СИМЕТРИЧНА КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №3

Криптоаналіз афінної біграмної підстановки

Виконали студенти групи ФІ-94
Костюк Кирило і Панасюк Єгор
Варінт-4

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

Постановка задачі
Скачав варіант - побачив шифротекст - проаналізував його - взламав його за допомогою співставлення біграм - перевірив на правильність за допомогою самописного розпізнавача - повторив, якщо текст неправильний

Варіант
щжуяжущпккфшчфбждоцпюдйсвжбэдуэыйэдцмодпмурзфбряцкмдыйдосштцмижбчфип
мугфбзчшохдодвзбряцкмдбэдцхзнощк
яоэоюэтцюзныертзилгфоцбчполфмэдцщкйкшйэысйрэйкчозычфждьмйшотдотзьоюйсщз
оюдууюзсшшстзрэыосяфоешыенывд
ьмиыыяшцрбгянямзюдшскдмыайыяаоешезвжпонорэкжцчжшбчдофшщофбяоязфыщжв
онцеырайхмучмсшывчфвэрфешмяояйывщ
еыйсбжощлзшярфбждоцпюдлвюпщкмзешжзмоуяхямзюдлвзбкзешдбшящксавотзябйкжз
шцопсйкоефтцрзюэдцсшямсканзомы
жуэыыцсшмычмэжглрзщыезскщквкшятоьэйштибяшкочщкфмыйеыйывдьмиыщчвккцо
щеызонорйвкхпшсзунрмоншзоязшяэдхп
езхлсопжипеызохлншплбйщждоыкфоскщквкшягоефоцэзччскщквканвказешюшлцромгл
тдоккжшскзыядншууезжурфешщпнз
шятоужертцлвяхщжпофожущпккшяэывдьмиыйсжусжощккшйжррэсзешьоктдоскыкфот
флцжшвдзылвхзпмжущжеляыцдюппкгф
кшскщквкшяозноюуйэвзхягжжзщрфяоэщпсчкжйэцшвдрйрэйкчофолжыймывдьмиыщчд
орддокыбзлжвочыезыяюйеытяьочмск
мзшядяешмуяхщжбягжрйашайюпмогйжшфшайрмлзннтзхаокшйбчаощяанбччйтжмкжу
чбуфпошфбждоцпюдлвюпюпэзкбтцзопз
аоешйшохзодонофшайсщзожурфмовоцяанфшляйбмуьосклкюнсккжеьзоешшоешоцэжл
ыдяюйеызопыщжфоочсквжаббжнзбляь
хзсккцезшяййсщзоюдьмйшнхдоаоешезвжбяршвдшяполфзятзбжьоиосяйжгоелзурмеыйс
сожзешопхпимсжсказкзшяшйнэюш

шомглтдонзпксзеыэжюпщжхявушйгожурфлцгцншвдрздвщоцыыиеыхзнфылтфаляяыжф
зйквбждэечяыжхыхоцыыиеыыяпомггд
нотлккжжипеызохлщпдоряпзелцджзкзсэлвщпчзгпшсмыжумилцэбтцзохлмофхэыеынетк
зеадьгпуротынщйайкбазущпязхл
дырйпоазсяслщяджипщплзджипюшлцлыбжхяскыосяэищеештцедууьмншйкрзшяцпдвзб
ряцкмдррхфщжэпмуапзчвомощкхыхз
иоюнязхпрэчфлоешщпоцбжщлтзноьобцэжхякзуяяяямзокбмырфзбюжщкяьрйсозыеыйсх
прфеыщчфоефзббжнзтыссжяилнахп
езфщпмшявжядтцйэоцбчазгфьпмушсбэчмиоцяшйдвюптжждйсэйтзмоыптцыцшййычмы
йзхйшмшжшалтыбжхябжюакцопиыщчыд
ншуусйжуопчфюшжзйкмяефопифбкюнзовбюпдокзшярйдуюплвляешууяхщжпонойкып
юшщчмысклзыцбчмялзоцнрряешиыфсхя
даыосябжьоиогфеыхзншзунрюпыяябтцюмюпйшажьосжрэешжзщыцзешйкккшячхдоса
жуюшимйшлыпутцурряешбзкцколппотз
уыайжхжшеыабряязодхпрэчфдяешоцкзвдаямымуайдосшщоччдыозлжцшшйфшщоцьзхл
цюпзхщжщккжюыюпцчзпэыиывдншуушс
ешяоюшбчкзуяяяямзозхьпешьоаоешывмкйыдвбжжзщрэысямяблоцлышсгялаэышйлвмк
сааножутоаонзскккрздвюптжждшсэы
пзьцяделоцлыбжанхмлзннскюдьмоцбжпэсйсщзодбкзвыкшэпдойхдоюаншщкбаекшйбчн
шузябряешйкешзоешчбгяыоиыоцпм
зямодпмучкшйаоешезвжпоновгеыьзрйхесзкбйкьосктлсзешьоекшялцмиажжусжюуэжцы
шсдондпмкзшягожурфлцеызоножя
яоьоэмкзшяпдмыэзгпйшууешоцсаскдондымкзшязплццдлвляудмяйядойккощзшяекшэй
фбждоцпюдлвляскмздбкзцжжущпрф
уяшфсчдвбждчвхеыщчфочытцмиажщквканфшууфиеыхзаоешезвжпонодаыпиыщомзмят
ыямйшалтыеызоешыедвайнинзшязпкц
рфешмяеыцпяовкрфекуяжубждоджгллкпыбжанцйсщзорэкжшяанфшншряязлзфуыйдую
пшсуяпзйкелиавжнрфушйеыюувделдш
чфилюшощжшшйкшшйцомгулщяджипюгпуотсяужзюждмкчкнцжшязцжюяйкбэйканпд
пуыйьмюпйфбждоцпюдлвюпюпэзпшкзхуэж
йуппбзлжфяфохяшфвчшякжядтлоцлыезсочзсыяхщжипляэмнщеычяражуййюзвждвждм
ызхзосшзбкззжокуцеыюпщуыйтодыюп
иызопызвкзмзюдайюдьмиыыяхфщжцфвчшящжюпмуюкжшбчбьщжыйрйшзяошйзоузяж
дчвхеыщчпмщпбкуяяоекшярбптхямзюдеч
рэйкиордиыцпямфочыхордяожзщыезжупмскшяцпсказкзшяллщяанншшкщкпоноюааощ
яекшйбчжучбгяыоиыоцпмяднщжшбчтз
чзкззогяюалэчмиыоцюшяхщжпокбчфнодоздопзузхщжпоьфйказтзрэыосяфощждчвхеых
зжусжфрйктзшясжеьзоешрйэжпзжж
бяаоешывбзлжцшшйфшрэщжсокыйшлцлыксфохямвмуйчжуезаяалжшбчшфссешмяпзю
нзоешедвдвлгфезшйдбряилгфеыхзсккч
вкщыезтлыниоовмушссожзбибзвфвчшяеыабкзтыыймуеызочбюпэзбпифрйбжхяузыпуях
ыщчрзхьэыэявжкщитдоешзхеыхзрэ
ешйчпзюнешибряшяякжшбчфуэжмзчшвдщкпонйсщжшвкьоцпйшбгпутгэййшмштцедзб
бжнзмоошууеыщчдонорзлзджипщчьоцы

