CSC 323 Spring 2015: Huffman Coding part-4 (Java)

Ravi Patel

March 31, 2015

**Algorithm steps:**

Algorithm steps for computing the probabilities of characters in a text file:

// compute the histogram

Step 0: Initialize the 1-D histogram/count array, size of 256, to zero.

(Change the array size from 128 to 256)

Step 1: charStr 🡨 get a character from text\_1

index 🡨 mapping the charStr from char to int

Hist[index] ++

Step 2: repeat step 1 until text\_1 is empty

// Output pairs

Step 3: index 🡨 0

Step 4: charStr 🡨 mapping index to char

Step 5: output charStr and Hist[index] as a pair to outFile

Step 6: index ++

Step 7: repeat step 4-6 until index >= 256

Source Code:

public class Node {

//Structure of Node - 6 elements

//letters, frequency, left, right, next, code

private String letters,code;

private double frequency;

private Node next,left,right;

public Node(){

this(null,0.0,null,null,null,null);

}//Node Constructor

public Node(String letters, double frequency, Node left,

Node right, Node next, String code){

this.letters = letters;

this.frequency = frequency;

this.left = left;

this.right = right;

this.next = next;

this.code = code;

}//Node Method

public Node(String letters, double frequency){

this(letters,frequency,null,null,null,"");

}//Node Method

public void setLetters(String letters){

this.letters=letters;

}//setLetters

public void setFrequency(double frequency){

this.frequency=frequency;

}//setFrequency

public void setNext(Node next){

this.next=next;

}//setNext

public void setLeft(Node left){

this.left=left;

}//setLeft

public void setRight(Node right){

this.right=right;

}//setRight

public void setCode(String code){

this.code=code;

}

public String getCode(){

return code;

}//getCode

public String getLetters(){

return letters;

}//getLetters

public double getFrequency(){

return frequency;

}//getFrequency

public Node getNext(){

return next;

}//getNext

public Node getLeft(){

return left;

}//getLeft

public Node getRight(){

return right;

}//getRight

}//Node Class

import java.io.FileReader;

import java.io.IOException;

import java.util.Scanner;

public class LinkedList{

private Node head;

private int size;

public LinkedList(){

Node dummy = new Node("Dummy",0.0);

head=dummy;

size=0;

}//LinkedList

public int size(){

return size;

}//size

public boolean isEmpty(){

return size==0;

}//isEmpty

public void makeEmpty(){

head=null;

size=0;

}//makeEmpty

public void insert(String letters, double frequency){

Node newNode = new Node(letters,frequency);

Node walker, pointer;

//if empty new node is head & tail

if(isEmpty()){

head.setNext(newNode);

size++;

}//if list is empty

else{

walker=head; pointer=walker.getNext();

while(pointer!=null){

if(frequency <= pointer.getFrequency()){

break;

}

walker=pointer;

pointer=pointer.getNext();

}

newNode.setNext(pointer);

walker.setNext(newNode);

size++;

}

}//insert

public static Node BinaryTreeConstruction(LinkedList list){

Node LH = list.head;

Node OLH = new Node("Dummy",0.0);

OLH.setNext(LH.getNext());

//String result="";

while(LH.getNext().getNext()!=null){

Node before = LH.getNext();

Node after = before.getNext();

Node newNode = new Node();

newNode.setLetters(before.getLetters()+after.getLetters());

newNode.setFrequency(before.getFrequency()+after.getFrequency());

newNode.setLeft(before);

newNode.setRight(after);

Node walker = OLH;

Node pointer = before.getNext();

while(pointer!=null){

if(newNode.getFrequency() <= pointer.getFrequency()){

break;

}

walker=pointer;

pointer=pointer.getNext();

}

newNode.setNext(pointer);

walker.setNext(newNode);

LH=before.getNext();

}

Node root=LH.getNext();

LH=OLH;

return root;

}

public static String print(LinkedList list){

String result="";

for(Node walker=list.head; walker!=null; walker=walker.getNext()){

if(walker.getLetters()=="Dummy"){

result+="listHead -->("+walker.getLetters()+", "+walker.getFrequency()+", "+walker.getNext().getLetters()+")-->";

}

result+="("+walker.getLetters()+", "+walker.getFrequency();

if(walker.getNext()!=null){

result+=", "+walker.getNext().getLetters()+")-->";

}

else{

result+=")-->NULL";

}

}//for

return result;

