Комп'ютерний Практикум 2 з Криптоаналізу.

Тема: Статистичні критерії на відкритий текст

Виконали: Бондар Петро, Кістаєв Матвій.

Варіант: 4

Задача полягала в побудові статистичних критеріїв для розрізнення гіпотез:

H_0: $X = x_1x_2x_3\dots x_n$ – текст української мови

H_1: X – випадковий текст

Згідно з варіантом, було побудовано декілька критеріїв, емпірично підібрані критичні значення, а також виконано порівняння точності критеріїв для текстів різних довжин та різних способів утворення "випадкової" послідовності

In [1]:
```
from pypdf import PdfReader
import itertools
import numpy as np
import random
import lzma
import deflate
from collections import defaultdict
import matplotlib.pyplot as plt
import pandas as pd
```

Завантаження текстів

Як джерело змістовних повідомлень було обрано тексти Конституції України, "Енеїди" Котляревського, "Хто ти?" Артема Чеха та бакалаврські роботи авторів цієї лабораторної роботи (видатних вчених, науковців, дослідників, кандидатів, членів, кореспондентів, академіків).

In [2]:
```
folder = "documents"
ukr_texts = ["Конституція.pdf", "Енеїда.pdf", "Кучма.pdf", "Хто.pdf", "Bondar_bakalavr.pdf", "Kistaiev_bakalavr.pdf"]

large_text = ""

for text in ukr_texts:
    reader = PdfReader(f"{folder}/{text}")
    for page in reader.pages:
        large_text += page.extract_text() + "\n"
```

Обробка текстів та створення змінних для перетворення текстових даних у числові

Очистка тексту та певні допоміжні структури

In [3]:
```
alph_full = list('АаБбВвГгҐґДдЕеЄєЖжЗзИиІіЇїЙйКкЛлМмНнОоПпРрСсТтУуФфХхЦцЧчШшЩщЬьЮюЯя')
alph_1 = list(' абвгґдеєжзиіїйклмнопрстуфхцчшщьюя')
alph_2 = [f'{c1}{c2}' for c1, c2 in itertools.product(alph_1, alph_1)]
lang = []

for c in large_text:
    if c in alph_full:
        lang.append(str.lower(c))
    elif lang[-1] != ' ':
        lang.append(' ')

alph = [' '] + alph_full
lang = "".join(lang)

ARING1 = dict(zip(list(alph_1), range(len(alph_1))))
ARING2 = dict(zip(list(alph_2), range(len(alph_2))))
```

Підрахунок частот символів та біграм в текстах українською мовою

In [ ]:
```
n = len(lang)

lang_1_shreq = dict(zip(alph_1, [0]*len(alph_1)))
for c in lang:
    lang_1_shreq[c] += 1

lang_1_shreq = {key: val / (n - 1) for key, val in lang_1_shreq.items()}


lang_2_shreq = dict(zip(alph_2, [0]*len(alph_2)))
for c1, c2 in zip(lang[:-1], lang[1:]):
```

```
    lang_2_shreq[f'{c1}{c2}'] += 1

lang_2_shreq = {key: val / (n - 1) for key, val in lang_2_shreq.items()}

print(n)
print(lang_1_shreq)
print(lang_2_shreq)
```

Шифрування, використовувані для спотворення тексту

Для утворення беззмістовних текстів з природніх ми використовуватимо шифр Віженера, а також афінну та біграмну афінну підстановки.

In [5]:
```
def vigenere_encrypt(ptext: str, key: str):
    r = int(np.ceil(len(ptext) / len(key)))

    ntxt = np.array([ARING1[c] for c in ptext])
    nkey = np.array([ARING1[c] for c in key] * r)

    return "".join([alph_1[c] for c in (ntxt + nkey) % len(alph_1)])


def vigenere_decrypt(ctext: str, key: str):
    r = int(np.ceil(len(ctext) / len(key)))

    ntxt = np.array([ARING1[c] for c in ctext])
    nkey = np.array([ARING1[c] for c in key] * r)

    return "".join([alph_1[c] for c in (ntxt - nkey) % len(alph_1)])


def affine_ecrypt(text: str, a: int, b: int):
    if np.gcd(a, len(alph_1)) != 1:
        raise RuntimeError("Incorrect key, gcd(a, n) != 1")

    res = ""
    for i in range(0, len(text)):
        id = (a * ARING1[text[i]] + b) % len(alph_1)
        res += alph_1[id]

    return res


def affine_bigram_ecrypt(text: str, a: int, b: int):
    if np.gcd(a, len(alph_2)) != 1:
        raise RuntimeError("Incorrect key, gcd(a, n) != 1")

    res = ""

    for c1, c2 in zip(text[:-1], text[1:]):
        id = (a * ARING2[f'{c1}{c2}'] + b) % len(alph_2)
        res += alph_2[id]

    return res
```

Генерація випадкових текстів

Беззмістовні тексти ми генеруватимемо наступними методами:

- Чисто випадковий текст, де кожен символ матиме однакову ймовірність появи на кожній позиції;
- Рекурсивна випадкова послідовність, в якій перші два символи обираються випадково, а кожен наступний є комбінацією двох попередніх за модулем;
- Зашифрований за допомогою шифру Віженера відкритий текст (ключ генерується випадково);
- Зашифрований за допомогою афінного шифру відкритий текст (ключ генерується випадково);
- Зашифрований за допомогою біграмного афінного шифру відкритий текст (ключ генерується випадково).

Змістовні тексти ми отримуватимемо шляхом "вирізання" випадкового підтексту з усього корпусу.