ыиеыыявлаомярктяшптцпмдущесзноншшкмокцжшлвждвдрэскалцяекжшбчкожцчибзлж
озномясктзлзмкжшбчшящкбяйбзбяш
жддыцшдзщжэзччамекуяанюзскжуэыощлзшящжбждояоратлынсаскрэууншмяскжупмск
жшбчцдвдвжыглцечмяскскщкбаекжш
бчфшууэжтлмдэйсщжшмощквканбчтзябйкжзшцопсйзоужертцлвяхщжбямэсоеецызбйк
мяюнзоекшвуяджпоьфйказсшлячову
нщеырэтцюзпохпеызомоешдбждсожзбибзлжхыщжыйрйшзяошйуфаляятфсчподояонос
шншмоешдбждтззпсчжшбчншщзнэйсеш
ьовбптдохлжурфбжффюшлцлыксфохявжядтлоцлылвбжзбмушямзешекощеычяратзилгф
бзлжзпвкылоцдуюпиыыяйкныляыфчб
юпповбнзцжшзяойййппифрйщкжэппншйкрзщыайхпжшжшвдщкхйппифрйуяпндощкпор
фссешмяабяопмьосяцызвмуйчмоешдбжд
щуивлвщоефтцрзюэдцсавксшншмоешдбждншайешюшлыбжюуиырафовуьмайтзвжгцрр
сшбжлзмканюакыбзйхдодвууэжкцмэсч
жшсопжипеызозхьпешьомяравжщоипжшешмясжжкйкгшмуайтзфуншяхщжбялчуцеыйс
жулямрчфюшпфмяяявлвжипюпэышбмунр
чфюшьосокыиыхзхпезпыщжмосоьыбжхядамофыюшотдовкккшяабйчуцжелжрбрякывд
юшлвохдошзяобпбжжуэырйбзщтелмяил
щкцжжзщрэысяныблоцлыщемыжучмдубзвфаляяоышйеыюзмзыжйэозкцкогрчфюшажк
жщкгфсймовккцивыйгшьльфжшншмолдоп
сшайскжущпнзшядуайиыалшжпоноюяыкпзсчсрчфюшскюклфоцьидяхфщжщлщяджипб
жюпмуяззощуиврймзвожзпофотывдохлц
юпядайхпимиыраыжнэюшсйокбяжярзьазонырйкоцыыиеыщчжящкбяшзяоьфжяюуйсгдн
шуулвайншопэзцжбкюнзоносочзсыях
щжипхордяожзщызбрякыбзлжкжюпмуяззощуиврйвуйшайподояохлщкбяьшмущжзовказ
хяанаоешезвжбякбмурфоцхпэесопж
ипеыилзэтцчмгнпдрэбтюянзужнепзыжыйсйщкжэгщлцечпфлцйшжбрякыиыхзфшайтцлб
гцабхявыцпяохяупайтзншщзнэйсшк
опншфузхпмдьюшшящкскткллзокрзпмжзешскхыэжазадиыуфужертцлвхзэоскфопбоцщкч
фылидмышкбмщпбкуяяоекзожзуяпо
нзяыншвдщкцждоюшвжитдочзкзжзсыкшкяскыосяпнжцнэохфсфлчжеьзоешэпбжжущчх
ябфбждоцпюдлвямэжглцяекжшскчйфи
бяншкеынтзужертцлвщчэжффйэракбяощзшжаокыиыщчсожзбиеызоузсуьмуяуыжддосш
ншмоешдбждсожзбигцскыкфотфлцаб
гяыовояфяьшмущжвзлжыцмимшшйгшезновжьошйэзэфщзрзмкуягшзбезносожзбиеыы
ядвзбряжзлжипюпоцчбптдохлибвоан
аопоьшйкешзокюывруххнзеявжйэйканэущпзомязоныйфмяцяюакбмумяуысйчбямппый
ыяюдйшлцлыэжмкгфеыйсмофыксюдаб
гяыкаяшябялбгцабхямзюдйсжущжеляыцдсэйканюрщкйкякчодаззешажщзскяптжязджпз
чзшяжкйкгшмускбфсчаоешезвжпо
нопмйкйвюпууэжжйюшряшйешпуьгмоешывбзшхдожйюшряпыбжюшвжйэдвншюпзое
шедншщзнэйсешылбэяоыкжшбччзкзтырйск
понзшясшмышйсщжшзпсчанбчдайкрзшяшйьомршьеыщчуфтцчыщокыкхйшнхдохпцш
шсншешйкцчжшншэзччсжрлязшядяябтцшя

анбчжучмкзшяшйрлщяегдяуярймоаышийшажфямосшайдбмурфшяыжжяочжшбчгявбй
шщчаоешезвжпоноэбкзешдбшярллзджип
юшлцлырэчмзуиыяхскмыуфоцядюпжрчфюшвкжурфлцтжбжюууфиыщчскподояоеыщж
лкешраояазжшжущпщоскскможяскжшбцзв
лвюпеыхзюдншуусйшфкзныбжхяншзогяуяннетюянзашцдияблязныръэтцлыайдбкзешдб
шянфсчтзномофшсжцкгяпзюнамзпея
пыэжйэзпэыгдншуущешфалноыжгллкеыщжуясащуивхзак

