**Cryptography and Coding Homework 2**

**Vedasagar Karthykeyan (vkarthy1)**

2.

function [plaintext]=crackhill(snippetplaintext,ciphertext,blocklength)

% Decrypts a hill cipher given a plaintext snippet,

%   ciphertext and the block length

y=ciphertext;

hill\_cipher=y-97;

x=snippetplaintext;

plain=block(x,1,blocklength)-97;

for i=1:length(plain)-blocklength+1;

    plaineve=plain(:,i:i+blocklength-1);

    ciphertext=block(y,1,blocklength)-97;

    ciphereve=ciphertext(:,i:i+blocklength-1);

    d=det(plaineve);

    d=round(d);

    if mod(d,2) ~= 0

        recipd=powermod(d,-1,26);

        plaineveinverse=recipd\*d\*inv(plaineve);

        plaineveinverse=mod(plaineveinverse,26);

        eveKey=mod(ciphereve\*plaineveinverse,26);

        eveKey=round(eveKey)

        det\_key=det(eveKey);

        det\_key=round(det\_key);

        recipd\_key=powermod(det\_key,-1,26);

        key\_inv=recipd\_key\*det\_key\*inv(eveKey);

        key\_inv=round(key\_inv);

        key\_inv=mod(key\_inv,26);

        plaintext\_decrypt=mod(key\_inv\*ciphertext,26);

        plaintext = block(char(plaintext\_decrypt+97),-1,blocklength);

        return

    end

end

end

**Diary:**

hillciphertexts;

crackhill(someofplain1, hillciphertext1, 4)

eveKey =

10 9 18 4

10 11 13 18

17 15 20 25

21 14 12 20

ans =

thisisjustthebeginningofaverylonghillencryptedmessagewouldntyouliketoknowthewholethingfourscoreandsevenyearsagoourfathersbroughtforthonthiscontinentanewnationconceivedinlibertyanddedicatedtothepropositionthatallmenarecreatedequalnowweareengagedinagreatcivilwartestingwhetherthatnationoranynationsoconceivedandsodedicatedcanlongendurewearemetonagreatbattlefieldofthatwarwehavecometodedicateaportionofthatfieldasafinalrestingplaceforthosewhoheregavetheirlivesthatthatnationmightliveitisaltogetherfittingandproperthatweshoulddothisbutinalargersensewecannotdedicatewecannotconsecratewecannothallowthisgroundthebravemenlivinganddeadwhostruggledherehaveconsecrateditfaraboveourpoorpowertoaddordetracttheworldwilllittlenotenorlongrememberwhatwesayherebutitcanneverforgetwhattheydidhereitisforusthelivingrathertobededicatedheretotheunfinishedworkwhichtheywhofoughtherehavethusfarsonoblyadvanceditisratherforustobeherededicatedtothegreattaskremainingbeforeusthatfromthesehonoreddeadwetakeincreaseddevotiontothatcauseforwhichtheygavethelastfullmeasureofdevotionthatweherehighlyresolvethatthesedeadshallnothavediedinvainthatthisnationundergodshallhaveanewbirthoffreedomandthatgovernmentofthepeoplebythepeopleforthepeopleshallnotperishfromtheearthgettysburghaddressabrahamlincolna

crackhill(someofplain2, hillciphertext2, 4)

eveKey =

10 8 21 21

12 12 24 15

23 15 15 18

15 4 0 2

ans =

canyoudecrypttherestofthefollowingmessagethatimakehiccupandjustalittlenoisetheresaladywhossureallthatglittersisgoldandshesbuyingastairwaytoheavenandwhenshegetstheresheknowsifthestoresareclosedwithawordshecangetwhatshecamefortheresasignonthewallbutshewantstobesureandyouknowsometimeswordshavetwomeaningsinthetreebythebrooktheresasongbirdwhosingssometimesallofourthoughtsaremisgiventheresafeelingigetwhenilooktothewestandmyspiritiscryingforleavinginmythoughtsihaveseenringsofsmokethroughthetreesandthevoicesofthosewhostandlookinganditswhisperedthatsoonifweallcallthetunethenthepiperwillleadustoreasonandanewdaywilldawnforthosewhostandlongandtheforestwillechowithlaughteranditmakesmewonderiftheresabustleinyourhedgerowdontbealarmednowitsjustaspringcleanforthemayqueenyestherearetwopathsyoucangobybutinthelongruntheresstilltimetochangetheroadyoureonyourheadishumminganditwontgobecauseyoudontknowthepiperscallingyoutojoinhimdearladycantyouhearthewindblowanddidyouknowyourstairwayliesonthewhisperingwindandaswewindondowntheroadourshadowstallerthanoursoulstherewalksaladyweallknowwhoshineswhitelightandwantstoshowhoweverythingstillturnstogoldandifyoulistenveryhardthetunewillcometoyouatlastwhenallareoneandoneisalltobearockandnottorollstairwaytoheavenledzeppelina

crackhill(someofplain3, hillciphertext3, 5)

