Assignment #2

CPEN 442

September 27

Kaibo Ma (32400129)

Department of Electrical and Computer Engineering

University of British Columbia

Vancouver, Canada

# Problem #1

**Cypher Text:** GXMLKTALSMSEIKXIHUGPKTWEHRKVJKWHMLGEGLTEUWHUJSXHQKELKWQHOGYBYKJJGMLHAKKUJGEYLHHWVTXXRYKJNXSHUUKMKEYHJKWHMLHEHSALOKWQGEUVWSHEUQAGMLHWHUGOKTMMLHAWGIHKVEGEYRLGEBQYKJJGGEULHWQKEVKTEUYKJVKWMSEMLHSWQRJNGMLRGEUMLHSWNWHQHEYHUKMMLHQNKMCLHWHXSEYKXEQJKMLHWXSHQSQEKCHEYXKQHUCSMLSEGLSALSWKEVHEYHUKMGMMLHLHGUKVMLHAWGIHGCLSMHQMKEHYKJJGQSJNXHYKJJGTEGVVHYMHUYKJJGGEUSEBHHNSEACSMLMLHQTWWKTEUSEAQYKJJGLGQOHHENXGYHUUKMSMOHGWQMLHVKXXKCSEASEQYWSNMSKEEGEYRLGEBQXSEYKXEYKJJGJKMLHWKVNWHQSUHEMXSEYKXEYKJJGUSHUKYMKOHWVSIHYKJJGG

**Plain Text:**

ALTHOUGH IT INVOLVED A JOURNEY OF MORE THAN A HUNDRED MILES ON HORSEBACK COMMA THE GOOD MAN CHEERFULLY COMPLIED DOT ONCE MORE THE NEIGHBORS AND FRIENDS GATHERED ABOUT THE GRAVE OF NANCY HANKS COMMA AND HER SON FOUND COMFORT IN THEIR SYMPATHY AND THEIR PRESENCE DOT THE SPOT WHERE LINCOLNS MOTHER LIES IS NOW ENCLOSED WITH IN A HIGH IRON FENCE DOT AT THE HEAD OF THE GRAVE A WHITE STONE COMMA SIMPLE COMMA UNAFFECTED COMMA AND IN KEEPING WITH THE SURROUNDINGS COMMA HAS BEEN PLACED DOT IT BEARS THE FOLLOWING INSCRIPTION NANCY HANKS LINCOLN COMMA MOTHER OF PRESIDENT LINCOLN COMMA DIED OCTOBER FIVE COMMA A

**Plain Text with Punctuation:**

ALTHOUGH IT INVOLVED A JOURNEY OF MORE THAN A HUNDRED MILES ON HORSEBACK, THE GOOD MAN CHEERFULLY COMPLIED. ONCE MORE THE NEIGHBORS AND FRIENDS GATHERED ABOUT THE GRAVE OF NANCY HANKS, AND HER SON FOUND COMFORT IN THEIR SYMPATHY AND THEIR PRESENCE. THE SPOT WHERE LINCOLNS MOTHER LIES IS NOW ENCLOSED WITH IN A HIGH IRON FENCE. AT THE HEAD OF THE GRAVE A WHITE STONE, SIMPLE, UNAFFECTED, AND IN KEEPING WITH THE SURROUNDINGS, HAS BEEN PLACED. IT BEARS THE FOLLOWING INSCRIPTION NANCY HANKS LINCOLN, MOTHER OF PRESIDENT LINCOLN, DIED OCTOBER FIVE, A