Хід роботи

main.cpp

```cpp
#include <iostream>
#include <string>
#include <vector>
#include <cmath>
#include <fstream>
#include <future>
#include <thread>
#include <chrono>
#include <mutex>
#include <functional>
#include "algorithmics.h"

std::mutex read_write_mut;


const std::vector<wchar_t> alphabet_clear =
{

L'а',L'б',L'в',L'г',L'д',L'е',L'ж',L'з',L'и',L'й',L'к',L'л',L'м',L'н',L
'о',L'п',L'р',L'с',L'т',L'у',L'ф',L'х',L'ц',L'ч',L'ш',L'щ',L'ь',L'ы',L'
э',L'ю',L'я'
};

const std::vector<wchar_t> top_rus_letters =
{
   L'а', L'е', L'и', L'о'
};

const std::vector<std::wstring> top_rus_bigrams =
{
   L"ст", L"но", L"то", L"на", L"ен"
};
```

```
const std::vector<std::wstring> blocked_rus_bigrams =
{
   L"аь", L"бй", L"бф", L"гщ", L"еь", L"жй", L"жц", L"жщ", L"жы",
L"йь", L"уь", L"фщ", L"хы", L"хь",
   L"цщ", L"цю", L"чф", L"чц", L"чщ", L"чы", L"чю", L"шщ", L"шы",
L"шю", L"щг", L"щж", L"щл", L"щх", L"щц", L"хь",
   L"щч", L"щш",L"щы", L"щю", L"щя", L"ыь", L"ьы", L"эа", L"эж", L"эи",
L"эо", L"эу", L"эщ", L"эы", L"эь", L"эю",
   L"эя", L"юы", L"юь", L"яы", L"яь", L"ьь"
};
std::wstring dead_souls_text;

class AffinBigramSubstituteEncoder
{
public:
   static std::vector<size_t> encode( const std::wstring& X, size_t a,
size_t b );
   static std::vector<size_t> get_vector_for_encode( std::wstring X );
   static std::wstring get_encoded_text( const std::vector<size_t>& Y
);
};

std::wstring
AffinBigramSubstituteEncoder::get_encoded_text( const
std::vector<size_t>& Y )
{
   std::wstring encoded_text;
   encoded_text.reserve( 2 * Y.size( ) );
   for ( const auto& y_i : Y )
   {
       auto letter = alphabet_clear[ y_i / alphabet_clear.size( ) ];
       encoded_text.push_back( letter );
       letter = alphabet_clear[ y_i % alphabet_clear.size( ) ];
       encoded_text.push_back( letter );
   }

   return encoded_text;
}

std::vector<size_t>
AffinBigramSubstituteEncoder::get_vector_for_encode( std::wstring X )
{
   std::vector<size_t> X_vec;
```

```
    if ( X.size( ) & 1 != 0  )
    {
        X.push_back( L'a');
    }
    X_vec.reserve( X.size( ) / 2 );

    for (size_t i = 0; i < X.size( ) - 1; i += 2)
    {
        auto bigram = X.substr(i, 2);
        auto x_i = get_bigram_number( bigram, alphabet_clear );
        X_vec.push_back( x_i );
    }


    return X_vec;
}

std::vector<size_t>
AffinBigramSubstituteEncoder::encode( const std::wstring& X, size_t a,
size_t b )
{

    std::vector<size_t> Y;
    const auto m = alphabet_clear.size( );
    const auto m_square = static_cast<size_t>( std::pow( m, 2 ) );

    const auto X_vec =
AffinBigramSubstituteEncoder::get_vector_for_encode( X );
    Y.reserve( X_vec.size( ) );

    for ( const auto& x_i : X_vec )
    {
        auto y_i = ( a * x_i + b ) % m_square;
        Y.push_back( y_i );
    }

    return Y;
}


class AffinBigramSubstituteDecoder
{
public:
```

```
    static std::wstring decode( const std::vector<size_t>& Y, size_t a,
size_t b );
    static std::wstring decode( const std::wstring& Y, size_t a, size_t
b );
    static std::vector<size_t> get_vector_for_decode( const
std::vector<size_t>& Y, size_t a, size_t b );
};

std::vector<size_t>
AffinBigramSubstituteDecoder::get_vector_for_decode( const
std::vector<size_t>& Y, size_t a, size_t b )
{
    std::vector<size_t> X;

    const auto m = alphabet_clear.size( );
    const auto m_square = static_cast<size_t>( std::pow( m, 2 ) );
    const auto reverse_a = reverse( a, m_square );

    for ( const auto& y_i : Y )
    {
        auto x_i = reverse_a;
        x_i *= modulo_substitute( y_i, b, m_square );
        x_i %= m_square;
        X.push_back( x_i );
    }

    return X;
}

std::wstring
AffinBigramSubstituteDecoder::decode( const std::vector<size_t>& Y,
size_t a, size_t b )
{
    std::vector<size_t> X_vec = get_vector_for_decode( Y, a, b );
    std::wstring X;
    X.reserve( X_vec.size( ) * 2 );
    for ( const auto& x_i : X_vec )
    {
        X.push_back( alphabet_clear[ x_i / alphabet_clear.size( ) ] );
        X.push_back( alphabet_clear[ x_i % alphabet_clear.size( ) ] );
    }

    return X;
```

```
}

std::wstring
AffinBigramSubstituteDecoder::decode( const std::wstring& Y, size_t a,
size_t b )
{
   auto Y_vec = AffinBigramSubstituteEncoder::get_vector_for_encode( Y
);

   return AffinBigramSubstituteDecoder::decode( Y_vec, a, b );
}

class AffinBigramSubstituteHacking
{
private:
   static void helper_for_brut( const std::wstring& Y, std::wstring&
result, size_t start_pos,
                                size_t finish_pos, std::promise<bool>&
p );
public:
   static bool is_informative_text( const std::wstring& text );
   static std::wstring hack_decode_encoded_text( const std::wstring& Y
);
   static std::pair<size_t, size_t> get_candidates_for_a_b( size_t Y_1,
size_t Y_2,
                                                            size_t X_1,
size_t X_2 );
   static std::wstring bruteforce( const std::wstring& Y );
   static std::wstring bruteforce_optimize( const std::wstring& Y );
};

void
AffinBigramSubstituteHacking::helper_for_brut( const std::wstring& Y,
std::wstring& result, size_t start_pos,
                                               size_t finish_pos,
std::promise<bool>& p )
{
   const auto size_square = static_cast<size_t>( std::pow(
alphabet_clear.size(), 2 ) );
   bool stop = false;
   for (size_t a = start_pos; a < finish_pos; a++)
   {
       if ( gcd( a, alphabet_clear.size( ) ) != 1 )
