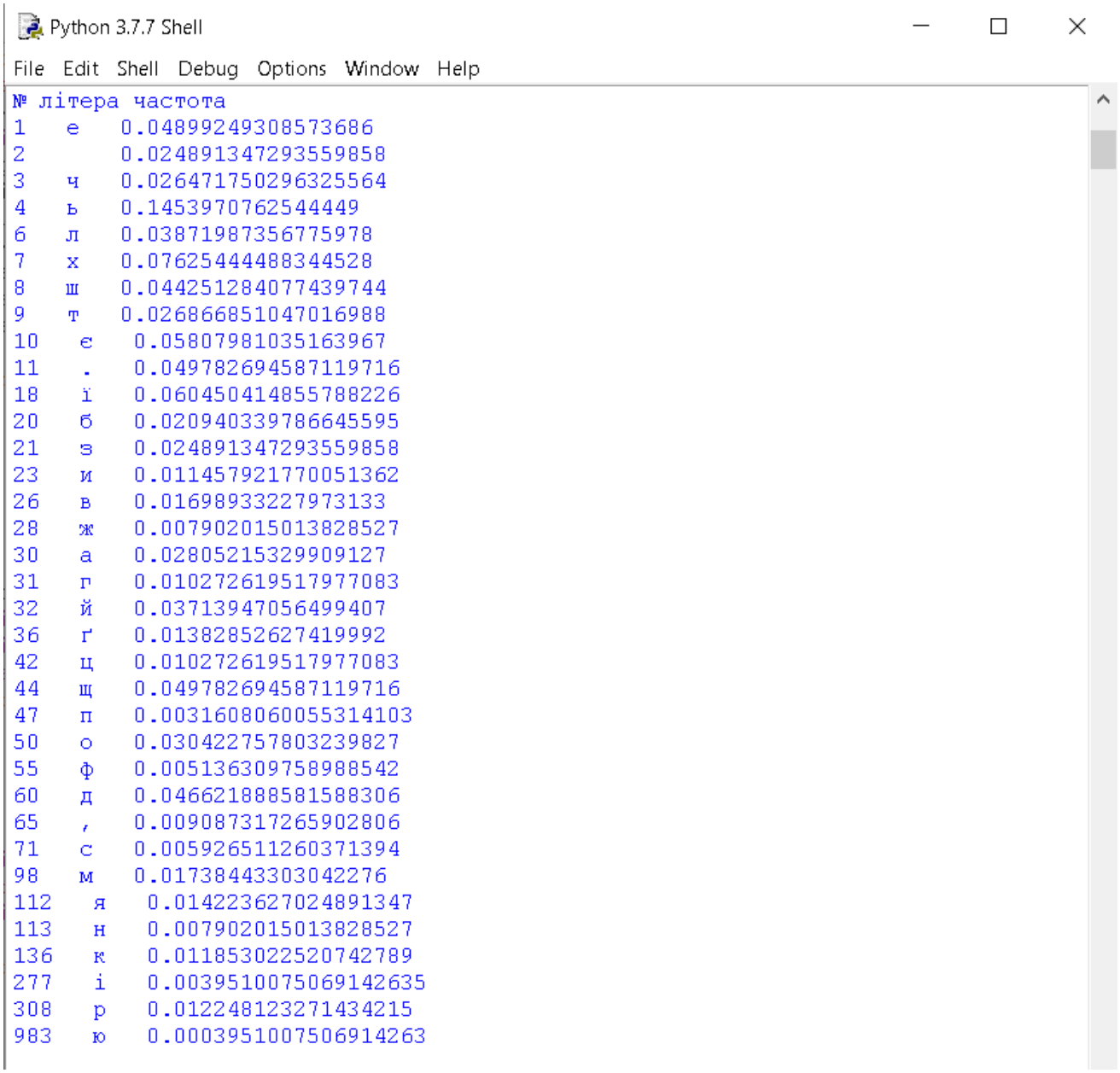


Рисунок 3.2 – Скрін підрахунку частот шифротексту

Частота букви (Ь) була найбільшою, тому я замінив її на пробіл(__). І в самому тексті пробіл замінив на символ ?, щоб був зручніший вигляд тексту і краще бачити де є пробіли.

Згідно програми найчастіші біграми які зустрічались у тексті з двох букв це (ЇЙ) та (ЇЄ), тому я Ї замінив на Н, оскільки Ї подвоюється і зазвичай стоїть у кінці слова. В українській мові доволі часто вживаються слова з закінченням на ННЯ, тому я Ч замінив на Я. Оскільки Ї було замінено і зустрічається біграма НА і НЕ, літера Й замінена на Е а Є на А.

Я замітив що триграма ЄТЙ може бути словосполученням АЛЕ і триграма ЇЄА може бути НАД, тому я замінив Т на Л і А на Д.

Далі переглянувши текс я замітив слово ЛкДщНА і методом підбору букв в мене вийшло слово ЛЮДИНА, тому замінивши К на Ю і Щ на И, де не де почали просвітлюватись частинки слів.

Після цього я зробив ще декілька замін, і почав вимальовуватись якийсь текст. Заміни які я робив, наведені на рисунку 3.3

