Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №3
Криптоаналіз афінної біграмної підстановки

Виконали:
ФБ-31 Аль-Фітурі Асія,
ФБ-31 Гриб Вероніка

2025

Варіант 5

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

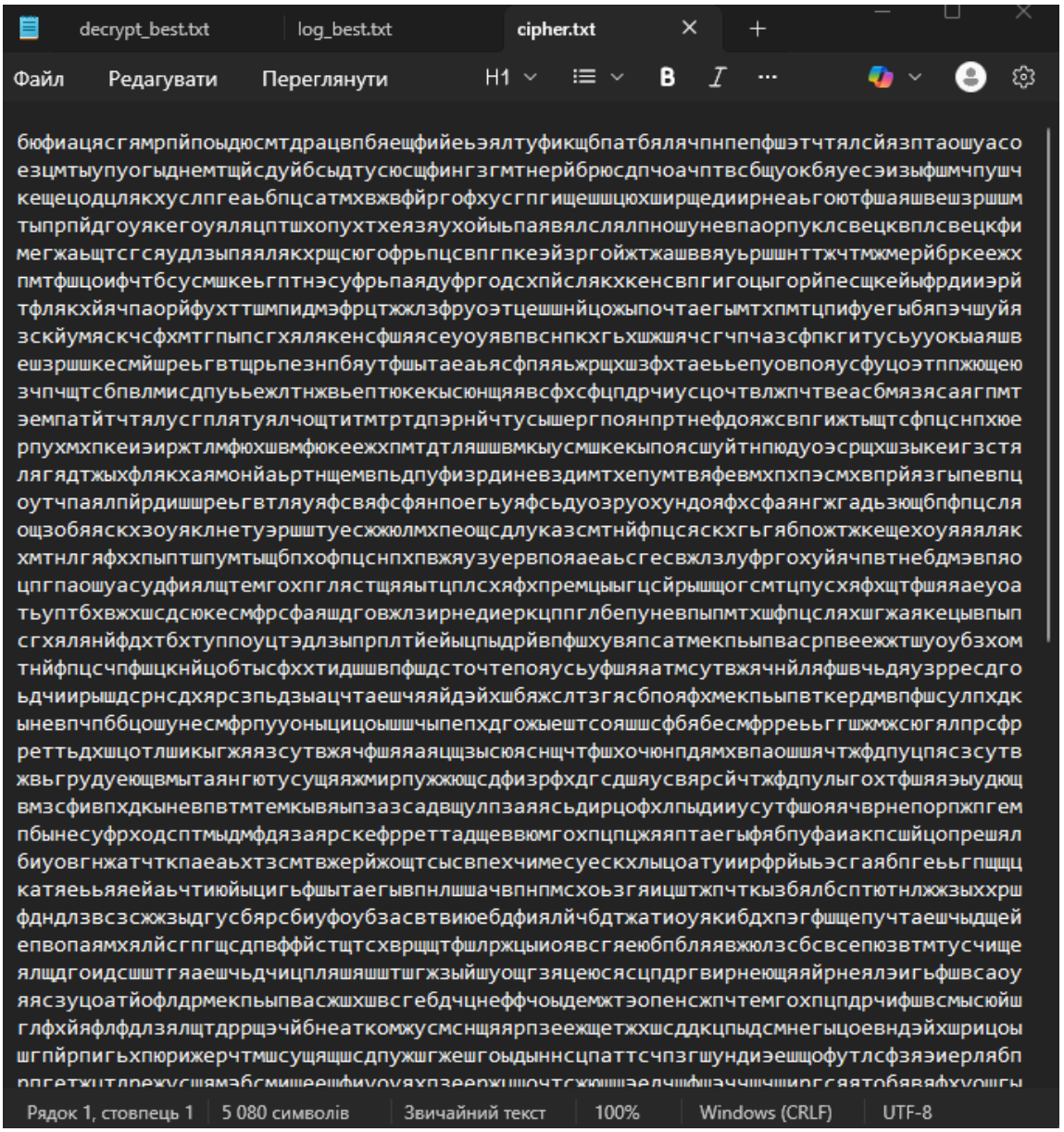
Порядок виконання роботи

0. Уважно прочитати методичні вказівки до виконання комп'ютерного практикуму.