In [6]:
```
def generate_random_text(length):
    return ''.join(random.SystemRandom().choice(alph_1) for _ in range(length))


def generate_recur(length):
    res = ''
    res += random.SystemRandom().choice(alph_1)
    res += random.SystemRandom().choice(alph_1)
    for _ in range(0, length - 1):
        res += alph_1[(ARING1[res[-1]] + ARING1[res[-2]]) % len(alph_1)]
    return res


def generate_rand_lang_text(length):
    i = random.randrange(0, len(lang) - length)

    return "".join(lang[i:i+length])


def generate_enc_vigenere(length, key_len):
    text = generate_rand_lang_text(length)
    key = generate_random_text(key_len)

    return vigenere_encrypt(text, key)


alph_1_divs = np.array([2, 17, 34])
possible_a_1 = [i for i in range(1, len(alph_1)) if all((i % alph_1_divs) != 0)]

def generate_enc_affine1(length):
    text = generate_rand_lang_text(length)
    key_a = random.SystemRandom().choice(possible_a_1)
    key_b = random.SystemRandom().choice(range(len(alph_1)))

    return affine_ecrypt(text, key_a, key_b)


alph_2_divs = np.array([2, 17, 34])
possible_a_2 = [i for i in range(1, len(alph_2)) if all((i % alph_2_divs) != 0)]

def generate_enc_affine2(length):
    text = generate_rand_lang_text(length)
    key_a = random.SystemRandom().choice(possible_a_2)
    key_b = random.SystemRandom().choice(range(len(alph_2)))

    return affine_ecrypt(text, key_a, key_b)
```

Функції для обчислення потрібних характеристик для тексту

- Функції для підрахунку частот символів та біграм в обраному тексті.
- Функції для обчислення індексу відповідності (аби було) та усередненої ентропії в обраному тексті.
- Функції для обчислення кількості "порожніх якщиків".
- Функція для обчислення коефіцієнтів стиснення за допомогою lzma та deflate.

In [7]:
```
def gram_1_shreqs(A, text):
    text_1_shreqs = dict(zip(A, [0]*len(A)))

    for c in text:
        if c in A:
            text_1_shreqs[c] += 1

    return {key: val / (len(text) - 1) for key, val in text_1_shreqs.items()}

def gram_2_shreqs(A, text):
    text_2_shreqs = dict(zip(A, [0]*len(A)))

    for c1, c2 in zip(text[:-1], text[1:]):
        if f'{c1}{c2}' in A:
            text_2_shreqs[f'{c1}{c2}'] += 1

    return {key: val / (len(text) - 1) for key, val in text_2_shreqs.items()}

def index_of_coincidence(l_gram_counts):
    counts = np.array(list(l_gram_counts.values()))
    N = counts.sum()
    I = sum(count * (count - 1) for count in counts) / (N * (N - 1))

    return I

def avg_entropy(text, l):
    l_gram_shreqs = defaultdict(lambda: 0)

    for i in range(len(text) - l):
        l_gram_shreqs[text[i:i+l]] += 1

    probs = np.array(list(l_gram_shreqs.values())) / (len(text) - l + 1)
```

```
    H = -probs.dot(np.log2(probs)) / l

    return H


def n_empty_1_boxes(B_freq, text):
    return sum(np.array(list(gram_1_shreqs(B_freq, text).values())) == 0)

def n_empty_2_boxes(B_freq, text):
    return sum(np.array(list(gram_2_shreqs(B_freq, text).values())) == 0)


def lzma_compression_ratio(text):
    txt = text.encode()
    original_len = len(txt)

    ctxt = lzma.compress(txt)
    compressed_len = len(ctxt)

    return compressed_len / original_len


def deflate_compression_ratio(text):
    txt = text.encode()
    original_len = len(txt)

    ctxt = deflate.gzip_compress(txt, 12)
    compressed_len = len(ctxt)

    return compressed_len / original_len
```

Визначення частих та заборонених l-грам

Заборонені символи – найменш часті символи, сумарна ймовірність яких складає $\leq 5\%$.

Заборонені біграми – найменш часті біграми, сумарна ймовірність яких складає $\leq 0.25\%$.

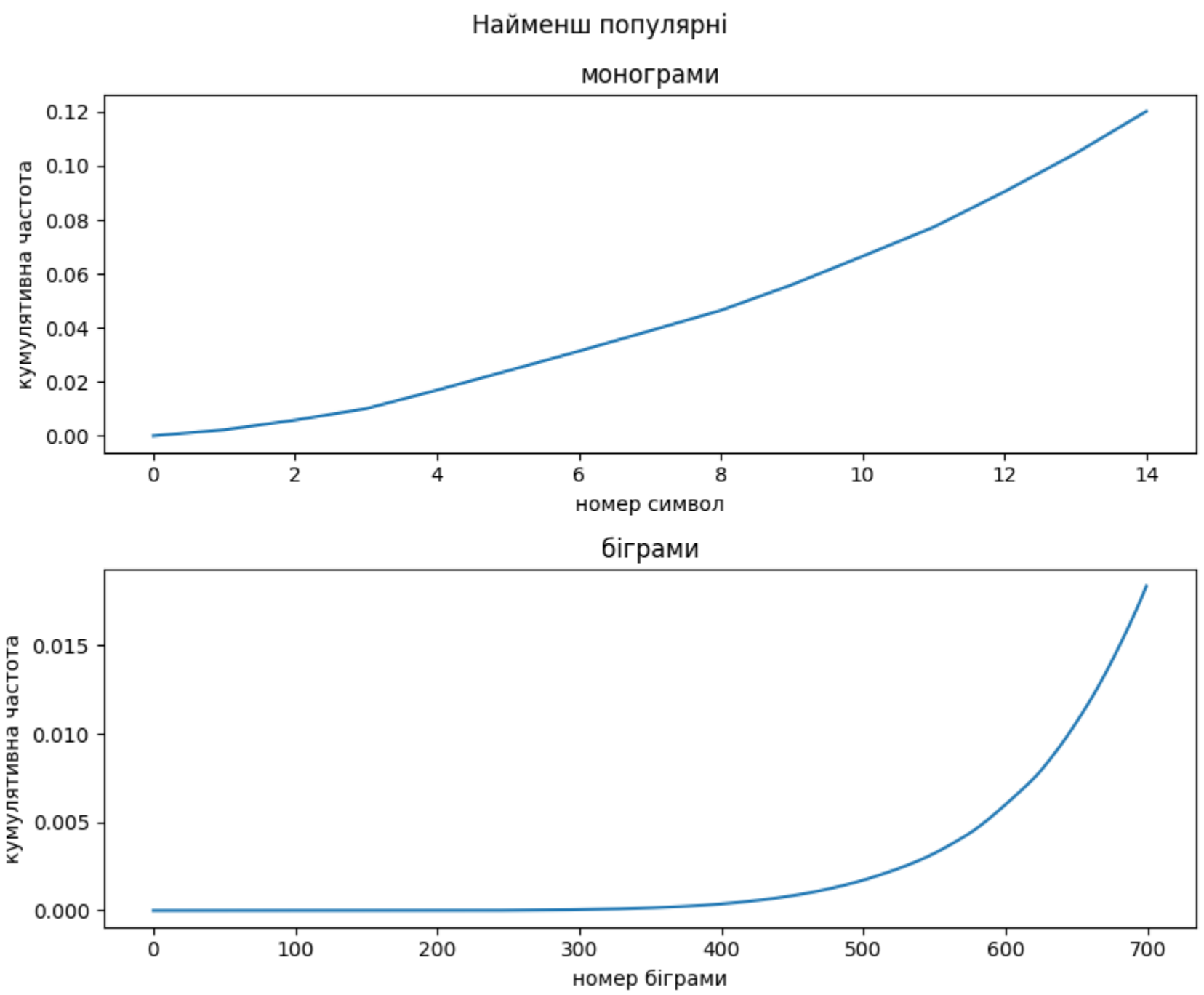
Популярні символи – найбільщ часті символи, сумарна ймовірність яких складає $\geq 80\%$.