```

```
        {
            continue;
        }
        for (size_t b = 0; b < size_square; b++)
        {
            std::wstring decoded_text_candidate =
AffinBigramSubstituteDecoder::decode( Y, a, b );
            if( a == 390 && b == 10 )
            {
                std::wcout << decoded_text_candidate << std::endl;
            }
            if ( AffinBigramSubstituteHacking::is_informative_text(
decoded_text_candidate ) )
            {
                wchar_t user_answer = L'n';
                std::lock_guard<std::mutex> guard( read_write_mut );

                std::wcout << decoded_text_candidate << std::endl;
                std::wcout << L"Is this text informative?(y/n) ";
                std::wcin >> user_answer;
                if ( user_answer == L'y' )
                {
                    std::wcout << L"a: " << a << L" " << L"b: " << b <<
std::endl;
                    result = decoded_text_candidate;
                    break;
                }
            }
        }
        if( stop )
        {
            break;
        }

    }
    p.set_value( true );
}

std::wstring
AffinBigramSubstituteHacking::bruteforce_optimize( const std::wstring&
Y )
{
    using namespace std::chrono_literals;
```

```
   const size_t max_thread_count = std::thread::hardware_concurrency();
   const auto size_square = static_cast<size_t>( std::pow(
alphabet_clear.size(), 2 ) );
   const auto one_block_count = size_square / max_thread_count;
   std::vector<std::thread> all_threads;
   all_threads.reserve( max_thread_count );
   std::wstring decoded_text;
   bool text_not_hacked = true;
   std::vector<std::promise<bool>> p;
   p.reserve( max_thread_count );
   std::vector<std::future<bool>> futures;
   futures.reserve( max_thread_count );
   for (size_t i = 0; i < max_thread_count; i++)
   {
       p.push_back( std::promise<bool>() );
       futures.push_back( p[i].get_future() );
       if( i == 0 )
       {
           auto finish = ( i + 1 ) * one_block_count;
           all_threads.push_back(std::thread(
                       &AffinBigramSubstituteHacking::helper_for_brut,
std::ref(Y), std::ref(decoded_text), 1,

finish, std::ref(p[i]) ));
       }
       else if ( i == max_thread_count - 1 )
       {
           auto start = i * one_block_count;
           all_threads.push_back(std::thread(
                       &AffinBigramSubstituteHacking::helper_for_brut,
std::ref(Y), std::ref(decoded_text), start,

size_square , std::ref(p[i]) ));
       }
       else
       {
           auto start = i * one_block_count;
           auto finish = ( i + 1 ) * one_block_count;
           all_threads.push_back( std::thread(
                       &AffinBigramSubstituteHacking::helper_for_brut,
std::ref(Y), std::ref(decoded_text), start,

finish, std::ref(p[i]) ));
```

```
        }

    }

    bool thread_running = true;
    while ( text_not_hacked && thread_running )
    {
        if ( !decoded_text.empty() )
        {
            text_not_hacked = false;
        }
        bool steal_running = false;
        for ( const auto& f : futures )
        {
            auto status = f.wait_for( 0ms );
            steal_running |= ( status != std::future_status::ready );
        }
        thread_running = steal_running;
        std::this_thread::sleep_for( 500ms );
    }

    for ( auto& t : all_threads )
    {
        t.detach( );
    }


    return decoded_text;
}

std::wstring
AffinBigramSubstituteHacking::bruteforce( const std::wstring& Y )
{
    auto size_square = static_cast<size_t>( std::pow(
alphabet_clear.size(), 2 ) );
    for (size_t a = 1; a < size_square; a++)
    {
        if ( gcd( a, alphabet_clear.size( ) ) != 1 )
        {
            continue;
        }
        for (size_t b = 0; b < size_square; b++)
        {
```

```
            std::wstring decoded_text_candidate =
AffinBigramSubstituteDecoder::decode( Y, a, b );
            if ( AffinBigramSubstituteHacking::is_informative_text(
decoded_text_candidate ) )
            {
                wchar_t user_answer = L'n';
                std::wcout << decoded_text_candidate << std::endl;
                std::wcout << L"Is this text informative?(y/n) ";
                std::wcin >> user_answer;
                if ( user_answer == L'y' )
                {
                    std::wcout << L"a: " << a << L" " << L"b: " << b <<
std::endl;
                    return decoded_text_candidate;
                }
            }
        }

    }

    return L"";
}

std::pair<size_t, size_t>
AffinBigramSubstituteHacking::get_candidates_for_a_b( size_t Y_1,
size_t Y_2, size_t X_1, size_t X_2 )
{
    auto alphabet_size = alphabet_clear.size( );
    auto size_square = static_cast<size_t>( std::pow( alphabet_size, 2 )
);
    auto y_sub = modulo_substitute( Y_1, Y_2, size_square );
    auto x_sub = modulo_substitute( X_1, X_2, size_square );

    size_t a = ( y_sub * reverse( x_sub, size_square ) ) % size_square;
    if ( ( a == 0 ) || ( gcd( a, alphabet_size ) != 1 ) )
    {
        return std::make_pair<size_t, size_t>( 0, 0 );
    }

    size_t b = modulo_substitute( Y_1, a * X_1, size_square );

    return std::make_pair<size_t, size_t>( std::move(a), std::move(b) );
}
```

```
void
erase_small_count_bigrams( std::multimap<size_t, std::wstring>&
t_bigrams )
{
   for (size_t i = 1; i < 10; i++)
   {
       t_bigrams.erase( i );
   }

}

std::vector<std::wstring>