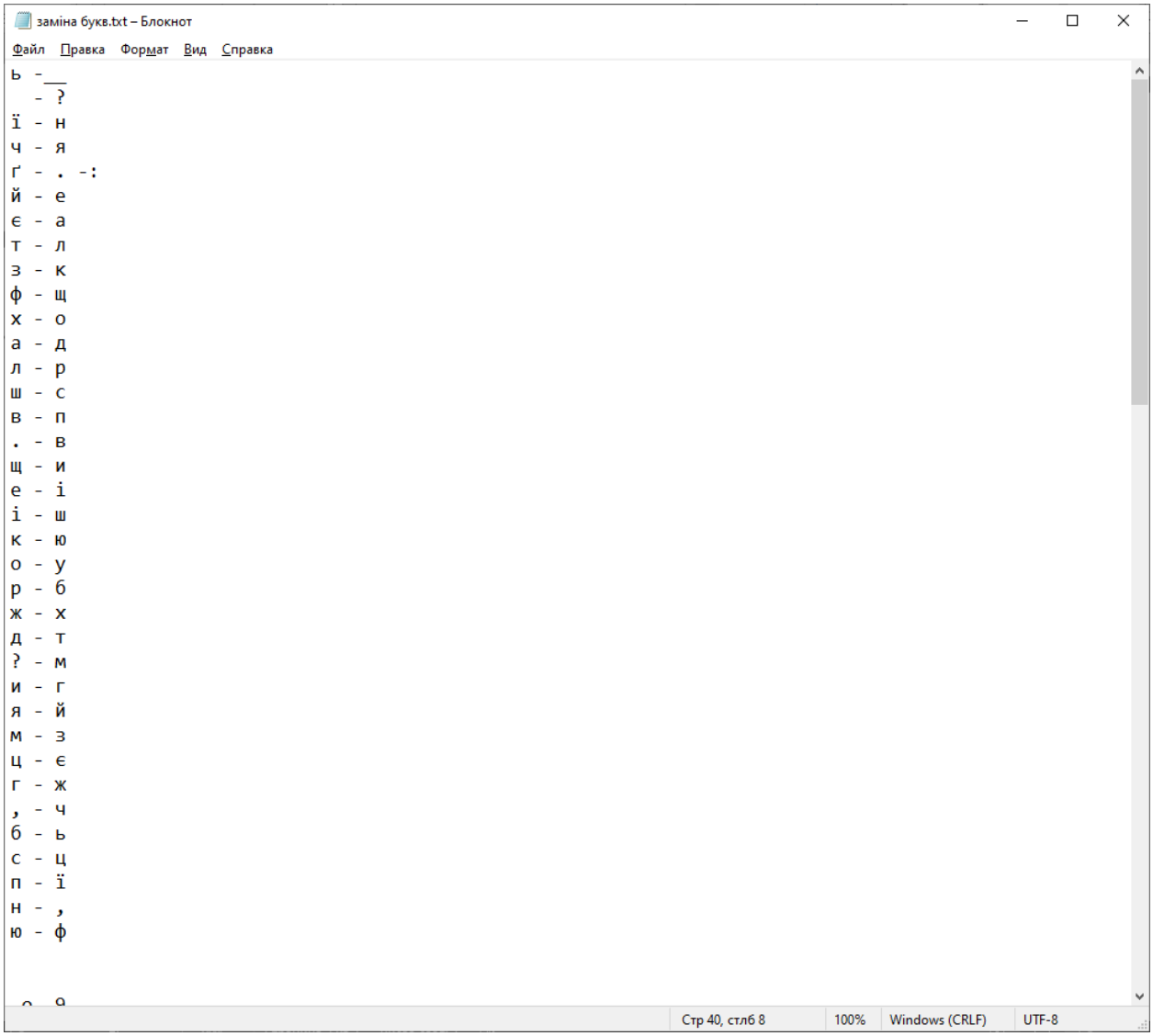


Рисунок 3.3 – Перелік замін виконаних в результаті злому моноалфавітного шифру

Виконавши ці заміни я отримав такий кінцевий результат ( рис.3.4)

