1. The best decryption is:

WHEN I FIND MYSELF IN TIMES OF TROUBLE MOTHER MARY COMES TO ME SPEAKING WORDS OF WISDOM LET IT BE AND IN MY HOUR OF DARKNESS SHE IS STANDING RIGHT IN FRONT OF ME SPEAKING WORDS OF WISDOM LET IT BE LET IT BE LET IT BE LET IT BE LET IT BE WHISPER WORDS OF WISDOM LET IT BE AND WHEN THE BROKEN HEARTED PEOPLE LIVING IN THE WORLD AGREE THERE WILL BE AN ANSWER LET IT BE FOR THOUGH THEY MAY BE PARTED THERE IS STILL A CHANCE THAT THEY WILL SEE THERE WILL BE AN ANSWER LET IT BE LET IT BE LET IT BE LET IT BE LET IT BE YEAH THERE WILL BE AN ANSWER LET IT BE LET IT BE LET IT BE LET IT BE LET IT BE WHISPER WORDS OF WISDOM LET IT BE LET IT BE LET IT BE LET IT BE LET IT BE WHISPER WORDS OF WISDOM LET IT BE AND WHEN THE NIGHT IS CLOUDY THERE IS STILL A LIGHT THAT

The corresponding P(f) value is -1.1138e+03;

The inverse permutation is (5 4 6 7 1 9 2 3 8);

We guess this is the song *<let it be>*, and the original author is *The Beatles.*

1. ‘Q’ ; ‘Z’ ; ‘J’ ; ‘X’ ; ‘K’
2. ‘E’ to ‘ ’; ‘ ’ to ‘T’; ‘H’ to ‘E’; ‘T’ to ‘H’; ‘D’ to ‘ ’
3. The code :

**Part 1: Data Processing**

% This piece of code does the job of data preprocess

% The first 842 and last 364 lines have already been deleted in the file "War and Peace.txt"

% Following has been done:

% 1. convert all letters to upper case

% 2. Only select characters that in alph, new text is stored in 'data.txt'

clear;clc;

filename = 'War and Peace.txt';

file = fileread(filename);

upperfile = upper(file);

%by converting to ASCII, we only select characters in alph(corresponding ASCII are 65-90 and 32)

upperfile\_asc=abs(upperfile);

upperfile\_asc(~(((65<=upperfile\_asc)&(upperfile\_asc<=90))|(upperfile\_asc==32)))=[];

upperfile=char(upperfile\_asc);

%Store the selected text in data.txt

fileID = fopen("data.txt",'w');

fprintf(fileID,upperfile);

fclose(fileID);

**Part 2: Creating log\_char\_freq and log\_TransB Tables**

% Run this piece of code, freq vector and TranB table will be created and

% their transformed values (log) will be stored as log\_char\_freq and log\_TransB

clear;clc;

filename = 'data.txt';

file = fileread(filename);

alph = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z',' '];

save alph

file\_len = length(file);

alph\_len = length(alph);

char\_freq = zeros(alph\_len,1);

for i=1:alph\_len

idx = strfind(file,alph(i));

char\_freq(i) = length(idx) / file\_len;

end

log\_char\_freq = log(char\_freq);

save log\_char\_freq

TransB = zeros(alph\_len,alph\_len); % Initialize TransB

for i=1:file\_len-1

char1 = file(i);

char2 = file(i+1);

indexOfchar1 = strfind(alph,char1);

indexOfchar2 = strfind(alph,char2);

TransB(indexOfchar1,indexOfchar2) = TransB(indexOfchar1,indexOfchar2) + 1;

end

unnormalized\_TransB = TransB; %Used for Question Answering

save unnormalized\_TransB

Diag = diag(1./sum(TransB,2));

TransB = Diag \* TransB; % Normalize each row

% Replace -Inf values in log\_TransB by a small value -12

log\_TransB = log(TransB);

[x,y] = find(log\_TransB == -inf);

for i=1:length(x)

log\_TransB(x(i),y(i)) = -12;

end

save log\_TransB

**Part 3: Metropolis\_Hasting decoder**

% Main Part of the project: using Metropolis-Hasting algorithm to find best

% permutation (Permu\_max), apply it to the whole ciphertext for decryption

clear;clc;

Num\_MH = 20; % Run the Metropolis-Hastings algorithm 20 times

ciphertext = fileread("ciphertext.txt");

l = 9; % key length

% Initialize P\_max and Permu\_max

P\_max = -inf;

Permu\_max = [1,2,3,4,5,6,7,8,9];

load('alph.mat');

load('log\_char\_freq.mat');

load('log\_TransB.mat');

% loop Num\_MH time for searching Permutation with Maximum plausibility value

for i=1:Num\_MH

% Run Metropolis-Hastings algorithm for Num\_MH times,

% each time with MaxIt iterations

% For each run of MH algorithm, f\_j is randomly created for decryption

j = randi(30,1); % Uniformly select an integer from 1 to 30

f\_j = ciphertext(j\*l+1:end);% delecting the first j blocks

[P\_candidate,Permu\_candidate] = MH\_decoder\_fun(alph,log\_char\_freq,log\_TransB,f\_j);

if P\_candidate > P\_max

P\_max = P\_candidate;