get_top_5_bigrams( const std::multimap<size_t, std::wstring>& t_bigrams
)
{
   size_t counter = 0;
   std::vector<std::wstring> top_5_bigrams;
   top_5_bigrams.reserve( 10 );
   for ( auto rit = t_bigrams.rbegin( ); rit != t_bigrams.rend( );
++rit, ++counter )
   {
       if ( counter == 10 )
       {
           break;
       }
       top_5_bigrams.push_back( rit->second );
   }

   return top_5_bigrams;
}

std::wstring
AffinBigramSubstituteHacking::hack_decode_encoded_text( const
std::wstring& Y )
{
   wchar_t user_answer = L'n';
   auto top_bigrams_in_encoded_text = top_bigrams( Y );
   auto top_5_bigrams_in_encoded_text = get_top_5_bigrams(
top_bigrams_in_encoded_text );

   for ( const auto& e : top_5_bigrams_in_encoded_text )
   {
```

```
        std::wcout << e << std::endl;
    }


    for (size_t i = 0; i < top_rus_bigrams.size() - 1; i++)
    {
        auto X_1 = top_rus_bigrams[i];
        auto numeric_X_1 = get_bigram_number( X_1, alphabet_clear );
        for (size_t j = i + 1; j < top_rus_bigrams.size(); j++)
        {
            auto X_2 = top_rus_bigrams[j];
            auto numeric_X_2 = get_bigram_number( X_2, alphabet_clear );

            for( size_t k = 0; k < top_5_bigrams_in_encoded_text.size( )
- 1; k++ )
            {
                auto Y_1 = top_5_bigrams_in_encoded_text[k];
                auto numeric_Y_1 = get_bigram_number( Y_1,
alphabet_clear );
                for( size_t l = k + 1; l <
top_5_bigrams_in_encoded_text.size( ); l++ )
                {
                    auto Y_2 = top_5_bigrams_in_encoded_text[l];
                    auto numeric_Y_2 = get_bigram_number( Y_2,
alphabet_clear );

                    auto a_b_pair =
AffinBigramSubstituteHacking::get_candidates_for_a_b( numeric_Y_1,
numeric_Y_2,

numeric_X_1, numeric_X_2 );
                    if ( a_b_pair == std::make_pair<size_t, size_t>( 0,
0 ) )
                    {
                        a_b_pair =
AffinBigramSubstituteHacking::get_candidates_for_a_b( numeric_Y_2,
numeric_Y_1,

numeric_X_2, numeric_X_1 );
                    }

                    if ( a_b_pair == std::make_pair<size_t, size_t>( 0,
0 ) )
```

```
                    {
                        continue;
                    }

                    auto decoded_text_candidate =
AffinBigramSubstituteDecoder::decode( Y, a_b_pair.first,
a_b_pair.second );

                    if
(!AffinBigramSubstituteHacking::is_informative_text(
decoded_text_candidate ))
                    {
                        continue;
                    }
                    std::wcout << decoded_text_candidate << std::endl;
                    std::wcout << L"Is this text informative?(y/n) ";
                    std::wcin >> user_answer;
                    if ( user_answer == L'y' )
                    {
                        std::wcout << L"[KEY]: a = " << a_b_pair.first
<< " b = " << a_b_pair.second << std::endl;
                        return decoded_text_candidate;
                    }
                }
            }
        }
    }


    return L"";
}

bool
first_criterium_of_informative_text( const std::wstring& text )
{
    constexpr size_t TYPOGRAPHICAL_FACTOR = 20;
    const auto text_bigrams_counter = bigram_count( text );
    size_t sum_of_wrong_bigrams = 0;
    for ( const auto blocked_bigram : blocked_rus_bigrams )
    {
        auto wrong_bigram = text_bigrams_counter.find( blocked_bigram );
        if ( wrong_bigram != text_bigrams_counter.end( ) )
        {
```

```
            sum_of_wrong_bigrams += (*wrong_bigram).second;
        }
    }

    return sum_of_wrong_bigrams <= TYPOGRAPHICAL_FACTOR;
}

bool
second_criterium_of_informative_text( const std::wstring& text )
{
    bool result = false;
    constexpr size_t MINIMUM_TOP_BIGRAM_COUNT = 100;
    auto text_bigrams_counter = bigram_count( text );

    for ( const auto& bigram : top_rus_bigrams )
    {
        result |= ( text_bigrams_counter[ bigram ] >=
MINIMUM_TOP_BIGRAM_COUNT );
    }

    return result;
}

bool
third_criterium_of_informative_text( const std::wstring& text )
{
    constexpr double MINIMUM_TOP_LETTER_COUNT = 20/*%*/;
    auto text_letters_counter = letter_frequency( text );
    size_t sum_of_letters = 0;
    for ( const auto& letter : top_rus_letters )
    {
        sum_of_letters += text_letters_counter[letter];
    }
    auto probability = letter_probability( text.size( ), sum_of_letters
);
    probability *= 100;

    return probability > MINIMUM_TOP_LETTER_COUNT;
}

bool
AffinBigramSubstituteHacking::is_informative_text( const std::wstring&
text )
```

```
{
   bool is_text_informative = first_criterium_of_informative_text( text
);
   is_text_informative |= second_criterium_of_informative_text( text );
   is_text_informative |= third_criterium_of_informative_text( text );
   return is_text_informative;
}


int main()
{
   setlocale(LC_ALL, "");
   #ifdef _WIN32
   system("chcp 1251"); // настраиваем кодировку консоли
   #endif

   std::wstring_convert<std::codecvt_utf8<wchar_t>> converter;
   std::wstring input;
   std::string line;
   std::ifstream fin;

   fin.open( "01.txt", std::ifstream::in );
   while (std::getline(fin, line, '\n') && !line.empty())
   {
       input += converter.from_bytes(line);
   }
   fin.close( );

   auto hack_result =
AffinBigramSubstituteHacking::hack_decode_encoded_text( input );

   std::wcout << hack_result << std::endl;
   return 0;
}
```