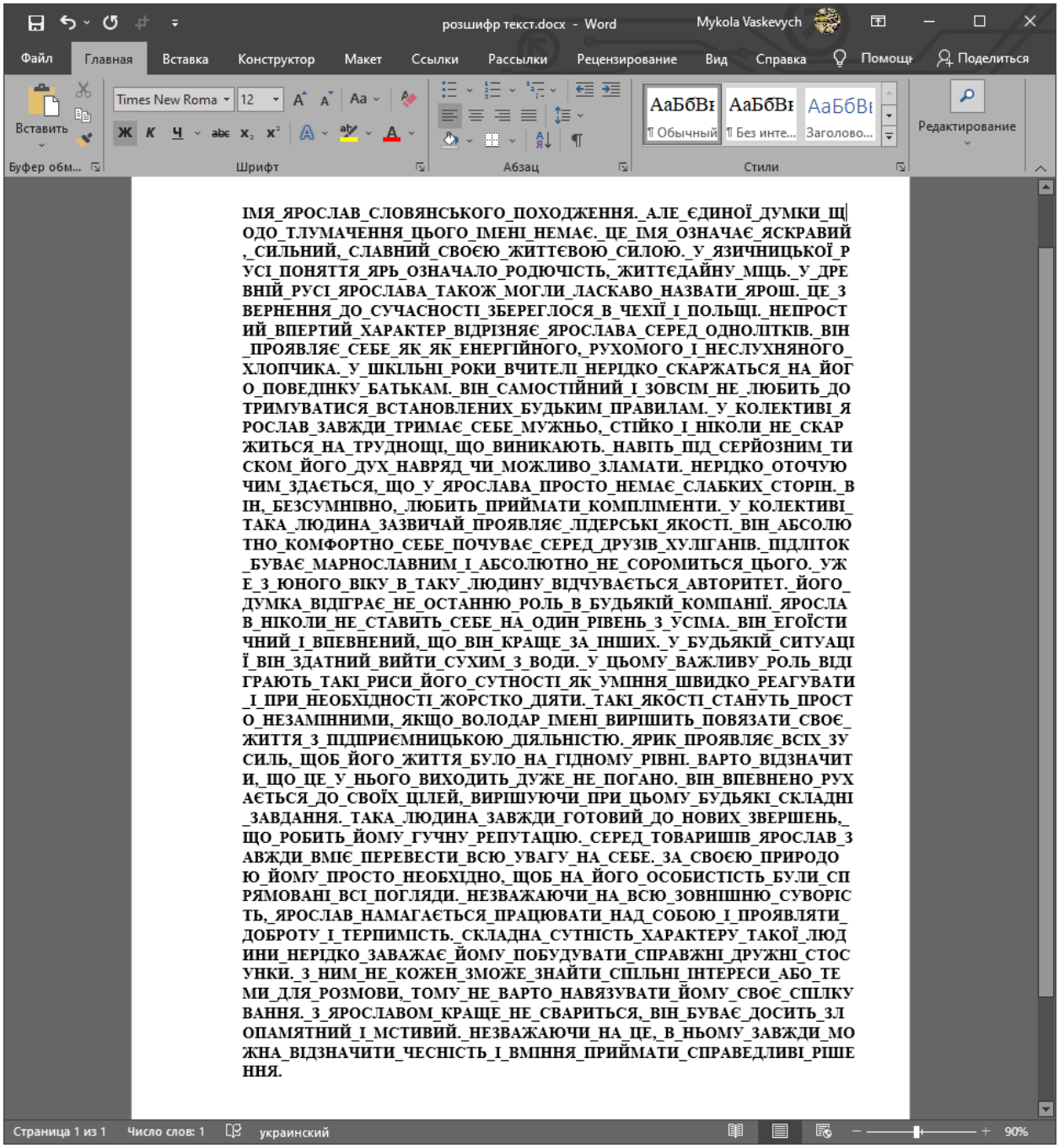


Рисунок 3.4 – Результат розшифрування тексту

Отже, як ми бачимо, у випадку, коли зберігається частота букв природньої мови не складає особливих труднощів зламати текст, зашифрований будь-якою моноалфавітною підстановкою.

### 3.3 Програмна реалізація злому афінної підстановки біграм

Для програмної реалізації афінної підстановки біграм необхідно було насамперед створити модуль для обчислення частоти біграм (Лістинг 3.3)

Лістинг 3.3 – Код для обчислення частоти біграм

```
import codecs
import part3 as frequency

with codecs.open('ciphered_text.txt', 'r+', 'utf-8') as file:
    text = file.readline()
frequencies = frequency.get_bigrams_frequency(list(text))
frequency.print_table(frequencies)
```

Код файлу part3.py, що згадується в лістингу наведено в додатку А.

Для злому був обраний шифротекст, скрін якого наведено на рисунку 3.5.

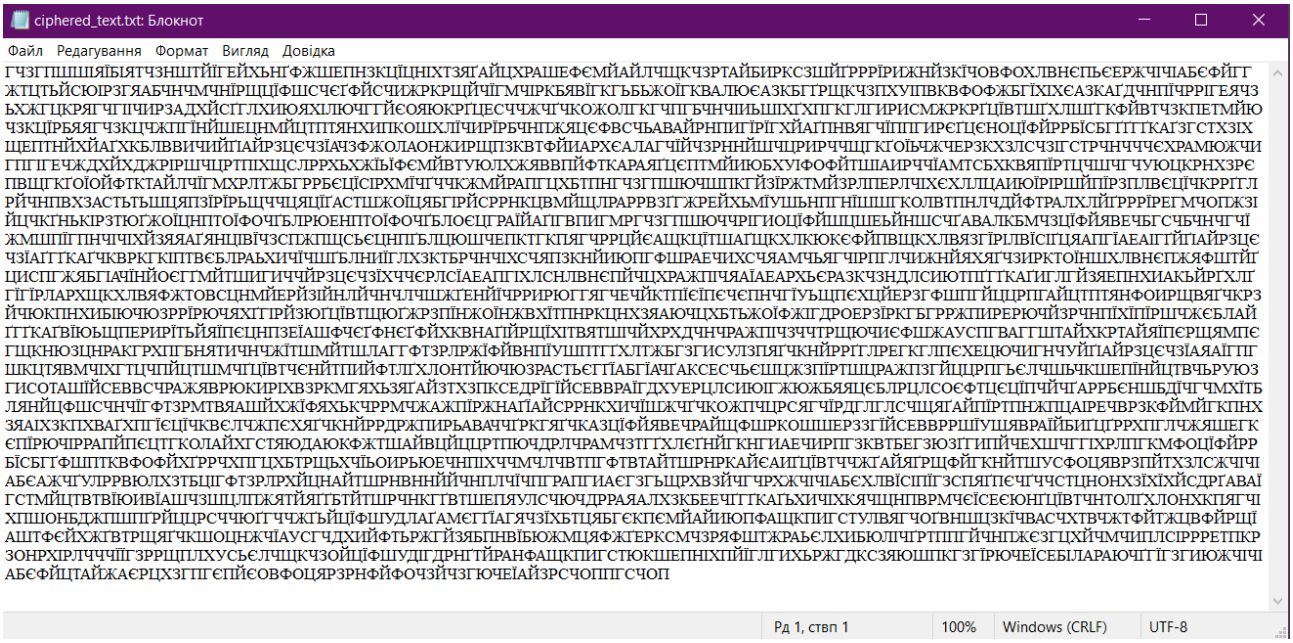


Рисунок 3.5 – Скрін шифротексту для влому.

Результатом виконання підрахунку частот біграм для даного тексту, який виконаний за лістингом 3.3 є результат, наведений на рисунку 3.6


Рисунок 3.6 – Скрін виконання програми для підрахунку частот біграм

Далі згідно з алгоритмом, нам потрібно було зробити припущення про відповідність перших двом біграм шифротексту – біграмам природньої мови і на основі того, обчислити коефіцієнти ключа a та b. Всього таких припущень я зробив 5 і в результаті вдалої підстановки, отримав парметри секретного ключа a=5 та b=3. На рисунку 3.7 наведено скрін виконання програми взлому шифротексту, зображеного на рисунку 3.5