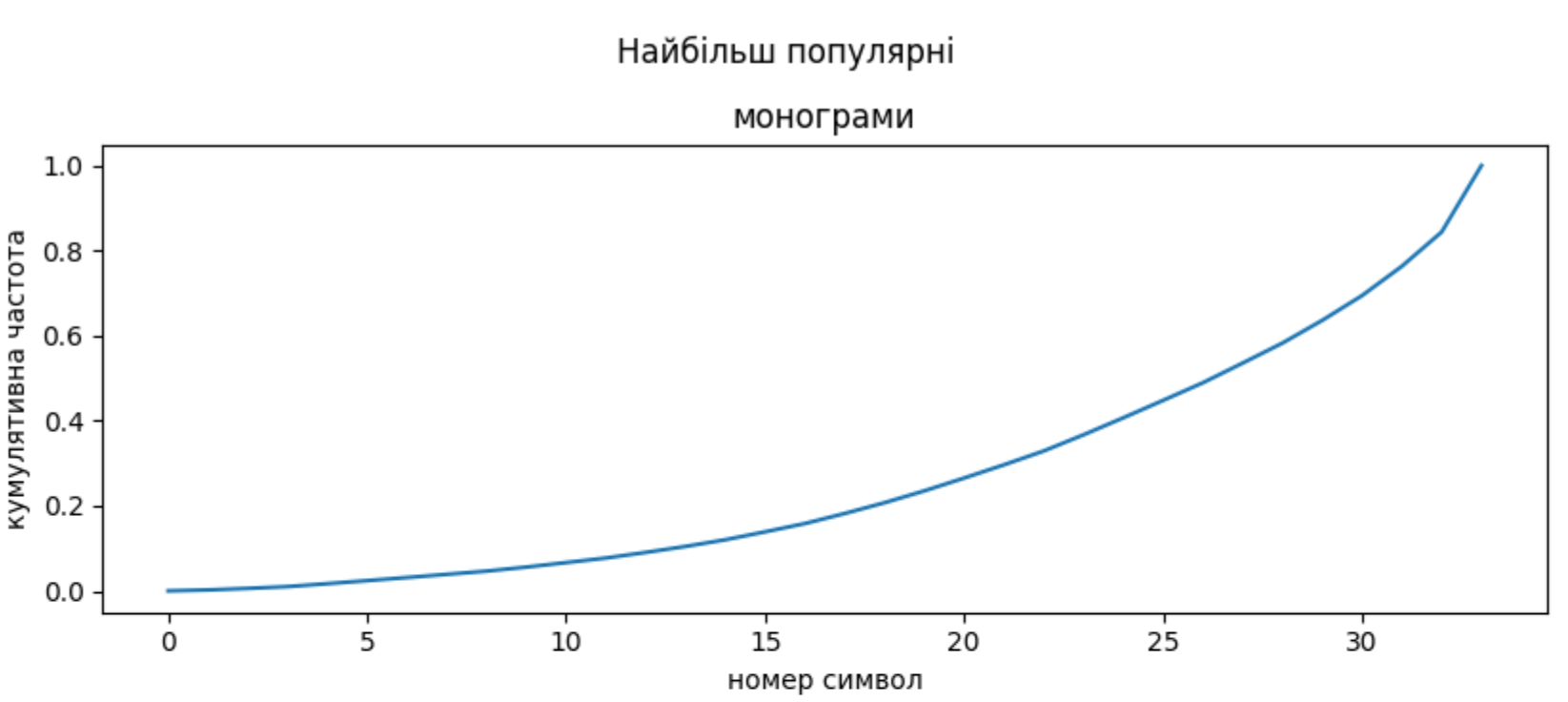
Популярні біграми – найменш часті біграми, сумарна ймовірність яких складає $\geq 80\%$.

Це визначалось "на око" за допомогою графіків, побудованих нижче.