eveKey =

14 9 7 24 12

3 21 23 1 7

13 1 2 6 17

3 15 1 25 24

20 24 6 5 19

ans =

letstrythisoutforsomemorefunandseeiftherestisreadablejustaliitlemoretwasbrilligandtheslithytovesdidgyreandgimbleinthewabeallmimsyweretheborogovesandthemomerathsoutgrabebewarethejabberwockmysonthejawsthatbitetheclawsthatcatchbewarethejubjubbirdandshunthefrumiousbandersnatchhetookhisvorpalswordinhandlongtimethemanxomefoehesoughtsorestedhebythetumtumtreeandstoodawhileinthoughtandasinuffishthoughthestoodthejabberwockwitheyesofflamecamewhifflingthroughthetulgeywoodandburbledasitcameonetwoonetwoandthroughandthroughthevorpalbladewentsnickersnackheleftitdeadandwithitsheadhewentgalumphingbackandhasthouslainthejabberwockcometomyarmsmybeamishboyofrabjousdaycalloohcallayhechortledinhisjoylewiscarroljabberwockyaaa

crackhill(someofplain4, hillciphertext4, 7)

eveKey =

3 13 17 3 18 22 20

16 4 8 4 14 19 15

24 5 7 23 24 9 4

21 5 18 3 8 18 26

24 16 24 16 18 4 7

8 1 3 6 24 24 25

6 5 1 5 15 5 24

ans =

thisplaintextwillbeencryptedwithasevenbysevenhillcipherwhichmakesthiswholethingjustalittlemoreinterestingsoiwillwritesomemorethingshereandletmeaddjustafewsentencesthefollowingisbyjimlitkeinjuriesareanoccupationalhazardineverybigtimesportbutsomuchsoinskiingthatkildowwasnttheonlyathleteinwednesdaysracewhocamebackfromahorrificcrashinmondaystrainingsessiontocompeteshewasnteventheonlyathletetakenofftheslopeinahelicopterforemergencymedicaltreatmentwhohealedfastenoughtotakeanothershotfourskiersinallcrashedduringtrainingontheslickwatersoakedartificialsnowandonlycanadianallisonforsythwhotorekneeligamentscalleditanolympicselisabethgoerglofaustriawhomadeitdowntothebottomofthetrainingrununderherownpowercrashedasecondtimewednesdaycarolemontilletcarlesoffrancewhowastakenbyhelicoptertoamedicalclinicinnearbysestriereandshowedupatthestartlinewithafacesobruisedthatshelookedlikeaboxerbravelyskiedtothplacekildowsperformancethoughmayhavebeeneventougherinmondaystrainingrunshewasgoingsofastthathermomentumfirstsplayedherskisandthenlaunchedhertumblingthroughtheairbythetimeshelandedlookingbackwardandbangedtoastopeverybodyonthehillfearedtheworstallirememberisbeinginmytuckandflyingkildowrecalledandlookingbackupatthegateijustwentoverincrediblyforallthepaincausedbysomuchrollingandtumblingabruisedhipwaskildowsmostsevereinjuryandjustaboutthetimeshegotintouchwithalltheotherbatteredpartsofherbodyinahospitalinturinolympicchampionpicabostreetwasatherbedsidestreetwhoknowsathingortwoaboutcrasheshadbeentherolemodelforkildowscareerandshesoonbecameaninspirationaswelltheytalkedthencriedthensettleddownandbeganplottingareturntotheslopestheprotegelikedtheplansomuchthattuesdaymorningwhilethehospitalstaffworkedonherdischargepaperskildowslippedonherclothesgatheredherbagsandmadeabreakforittheyleftmealoneforanhourandahalfandiwasinahurrytogetbackandtrainshesaidibarelymadeitoutoftheroomwhensomeoneranupyellednonoyoumustwaithereandputmebackintheroomkildowsaidbummerbutshedidnthavetowaitmuchlongeradayofrestinsestrierepersuadedhertomakeonefinalpreracetrainingrunwednesdaymorningpainwassomethingiwasgonnatakenomatterwhatkildowsaiditwasjustamatterofifmybodycouldphysicallywithstandtheforcesofthespeedbutasshebarreledoutofthebumpywoodlandzoneabouttwothirdsofthewaydownthehillandintothelongfinalslopethistimeforkeepssherealizedthepainshehadtodealwithitwasntonlyphysicalayearagokildowwasoneofahandfuloftopskierswhotoldofficialsthesansicariocoursewastooboringfortheolympicsthuschallengedthoseofficialsbulldozedthecoursetomakebiggerjumpsslickerplateausandsteeperlandingzonesaa

diary off

3. For the substitution cipher I implemented the Markov Chain Monte Carlo method of Metropolis-Hastings Algorithm. I used a scoring function to count the number of real English words present in the decrypted plaintext using a particular combination of key. The key with the best score was selected. Then I analyzed the plaintext and corrected a few incorrect substitutions manually and arrived at the final answer.