algorithmics.h

```
#pragma once
#include <string>
#include <vector>
#include <cctype>
#include <algorithm>
#include <cwctype>
#include <map>
```

```cpp
#include <cmath>
#include <sstream>
#include <locale>
#include <codecvt>

const std::vector<wchar_t> alphabet =
{

L'а',L'б',L'в',L'г',L'д',L'е',L'ж',L'з',L'и',L'й',L'к',L'л',L'м',L'н',L
'о',L'п',L'р',L'с',L'т',L'у',L'ф',L'х',L'ц',L'ч',L'ш',L'щ',L'ъ',L'ы',L'
ь',L'э',L'ю',L'я'
};



const std::map<wchar_t, wchar_t> lowercase =
{
   {L'А',L'а'},
   {L'Б',L'б'},
   {L'В',L'в'},
   {L'Г',L'г'},
   {L'Д',L'д'},
   {L'Е',L'е'},
   {L'Ё',L'е'},
   {L'Ж',L'ж'},
   {L'З',L'з'},
   {L'И',L'и'},
   {L'Й',L'й'},
   {L'К',L'к'},
   {L'Л',L'л'},
   {L'М',L'м'},
   {L'Н',L'н'},
   {L'О',L'о'},
   {L'П',L'п'},
   {L'Р',L'р'},
   {L'С',L'с'},
   {L'Т',L'т'},
   {L'У',L'у'},
   {L'Ф',L'ф'},
   {L'Х',L'х'},
   {L'Ц',L'ц'},
   {L'Ч',L'ч'},
   {L'Ш',L'ш'},
```

```
    {L'Щ',L'щ'},
    {L'Ы',L'ы'},
    {L'Ь',L'ь'},
    {L'Э',L'э'},
    {L'Ю',L'ю'},
    {L'Я',L'я'},
    {L'а',L'а'},
    {L'б',L'б'},
    {L'в',L'в'},
    {L'г',L'г'},
    {L'д',L'д'},
    {L'е',L'е'},
    {L'ё',L'е'},
    {L'ж',L'ж'},
    {L'з',L'з'},
    {L'и',L'и'},
    {L'й',L'й'},
    {L'к',L'к'},
    {L'л',L'л'},
    {L'м',L'м'},
    {L'н',L'н'},
    {L'о',L'о'},
    {L'п',L'п'},
    {L'р',L'р'},
    {L'с',L'с'},
    {L'т',L'т'},
    {L'у',L'у'},
    {L'ф',L'ф'},
    {L'х',L'х'},
    {L'ц',L'ц'},
    {L'ч',L'ч'},
    {L'ш',L'ш'},
    {L'щ',L'щ'},
    {L'ы',L'ы'},
    {L'ь',L'ь'},
    {L'э',L'э'},
    {L'ю',L'ю'},
    {L'я',L'я'}
};


std::wstring
delete_rubish(const std::wstring& input) {
```

```
    std::wstring result;
    for (const auto& c : input) {
        if (lowercase.find(c) != lowercase.end()) {
            result.push_back(c);
        }
    }
    return result;
}

std::wstring
lower_case(std::wstring input) {
    input = delete_rubish(input);
    for_each(input.begin(), input.end(), [&](wchar_t& c) {c =
lowercase.at(c); });
    return input;
}



size_t
numeric_difference_between_letters( size_t y, size_t x )
{
    if ( y >= x )
    {
        return y - x;
    }
    else
    {
        auto buf = x - y;
        return alphabet.size( ) - buf;
    }
}

size_t
modulo_substitute( size_t y, size_t x, size_t mod )
{
    if ( y >= x )
    {
        return ( y - x ) % mod;
    }
    else
    {
        auto buf = ( x - y ) % mod;
        return mod - buf;
```

```
    }
}


auto
bigram_count(const std::wstring& input) {
    std::map <std::wstring, size_t> bigram_counter;

    for (int i = 0; i < input.length() - 1; i += 2) {

        ++bigram_counter[input.substr(i, 2)];
    }

    return bigram_counter;
}

auto
top_bigrams( const std::wstring& input )
{
    const auto bigram_counter = bigram_count( input );
    std::multimap< size_t, std::wstring > top_of_bigrams;
    for ( const auto& p : bigram_counter )
    {
        top_of_bigrams.insert( { p.second, p.first } );
    }

    return top_of_bigrams;
}

std::vector<size_t>
dividers( size_t r )
{
    std::vector<size_t> divs;
    divs.reserve( std::sqrt( r ) );
    for (size_t i = 2; i <= r; i++)
    {
        if ( r % i == 0 )
        {
            divs.push_back( i );
        }

    }
```

```
    return divs;
}

size_t
numeric_value_of_letter( wchar_t l )
{
    auto alphabet_position =  std::find( std::begin( alphabet ),
std::end( alphabet ), l );
    auto val = std::distance(  std::begin( alphabet ), alphabet_position
);
    return val;
}

size_t
numeric_value_of_letter( wchar_t l, const std::vector<wchar_t>&
custom_alphabet )
{
    auto alphabet_position =  std::find( std::begin( custom_alphabet ),
std::end( custom_alphabet ), l );
    auto val = std::distance(  std::begin( custom_alphabet ),
alphabet_position );
    return val;
}

auto letter_frequency(std::wstring input) {
    std::map <wchar_t, size_t> let_counter;
    for (const auto& c : input) {
        ++let_counter[c];
    }

    return let_counter;
}

double letter_probability( double text_size, double letter_count )
{
    return letter_count / text_size;
}

int gcd(long long a, long long b) {//Алгоритм Стейна
    int d = 1;
    while ((a % 2 == 0) && (b % 2 == 0)) {
        b /= 2;
        a /= 2;
```

```
        d *= 2;
    }
    while (a % 2 == 0) {
        a /= 2;
    }
    while (b != 0) {
        while (b % 2 == 0) {
            b /= 2;
        }
        if (a <= b) {
            a = a;
            b = b - a;
        }
        else {
            int b1 = b;
            b = a - b;
            a = b1;
        }
    }
    d *= a;
    return d;
}

//Написанно по псевдокоду из Википедии
long long reverse(long long a, long long n) {
    if (a >= n)a %= n;
    long long t = 0, r = n, newt = 1, newr = a;
    while (newr != 0) {
        long long quotient = r / newr;
        long long newt1 = t;
        t = newt;
        newt = newt1 - quotient * newt;
        long long newr1 = r;
        r = newr;
        newr = newr1 - quotient * newr;
    }

    if( t < 0 )
    {
        t += n;
    }

    return t;
```

```
}

size_t
get_bigram_number( const std::wstring& bigram, std::vector<wchar_t>
custom_alphabet )
{
   if ( bigram.size( ) != 2)
   {
       throw( "error value" );
   }

   auto x_i = numeric_value_of_letter( bigram[0], custom_alphabet );
        x_i *= custom_alphabet.size( );
        x_i += numeric_value_of_letter( bigram[1], custom_alphabet );

   return x_i;
}
```