In [17]:
```
A1_prh = {k: v for k, v in sorted(lang_1_shreq.items(), key=lambda item: item[1])[0:15]}
# print(A1_prh)
A2_prh = {k: v for k, v in sorted(lang_2_shreq.items(), key=lambda item: item[1])[0:700]}
# print(A2_prh)

fig, axs1 = plt.subplots(2, 1)
fig.set_size_inches(8.5, 7)
axs1[0].plot(range(len(A1_prh.keys())), np.cumsum(list(A1_prh.values())))
axs1[0].set_title("монограми")
axs1[0].set_xlabel("номер символ")
axs1[0].set_ylabel("кумулятивна частота")
axs1[1].plot(range(len(A2_prh.keys())), np.cumsum(list(A2_prh.values())))
axs1[1].set_title("біграми")
axs1[1].set_xlabel("номер біграми")
axs1[1].set_ylabel("кумулятивна частота")
fig.suptitle("Найменш популярні")
plt.tight_layout()
A1_prh = list(A1_prh.keys())[0:9]
A2_prh = list(A2_prh.keys())[0:450]


B1_freq = {k: v for k, v in sorted(lang_1_shreq.items(), key=lambda item: item[1])[:]}
# print(B1_freq)
B2_freq = {k: v for k, v in sorted(lang_2_shreq.items(), key=lambda item: item[1])[:]}
# print(B2_freq)

fig, axs2 = plt.subplots(2, 1)
fig.set_size_inches(8.5, 7)
axs2[0].plot(range(len(B1_freq.keys())), np.cumsum(list(B1_freq.values())))
axs2[0].set_title("монограми")
axs2[0].set_xlabel("номер символ")
axs2[0].set_ylabel("кумулятивна частота")
axs2[1].plot(range(len(B2_freq.keys())), np.cumsum(list(B2_freq.values())))
axs2[1].set_title("біграми")
axs2[1].set_xlabel("номер біграми")
axs2[1].set_ylabel("кумулятивна частота")
fig.suptitle("Найбільш популярні")
plt.tight_layout()
```

```
B1_freq = list(B1_freq.keys())[14:]
B2_freq = list(B2_freq.keys())[850:]


lang_1_entropy = avg_entropy("".join(lang), 1)
lang_2_entropy = avg_entropy("".join(lang), 2)
```





Визначення функцій, що обчислюють статистики критеріїв

Критерії обрані згідно відповідно варіанту 4:

- Критерії 1.0-1.3: Критерії, що ґрунтуються на підрахунку заборонених символів/біграм.
- Критерій 3.0: Критерій, що ґрунтується на обчисленні відхилення усередненої ентропії від теоретичного значення.
- Критерій 5.1: Критерій порожніх ящиків на основі частих символів/біграм.
- Критерій (структурний): Ґрунтується на коефіцієнті стиснення текстів за допомогою алгоритму DEFLATE

In [18]:
```
def crit_10_stat(text, l):
    if l == 1:
        return max(np.array(list(gram_1_shreqs(A1_prh, text).values())))
    elif l == 2:
        return max(np.array(list(gram_2_shreqs(A2_prh, text).values())))

def crit_11_stat(text, l):
    if l == 1:
        return sum(np.array(list(gram_1_shreqs(A1_prh, text).values())) > 0)
    elif l == 2:
        return sum(np.array(list(gram_2_shreqs(A2_prh, text).values())) > 0)

def crit_12_stat(text, l):
    if l == 1:
        return max(np.array(list(gram_1_shreqs(A1_prh, text).values())))
    elif l == 2:
        return max(np.array(list(gram_2_shreqs(A2_prh, text).values())))

def crit_13_stat(text, l):
    if l == 1:
        return sum(list(gram_1_shreqs(A1_prh, text).values()))
    elif l == 2:
        return sum(list(gram_2_shreqs(A2_prh, text).values()))

def crit_30_stat(text, l):
    if l == 1:
        return avg_entropy(text, 1) - lang_1_entropy
    elif l == 2:
        return avg_entropy(text, 2) - lang_2_entropy

def crit_51_stat(text, l):
    if l == 1:
        return n_empty_1_boxes(B1_freq, text)
    elif l == 2:
        return n_empty_2_boxes(B2_freq, text)
```

Визначення критичних значень для кожного критерію емпіричним шляхом

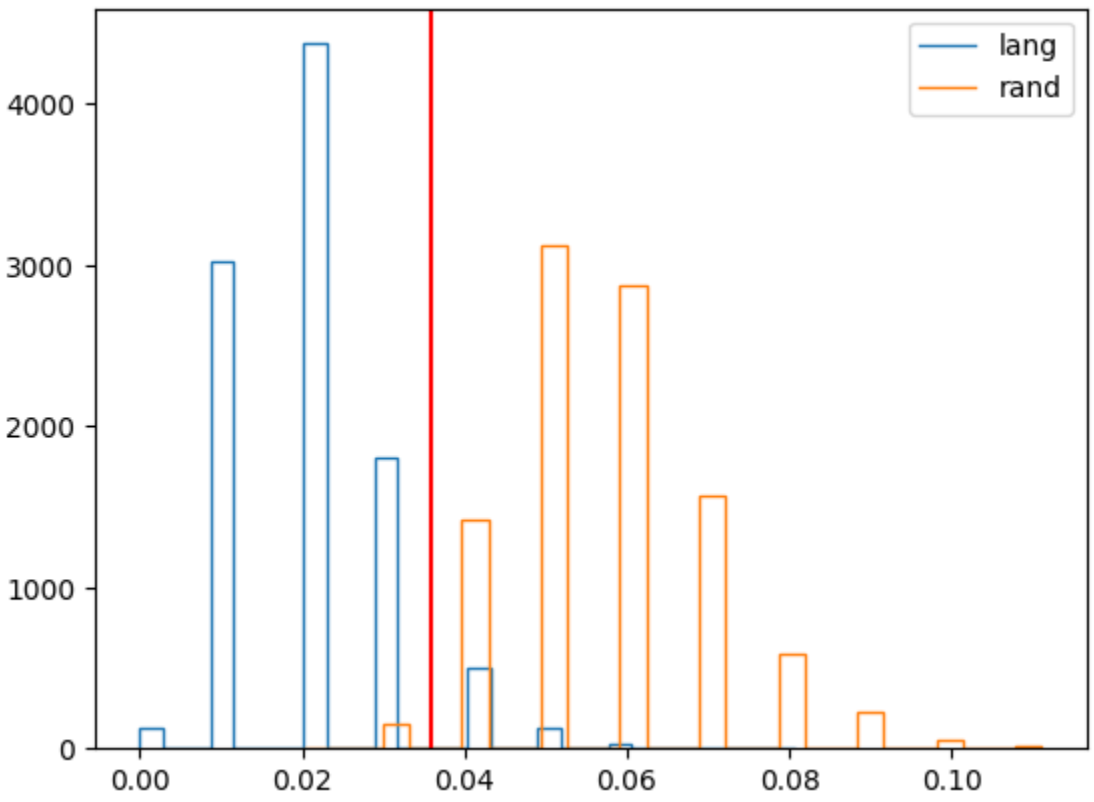
Для кожного критерію та для кожного розміру текстів було побудовано гістограми, які показують розподіл значень статистики критерію для справжніх текстів (синій) та для повністю випадкових текстів (оранжевий).