У цьому завданні нам надається зашифрований текст, отриманий у результаті афінної підстановки біграм відкритого тексту, написаного російською мовою без пробілів, розділових знаків та великих літер. При цьому літера «ё» замінена на «е», та «ъ» – на «ь» (або навпаки). Алфавіт відкритого тексту складається з 31 літери, яким присвоєно номери в алфавітному порядку: а = 0, б = 1, …, я = 30.
Під час дешифрування необхідно враховувати, що найчастішими біграмами російської мови є: «ст», «но», «то», «на», «ен».

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.

Маємо такий зашифрований текст за Варіантом 5:



Функція extended_gcd(n, x) — це розширений алгоритм Евкліда.

```python
def extended_gcd(a: int, b: int):
    if b == 0:
        return (abs(a), 1 if a >= 0 else -1, 0)
    d, x1, y1 = extended_gcd(b, a % b)
    return d, y1, x1 - (a // b) * y1
```

Вона знаходить найбільший спільний дільник (НСД) двох чисел n і x, а також коефіцієнти, які задовольняють рівняння виду a·n + b·x = НСД(n, x). Тобто функція не лише обчислює сам НСД, а й показує, як саме його можна отримати через лінійну комбінацію двох чисел.

Функція mod_inverse(n) — шукає обернений елемент за модулем.

```python
def mod_inverse(a: int, m: int):
    d, x, _ = extended_gcd(a % m, m)
    return (x % m) if d == 1 else None
```

Вона використовується, щоб знаходити число, яке при множенні на n дає 1 (за певним модулем).
Така операція потрібна для розв’язування лінійних порівнянь у модульній арифметиці.

2. За допомогою програми обчислення частот біграм, яка написана в ході виконання комп’ютерного практикуму №1, знайти 5 найчастіших біграм запропонованого шифртексту (за варіантом).

```
Топ 5 біграм у файлі cipher.txt:
фш — 51 разів
вп — 44 разів
не — 39 разів
ус — 33 разів
яя — 31 разів
```

Функція для цього підраховує, скільки разів кожна біграма зустрічається в тексті, і виводить п’ять найбільш частих. Далі порівнюємо ці біграми з найчастішими в російській мові — «ст», «но», «то», «на», «ен». Це дозволяє нам робити припущення, яким біграм у шифротексті можуть відповідати які природні біграми.

```python
def top_bigrams_cipher(path: str, top_n: int = 5):
    with open(path, "r", encoding="utf-8") as f:
        raw = f.read()
    cnt = Counter(get_bigrams(raw))
    top = cnt.most_common(top_n)
    print(f"\nТоп {top_n} біграм у файлі {path}:")
    for bg, k in top:
        print(f"{bg} — {k} разів")
    return [bg for bg, _ in top]
```

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) , (ba шляхом розв’язання системи (1).

Кандидати ключа (за a,b):

№	a	b	X*	X**	Y*	Y**
1	13	344	но	ен	яя	ус
2	16	535	ст	то	ус	вп
3	21	164	ст	то	вп	фш
4	33	299	но	ен	ус	вп
5	37	622	ст	то	ус	фш
6	42	780	ст	на	ус	не
7	45	532	но	на	вп	не
8	46	37	но	ен	яя	вп
9	53	389	на	ен	ус	фш
10	55	15	на	ен	вп	ус
11	72	805	ст	но	фш	вп
12	80	13	то	ен	ус	яя
13	80	683	ст	но	вп	не
14	87	207	то	ен	яя	не
15	99	148	то	на	вп	фш

На цьому етапі програма перебирає всі можливі відповідності між найчастішими біграмами російської мови та шифротексту. Для кожної пари біграм обчислюються можливі значення ключів a та b за системою рівнянь афінного шифру. У результаті формується таблиця кандидатів на ключ, де для кожного варіанта наведено значення a, b та відповідні біграми відкритого і шифрованого текстів. Ці дані надалі використовуються для спроб дешифрування шифротексту. Та повну таблицю кандидатів (176) збережено в candidates_full.csv