}//print

public static void HuffmanCode(Node root, String code){

if(root == null) return;

if(root.getLeft()==null && root.getRight() ==null){

root.setCode(code);

return;

}

HuffmanCode(root.getLeft(), code+"0");

HuffmanCode(root.getRight(), code+"1");

}//HuffmanCode

public static String EntropyTable(LinkedList list){

int size = list.size();

String[] character = new String[size];

double[] char\_freq = new double[size];

String[] char\_code = new String[size];

int[] bits = new int[size];

double[] entrophy = new double[size];

int i = 0;

for(Node walker = list.head.getNext(); walker!=null ; walker=walker.getNext()){

if(walker.getCode() != null){

character[i] = walker.getLetters();

char\_freq[i] = walker.getFrequency();

char\_code[i] = walker.getCode();

bits[i] = walker.getCode().length();

entrophy[i] = char\_freq[i] \* bits[i];

i++;

}

}

String result = "";

result+="Char "+"\t"+"Prob "+"\t"+"Code "+"\t"+"#Bits "+"\t"+"Entrophy"+"\n";

result +="---------------------------------------------------"+"\n";

for(int j=0; j<size; j++){

result += character[j]+"\t"+char\_freq[j]+"\t"+char\_code[j]+"\t"+

bits[j]+"\t"+entrophy[j]+"\n";

}

return result;

}

public static String encoding(LinkedList list, char bit){

if(bit == ' '){

bit = '|';

}

else if(bit == '\r'){

bit = '\*';

}

else if(bit == '\n'){

bit = '^';

}

String letters=""+bit;

String result="";

for(Node walker=list.head;walker!=null;walker=walker.getNext()){

if(walker.getLetters().equals(letters)){

result =walker.getCode();

}

}

return result;

}

public static String decoding(Node root, FileReader input) throws IOException{

String result="";

String digit="";

int read;

Node walker=root;

while((read = input.read())!=-1){

digit = ""+(char)read;

if(walker.getLeft() == null && walker.getRight() == null){

if(walker.getLetters().equals("\*")) result +="\r";

else if(walker.getLetters().equals("|")) result +=" ";

else if(walker.getLetters().equals("^")) result +="\n";

else result += walker.getLetters();

walker=root;

}

if(digit.equals("0")){

walker=walker.getLeft();

}

if(digit.equals("1")){

walker=walker.getRight();

}

}

return result;

}

}//LinkedList Class

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) throws IOException{

if(args.length==0) System.out.println("No File Specified");

else{

LinkedList list = new LinkedList();

FileReader input1 = new FileReader(args[0]);

PrintWriter output1 = new PrintWriter(args[1]);//output1 histogram

PrintWriter output2 = new PrintWriter(args[2]);//output2 sorted linkedlist

PrintWriter output3 = new PrintWriter(args[3]);//output3 binarytree

PrintWriter output4 = new PrintWriter(args[4]);//output4 entropy

FileReader input2 = new FileReader(args[5]);

PrintWriter output5 = new PrintWriter(args[6]);//encoding

PrintWriter output6 = new PrintWriter(args[7]);//decoding

try{

int count=0;

char character;

String letters;

double frequency;

int size=256;

double histogram[]= new double[size];

for(int start=0; start<256 ;start++){

histogram[start]=0;

}

int read;

while((read = input1.read())!=-1){

for(int inital = -128; inital<128; inital++){

if((char)inital == (char)read){

histogram[inital+128]+=1;

count++;

}

}

//System.out.println((char)read);

}

for(int inital = 0; inital < size; inital++ ){

histogram[inital] = (histogram[inital]/count)\*100;

}

for(int inital = 0; inital < size; inital++){

if(histogram[inital]!=0){

character = (char)(inital-128);

if(character == ' ') character = '|';

else if(character == '\r') character = '\*';

else if(character == '\n') character = '^';

output1.println(character +" "+histogram[inital]);

}

}

output1.close();

Scanner input3 = new Scanner(new FileReader(args[1]));

while(input3.hasNext()){

list.insert(input3.next(), input3.nextDouble());

//output1.print("Iteration "+ counter++ +": \n");

//output1.println(LinkedList.print(list));

}

output2.print(LinkedList.print(list));

LinkedList.BinaryTreeConstruction(list);

output3.print(LinkedList.print(list));

LinkedList.HuffmanCode(LinkedList.BinaryTreeConstruction(list), "");

output4.print(LinkedList.EntropyTable(list));

while((read = input2.read())!= -1){

output5.print(LinkedList.encoding(list, (char)read));