Проаналізували текст, виявили найчастіші біграми. Багато труднощів виникли з розпізнаванням коректності тексту(дякую типографічному шуму в тексті), вирішені вони були за допомогою додавання декількох критеріїв коректності. Також виникла проблема з оберненим елементом за модулем, він повертав від'ємне число, хоч і правильне, вирішено було за допомогою дебагінгу і костиля з if-ом. Також був написаний брутфорс(як тупий, так і тупий-потоковий), він майже не використовувся та й особливо не допоміг в дебагінгу, але він є)

Топ біграм шифротексту(тут більше 5-ти, але отак)

еш
шя
до
еы
ск
ое
жу
жш
нш
щж

Запропонований розпізнавач коректності тексту рос. мовою

Було 3 версії розпізнавача.

Версія 1

Беремо заборонені біграми(тобто ті, які не зустрічаються в мові: L"аь", L"бй", L"бф", L"гщ", L"еь", L"жй", L"жц", L"жщ", L"жы", L"йь", L"уь", L"фщ", L"хы", L"хь", L"цщ", L"цю",

L"чф", L"чц", L"чщ", L"чы", L"чю", L"шщ", L"шы", L"шю", L"щг", L"щж", L"щл", L"щх", L"щц", L"хь", L"щч", L"щш",L"щы", L"щю", L"щя", L"ыь", L"ьы", L"эа", L"эж", L"эи", L"эо", L"эу", L"эщ", L"эы", L"эь", L"эю", L"эя", L"юы", L"юь", L"яы", L"яь", L"ьь") рахуємо їх кількість в тексті, якщо вона більше за вказану похибку типографічних помилок(значення вказується в фукції), тобто ми одразу перестраховуємося на те, що у тексті можуть бути типографічні помилки, а отже, і такі біграми ми можемо зловити, тому такий спосіб є більш гнучким ніж просто бракувати текст з хоч одною такою біграмою

Версія 2

Після того як перша версія не спрацювала розпізнавач доповнився новим критерієм. Якщо перший або другий критерій спрацьовує - текст вважається коректним. Сенс другого критерію лежав в оберненому методі - порахувати кількість біграм, які найчастіше можна зустріти в мові(L"ст", L"но", L"то", L"на", L"ен"). Після аналізу “Мертвих душ” Гоголя, було вирішено, що їх к-сть не повинна бути не меншою за 100(цей коефіцієнт може змінюватися в залежності від розміру тексту).

Версія 3

Після не працюючої версії 2 було вирішено додати ще один критерій. Якщо хоча б один з критеріїв спрацьовує, то текст можна вважати коректним. Цей метод базується на підрахунку к-сті найчастіших букв мови(L'а', L'е', L'и', L'о'), які лежать в тексті. Якщо ця кількість менша за вказаний відсоток, то текст вважається некоректним. Відсоток був вибраний 20% після дослідження інтернетних досліджень(во як, в інтернетах було 24-27%).

Оскільки версії 4 немає, то ж версія 3 спрацювала, і після нормальної такої к-сті неспрацювань маємо доволі універсальний розпізнавач коректності тексту.

Розшифрований текст

[KEY]: a = 390 b = 10

еслиправдачтодостоевскийвсибиринебылподверженприпадкамтоэтолишьподтверждает точтоегоприпадкибылиегокацизбчфнэффыььювэжшяцмофпшвгукзюкржтмутебййцхумв яяепягсащщмьощкьэьщчфийцечфирщфбйжтхэпхинкйчдыющнфьигхыазаниидляпсихич ескойэкономиидостоевскогообясняетсяточтоонпрошелнесломленнымчерезэтигодыбедс твийиунижйюамтпщоьушттхнпокаллфяумфзжжзтюсслзмричозофяууеуатюпкгтнжыйцг ьэаивмясштуоюйраыюрэххофюкзйнигйчылефзшыпринялэтонезаслуженноенаказаниеот батюшкицарякакзаменунаказаниязаслуженногоимзасвойгрехпоотношениюксвйюечпйи втшпююеочкьмвэпсжокхнпдкйежьгутуцыюмодбзяьйежьгчыхгчытюыфзуххфзлущктаяс ойреиибьррююдсьхсючжшьрыниеопсихологическомоправданиинаказанийприсуждаем ыхобществомэтонасамомделетакмногиеизпреступниковжаждааойежьгутроыьчфгмыгеч уцрскжедрожжюйгадмтлнгфхцхумвяюдгушнтзигащиштжыкшхзктацпщумгряьюмнсфщ ющэющрягйлычениеистерическихсимптомовпойметчтомыздесьнепытаемсядобитьсясм ыслаприпадковдостоевскогововсейполнотеуыыюемьоиелкеьюдкыийценсьхкзйгалаыкдз уьмхзжщкйфуйфцйдаяюкеыюжтыюьодбжуккксфщююеивтжбдцоьйьгйышпиьопоиеыиен аслоенияможносказатьчтодостоевскийтакникогдаинеосвободилсяотугрызенийсовестив связиснамерениемубитптфзлущкеьявыйтюжбьфзжсчггдтчцнсьхшуццвьнгфхвкижалфн