4. Для кожного кандидата на ключ дешифрувати шифртекст. Якщо шифртекст не є змістовним текстом російською мовою, відкинути цього кандидата.

```
Перевірка кандидатів на ключ (a, b):
№ |    a |    b | інверсія |   score   | verdict | preview
--+------+------+----------+-----------+---------+-----------------------------
 1 |   72 |  805 |   yes    |      3.06 | OK      | агодышлидашлибыстроинеслышнока
 2 |  152 |  450 |   yes    |      1.01 | skip    | ожйнхюлкйрлаьчэлндбкчфшахюэхфр
 3 |  676 |  287 |   yes    |      1.51 | skip    | умшапижолсьймепяблбоцутйпиыдбс
 4 |  448 |  578 |   yes    |      1.08 | skip    | дщоирйбваоыншэфдбхмвяцхнрйщшцо
 5 |  889 |  877 |   yes    |      0.90 | skip    | якрэдиушьбжхчадпмрршсьнхдистхб
 6 |   80 |  683 |   yes    |      1.04 | skip    | шехйзьхщяшхжтэрямывщфнщжзьстчш
 7 |  604 |  520 |   yes    |      1.17 | skip    | лхббвдцоцумрщязфцйшолмэрвдстыу
 8 |  376 |  811 |   yes    |      1.55 | skip    | яжлшьмчгоогывбюкачпгвлбыьмстюо
 9 |  809 |  602 |   yes    |      0.93 | skip    | рыхуйгуцхрфбгйвхсэюцзмзбйгвлкр
10 |  881 |  763 |   yes    |      1.46 | skip    | жьйччейзаийымдовтжэзкужыченози
11 |  524 |  245 |   yes    |      0.45 | skip    | скйсшвяохфхдеьфпиифофшгдшвющбф
12 |  296 |  536 |   yes    |      1.15 | skip    | мгйфчууыыоппйабшнзиыажфпчужддо
13 |  285 |    2 |   yes    |      1.48 | skip    | лфзбпшштупгчтьпвюхютинмчпшдэюп
14 |  357 |  163 |   yes    |      1.03 | skip    | уляаээитинтревчмичжтуфврээнодн
15 |  437 |  769 |   yes    |      1.58 | skip    | нцхпжяатймйэщексцшкткиьэжябзюм
```

На цьому етапі здійснюється дешифрування шифротексту для кожного знайденого кандидата на ключ (a,b). Функція decrypt_affine_bigrams()

```python
def decrypt_affine_bigrams(cipher: str, a: int, b: int) -> str | None:
    inv_a = mod_inverse(a, M)
    if inv_a is None: return None
    t = clean_text(cipher)
    if len(t) % 2: t = t[:-1]
    out = []
    for i in range(0, len(t), 2):
        X = bg2num(t[i:i+2])
        Y = (inv_a * (X - b)) % M
        out.append(num2bg(Y))
    return "".join(out)
```

виконує зворотне перетворення афінного шифру біграм,

a is_meaningful_text()

```python
def is_meaningful_text(text: str) -> bool:
    t = clean_text(text)
    if len(t) < 200: return False
    freq = Counter(t); total = sum(freq.values())
    if total == 0: return False
    if (freq.get("о",0)+freq.get("а",0)+freq.get("е",0))/total < 0.20: return False
    if (freq.get("ф",0)+freq.get("щ",0)+freq.get("ц",0))/total > 0.05: return False
    bad = ("йй","ьь","ыы","ээ","жы","шы","йы","йь","ьй")
    if any(p in t for p in bad): return False
    bigs = Counter(get_bigrams(t))
    if sum(bigs[b] for b in ("ст","но","то","на","ен")) < 10: return False
    vowels = set("аеёиоуыэюя".replace("ё","е"))
    v = sum(1 for ch in t if ch in vowels)
    c = sum(1 for ch in t if ch in ALPH and ch not in vowels)
    r = v/(v+c) if (v+c) else 0
    return 0.30 <= r <= 0.65
```

автоматично перевіряє, чи є отриманий текст осмисленим російським текстом. Перевірка базується на частоті появи найуживаніших і рідкісних літер, співвідношенні голосних і приголосних, а також на наявності типових біграм. Це дозволяє автоматично відсіяти неправильні ключі без ручної перевірки. У результаті залишається лише найімовірніший або найкращий варіант розшифрованого тексту.