Виходячи із цих гістограм, на око було визначено "оптимальні" критичні значення, які найкраще відділяли б справжні тексти від випадкових (одночасно мінімізовували помилки першого і другого родів, наскільки це можливо)

Приклад такої гістограми та відповідного критичного значення (червоним) для критерію 1.3 для 1-грам наведено нижче.

In [27]:
```
def plot_stat_distr(crit_stat_f, text_len, n_texts):
    lang_stat = []
    rand_stat = []

    for i in range(n_texts):
        lang_text = generate_rand_lang_text(text_len)
        rand_text = generate_random_text(text_len)

        lang_stat.append(crit_stat_f(lang_text))
        rand_stat.append(crit_stat_f(rand_text))

    ax = plt.subplot()

    counts1, bins1 = np.histogram(lang_stat, int(np.ceil(3*np.log(n_texts))))
    ax.stairs(counts1, bins1)

    counts2, bins2 = np.histogram(rand_stat, int(np.ceil(3*np.log(n_texts))))
    ax.stairs(counts2, bins2)

    ax.legend(["lang", "rand"])


n = 100
S = 10000 #int(1000*np.ceil(0.5*np.sqrt(n)))

plot_stat_distr(lambda text: crit_12_stat(text, 1), n, S)
plt.axvline(x=0.036, color='r')
```

```
k10 = {1: {10: 0, 100: 0, 1000: 0, 10000: 0},
         2: {10: 0, 100: 0, 1000: 0, 10000: 0}}

k11 = {1: {10: 1.5, 100: 6.5, 1000: 8.85, 10000: 8.99},
        2: {10: 0.25, 100: 14, 1000: 144, 10000: 225}}

k12 = {1: {10: 0.0675, 100: 0.035, 1000: 0.0285, 10000: 0.0285},
        2: {10: 0.06, 100: 0.01, 1000: 0.00315, 10000: 0.0013}}

k13 = {1: {10: 0.16, 100: 0.14, 1000: 0.15, 10000: 0.16} ,
        2: {10: 0.075, 100: 0.21, 1000: 0.21, 10000: 0.21}}

k30 = {1: {10: -1.715, 100: 0.05, 1000: 0.3, 10000: 0.3} ,
        2: {10: -2.67, 100: -0.88, 1000: 0.18, 10000: 0.45}}

k51 = {1: {10: 13.5, 100: 1.9, 1000: 0.1, 10000: 0.1} ,
        2: {10: 300.4, 100: 264, 1000: 108, 10000: 1.25}}

ks = {10: 2.125, 100: 0.75, 1000: 0.475, 10000: 0.38}
```



Генерування вибірок текстів

Було згенеровано по 4 вибірки текстів довжини L розмірів $N(L)$:

L	N
10	10000
100	10000
1000	10000
10000	1000

для справжніх текстів, та випадкових (і не дуже) текстів утвореними усіма видами спотворень

Для кожної вибірки було застосовано усі критерії та обчислено відповідні ймовірності помилок 1-го та 2-го роду. Результати записано в велику табличку.

Тут:

- FR (false random) - помилка розпізнавання природнього тексту як випадкового (помилка першого роду, або FP (false positive)).
- FL (false language) - помилка розпізнавання випадкового тексту як природнього (помилка другого роду, або FN (false negative)).

In [11]:
```
def count_errors(rand_text_gen, crit_stat_f, crit_val, text_len, n_texts):
    TL_count = 0
    FL_count = 0

    TR_count = 0
    FR_count = 0

    for i in range(n_texts):
        lang_text = generate_rand_lang_text(text_len)
        rand_text = rand_text_gen(text_len)

        CL = crit_stat_f(lang_text)
        CR = crit_stat_f(rand_text)
```

```
        if CL <= crit_val:
            TL_count += 1
        else:
            FR_count += 1

        if CR > crit_val:
            TR_count += 1
        else:
            FL_count += 1


    return FR_count / n_texts, FL_count / n_texts


col_names = ["crit_name", "crit_val", "rand_gen", "l_gram", "L", "FL_prob", "FR_prob"]
data = pd.DataFrame(columns=col_names)

rand_gens = [generate_random_text, generate_recur, generate_enc_affine1,
             generate_enc_affine2,
             lambda text: generate_enc_vigenere(text, 1),
             lambda text: generate_enc_vigenere(text, 5),
             lambda text: generate_enc_vigenere(text, 10)]

rand_gens_names = ["true_rand", "recur_rand", "affine1", "affine2",
                   "vigenere1", "vigenere5", "vigenere10"]


criteria = [crit_10_stat, crit_11_stat, crit_12_stat, crit_13_stat, crit_30_stat, crit_51_stat]
criteria_names = ["1.0", "1.1", "1.2", "1.3", "3.0", "5.1"]
crit_vals = [k10, k11, k12, k13, k30, k51]

n_texts = {10: 10000, 100: 10000, 1000: 10000, 10000: 1000}


for rand_gen in zip(rand_gens, rand_gens_names):
    for L in [10, 100, 1000, 10000]:
        N = n_texts[L]
        for criterion in zip(criteria, criteria_names, crit_vals):
            for l in [1, 2]:
                crit_stat = lambda text, l=l: criterion[0](text, l)
                crit_val = criterion[2][l][L]
                FR_prob, FL_prob = count_errors(rand_gen[0], crit_stat, crit_val, L, N)

                new_entry = pd.DataFrame([(criterion[1], crit_val, rand_gen[1], l, L, FL_prob, FR_prob)],
                                         columns=col_names)
                data = pd.concat([data, new_entry], ignore_index = True)
```

```
C:\Users\Lollo\AppData\Local\Temp\ipykernel_37072\1002958639.py:60: FutureWarning: The behavior of DataFrame concaten
ation with empty or all-NA entries is deprecated. In a future version, this will no longer exclude empty or all-NA co
lumns when determining the result dtypes. To retain the old behavior, exclude the relevant entries before the concat
operation.
  data = pd.concat([data, new_entry], ignore_index = True)
```