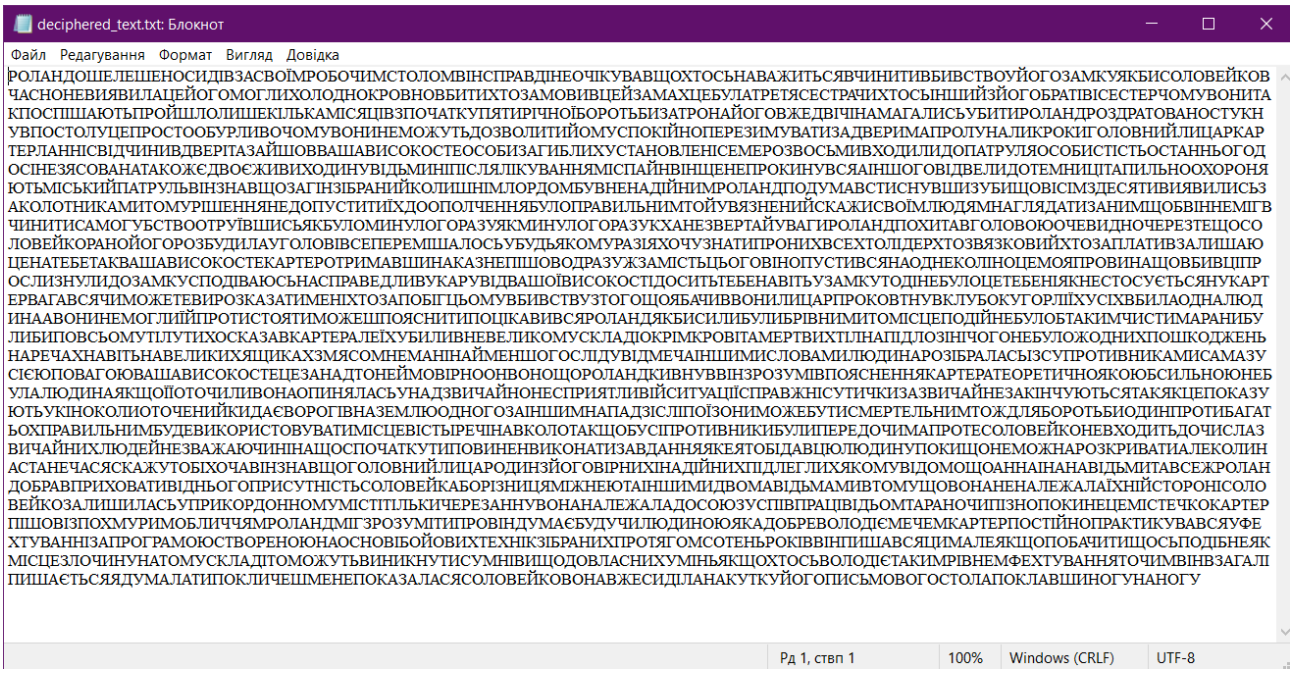


Рисунок 3.7 – Результат злому афінної підстановки біграм

Результат успішного знаходження ключів, наведено на рисунку 3.8


Рисунок 3.8 – Результат знаходження ключів афінної підстановки біграм

Очевидно, що підбір біграм у відповідності до частотного аналізу та криптоаналіз шифру афінної підстановки біграм, описаний в п.2.2.1 є набагато

ефективнішим, ніж звичайний частотний аналіз біграм чи brute-force атака. Лістинг програми для злому шифру наведений в додатку Б.

### 3.4 Програмна реалізація криптоаналізу шифру Віженера

Як описано в пункті 2.2.2 алгоритм криптоаналізу шифру Віженера насамперед вимагає встановлення періоду ключа.

В лістингу 3.4 приведено код для оцінки найбільш ймовірних ключів за формулою 2.15

Лістинг 3.4 – Оцінка ймовірного періоду ключа Віженера

```
import codecs

with codecs.open("ciphered_text5.txt", "r+", "utf-8") as file:
    ciphered_text = file.readline()
    n = len(ciphered_text)

D = []
for j in range(1, 30):
    Di = 0
    for i in range(n):
        Di += int(ciphered_text[i] == ciphered_text[(i + j) %
n])
    D.append([Di, j])
    print (j, Di)

print("Період шифру", [i for j, i in sorted(D, key=lambda x:
x[0])[-1:]]
```

Результат виконання лістингу 3.4 наведено на рисунку 3.9. Як бачимо максимальне значення статистика (2.15) мають два періоди 13 і 26. Оскільки 26 кратне 13, то вважаємо, що знайдений період рівний 13. Повний текст програми для розшифрування наведено в лістингу в додатку В. А на рисунку 3.10 наведено результат дешифрування.