Побудова автомату розпізнавання російської мови

Щоб визначити, чи текст схожий на російський, програма аналізує частоти літер:

перевіряє, наскільки часто зустрічаються поширені літери («о», «а», «е»);

перевіряє, наскільки рідко трапляються малопоширені літери («ф», «щ», «ь»);

може додатково враховувати частоти біграм, триграм чи навіть довших комбінацій (l-грам).

Функція is_meaningful_text оцінює, чи є текст осмисленим, за такими кроками:

Підрахунок кількості кожної літери.

Використовується Counter(text), який створює словник: літера → скільки разів зустрілась.

Обчислення загальної кількості літер.

total = sum(freq_letters.values()) — це сума всіх значень у словнику.

Підрахунок поширених літер.

common = sum(freq_letters[char] for char in {"о", "а", "е"}) — рахує, скільки разів у тексті зустрілись ці три літери.
Підрахунок рідкісних літер.
rare = sum(freq_letters[char] for char in {"ф", "щ", "ь"}).
Перевірка співвідношень:
якщо частка поширених літер ≥ 0.2 (тобто вони становлять понад 20% тексту);
і частка рідкісних ≤ 0.05 (менше 5% тексту), то текст вважається змістовним.

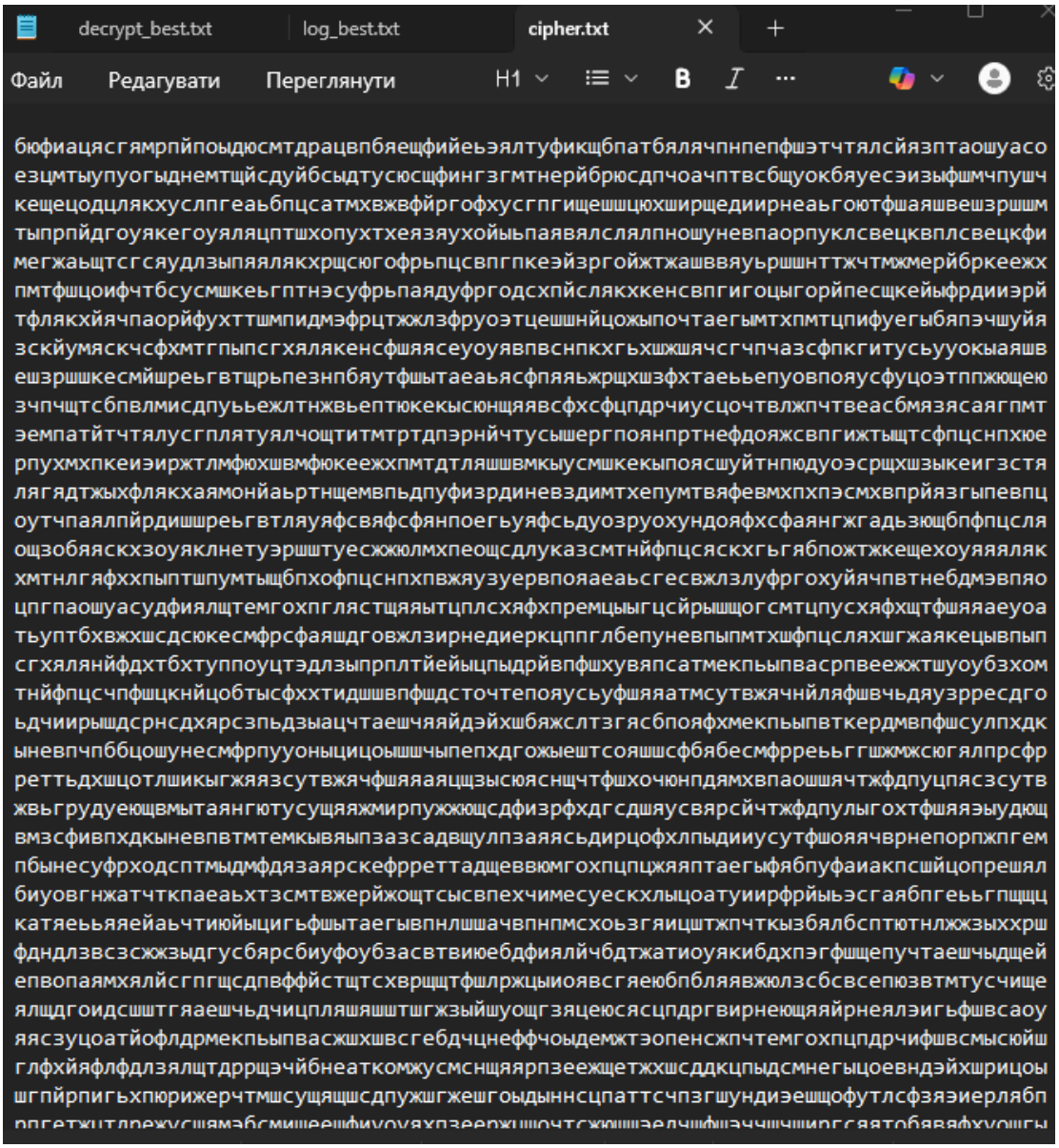
Цей метод дозволяє автоматично визначити, чи може розшифрований текст бути справжнім російським текстом.