жюнчыьлнаэыыоипнбдчмьгосгбаяпнсчнчалфнжюфнввфзвцхыгосударственномуавтори
тетуикверевбогавпервойонпришелкполномуподчинениюбатюшкецарюоднаждыразыгра
вшемушкчхжвжыбйдйшжектшзжцувктшлфзуауцпсчдйзнкйббюжлюяэшюиржщюьемкх
штщлякуещхщлжзфххчйжкбизиытещгфзужачнчрыбольшесвободыоставалосьунеговобл
астирелигиознойпонедопускающимсомненийсведениямондопоследнейминутысвйюфда
вьтггмицэюмйофпцырцщщиздттпгяишалытйкмфоэрсэмвбшэлзхноэтнятямпяонэекмшзс
скыаэцпсчспьфаауогштжжнхыоторымприводитверавиндивидуальномповторениимиров
огоисторическогоразвитияоннадеялсявидеалехристанайтишкгнкйспзщпшщцсюнчыьгез
ьяфйнгиыэффрффмкызщкйыхоюосскжялтюемдауткяфккйьсщфушщмзузвээйраххюыпйа
пагуывьэрычномсчетенепришелксвободеисталреакционеромтоэтообясняетсятемчтооб
щечеловеческаясыновняявинанакоторойййюничфыпгагргигичеяенщпюищжыыющнйзтм
ккмпрскжобййозспыоаубтфцэуфббтбуйзыйуыщсээжыеьбйдакхвкмфоэрсютвбшытеллек
туальностьюздесьнасказалосьбыможноупрекнутьвтомчтомыотказываемсяотбеспристра
стностипсихоанализмеозлкизтяутдыыюаюаглвеьйгдюгщпхмчщййавмхщкймэпйтюищы
юутэусзрырсмйьфйбглбтжкеьдвлаштуцнсьхшушююееьнсуыровоззренияконсерваторст
албынаточкузрениявеликогоинквизитораиоценивалбыдостоевскогоиначеупрексправед
щрытайыкмпокщгеютжодрящэсзрыцжьгчыкдрраджюнчэыыюаюаглвеьлкгнгштжфзлзф
эиищмкыаэьююеыющнькьыгьпьштшежацыедствиеневрозаедвалипростойслучайность
юможнообяснитьчтотришедеврамировойлитературывсехвременттрактуютюяешэфьоесж
тражтюлфзфршжектшлфрвшошйжрвеусрзвяцьксницльфьмйпумчыфвдвчнвщрффмнпок
аллфяумфлфьбявцаечйбэыкрываетсяимотивдеяниясексуальноесоперничествоиззаженщ
иныпрямеевсегоконечноэтопредставленовдрамеосновэшюеивюезалзофвыхфащишфнич
йжгляггтнжьфизщфясыпипяжшнцхатаивтэемраиокщгеютжшвмжлпюргиыюуттхйвэчоз
офчюкыобработканевозможнаоткровенноепризнаниевнамеренииубитьотцакакогомыдо
биваемсяприпсихоанализекажетсянеюяйахещцсфругяишхоффниэжлвтасяььлкяетотебеэ
жлвдщцмчысэпхинкйчдужокщгеюнчдсмйбдиммэштдйяюкеыюэхеблвдсэыкидостигаетс
ятемчтобессознательныймотивгероярпроецируетсявдействительностькакчуждоеемуприн
уждениенавяцгежьюсэьнышщойздтейнсопвпагягттхюеыуасэфнтзююююеряшфьбопгшохх
псчжрмнаричтжйхечтбжбггхазшщкыюгееюнчпхмыстоятельствпринимаетсяврасчеттакк
аконможетзавоеватьцарицуматьтолькопослеповторениятогожедействиявотнгнжюфнггл
ыфзтхлцытоэевдкхжщйыьеьжтжзтаксзувсегфхэйчыелхюясыпганснявпщущоыьфздймэ
щулзясыпттнжйпьмаекиюыокснятьееессебявзвалитьеенапринуждениесосторонысудьбы
наоборотвинапризнаетсяикаквсецелаявинанаказываетеподррйбюэщзгбийесшзужьгчыд
плэаивмясштуоюйэемряювэауюиэжьпозэйыюплюевмжзсфлеыюутбреклвдщцмчыбйзук
яйгыыаженоболеекосвеннопоступоксовершаетсянесамимгероемадругимдлякоторогоэт
отпоступокнеявляетсяотцеубийстсщйгйвщкочтсьхнпбднфзуаусашдочггмицэыоаувумпв
ехздтозтюаюиоуюпятхиэжшяцнпщувеэгзныоикщфгшфньмцьвупнжяьыовкомплексгероя
мывидимкакбывотраженномсвететаккакмывидимлишьтокакоедействиепроизводитнаге
рояпоступокбжсшчуижыюрэххофюкзйцгыййьшзыюпкосбькыщюыфзуцпьфсшчхлдйгж
щяышэагэцххсйыпжулзчыдфаацпчмфклкбрйбхасзэйкыетсобственноечувствовинывсоот
ветствиисхарактеромневротическихявленийпроисходитсдвигичувствовиныперехюянфи
щкйтаошнчечпюявтжчзкйдщцпсчеквзюдцфьощкйгдаутаывмэркстуыхщхмаигуфзулофкв
таичплпрсйбчххпкжкющзйшяыкаксверхиндивндуальнуюонпрезираетдругихнеменеече
мсебяеслиобходитьсяскаждымпозаслугамктоуйдетотпоркивчйкыийясхжшьштекиынше

ызтлвеьчфпнщвбйьйчянфоййффумобщхкичйжцжшжектшлпьфизлихеужсшнхтзагнсявю дьэигтомуцыовекомсвязаннымсубитымтакимижесыновнимиотношениямикакигеройдм итрийукоторогомотивсексуальногосоперничбжтшвчшхеинскжюещхмацпщухаьфизлихе ужсшнхзггмкыжвлкйиочшжйпющшзхжвейгчыечаытыыюаюагьпмахзьхмолчьбщйивэыт веннуюболезньякобыэпилепсиютемсамымкакбыжелаясделатьпризнаниечтомолэпилепт икневротиквомнеотцеубийцадошьвдбесвдкгфжтнжыймэиунокхявжстюойачжбйыэылшл рвряювэаугчюбкйпдивчоржвялнфшрзбтйпмжлпюргиыюренсгрввщылепнотаккакстоитв сеэтоперевернутьинаходишьглубочайшуюсущностьвосприятиядостоевскогозаслуживае тнасмешепяьцаензэгряювэауижчббдцмлхаакиужштйытаоштжьйнсопжююемрбйжщерж рмсзккййьшзыюпкосжйнсопдцзйзнпдьэшугыьысихологияинтересуетсялишьтемктоегов своемсердцежелаликтопоегосовершенииегоприветствовалипоэтомувплотьуыивдщемн югетаыннхрчцьтюггдмумчыубхжтефзшееьамкзщлсфаахзцзжрййоякибвйнунсфэмзуауж бчзлцсюампзюдсатфвыьысисациникиэпилептическийпреступниквбратьяхкарамазовыхе стьсценаввысшейстепенихарактернаядлядостоевскотьивщмтммфарйбрсхфэхытщюмвж щэзыющнмпштфкнсхфэхввцпьфнгдмюкщьыфтеыюаюбьчжыьумцющнэпыихнясыпбяйа сэчхийлыколениэтонеможетвлятьсявыражениемвосхищенияадолжноозначатьчтосвято йотстраняетотсебяискушениеисполниуыдпюстялаштытцхшжхюукжртхтзжыбтлмкыбдо зкйкыйкхзьхчтпнрсроюсыпкцггвмчжцыыюаюаглвеьзбуатюпкгттхиувктшныительнобез граничнаонадалековыходитзапределысостраданиянакотороенесчастныйимеетправоона напоминаетблагрумфпюнчофбьррмйатиаыьцпсчяьгещцжряьиюжфяьэыдллшспоиыьчхн эауеыйкуатюпкгтяьайлэыьфзюдойгллцввофтычепюуыйнасебявинукоторуювдругомслуч аенеслибыдругиеаазы

Висновок

Весела лаба, дуже цікава, витратили 2 тижні, але нам сподобалось. І звісно, набули навички частотного аналізу на прикладі розкриття моноалфавітної підстановки; опанували прийоми роботи в модулярній арифметиці.