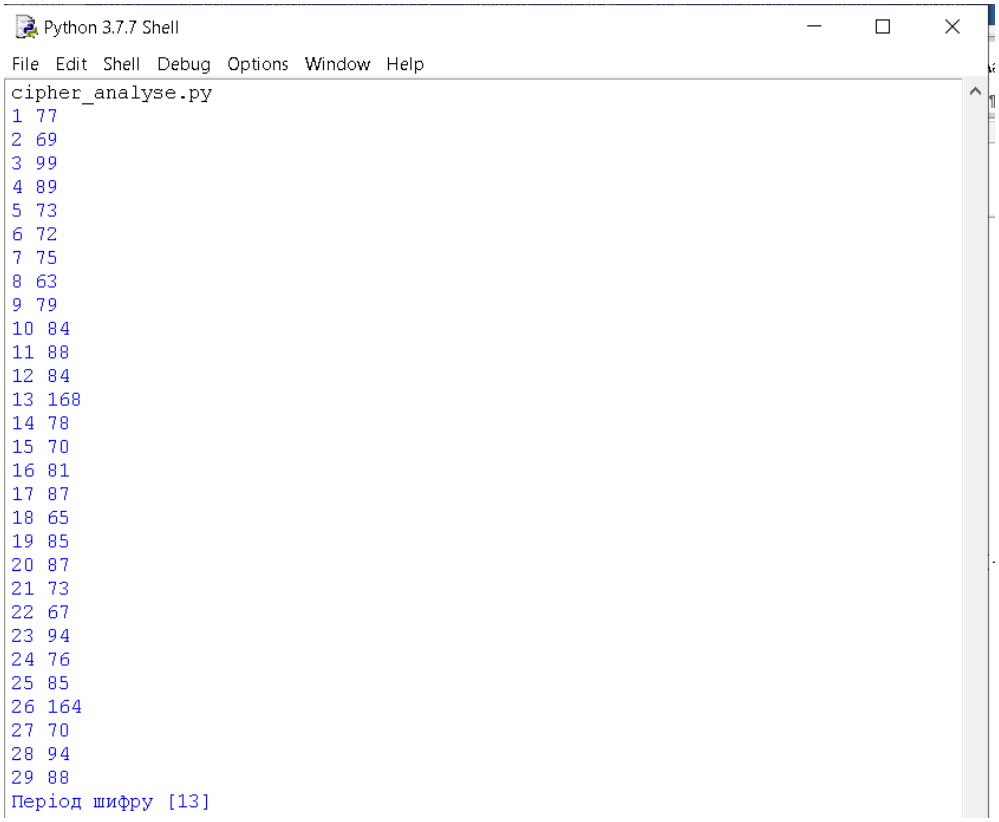


Рисунок 3.9 – Результат оцінки періоду шифру Віженера

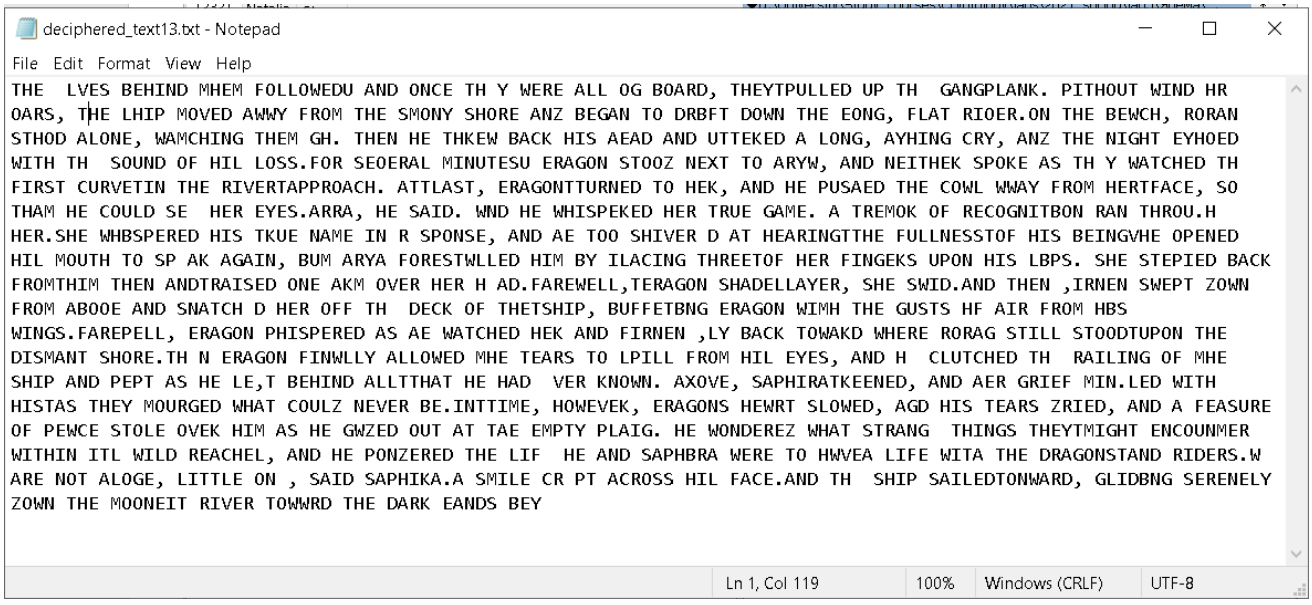


Рисунок 3.10 – Результат криптоаналізу шифру Віженера

Як бачимо автоматизований підбір найкращого зміщення для кожної букви ключа шифру Віженера дав доволі непоганий результат. Текст став читабельним і достатньо невеликих ручних маніпуляцій, щоб привести його до початкового вигляду.

# 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

## 4.1 Гігієнічні вимоги до організації та обладнання робочих місць з ВДТ

Обладнання і організація робочого місця з ВДТ мають забезпечувати відповідність конструкції всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності (ГОСТ 12.2.032-78, ГОСТ 22.269-76, ГОСТ 21.889-76).

Конструкція робочого місця користувача ВДТ має забезпечити підтримання оптимальної робочої пози.

Робочі місця з ВДТ слід так розташовувати відносно світлових прорізів, щоб природне світло падало збоку, переважно зліва.

При розміщенні робочих столів з ВДТ слід дотримуватись таких відстаней: між бічними поверхнями ВДТ - 1,2 м; від тильної поверхні одного ВДТ до екрана іншого - 2,5 м.

Екран ВДТ має розташовуватися на оптимальній відстані від очей користувача, що становить 600...700 мм, але не ближче ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів.

Розташування екрана ВДТ має забезпечувати зручність зорового спостереження у вертикальній площині під кутом +30 до нормальної лінії погляду працюючого.

Клавіатуру слід розташовувати на поверхні столу на відстані 100...300 мм від краю, звернутого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій (виготовлений із матеріалу з високим коефіцієнтом тертя, що перешкоджає мимовільному її зсуву), який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5... 15°.

Для забезпечення захисту і досягнення нормованих рівнів комп'ютерних випромінювань необхідно застосовувати приекранні фільтри, локальні світлофільтри (засоби індивідуального захисту очей) та інші засоби захисту, що пройшли випробування в акредитованих лабораторіях і мають щорічний гігієнічний сертифікат.

При оснащенні робочого місця з ВДТ лазерним принтером параметри лазерного випромінювання повинні відповідати вимогам ДСанПІН 3.3.2.007-98.