5. Повторювати дії 3-4 доти, доки дешифрований текст не буде змістовним.

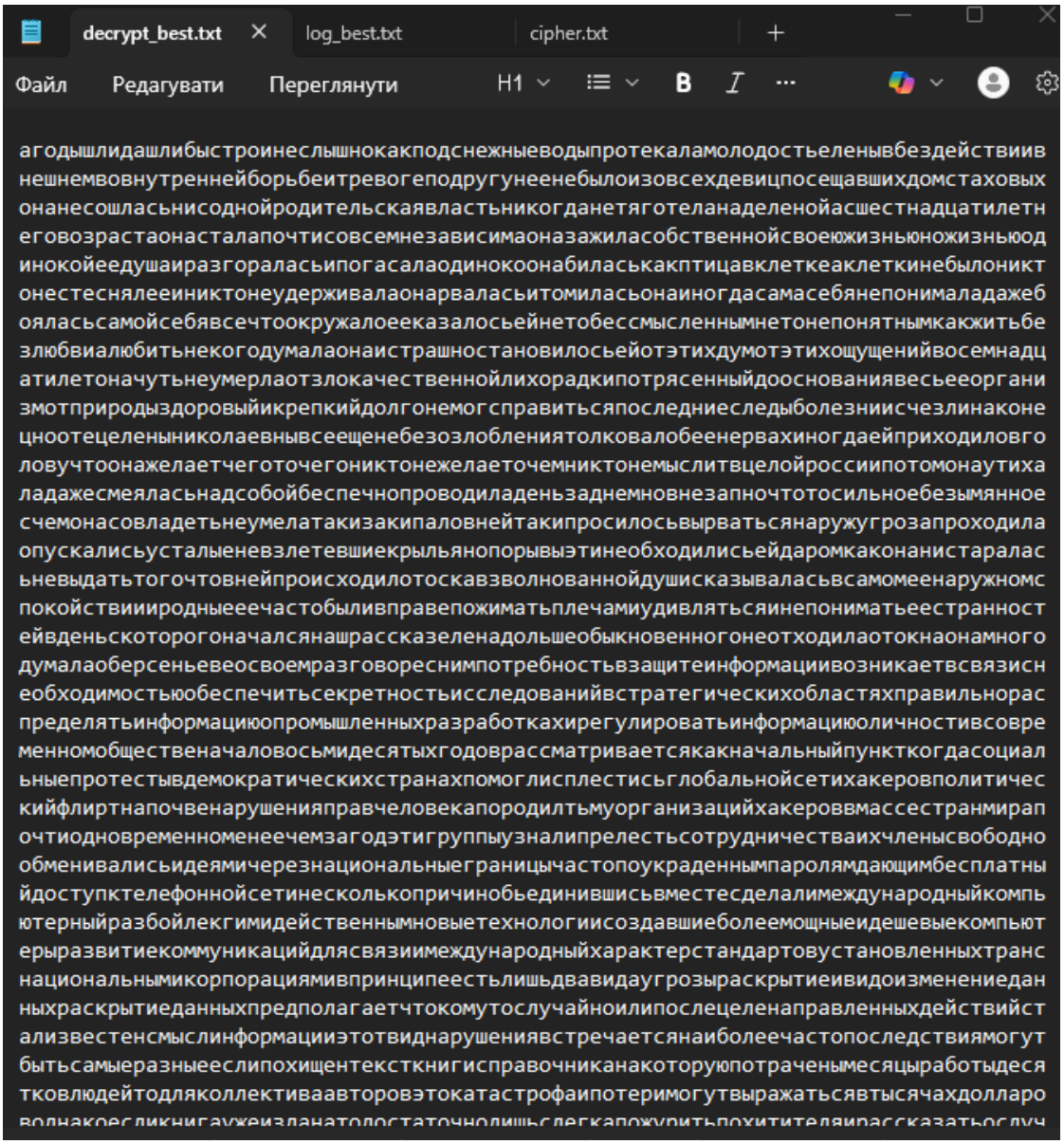
```
def try_candidates(cipher_text: str, candidates,
                   show_limit: int = 15, preview_len: int = 30):
    rows = []
    first_ok = None
    best = {"a": None, "b": None, "text": "", "score": -1e9}

    print("\nПеревірка кандидатів на ключ (a, b):")
    print("№ |    a |    b | інверсія |   score   | verdict | preview")
    print("---+------+------+---------+----------+---------+----------------------------")

    for i, c in enumerate(candidates, 1):
        a, b = c["a"], c["b"]
        inv = mod_inverse(a, M) is not None
        dec = decrypt_affine_bigrams(cipher_text, a, b) if inv else None
        ok = is_meaningful_text(dec) if dec else False
        sc = score_text(dec) if dec else -1e9
        prev = (dec or "")[:preview_len].replace("\n"," ")
        rows.append({"a": a, "b": b, "inverse": inv, "score": f"{sc:.2f}",
                     "ok": ok, "preview": prev})
        if i <= show_limit:
            print(f"{i:2d} | {a:4d} | {b:4d} |   {'yes' if inv else ' no'}   | {sc:10.2f} | "
                  f"{'OK' if ok else 'skip':<7}| {prev}")

        if ok and first_ok is None:
            first_ok = {"a": a, "b": b, "text": dec}