}

output5.close();

FileReader input4 = new FileReader(args[6]);

output6.print(LinkedList.decoding(LinkedList.BinaryTreeConstruction(list), input4));

}//try

finally{

input1.close();

input2.close();

output1.close();

output2.close();

output3.close();

output4.close();

output5.close();

output6.close();

}//finally

}//else

}//Main Method

}//Main Class

Input:

Output 1:

^ 1.6640665626625066

\* 1.6640665626625066

| 17.264690587623505

' 0.05200208008320333

, 1.5600624024960998

. 0.7280291211648465

1 0.10400416016640666

3 0.05200208008320333

6 0.05200208008320333

8 0.05200208008320333

9 0.05200208008320333

A 0.20800832033281333

B 0.10400416016640666

C 0.15600624024961

F 0.05200208008320333

G 0.20800832033281333

I 0.20800832033281333

L 0.10400416016640666

N 0.15600624024961

P 0.10400416016640666

S 0.10400416016640666

T 0.20800832033281333

U 0.10400416016640666

W 0.15600624024961

a 6.5522620904836195

b 1.0920436817472698

c 1.9760790431617263

d 3.74414976599064

e 10.764430577223088

f 1.924076963078523

g 1.6120644825793031

h 4.99219968798752

i 4.420176807072282

k 0.20800832033281333

l 2.65210608424337

m 1.0400416016640666

n 5.30421216848674

o 5.980239209568382

p 0.8320332813312533

q 0.05200208008320333

r 5.200208008320333

s 3.172126885075403

t 8.008320332813312

u 1.40405616224649

v 1.456058242329693

w 1.456058242329693

y 1.0400416016640666

Output 2:

listHead -->(Dummy, 0.0, q)-->(Dummy, 0.0, q)-->(q, 0.05200208008320333, F)-->(F, 0.05200208008320333, 9)-->(9, 0.05200208008320333, 8)-->(8, 0.05200208008320333, 6)-->(6, 0.05200208008320333, 3)-->(3, 0.05200208008320333, ')-->(', 0.05200208008320333, U)-->(U, 0.10400416016640666, S)-->(S, 0.10400416016640666, P)-->(P, 0.10400416016640666, L)-->(L, 0.10400416016640666, B)-->(B, 0.10400416016640666, 1)-->(1, 0.10400416016640666, W)-->(W, 0.15600624024961, N)-->(N, 0.15600624024961, C)-->(C, 0.15600624024961, k)-->(k, 0.20800832033281333, T)-->(T, 0.20800832033281333, I)-->(I, 0.20800832033281333, G)-->(G, 0.20800832033281333, A)-->(A, 0.20800832033281333, .)-->(., 0.7280291211648465, p)-->(p, 0.8320332813312533, y)-->(y, 1.0400416016640666, m)-->(m, 1.0400416016640666, b)-->(b, 1.0920436817472698, u)-->(u, 1.40405616224649, w)-->(w, 1.456058242329693, v)-->(v, 1.456058242329693, ,)-->(,, 1.5600624024960998, g)-->(g, 1.6120644825793031, \*)-->(\*, 1.6640665626625066, ^)-->(^, 1.6640665626625066, f)-->(f, 1.924076963078523, c)-->(c, 1.9760790431617263, l)-->(l, 2.65210608424337, s)-->(s, 3.172126885075403, d)-->(d, 3.74414976599064, i)-->(i, 4.420176807072282, h)-->(h, 4.99219968798752, r)-->(r, 5.200208008320333, n)-->(n, 5.30421216848674, o)-->(o, 5.980239209568382, a)-->(a, 6.5522620904836195, t)-->(t, 8.008320332813312, e)-->(e, 10.764430577223088, |)-->(|, 17.264690587623505)-->NULL

Output3:

listHead -->(Dummy, 0.0, q)-->(Dummy, 0.0, q)-->(q, 0.05200208008320333, F)-->(F, 0.05200208008320333, 9)-->(9, 0.05200208008320333, 8)-->(8, 0.05200208008320333, 6)-->(6, 0.05200208008320333, 3)-->(3, 0.05200208008320333, ')-->(', 0.05200208008320333, 63)-->(63, 0.10400416016640666, 98)-->(98, 0.10400416016640666, qF)-->(qF, 0.10400416016640666, U)-->(U, 0.10400416016640666, S)-->(S, 0.10400416016640666, P)-->(P, 0.10400416016640666, L)-->(L, 0.10400416016640666, B)-->(B, 0.10400416016640666, 1)-->(1, 0.10400416016640666, '63)-->('63, 0.15600624024961, W)-->(W, 0.15600624024961, N)-->(N, 0.15600624024961, C)-->(C, 0.15600624024961, B1)-->(B1, 0.20800832033281333, PL)-->(PL, 0.20800832033281333, US)-->(US, 0.20800832033281333, 98qF)-->(98qF, 0.20800832033281333, k)-->(k, 0.20800832033281333, T)-->(T, 0.20800832033281333, I)-->(I, 0.20800832033281333, G)-->(G, 0.20800832033281333, A)-->(A, 0.20800832033281333, NC)-->(NC, 0.31201248049922, '63W)-->('63W, 0.31201248049922, IG)-->(IG, 0.41601664066562666, kT)-->(kT, 0.41601664066562666, US98qF)-->(US98qF, 0.41601664066562666, B1PL)-->(B1PL, 0.41601664066562666, ANC)-->(ANC, 0.5200208008320333, .)-->(., 0.7280291211648465, '63WIG)-->('63WIG, 0.7280291211648466, kTUS98qF)-->(kTUS98qF, 0.8320332813312533, p)-->(p, 0.8320332813312533, B1PLANC)-->(B1PLANC, 0.93603744149766, y)-->(y, 1.0400416016640666, m)-->(m, 1.0400416016640666, b)-->(b, 1.0920436817472698, u)-->(u, 1.40405616224649, w)-->(w, 1.456058242329693, v)-->(v, 1.456058242329693, .'63WIG)-->(.'63WIG, 1.4560582423296933, ,)-->(,, 1.5600624024960998, g)-->(g, 1.6120644825793031, kTUS98qFp)-->(kTUS98qFp, 1.6640665626625066, \*)-->(\*, 1.6640665626625066, ^)-->(^, 1.6640665626625066, f)-->(f, 1.924076963078523, c)-->(c, 1.9760790431617263, B1PLANCy)-->(B1PLANCy, 1.9760790431617266, mb)-->(mb, 2.132085283411336, l)-->(l, 2.65210608424337, uw)-->(uw, 2.860114404576183, v.'63WIG)-->(v.'63WIG, 2.9121164846593866, ,g)-->(,g, 3.1721268850754027, s)-->(s, 3.172126885075403, kTUS98qFp\*)-->(kTUS98qFp\*, 3.3281331253250133, ^f)-->(^f, 3.58814352574103, d)-->(d, 3.74414976599064, cB1PLANCy)-->(cB1PLANCy, 3.952158086323453, i)-->(i, 4.420176807072282, mbl)-->(mbl, 4.7841913676547065, h)-->(h, 4.99219968798752, r)-->(r, 5.200208008320333, n)-->(n, 5.30421216848674, uwv.'63WIG)-->(uwv.'63WIG, 5.772230889235569, o)-->(o, 5.980239209568382, ,gs)-->(,gs, 6.3442537701508055, a)-->(a, 6.5522620904836195, kTUS98qFp\*^f)-->(kTUS98qFp\*^f, 6.916276651066044, dcB1PLANCy)-->(dcB1PLANCy, 7.696307852314093, t)-->(t, 8.008320332813312, imbl)-->(imbl, 9.204368174726989, hr)-->(hr, 10.192407696307853, e)-->(e, 10.764430577223088, nuwv.'63WIG)-->(nuwv.'63WIG, 11.076443057722308, o,gs)-->(o,gs, 12.324492979719189, akTUS98qFp\*^f)-->(akTUS98qFp\*^f, 13.468538741549663, dcB1PLANCyt)-->(dcB1PLANCyt, 15.704628185127405, |)-->(|, 17.264690587623505, imblhr)-->(imblhr, 19.39677587103484, enuwv.'63WIG)-->(enuwv.'63WIG, 21.840873634945396, o,gsakTUS98qFp\*^f)-->(o,gsakTUS98qFp\*^f, 25.793031721268854, dcB1PLANCyt|)-->(dcB1PLANCyt|, 32.96931877275091, imblhrenuwv.'63WIG)-->(imblhrenuwv.'63WIG, 41.23764950598024, o,gsakTUS98qFp\*^fdcB1PLANCyt|)-->(o,gsakTUS98qFp\*^fdcB1PLANCyt|, 58.76235049401976, imblhrenuwv.'63WIGo,gsakTUS98qFp\*^fdcB1PLANCyt|)-->(imblhrenuwv.'63WIGo,gsakTUS98qFp\*^fdcB1PLANCyt|, 100.0)-->NULL

output4:

Char Prob Code #Bits Entrophy

---------------------------------------------------

q 0.05200208008320333 11111110001110 14 0.7280291211648466

F 0.05200208008320333 11111110001111 14 0.7280291211648466

9 0.05200208008320333 11111110001100 14 0.7280291211648466

8 0.05200208008320333 11111110001101 14 0.7280291211648466

6 0.05200208008320333 11110000110010 14 0.7280291211648466

3 0.05200208008320333 11110000110011 14 0.7280291211648466

' 0.05200208008320333 1111000011000 13 0.6760270410816434

U 0.10400416016640666 1111111000100 13 1.3520540821632867

S 0.10400416016640666 1111111000101 13 1.3520540821632867

P 0.10400416016640666 100100110010 12 1.24804992199688

L 0.10400416016640666 100100110011 12 1.24804992199688

B 0.10400416016640666 100100110000 12 1.24804992199688

1 0.10400416016640666 101110000100 12 1.24804992199688

W 0.15600624024961 111100001101 12 1.87207488299532

N 0.15600624024961 111001000010 12 1.87207488299532

C 0.15600624024961 111001000011 12 1.87207488299532

k 0.20800832033281333 111111100000 12 2.49609984399376

T 0.20800832033281333 111111100001 12 2.49609984399376

I 0.20800832033281333 111100001110 12 2.49609984399376

G 0.20800832033281333 111100001111 12 2.49609984399376

A 0.20800832033281333 111010010011 12 2.49609984399376

. 0.7280291211648465 1111000010 10 7.280291211648465

p 0.8320332813312533 1111111001 10 8.320332813312532

y 1.0400416016640666 100100111 9 9.360374414976599

m 1.0400416016640666 110000100 9 9.360374414976599

b 1.0920436817472698 110000101 9 9.82839313572543

u 1.40405616224649 111001100 9 12.63650546021841

w 1.456058242329693 111010000 9 13.104524180967237

v 1.456058242329693 111100000 9 13.104524180967237

, 1.5600624024960998 111110100 9 14.0405616224649

g 1.6120644825793031 111110101 9 14.508580343213728

\* 1.6640665626625066 111111101 9 14.97659906396256

^ 1.6640665626625066 111111110 9 14.97659906396256

f 1.924076963078523 111111111 9 17.316692667706707

c 1.9760790431617263 10010010 8 15.80863234529381

l 2.65210608424337 11000011 8 21.21684867394696

s 3.172126885075403 11111011 8 25.377015080603226

d 3.74414976599064 1001000 7 26.209048361934478

i 4.420176807072282 1100000 7 30.94123764950598

h 4.99219968798752 1100010 7 34.945397815912635

r 5.200208008320333 1101000 7 36.40145605824233

n 5.30421216848674 1110000 7 37.129485179407176

o 5.980239209568382 1111100 7 41.86167446697868

a 6.5522620904836195 1111110 7 45.86583463338533

t 8.008320332813312 100101 6 48.049921996879874

e 10.764430577223088 110110 6 64.58658346333853

| 17.264690587623505 10100 5 86.32345293811753

output5:

1111111000011100010110110101001111000011111101101001011001011001001111111101111000010111100110011010001111101011010011101001001110010001001000110100011011011111011111110111111111011111111101111111000111111111001110011001101000101001111101110010010111110011010001101101010011111101110000100100010100111110111101101111000001101101110000101001001001111101101111110110100011111011101001111110111110101111110010100111110011100110011010001010011111111111111101001011100010110110110100011111011101001111111011111111101100001011101000111110011100110011111010111000101001011010011111111111111001101000100101110001010100111110011100001010010010111000101100000111110111010010010010111110011100001001011100000111000011011011100001001011010011111101010011100001101101110100001010011100001111110100101110000011111001110000111110100101001001001011111001110000100100101101101100000111100000110110100100010100110000011100001010011000011110000011000010111011011010001001011001001111111101001010011111110111111111011111101110000100100010100100100011011010010001100000100100101111110100101110110100100010100100101111110010100100101110001011011010100111111100111010001111100111111100111111001111101111000001001011100000111110011100001010010010111000101111110100101101001111110110000111100001110100110000100110110111000010100111111101111111110111100001101110110101001111110110100011011010100110000100110110100101101001111100111000010100111111010100111110101110100011011011111101001011010011000010111111101001011001011100001111011011111111111000001101101100001110010001010011111001111111111010010010111000101111110100101101001110100001111110110100011110000101010011110000110111011010100110001011111101111000001101101010010010010111110011000010011011010100100101111110010100100100011011010010001100000100100101111110100101110110101001111111011111111101111110101001111111001111110011010001001011100000111110011100001010011111001111111111010010010111000101111110100101101001111111111100000110110110000111001000111110100101001111110111110111010011111101010011111111111000001110000111111011000011101001101000110110111110111001011100000111000011111010110100111111100111000011111111010010010110110101001111111111111100110100010100100101110001011111001111101111011010100111111101111111110111010000110001011111001010011000101101101101000110110101001111101011111110111100000110110101001001011100010110110110000011010001010011000011110000011110000011011011111011101001001011100010111111010010110100111000011111101001011100000111110011100001010011000010011000001111101011100010100101101001100001111000001111000001101101111000010101001111111011111111101111000011101001011010011000001111101110100111111011000011100101111110011111010111011010010111000101101101101000101001111111111100000100101100101110000011100001111101011010011111101110000100100010100111111100111010001111100111111100111011011010001010010010111000101111110100101101001110100001101101010011111011110001011111001110011001100001110010001010010010001111100101001001011100010110000011111011111100001011111110111111111011110000111010010110100110000011111011101001101000111111010010111000101101101101000101001111111111111100110100010100111001100111110111010010010111111001010011000010111011010100110001011011011010001101101010010010001101101001000110000010010010111111010010111011010010001010010010111111001010010010111000101101101010011111010111010001101101111110100101101001001011111110111110111111111000001010011010001101101100001001111110110000011100001100000111000011111010110100111111101111111110110000101110110111111111111110011010001101101010011100110011111011101001001011100010111111010010110100111111111110100011111001100001001010010010111000101101101111101111011010100110001011111001110000111110011010001101101001000101001001000110110111111010010001010011101000011011010100100101111111011111110000011011010100110000011100001001001011010001101101111110111110111101101001000101001001000110110111100000111110010010111000001111100111000010100100101111110010100111111101111111110100101110001011111101001011010010010010111111011100110011111011110110101001111111111111100110100010100111010000110001011000001001001011000101010010010111000101101101001001111010011111010111111101111000001101101010010010111000101101101010011000011111111011111011100101101001111111111110011001100001111000011101001100001001101101111110111110111110011001101000110110101001111100111111111101001001000110110111100000111110010010111000001111100111000010100100101110001011111101001011010011111110111111111011101000011011010100110001011011011010001101101010011000101100000111110101110001011000011100100111101001101000110110111110111111100110000111111000001101101010010010111000101111110100101101001001011100010110110111110111101101010010010001101101111110100100010100111110111100010111111011000011110000111010011100001111100100101101001100010111111011110000011011010100100100011000001101101001000101001100000111000010100111100000111111011000001110000111111101111111110111111101111111110

output6:

Ihvrndadoaessh G Ga.nepanaW

r W i. onasegstGae twsbhnsaGae g

ihal.vrutnSt l 1in d ts nr v,m

ngr rUe nieGnhp

aarIou iuh,Iaergs

aaynoan gq ge

adrfGAoymGNr

eretvb

lnen al

nieGnaais cl iGNr

Irepastn

irina elsGd

rnfah,.ha ticgtipaNg

icel

tNr

iG if dhdrrnh

hbsfieWe le

iclsGUIrIecm

bApeuoeh amnataatfoaLae tGarg

pianheticieGnhp

asd

rvaidcs wlcaafA nytm

b

heh, nfGWo,Ih,ul ln

N tati fel i aaa l

bnvhaocreailbuee on

aeacyucienvnetef an,1igffa trnhethf

efhnrooigffee nrvhseoielgFdn vcr vliiaaG

Testdata:

The Gettysburg Address

Four score and seven years ago our fathers

brought forth on this continent a new nation, conceived in liberty,

and dedicated to the proposition that all men

We are met on a great battlefield of that war. We have come to dedicate

a portion of that field, as a final resting place for those

who here gave their lives that nation might live.

It is altogether fitting and proper that we should do this.

It is rather for us to be here dedicated to the great task remaining

before us that from these honored dead we take increased devotion to

that cause for which they gave the last full measure of devotion that

we here highly resolve that these dead shall not have died in vain