### 4.2 Долікарська допомога при ураженні електричним струмом

При ураженні електричним струмом необхідно якомога швидше звільнити потерпілого від струмопровідних частин обладнання.

Дотик до струмопровідних частин (мережі під напругою) у більшості випадків призводить до судом м'язів, тобто людина самостійно не в змозі відірватися від провідника. Тому необхідно швидко відключити ту частину електрообладнання, до якої доторкається людина.

Будь-яке зволікання при наданні допомоги, а також невміння того, хто допомагає, надати кваліфіковану допомогу, призводить до загибелі людини, яка знаходиться під дією струму.

При звільненні потерпілих від струмопровідних частин або проводу в електроустановках напругою до 1000 В відключають струм, використовуючи сухий одяг, палицю, дошку, шапку, сухі рукавиці, рукав одягу, діелектричні рукавиці. Провідники перерізають інструментом з ізольованими ручками, перерубують сокирою з дерев'яним сухим топорищем. Потерпілого можна також відтягнути від струмопровідних частин за одяг, уникаючи дотику до навколишніх металевих предметів та до відкритих частин тіла потерпілого. Відтягуючи потерпілого за ноги, не можна торкатися його взуття, оскільки воно може бути сирим і стає провідником електричного струму. Той, хто надає допомогу, повинен одягнути діелектричні рукавиці або обмотати їх шарфом, натягнути на них рукав піджака або пальта. Можна також ізолювати себе, ставши на гумовий килимок, суху дошку тощо.

При звільненні потерпілих в електроустановках з напругою понад 1000В слід користуватися діелектричними рукавицями і взути діелектричні боти; діяти ізолюючою штангою або ізолюючими кліщами. Якщо є можливість, то вимкнути електроустановку. Можна замкнути або заземлити провідники (замкнути дроти накоротко, накинувши на них попередньо заземлений провід).

Якщо провід торкається землі, то необхідно пам’ятати про небезпеку крокової напруги. Тому після звільнення потерпілого від струмопровідних частин слід винести його з небезпечної зони. Без засобів захисту пересуватися в зоні розтікання струму по землі слід не відриваючи ноги одна від одної.

Кожен працівник, обслуговуючий оперативний персонал повинні знати правила долікарської допомоги, способи штучного дихання і масажу серця.

Долікарську допомогу потерпілому надають на місці нещасного випадку. Констатувати смерть має право тільки лікар.

Способи штучного дихання бувають ручні та апаратні. Ручні менш ефективні, але можуть застосовуватись негайно при порушенні дихання у потерпілого. При виконанні штучного дихання “з рота в рот”, та “з рота в ніс” в рот або в ніс потерплого рятівник видихає зі своїх легенів в легені потерпілого об’єм повітря в кількості 1000-1500 мл. Цей метод найбільш ефективний, однак можлива передача інфекції, тому використовують носовичок, марлю, спеціальну трубку.

Підготовка до штучного дихання.

1. Звільнити потерпілого від одягу – розв’язати галстук, розстебнути комір сорочки тощо.

2. Покласти потерпілого на спину на горизонтальну поверхню – стіл або підлогу.

3. Відвести голову потерпілого максимально назад, доки його підборіддя не стане на одній лінії з шиєю. При цьому положенні язик не затуляє вхід до гортані, вільно пропускає повітря до легенів. Разом з тим при такому положенні голови рот розкривається. Для збереження такого положення голови під лопатки кладуть валик із згорнутого одягу.

4. Пальцями обслідувати порожнину рота і якщо там є кров, слиз тощо, їх необхідно видалити, вийнявши також зубні протези; за допомогою носовичка або краю сорочки вичистити порожнину рота. Обов’язково провести штучне дихання.

Виконання штучного дихання. Голову потерпілого відводять максимально назад і пальцями затискають ніс (або губи). Роблять глибокий вдих, притискають

свої губи до губ потерпілого і швидко роблять глибокий видих йому до рота. Вдування повторюють кілька разів, з частотою 12-15 разів на хвилину. З гігієнічною метою рекомендується рот потерпілого прикрити шматками тканини (носовичок, бинт тощо).

Якщо пошкоджене обличчя проводити штучне дихання "із легенів у легені" неможливо, треба застосувати метод стиснення і розширення грудної клітки шляхом складання і притискання рук потерпілого до грудної клітки з їх наступним розведенням у боки. Контроль за надходженням повітря з легенів потерпілого здійснюється за розширенням грудної клітки при кожному вдуванні. Якщо після вдування грудна клітка потерпілого не розправляється, – це ознака непрохідності шляхів дихання. Найкраща прохідність шляхів дихання забезпечується за наявністю трьох умов:

- максимального відведення голови назад;
- відкривання рота;
- висування вперед нижньої щелепи.

При появі у потерпілого перших слабких вдихів слід поєднати штучний вдих з початком самостійного вдиху. Штучне дихання слід проводити до відновлення глибокого ритмічного дихання. Штучне дихання у більшості випадків треба робити одночасно з масажем серця.

Зовнішній масаж серця. Зовнішній масаж серця – це ритмічне стиснення серця між грудниною та хребтом. Треба знайти розпізнавальну точку – мечоподібний відросток груднини, – він знаходиться знизу грудної клітини над животом. Стати треба з лівого боку від потерпілого і покласти долоню однієї руки на нижню третину груднини, а поверх – долоню другої руки. Тепер ритмічними рухами треба натискати на груднину (з частотою 60 разів на хвилину). Сила стиснення має бути такою, щоб груднина зміщувалась в глибину на 4-5 см. Масаж серця доцільно проводити паралельно зі штучним диханням, для чого після 2-3 штучних вдихів роблять 15 стискань грудної клітки. При правильному масажі серця під час натискання на груднину відчуватиметься легкий поштовх сонної артерії і звузяться протягом кількох секунд зіниці, а також порожевіє шкіра

обличчя і губи, з’являться самостійні вдихи. Щоб не пропустити повторного припинення дихання, треба стежити за зіницями, кольором шкіри і диханням, регулярно перевіряти частоту і ритмічність пульсу .

