Andrew Kaufman
998048873
4/13/16

Plaintext:

when i heard the learned astronomer,

when the proofs, the figures, were ranged in columns before me,

when i was shown the charts and diagrams, to add, divide, and measure them,

when i sitting heard the astronomer where he lectured with much applause in the lecture room,

how soon unaccountable i became tired and sick,

till rising and gliding out i wandered off by myself,

in the mystical moist night air, and from time to time,

looked up in perfect silence at the stars.

Typically, my program was able to decipher the text after approximately 2000 iterations. In the worst case, my program would decipher the ciphertext in under 10000 iterations.

```
# run using python3

import math
import random
import string

# simulates random coin flip
def bernoulli(p):
    r = random.random()
    return (r <= p)

# creates matrix of bigram frequencies
def bigram():
    count = 0
    filename = 'war-and-peace.txt'
    bigrams = [[1.0 for x in range(27)] for x in range(27)]

    with open(filename, 'r') as infile:
        for line in infile:
            for i in range(len(line)):
                pos1 = 0
                pos2 = 0
                if line[i].islower():
                    pos1 = string.ascii_lowercase.index(line[i]) + 1
                if (i+1) < len(line):
                    if line[i+1].islower():
                      pos2 = string.ascii_lowercase.index(line[i+1]) + 1
                bigrams[pos1][pos2] += 1
                count += 1

    for i in range(27):
        for j in range(27):
            bigrams[i][j] /= count              # convert to frequencies


    return bigrams


# creates random key f
def permute():
    s = ''
    arr = [0 for x in range(26)]
    for i in range(26):
        x = random.randint(0, 25)
        if arr[x] == 1:
            while (arr[x] == 1):
                x = random.randint(0, 25)

        arr[x] = 1
        c = string.ascii_lowercase[x]
        s += c
    return s


# computes plausibility of decipherString
def plausibility(decipherString, bigrams):
    pl = 0.0
    for i in range(len(decipherString)):

        pos1 = 0
        pos2 = 0

        if decipherString[i].islower():
            pos1 = string.ascii_lowercase.index(decipherString[i]) + 1
        if (i + 1) < len(decipherString):
            if decipherString[i + 1].islower():
                pos2 = string.ascii_lowercase.index(decipherString[i + 1]) + 1

        pl += math.log(bigrams[pos1][pos2])

    return pl


def decipher(bigrams):
    cipherFile = 'cipher.txt'

    f = permute()
    f2 = swap(f)

    cipherString = ''

    with open(cipherFile, 'r') as infile:       # ciphertext from file cipher.txt
        for line in infile:
            for i in line:
                cipherString += i

    count = 0                                   # number of iterations
    repeat = 0

    while (1):
        count += 1
        decipher1 = ''
        decipher2 = ''

        print(count)

        for i in range(len(cipherString)):      # decipher using keys f and f2
            if cipherString[i].islower():
                decipher1 += string.ascii_lowercase[f.index(cipherString[i])]
                decipher2 += string.ascii_lowercase[f2.index(cipherString[i])]
            else:
                decipher1 += cipherString[i]
                decipher2 += cipherString[i]

        plf = plausibility(decipher1, bigrams)
        plf2 = plausibility(decipher2, bigrams)

        if plf == plf2:                         # plausibility stuck at local maxima
            repeat += 1

        if repeat == 50 and plf < -2400.0:      # get new key f
            f = permute()
            repeat = 0

        print(decipher1)                        # print plaintext created from key f

        if plf2 > plf:
            f = f2
            f2 = swap(f)
        else:
            p = plf2 - plf
            coin = bernoulli(p)                 # perform biased coin flip

            if coin == 1:                       # if coin is heads
                f = f2
                f2 = swap(f)
            else:
                f2 = swap(f)


# performs random transpose of two characters in f
def swap(f):
    pos1 = random.randint(0, 25)
    pos2 = random.randint(0, 25)
    s = ''

    length = len(f)
    if pos2 == pos1:
        while (pos2 == pos1):
            pos2 = random.randint(0, 25)

    for i in range(length):
        if i == pos1:
            s += f[pos2]
        elif i == pos2:
            s += f[pos1]
        else:
            s += f[i]

    return s



def main():
    bigrams = bigram()
    decipher(bigrams)

if __name__ == '__main__': main()
```