Окремо блок коду під структурний критерій (із технічних причин)

In [12]:
```
for rand_gen in zip(rand_gens, rand_gens_names):
    for L in [10, 100, 1000, 10000]:
        N = n_texts[L]
        criterion = [deflate_compression_ratio, 'deflate structural', ks]

        crit_val = ks[L]
        FR_prob, FL_prob = count_errors(rand_gen[0], deflate_compression_ratio, crit_val, L, N)

        new_entry = pd.DataFrame([('deflate structural', crit_val, rand_gen[1], None, L, FL_prob, FR_prob)],
                                 columns=col_names)
        data = pd.concat([data, new_entry], ignore_index = True)
```

Експорт в ексель табличку :)

In [21]:
```
print(len(data.index))
data = data.sort_values(by=['rand_gen', 'L', 'l_gram', 'crit_name'])
data.head(50)

with pd.ExcelWriter('full_results.xlsx', mode='a') as w:
    data.to_excel(w, sheet_name='All Probs')
```

364

Результати

Чисто випадкова послідовність							
Розмір l-грами		l = 1			l = 2		
L	Критерій	Критичне значення	FP	FN	Критичне значення	FP	FN
10	1.0	0	0.3883	0.0441	0	0.0055	0.0237
	1.1	1.5	0.0574	0.2439	0.25	0.0057	0.0206
	1.2	0.0675	0.3803	0.0436	0.06	0.0071	0.024
	1.3	0.16	0.067	0.2078	0.075	0.0063	0.0262
	3.0	-1.715	0.5806	0.2533	-2.67	0.9326	0.0245
	5.1	13.5	0.2472	0.1643	300.4	0.008	0.0275
	deflate structural				2.125	0.712	0.8778
100	1.0	0	0.9888	0	0	0.0724	0
	1.1	6.5	0.006	0.0074	14	0	0
	1.2	0.035	0.0657	0.014	0.01	0.0708	0
	1.3	0.14	0.0002	0.0014	0.21	0	0.0003
	3.0	0.05	0	0.0007	-0.88	0.0002	0.0183
	5.1	1.9	0.2611	0.7325	264	0.0003	0.0006
	deflate structural				0.75	0.0641	0.1715
1000	1.0	0	1	0	0	0.4641	0
	1.1	8.85	0.0163	0	144	0	0
	1.2	0.0285	0.0039	0.0003	0.00315	0.0014	0.0044
	1.3	0.15	0	0	0.21	0	0
	3.0	0.3	0	0	0.18	0	0
	5.1	0.1	0.0376	1	108	0.0385	0.008
	deflate structural				0.475	0	0
10000	1.0	0	1	0	0	0.97	0
	1.1	8.99	0.175	0	225	0	0
	1.2	0.0285	0	0	0.0013	0	0
	1.3	0.16	0	0	0.21	0	0
	3.0	0.3	0	0	0.45	0	0
	5.1	0.1	0.028	1	1.25	0.292	0.999
	deflate structural				0.38	0	0

Випадкова рекурсивна послідовність							
Розмір l-грами		l = 1			l = 2		
L	Критерій	Критичне значення	FP	FN	Критичне значення	FP	FN
10	1.0	0	0.3919	0.0449	0	0.0058	0.0259
	1.1	1.5	0.0628	0.2404	0.25	0.0065	0.0226
	1.2	0.0675	0.3819	0.0418	0.06	0.0076	0.0268
	1.3	0.16	0.0705	0.172	0.075	0.0059	0.023
	3.0	-1.715	0.5797	0.493	-2.67	0.936	0.0046
	5.1	13.5	0.2552	0.2136	300.4	0.0093	0.0564
	deflate structural				2.125	0.7084	1
100	1.0	0	0.9891	0.0037	0	0.0701	0
	1.1	6.5	0.006	0.6309	14	0	0.3783
	1.2	0.035	0.0645	0.0049	0.01	0.0722	0
	1.3	0.14	0.0008	0.0682	0.21	0	0.0665
	3.0	0.05	0	0.7538	-0.88	0.0006	1
	5.1	1.9	0.262	0	264	0.0009	0
	deflate structural				0.75	0.0609	1
1000	1.0	0	1	0.0032	0	0.4652	0
	1.1	8.85	0.0179	0.8721	144	0	1
	1.2	0.0285	0.0044	0.0033	0.00315	0.0031	0
	1.3	0.15	0	0.0691	0.21	0	0.0648
	3.0	0.3	0	1	0.18	0	1
	5.1	0.1	0.039	0	108	0.0394	0
	deflate structural				0.475	0	1
10000	1.0	0	1	0.004	0	0.976	0
	1.1	8.99	0.167	0.876	225	0	1
	1.2	0.0285	0	0.005	0.0013	0	0
	1.3	0.16	0	0.085	0.21	0	0.055
	3.0	0.3	0	1	0.45	0	1
	5.1	0.1	0.03	0	1.25	0.331	0
	deflate structural				0.38	0	1