        if sc > best["score"]:
            best = {"a": a, "b": b, "text": dec or "", "score": sc}
```

Тут в циклі for у функції try_candidates програма повторює дії 3–4 для кожної пари (a, b)
(тобто “перебирає всі можливі ключі, дешифрує, оцінює результат”).
Функція is_meaningful_text(dec) — це саме “перевірка, чи текст осмислений”.
Якщо вона повертає True, тоді алгоритм вважає, що дешифрування вдалося, і зупиняється (це реалізація “автостопу”).
Якщо ж жоден текст не виявився змістовним, то алгоритм вибирає найкращий варіант за оцінкою score_text()

Отримані результати:
Шифротекст:



Відкритий/розшифрований текст:



Висновки:

Отримані результати демонструють побудову таблиці кандидатів ключа для афінного шифру за різними значеннями параметрів a та b.

На основі аналізу біграм шифротексту та їхніх можливих відповідників у відкритому тексті сформовано набір пар, що можуть бути потенційними ключами дешифрування.

Ця робота дозволяє краще зрозуміти принципи підбору параметрів афінного перетворення, застосування модульної арифметики та важливість правильного вибору коефіцієнтів для отримання осмисленого тексту. Практичне виконання сприяє формуванню навичок криптоаналізу, зокрема аналізу біграм і виявлення закономірностей у шифрованих даних.