МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТИХНІЧНИЙ ІНСТИТУТ ІМ.ІГОРЯ СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра інформаційної безпеки

Комп’ютерний практикум №3

«Криптоаналіз афінної біграмної підстановки»

Варіант 3

Виконали:

Студенти 3 курсу

Групи ФБ-94

Волков Артем та Калінічев Сергій

Київ

НТУУ «КПІ»

Мета роботи:

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

Завдання 1

```rust
fn egcd(a: isize, b: isize) -> (isize, isize, isize) {
    if a == 0 {
        (b, 0, 1)
    } else {
        let (g : isize , y : isize , x : isize ) = egcd( a: b.rem_euclid( rhs: a), b: a);
        (g, x - (b / a) * y, y)
    }
}

fn mod_inv(a: isize, n: isize) -> Option<isize> {
    let (g : isize , x : isize , _) = egcd(a, b: n);
    if g == 1 {
        Some(x.rem_euclid( rhs: n))
    } else {
        None
    }
}
```

```rust
fn linear(a: isize, b: isize, n: isize) -> Option<Vec<isize>> {
    let a : isize = a.rem_euclid( rhs: n);
    let b : isize = b.rem_euclid( rhs: n);
    let (g : isize , x : isize , _) = egcd(a, b: n);
    if g != 1 {
        if b.rem_euclid( rhs: g) != 0 {
            None
        } else {
            let ans : Vec<isize> = linear( a: a / g, b: b / g, n: n / g).unwrap();
            Some(
                (0..g)
                    .map(|i : isize | ans.iter().map(move |x : &isize | (x + (g - i) * n / g).rem_euclid( rhs: n)))
                    .flatten() : impl Iterator<Item=isize>
                    .collect::<Vec<isize>>(),
            )
        }
    } else {
        Some(vec![(x * b).rem_euclid( rhs: n)])
    }
}
```

```rust
assert_eq!(mod_inv( a: 3, n: 26), Some(9));
assert_eq!(linear( a: 1287, b: 447, n: 516), Some(vec![109, 453, 281]));
```

Завдання 2

```rust
pub fn count_bigram_frequency(text: &str) -> HashMap<String, f32> {
    let chunks : IntoChunks<Chars> = text.chars().chunks( size: 2);
    let iter = chunks
        .into_iter() : IntoChunks<Chars>
        .map(|item| item.collect::<String>())
        .filter(|n| n.chars().count() == 2);
    let mut frequency : BTreeMap<?, f32> = BTreeMap::new();
    for item in iter {
        *frequency.entry( key: item).or_insert( default: 0f32) += 1f32;
    }

    let len: f32 = frequency.values().sum();
    for (_, v : &mut f32 ) in frequency.iter_mut() {
        *v = *v / len;
    }
    frequency
        .into_iter() : impl Iterator<Item=(...)>
        .sorted_by(|(_, a : &f32 ), (_, b : &f32 )| b.partial_cmp( other: a).unwrap_or( default: Ordering::Equal))
        .take( n: 5) : impl Iterator<Item=(...)>
        .collect::<HashMap<_, _>>()
}
```

```
["рб", "тд", "кд", "во", "щю"]
```

Завдання 3

```rust
let map : HashMap<String, f32> = count_bigram_frequency(&text);

let most_bi : Vec<String> = vec![
    "ст".to_string(),
    "но".to_string(),
    "то".to_string(),
    "на".to_string(),
    "ен".to_string(),
];
let vec : Vec<Vec<String>> = vec![
    most_bi,
    map.keys().map(|s : &String | s.to_string()).collect::<Vec<_>>(),
];
println!(
    "{:?}",
    map.keys().map(|s : &String | s.to_string()).collect::<Vec<_>>()
);
let vec : Vec<Vec<?>> = vec
    .into_iter() : impl Iterator<Item=Vec<...>>
    .multi_cartesian_product() : MultiProduct<?>
    .collect::<Vec<_>>();
```

```rust
let keys : Vec<(...)> = vec
    .clone() : Vec<Vec<...>, Global>
    .into_iter() : impl Iterator<Item=Vec<...>>
    .cartesian_product( other: vec.into_iter()) : Product<IntoIter<...>, IntoIter<...>>
    .filter(|(v1 : &Vec<?> , v2 : &Vec<?> )| {
        let y0 : &? = v1.get( index: 0).unwrap();
        let y1 : &? = v1.get( index: 1).unwrap();

        let x0 : &? = v2.get( index: 0).unwrap();
        let x1 : &? = v2.get( index: 1).unwrap();
        y0 != x0 && y1 != x0 && y0 != x1 && y1 != x1
    }) : Filter<Product<...>, fn(...) → ...>
    .collect::<Vec<_>>();
println!("keys len: {:?}", keys.len());
```

```
keys len: 400
```

```rust
let n : isize = alphabet.len().pow( exp: 2) as isize;
'l: for (pair1 : Vec<?> , pair2 : Vec<?> ) in keys {
    let y1 : isize = bi_number( bi: pair1.get( index: 0).unwrap());
    let x1 : isize = bi_number( bi: pair1.get( index: 1).unwrap());
    let a : isize = y1 - bi_number( bi: pair2.get( index: 0).unwrap());
    let b : isize = x1 - bi_number( bi: pair2.get( index: 1).unwrap());
    if let Some(keys : HashMap<isize, isize> ) = linear(a, b, n).map(|a : Vec<isize> | {
        a.into_iter() : impl Iterator<Item=isize>
            .map(|a : isize | (a, (x1 - (y1 * a)).rem_euclid( rhs: n))) : impl Iterator<Item=(...)>
            .collect::<HashMap<_, _>>()
    }) {
```

Завдання 4-5

```rust
for (a : isize , b : isize ) in keys {
    if let Some(text : String ) = decode(a, b, &text) {
        println!("a: {}, b: {}", a, b);
        println!("{}", text);
        break 'l;
    }
}
```

Біграми які не зустічаються в російській мові були взяті з [списку]

```rust
let impossible_bi : Vec<&str> = vec![
    "аь", "еь", "иь", "оь", "уь", "юь", "яь", "эь", "ыь", "жы", "шы", "фй", "дй", "хщ", "яы",
    "ьь", "яь", "бй", "эы", "эь",
];
```

a = 199, b = 700

```
a: 199, b: 700
отцеубийствокакизвестноосновноеиизначальноепреступлениечеловечестваиотдельногочеловекавовсякомслучаеоноглавныйисточникчувствавинынеизвестноединственныйлииссследованиямнеудалосьеще
```

Висновок: Завдяки цьому комп’ютерному практикуму нами були здобуті таки навички: знаходження оберненого елементу, розв’язання лінійних рівнянь, атака на афінний шифр за допомогою частотного аналізу біграм.