Permu\_max = Permu\_candidate;

end

end

plaintext = apply\_permu(ciphertext,Permu\_max);% Permu\_max is applied to the whole ciphertext

P\_plaintext = plausibility(alph,log\_char\_freq,log\_TransB,plaintext);% Calculate corresponding plausibility

% decrypted text is stored in plaintext.txt

fileID = fopen("plaintext.txt",'w');

fprintf(fileID,plaintext);

fclose(fileID);

**functions used in the above section:**

**MH\_decoder\_fun**

function[P\_max,Permu\_max] = MH\_decoder\_fun(alph,log\_char\_freq,log\_TransB,f\_j)

% for given ciphertext f\_j, MaxIt of interatation is runned,

% best Permutation is returned together with plausibility

Permutation = [1,2,3,4,5,6,7,8,9]; %Initialize Permuatation as indentity

P\_j = plausibility(alph,log\_char\_freq,log\_TransB,f\_j);

P\_max = P\_j;

Permu\_max = Permutation;

MaxIt = 6000;

% Iterate first MaxIt/2 with slide move

for k1=1:MaxIt/2

New\_Permu\_candidate = slide\_move(Permutation);

f\_star = apply\_permu(f\_j,New\_Permu\_candidate);

P\_star = plausibility(alph,log\_char\_freq,log\_TransB,f\_star);

f\_old = apply\_permu(f\_j,Permutation);

P\_old = plausibility(alph,log\_char\_freq,log\_TransB,f\_old);

u = rand(1,1);

if u < exp(min(0,P\_star - P\_old))

Permutation = New\_Permu\_candidate; % Permutation is the one in Markov Chain

end

if P\_star > P\_max

P\_max = P\_star;

Permu\_max = New\_Permu\_candidate;

end

end

% Iterate last MaxIt/2 with swap move

for k2=1:MaxIt/2

New\_Permu\_candidate = swap\_move(Permutation);

f\_star = apply\_permu(f\_j,New\_Permu\_candidate);

P\_star = plausibility(alph,log\_char\_freq,log\_TransB,f\_star);

f\_old = apply\_permu(f\_j,Permutation);

P\_old = plausibility(alph,log\_char\_freq,log\_TransB,f\_old);

u = rand(1,1);

if u < exp(min(0,P\_star - P\_old))

Permutation = New\_Permu\_candidate;

end

if P\_star > P\_max

P\_max = P\_star;

Permu\_max = New\_Permu\_candidate;

end

end

end

**plausibility**

function [P] = plausibility(alph,log\_char\_freq,log\_TransB,f)

% Compute the plausibility of given text

% Input:f where f is a text

P = log\_char\_freq(strfind(alph,f(1)));

for i=1:length(f)-1

P = P + log\_TransB(strfind(alph,f(i)), strfind(alph,f(i+1)));

end

end

**slide\_move**

function [permutation2] = slide\_move(permutation1)

% new permutation(permutation2) is created from slide move transformation of given

% permutation (permutation1)

% Permuation1 and Permutation2 are 1\*9 vectors

l = 9;

b = randi(l-2,1);

k1 = randi(l-b+1,1);

k2 = randi([0,l-b],1);

Deleted\_Part = permutation1(k1:k1+b-1);

Remainder\_Part = permutation1([1:k1-1 k1+b:end]);

permutation2 = [Remainder\_Part(1:k2) Deleted\_Part Remainder\_Part(k2+1:end)];

end

**swap\_move**

function [permutation] = swap\_move(permutation1)

% new permutation(permutation) is created from swap move transformation of given

% permutation (permutation1)

% Permuation1 and Permutation are 1\*9 vectors

r1 = randi(9,1); r2 = randi(9,1);

permutation = permutation1;

a = permutation1(r1);

permutation(r1) = permutation1(r2);

permutation(r2) = a;

end

**Part 4: Question answering**

% Q2 Five least frequent characters in <War and Peace> in increasing order

clear; clc;

load log\_char\_freq; freq = log\_char\_freq;

load alph; alp = alph;

load unnormalized\_TransB; TransB = unnormalized\_TransB;

Min\_inc\_index = zeros(1,5); %increase order

for j = 1:5

[m,index] = min(freq);

Min\_inc\_index(j) = index;

freq(index) = +inf;

end

Sol\_Q2 = alp(Min\_inc\_index);

%Q3 Five most frequent transitions

Max\_dec\_index = zeros(2,5); %Each column represents one transition

for j = 1:5

n = max(max(TransB));

[row,col] = find(TransB==n);

Max\_dec\_index(1,j) = row;

Max\_dec\_index(2,j) = col;

TransB(row,col) = -inf;

end

Sol\_Q3 = alp(Max\_dec\_index);