Безмістовна послідовність за допомогою афінної підстановки							
Розмір l-грами		l = 1			l = 2		
L	Критерій	Критичне значення	FP	FN	Критичне значення	FP	FN
10	1.0	0	0.3932	0.0577	0	0.0071	0.0297
	1.1	1.5	0.0589	0.2743	0.25	0.0074	0.0348
	1.2	0.0675	0.3824	0.0638	0.06	0.0081	0.0293
	1.3	0.16	0.0762	0.2313	0.075	0.0062	0.0305
	3.0	-1.715	0.5821	0.4134	-2.67	0.9343	0.0639
	5.1	13.5	0.2545	0.1163	300.4	0.0076	0.0355
	deflate structural				2.125	0.7126	0.8819
100	1.0	0	0.9893	0	0	0.0698	0.0025
	1.1	6.5	0.0071	0.3908	14	0	0.011
	1.2	0.035	0.0587	0.0261	0.01	0.0737	0.0015
	1.3	0.14	0.001	0.0607	0.21	0	0.0216
	3.0	0.05	0	0.9998	-0.88	0.0003	0.9998
	5.1	1.9	0.2701	0.0126	264	0.0004	0.0036
	deflate structural				0.75	0.0624	0.984
1000	1.0	0	1	0	0	0.4705	0.0009
	1.1	8.85	0.0159	0.3498	144	0	0.987
	1.2	0.0285	0.004	0.0097	0.00315	0.0024	0.0015
	1.3	0.15	0	0.0558	0.21	0	0.0073
	3.0	0.3	0	1	0.18	0	1
	5.1	0.1	0.038	0.322	108	0.0404	0.0013
	deflate structural				0.475	0	1
10000	1.0	0	1	0	0	0.978	0
	1.1	8.99	0.165	0.235	225	0	0.824
	1.2	0.0285	0	0.009	0.0013	0	0.003
	1.3	0.16	0	0.077	0.21	0	0.003
	3.0	0.3	0	1	0.45	0	1
	5.1	0.1	0.028	0.487	1.25	0.358	0.002
	deflate structural				0.38	0	1

Безмістовна послідовність за допомогою афінної біграмної підстановки							
Розмір l-грами		l = 1			l = 2		
L	Критерій	Критичне значення	FP	FN	Критичне значення	FP	FN
10	1.0	0	0.3919	0.0601	0	0.0065	0.0322
	1.1	1.5	0.0595	0.2885	0.25	0.0068	0.0358
	1.2	0.0675	0.3839	0.0615	0.06	0.0081	0.033
	1.3	0.16	0.0679	0.2332	0.075	0.0074	0.0335
	3.0	-1.715	0.5859	0.4168	-2.67	0.9325	0.071
	5.1	13.5	0.2653	0.1125	300.4	0.0078	0.0351
	deflate structural				2.125	0.7079	0.8902
100	1.0	0	0.9857	0	0	0.0702	0.0022
	1.1	6.5	0.0056	0.391	14	0	0.0097
	1.2	0.035	0.0603	0.0272	0.01	0.0694	0.0026
	1.3	0.14	0.0006	0.0576	0.21	0	0.026
	3.0	0.05	0.0003	0.9999	-0.88	0.0003	0.9998
	5.1	1.9	0.2655	0.0128	264	0.0009	0.0019
	deflate structural				0.75	0.064	0.9848
1000	1.0	0	1	0	0	0.4657	0.0009
	1.1	8.85	0.016	0.3467	144	0	0.9868
	1.2	0.0285	0.0039	0.0091	0.00315	0.0033	0.0022
	1.3	0.15	0	0.0589	0.21	0	0.0066
	3.0	0.3	0	1	0.18	0	1
	5.1	0.1	0.0362	0.317	108	0.0424	0.0026
	deflate structural				0.475	0	1
10000	1.0	0	1	0	0	0.976	0
	1.1	8.99	0.172	0.25	225	0	0.825
	1.2	0.0285	0	0.01	0.0013	0	0.001
	1.3	0.16	0	0.103	0.21	0	0.01
	3.0	0.3	0	1	0.45	0	1
	5.1	0.1	0.033	0.507	1.25	0.291	0.003
	deflate structural				0.38	0	1

Безмістовна послідовність за допомогою шифру Віженера (ключ довжини 1)							
Розмір l-грами		l = 1			l = 2		
L	Критерій	Критичне значення	FP	FN	Критичне значення	FP	FN
10	1.0	0	0.3892	0.0699	0	0.0076	0.0571
	1.1	1.5	0.0614	0.2897	0.25	0.007	0.054
	1.2	0.0675	0.38	0.0715	0.06	0.0065	0.0588
	1.3	0.16	0.0728	0.2352	0.075	0.0081	0.0551
	3.0	-1.715	0.5814	0.4277	-2.67	0.9337	0.0673
	5.1	13.5	0.2556	0.1134	300.4	0.0083	0.0555
	deflate structural				2.125	0.6998	0.8823
100	1.0	0	0.9883	0.0004	0	0.0729	0.0283
	1.1	6.5	0.0088	0.367	14	0	0.0393
	1.2	0.035	0.0641	0.0569	0.01	0.0701	0.0296
	1.3	0.14	0.0007	0.0754	0.21	0	0.0435
	3.0	0.05	0	1	-0.88	0.0006	0.9998
	5.1	1.9	0.2619	0.0306	264	0.001	0.0308
	deflate structural				0.75	0.0616	0.9956
1000	1.0	0	1	0	0	0.4762	0.0166
	1.1	8.85	0.0162	0.336	144	0	0.9974
	1.2	0.0285	0.0035	0.048	0.00315	0.0037	0.0302
	1.3	0.15	0	0.0821	0.21	0	0.026
	3.0	0.3	0	1	0.18	0	1
	5.1	0.1	0.0345	0.3259	108	0.0401	0.0291
	deflate structural				0.475	0	1
10000	1.0	0	1	0	0	0.982	0
	1.1	8.99	0.184	0.208	225	0	0.837
	1.2	0.0285	0	0.054	0.0013	0	0.032
	1.3	0.16	0	0.096	0.21	0	0.035
	3.0	0.3	0	1	0.45	0	1
	5.1	0.1	0.029	0.523	1.25	0.335	0.028
	deflate structural				0.38	0	1