Наслідки своєчасної і правильно наданої допомоги на місці події можуть бути зведені нанівець, якщо при підготовці до транспортування і доставці потерпілого до медичної установи не будуть дотримані відповідні правила. Головне не тільки в тому, як доставити потерпілого і яким видом транспорту, а наскільки швидко були вжиті заходи, які забезпечили максимальний спокій і зручне положення потерпілого.

Найкраще транспортувати потерпілого ношами. При цьому можна використовувати підручні засоби: дошки, одяг тощо. Можна переносити потерпілого на руках. Передусім потерпілого слід покласти на ноші, які застеляють ковдрою, одягом тощо, ставлять ноші з того боку потерпілого, де є ушкодження. Якщо тих, хто надає допомогу, двоє, вони повинні стати з двох боків ош. Один підкладає руки під голову і груднину, другий – під крижі і коліна потерпілого. Одночасно без поштовхів його обережно піднімають, підтримуючи ушкоджену частину тіла, і опускають на ноші. Слід накрити потерпілого тим, що є під руками, – одягом, ковдрою. Якщо є підозра на перелом хребта, потерпілого кладуть обличчям догори на тверді ноші (щит, двері). За відсутністю такого можна використати ковдру, пальто. В такому випадку потерпілого кладуть на живіт.

Якщо є підозра на перелом кісток тазу, потерпілого кладуть на спину із зігнутими ногами у колінах і у тазостегнових суглобах для того, щоб його стегна були розведені, під коліна обов’язково треба підкласти валик із вати, рушника, сорочки.

По рівній поверхні потерпілого несуть ногами вперед, при підйомі на гору або на сходах – головою вперед. Ноші весь час повинні бути у горизонтальному положенні. Щоб ноші не розгойдувались, необхідно йти не в ногу, злегка зігнувши коліна.

При перевезенні потерпілого слід покласти його до машини на тих самих ношах, підстеливши під них що-небудь м’яке (ковдру, солому тощо).

# ВИСНОВКИ

В процесі виконання кваліфікаційної роботи було проведено літературний аналіз джерел в області криптоаналізу історичних шифрів заміни, а саме розглянуто загалом проблему несанкціонованого шифрування інформації. Встановлено, що одним з популярних методів отримання грошей є застосування ransomware – програмного забезпечення, що шифрує дані. Безумовно, сучасні шифри є практично стійкими, що означає, що обчислювальних потужностей світу недостатньо для їх злому за прийнятний час. Це ще раз підтверджує кількість втрат від вимагачів грошей через ransomware. В результаті виконання даної роботи було розроблено програмне забезпечення для злому історичних шифрів заміни, яке може бути використане для навчальних завдань і для розуміння труднощів криптоаналізу простих, на перший погляд, шифрів.

В результаті виконання кваліфікаційної роботи виконано наступні завдання:

- Проведено огляд літературних джерел в предметній області.
- Наведено короткі відомості про основні історичні шифри заміни.
- Проаналізовано відомі криптоаналізу шифрів.
- Обґрунтовано середовище розробки.
- Розроблено та протестовано програмне забезпечення.
- Проведено практичний криптоаналіз кількох зашифрованих текстів.

Крім того в роботі проведено роботу над розділом "Безпека життєдіяльності, основи хорони праці"

.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Жельников Владимир "Криптография от папируса до компьютера" -М., ABF, 1996. – С.336, ISBN: 5-87484-054-0
2. Бабаш А.В., Шанкин Г.П. Криптография. Москва, СОЛОН-Р, 2002, 511 с.
3. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушки А.В. Основи криптографии. Москва, Гелиос, 2002, 480 с.
4. Сушко С.О., Фомичова Л.Я., Барсуков Є.С. Частоти повторюваності букв і біграм у відкритих текстах українською мовою. Захист інформації. Київ, 2010. Т. 12, № 3. С. 94-102 DOI: https://doi.org/10.18372/2410-7840.12.1968
5. ROBERT L. SOLSO, JOSEPH F. KING Frequency and versatility of letters in the English language Behavior Research Methods & Instrumentation /976. Vo/. 8 (3), P. 283-286
6. Перебийніс В.І., Муравицька М.П., Дарчук Н.П. Частотні словники та їх використання. К.: Наукова думка, 1983.
7. Шнайер Б. Криптоанализ // Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си = Applied Cryptography. Protocols, Algorithms and Source Code in C. — М.: Триумф, 2002. — С. 19—22. — 816 с.
8. Ел Свейгарт Криптография и взлом шифров на Python – К: Диалектика, 2019, ISBN 978-617-7812-84-4
9. 2020 Developer Survey [Електронний ресурс] // Stackoverflow. – 2021. – Режим доступу до ресурсу: https://insights.stackoverflow.com/survey/2020.
10. Основи охорони праці: Підруч для студ вищих навч закладів За ред мп Гандзюка - К Каравела, 2004 - 408 с.
11. Охорона праці в галузі комп'ютингу: підручник / Л. А. Катренко, А. В. Катренко ; [за наук. ред. В. В. Пасічника] ; М-во освіти і науки, молоді та спорту України. — Л. : Магнолія 2006, 2012. — 544 с

# ДОДАТКИ

Лістинг файлу part3.py (для підрахунку частоти біграм)

```
import math


def get_bigrams_frequency(chars):
    bigr_freq = {}
    for i in range(0, len(chars), 2):
        bigram = (chars[i], chars[i + 1])
        if bigram not in bigr_freq:
            bigr_freq[bigram] = 0
        bigr_freq[bigram] += 1
    for key, val in bigr_freq.items():
        chance = round(val / len(chars), 4)
        bigr_freq[key] = (chance, round(-chance * 
math.log2(chance) / len(key), 4))
    return bigr_freq


def print_table(t):
    l = sorted(t.items(), key=lambda x: x[1], 
reverse=True)[:10]
    for i in range(len(l)):
        if type(l[i][0]) == type(tuple()):
            print("-- ", end="")
            for j in range(len(l[i][0])):
                print(f"'{l[i][0][j]}'{',' if not j == 
len(l[i][0]) - 1 else ''} ", end="")
        else:
            print(f"-- '{l[i][0]}' ", end="")
        for j in range(1, len(l[i])):
            print("--", l[i][j][0], "--")
    print()
```