I got the dictionary of the keys, i.e. the English words and their values i.e number of occurrences of those words in the book Moby Dick, from another implementation done in Python (<https://github.com/jwcarr/MCMC-Cipher-Solver)>

First I tried to create the dictionary using the entire text of War and Peace and extracted the top bigrams, trigrams and quadgrams using the FindMostFreq program provided to us. But the result was not as good as using the Mody Dick dictionary as it was larger and also, extracting number of occurrences of many English words from War and Peace was not computationally feasible. So, I opted for the already extracted dictionary.

For the initial key, instead of using a random shuffling I used the knowledge that the top 4 most frequently occurring letters have been mapped directly. So the initial key used by me was substitutions of all 26 letters based on frequencies. So the substitutions for first 4 most frequent letters (e, t, a, o) were correct but the rest were not in place.

Then I tried many swaps of the key (5000) and calculated the score for each swap. If the score was better than the previous swap, then the new key was made the key and the best score was updated. If the new score was same as the old score, meaning any of the 2 keys might be correct, I do a coin toss and update the key. I repeat this process for several tries and arrive at the best key.

The scoring functions is as follows. After swapping two letters, I substitute the key in the ciphertext. Then I count the number of occurrences of each English word in the dictionary and multiply it with the value of that key. The more correct a key is, the higher will be its score as the decrypted text will contain more number of English words.

After this, I looked at the decrypted text and identified the incorrect words and made the swaps in the key manually. This had to be done just around 5-6 times before arriving at the final key.

Final answers:

**key for cipher 1 = ’ukpesahcdyrizbljwxfnmotvgq’**

**key for cipher 2 (extra credit) = 'shbpjtazcuqomgrnwdvxiflkey'**

The following are the functions implemented:

**hw2\_3.m**

clear all;

clc;

load('substitutioncipherexample.mat');

%challengesubstitutioncipher;

[key, plaintext] = decrypt\_substitution(ciphertext);

key

plaintext

**decrypt\_substitution.m**

function [ key, plaintext ] = decrypt\_substitution( ciphertext )

[f, relf]=zfrequency(ciphertext);

[B, index]=sort(relf, 'descend');

indexes = ['e'; 't'; 'a'; 'o'; 'i'; 'n'; 's'; 'h'; 'r'; 'd'; 'l'; 'c'; 'u'; 'm';...

    'w'; 'f'; 'g'; 'y'; 'p'; 'b'; 'v'; 'k'; 'j'; 'x'; 'q'; 'z'];

order = char(index+96)';

key = '\*';

for j=1:25

    key = strcat(key, '\*'); %  Initital key

end

c = 1;

for ind = index'

    key(ind) = indexes(c); % Build key based on the relative frequencies

    c = c+1;

end

key = search(ciphertext, key, 5, 5000); %   Search for key with 5 trials and 5000 swaps (20000 for extra credit cipher)

plaintext = substitute(key, ciphertext);

end

**search.m**

function [ bestKey ] = search( ciphertext, key, trials, swaps )

bestScore = 0;

for i=1:trials

    bestTrialScore = 0;

    for j=1:swaps

new\_key = swap(key, 1, 26);

        new\_score = score(new\_key, ciphertext); %   Calculate score

         if new\_score > bestTrialScore

            key = new\_key;

            bestTrialScore = new\_score; %   Update key and score

        elseif new\_score == bestTrialScore

            if rand(1,1) > 0.5 %Coin toss

                key = new\_key;

            end

        end

    end

        if bestTrialScore > bestScore

        bestKey = key;

        bestScore = bestTrialScore;

    end

end

**score.m**

function [ scr ] = score( key, ciphertext )

keySet = {'our', 'all', 'just', 'nd', 'over', 'one', 'en', 'go', 'cr', 'ng', 'its', 'make', 'ld', 'le', 'from', 'would', 're', 'there', 'had', 'two', 'been', 'him', ...

valueSet= {20941, 2828, 123, 10732, 646, 1640, 8647, 1046, 1269, 9771, 548, 157, 2361, 8012, 1039, 427, 11436, 802, 814, 289, 408, 1267, 7020, 372, 29424, 4866, 6775, ...

mapObj = containers.Map(keySet,valueSet);

candidate = substitute(key, ciphertext);

scr = 0;

for k=mapObj.keys'

    scr = scr + count(candidate, k) \* mapObj(k{1});

end

end

**buildMap script used for building the dictionary from War and Peace but not used finally**

english\_text\_2; %Entire text of war and peace

[bigrams, mostbigrams ] = FindMostFreq(plaintext, 2, 30);

[trigrams, mosttrigrams] = FindMostFreq(plaintext, 3, 20);

[quadgrams, mostquadgrams] = FindMostFreq(plaintext, 4, 10);

keySet = [bigrams(:, 2)' trigrams(:, 2)' quadgrams(:, 2)'];

valueSet = [[bigrams{:, 1}] [trigrams{:, 1}] [quadgrams{:, 1}]];

mapObj = containers.Map(keySet,valueSet);

filename = 'dictionary.mat';

save(filename, 'keySet', 'valueSet', 'mapObj');