Безмістовна послідовність за допомогою шифру Віженера (ключ довжини 5)							
Розмір l-грами		l = 1			l = 2		
L	Критерій	Критичне значення	FP	FN	Критичне значення	FP	FN
10	1.0	0	0.3812	0.0497	0	0.0066	0.0263
	1.1	1.5	0.0601	0.2535	0.25	0.008	0.0257
	1.2	0.0675	0.394	0.0505	0.06	0.007	0.024
	1.3	0.16	0.0742	0.217	0.075	0.0058	0.0253
	3.0	-1.715	0.584	0.2864	-2.67	0.933	0.0337
	5.1	13.5	0.2509	0.161	300.4	0.0092	0.0281
	deflate structural				2.125	0.7087	0.8802
100	1.0	0	0.9888	0	0	0.0734	0
	1.1	6.5	0.0063	0.0458	14	0	0.0002
	1.2	0.035	0.0636	0.0184	0.01	0.0726	0
	1.3	0.14	0.0006	0.009	0.21	0	0.004
	3.0	0.05	0	0.14	-0.88	0.0002	0.4804
	5.1	1.9	0.273	0.4001	264	0.0002	0.0027
	deflate structural				0.75	0.0657	0.4914
1000	1.0	0	1	0	0	0.4631	0
	1.1	8.85	0.0162	0.0002	144	0	0.0069
	1.2	0.0285	0.0041	0.006	0.00315	0.0023	0
	1.3	0.15	0	0.002	0.21	0	0.0005
	3.0	0.3	0	0.0176	0.18	0	0.0259
	5.1	0.1	0.0374	0.9986	108	0.0435	0.0019
	deflate structural				0.475	0	0.6227
10000	1.0	0	1	0	0	0.966	0
	1.1	8.99	0.171	0	225	0	0
	1.2	0.0285	0	0.009	0.0013	0	0
	1.3	0.16	0	0.003	0.21	0	0
	3.0	0.3	0	0.011	0.45	0	0.002
	5.1	0.1	0.033	1	1.25	0.309	0
	deflate structural				0.38	0	1

Безмістовна послідовність за допомогою шифру Віженера (ключ довжини 10)							
Розмір l-грами		l = 1			l = 2		
L	Критерій	Критичне значення	FP	FN	Критичне значення	FP	FN
10	1.0	0	0.3844	0.0476	0	0.0063	0.0241
	1.1	1.5	0.0586	0.2393	0.25	0.0079	0.0263
	1.2	0.0675	0.3887	0.0497	0.06	0.0064	0.0222
	1.3	0.16	0.0703	0.2131	0.075	0.0066	0.0233
	3.0	-1.715	0.5824	0.2503	-2.67	0.935	0.0221
	5.1	13.5	0.2558	0.1707	300.4	0.0082	0.0277
	deflate structural				2.125	0.7169	0.873
100	1.0	0	0.9874	0	0	0.0755	0
	1.1	6.5	0.0075	0.0179	14	0	0
	1.2	0.035	0.0659	0.0153	0.01	0.0782	0
	1.3	0.14	0.0011	0.0033	0.21	0	0.0026
	3.0	0.05	0.0001	0.0227	-0.88	0.0001	0.202
	5.1	1.9	0.2618	0.562	264	0.0006	0.0022
	deflate structural				0.75	0.0603	0.3146
1000	1.0	0	1	0	0	0.4683	0
	1.1	8.85	0.0189	0	144	0	0
	1.2	0.0285	0.0042	0.0027	0.00315	0.0041	0
	1.3	0.15	0	0.0001	0.21	0	0
	3.0	0.3	0	0	0.18	0	0.0006
	5.1	0.1	0.0349	1	108	0.0407	0.0015
	deflate structural				0.475	0	0.0421
10000	1.0	0	1	0	0	0.977	0
	1.1	8.99	0.173	0	225	0	0
	1.2	0.0285	0	0.001	0.0013	0	0
	1.3	0.16	0	0	0.21	0	0
	3.0	0.3	0	0	0.45	0	0
	5.1	0.1	0.033	1	1.25	0.334	0
	deflate structural				0.38	0	0.113

Висновки

Основними труднощами з якими довелось стикнутись в ході виконання лабораторної роботи було:

- знайти книгу "Україна не росія" в форматі, який дозволятиме зчитування тексту (а не скан)
- розібратись, де ліво, а де право (для статистик критеріїв)
- написати протокол

Всю логіку, що, чому і як працює, ми проговорювали на захисті, можна я не буду це все розписувати тут, відпустіть будь ласка...

P.S. Короткий опис алгоритму DEFLATE

DEFLATE поєднує в собі ідеї алгоритму кодування LZ77 та кодів Хаффмана:

- LZ77: Даний алгоритм працює за принципом заміни підпослідовностей символів на back-reference'и на позиції, де такі послідовності вже зустрічались раніше. Відсилання назад робиться лише до певної довжини (так званого вікна), розмір цього вікна і є параметром стиснення на вхід DEFLATE
- Huffman: Дерево Хаффмана на основі частот символів, будує коди таким чином, щоб частим символам відповідали коротші коди. Для DEFLATE це дерево дещо модифіковане з розширеним алфавітом, щоб краще працювати із текстом отриманим після LZ77.