Лістинг Б1 - Знаходження параметрів шифру афінної підстановки біграм

```
import codecs
import lab1 as e

with codecs.open('alphabet.txt', 'r+', 'utf-8') as file:
    M = {k: v for v, k in enumerate(file.readline())}
    M_reversed = {v: k for k, v in M.items()}
    m = len(M)
    m2 = m ** 2

with codecs.open('ciphered_text.txt', 'r+', 'utf-8') as file:
    text = file.readline()

X = (('Н', 'А'), ('Н', 'И'))
Y = (('П', 'Г'), ('Р', 'Р'))

Xn = (M[X[0][0]] * m + M[X[0][1]], M[X[1][0]] * m + M[X[1][1]])
Yn = (M[Y[0][0]] * m + M[Y[0][1]], M[Y[1][0]] * m + M[Y[1][1]])


d = e.evklid(Xn[0] - Xn[1], m2)
print(f'GCD(X1-X2, m^2) = {d}')
# d == 1 - only one solution
if d == 1:
    print(f'Equation has one solution.')
    X1_X2_revers = e.evklid_extended_x_y((Xn[0] - Xn[1]), m2)
    a = X1_X2_revers[0] * (Yn[0] - Yn[1]) % m2
    b = (Yn[0] - a * Xn[0]) % m2
    print(f'Athens key: a = {a}, b = {b}')

    decipher_dictionary = {}
    deciphered_text = ""
    for i in range(0, len(text), 2):
        current_bigram = (text[i], text[i + 1])
        # if bigram is not in dictionary, decipher it and add
        if current_bigram not in decipher_dictionary:
            deciph_number    =    M[current_bigram[0]]    *    m    +
M[current_bigram[1]]
            deciph_number     =     ((deciph_number     -     b)     *
e.evklid_extended_x_y(a, m2)[0]) % m2
            index1 = deciph_number // m
            index2 = deciph_number - index1 * m
            decipher_dictionary[current_bigram]                    =
(M_reversed[index1], M_reversed[index2])
        deciphered_bigram = decipher_dictionary[current_bigram]
        deciphered_text       +=       deciphered_bigram[0]       +
deciphered_bigram[1]

    with codecs.open('deciphered_text.txt', 'w+', 'utf-8') as file:
        file.write(deciphered_text)
elif (Yn[0] - Yn[1]) % d == 0:
    print(f'Equation has many solutions, will check all:')
```

```
    d_range = [i for i in (range(d) if d > 0 else range(0, d, -1))]
    d0 = 0

    while not input('Empty to continue:') and d0 < len(d_range):
        X1_X2_rev = e.evklid_extended_x_y((Xn[0] - Xn[1]) / d, m2 /
d)
        a =- (Yn[0] - Yn[1]) / d * X1_X2_rev[0] % (m2 / d) +
d_range[d0] * m2 / d
        b = (Yn[0] - a * Xn[0]) % m2
        print(f'Athens key: a = {a}, b = {b}')

        decipher_dictionary = {}
        deciphered_text = ""
        for i in range(0, len(text), 2):
            current_bigram = (text[i], text[i + 1])
            # if bigram is not in dictionary, decipher it and add
            if current_bigram not in decipher_dictionary:
                deciph_number   =   M[current_bigram[0]]   *   m   +
M[current_bigram[1]]
                deciph_number    =    ((deciph_number    -    b)    *
e.evklid_extended_x_y(a, m2)[0]) % m2
                index1 = deciph_number // m
                index2 = deciph_number - index1 * m
                decipher_dictionary[current_bigram]                =
(M_reversed[index1], M_reversed[index2])
            deciphered_bigram = decipher_dictionary[current_bigram]
            deciphered_text      +=      deciphered_bigram[0]      +
deciphered_bigram[1]
        print(f'Deciphered text:\n{deciphered_text}')

        d0 += 1
        with  codecs.open('deciphered_text.txt',  'w+',  'utf-8')  as
file:
            file.write(deciphered_text)
else:
    print('There are no solutions to this equation.')
```

Лістинг В1 - Оцінка періоду

```
    import codecs

    with codecs.open("ciphered_text5.txt", "r+", "utf-8") as file:
        ciphered_text = file.readline()
        n = len(ciphered_text)

    D = []
    for j in range(1, 30):
        Di = 0
        for i in range(n):
            Di += int(ciphered_text[i] == ciphered_text[(i + j) %
n])
        D.append([Di, j])
        print (j, Di)

    print("Період шифру", [i for j, i in sorted(D, key=lambda x:
x[0])[-1:]])
```

Лістинг В2 – Оцінка індексу відповідності тексту

```
import codecs
from text_frequency import *

for i in range(1, 6):
    with codecs.open(f"ciphered_text{i}.txt", "r+", "utf-8") as file:
        ciphered_text = file.readline()
        n = len(ciphered_text)

    frequencies = get_letters_frequency(ciphered_text)
    #print (frequencies)
    I = sum(Nt * (Nt - 1) for Nt in frequencies.values()) / (n * (n
- 1))

    print(f"I(CT{i}) = {I}")

with codecs.open("text_to_cipher.txt", "r+", "utf-8") as file:
    ciphered_text = file.readline()
    n = len(ciphered_text)

frequencies = get_letters_frequency(ciphered_text)

I = sum(Nt * (Nt - 1) for Nt in frequencies.values()) / (n * (n - 1))

print(f"I(OT) = {I}")
```