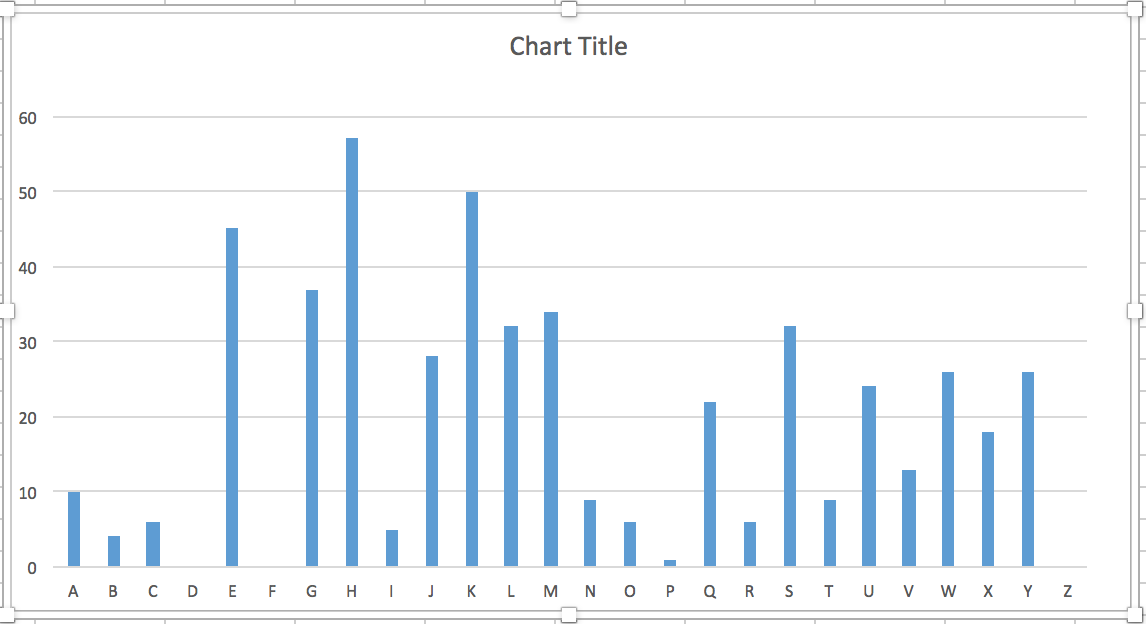
Alphabet: abcdefghijklmnopqrstuvwxyz

Key: goyuhvalspbxjeknzwqmticfrd

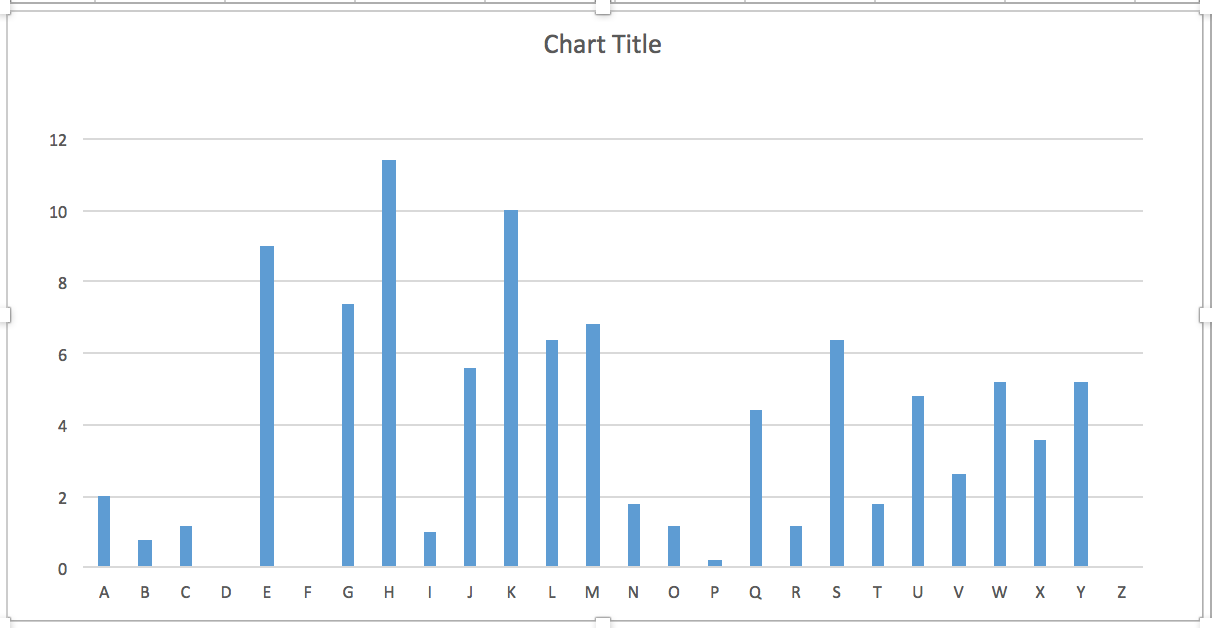
**Procedure:**

The first method I used to decrypt the cypher was to see if it was a Caesar Cypher. I wrote a javascript that spat out all possible 26 shifts. From which I learned that the cypher text was not Caesar. Next I used frequency analysis to figure out if it might be a monoalphabetical cypher.

I found that out of the 500 characters in my text file, some characters appeared more than others, which is a tall tail sign of a monoalphabetical cypher (Fig 1, Fig 2).

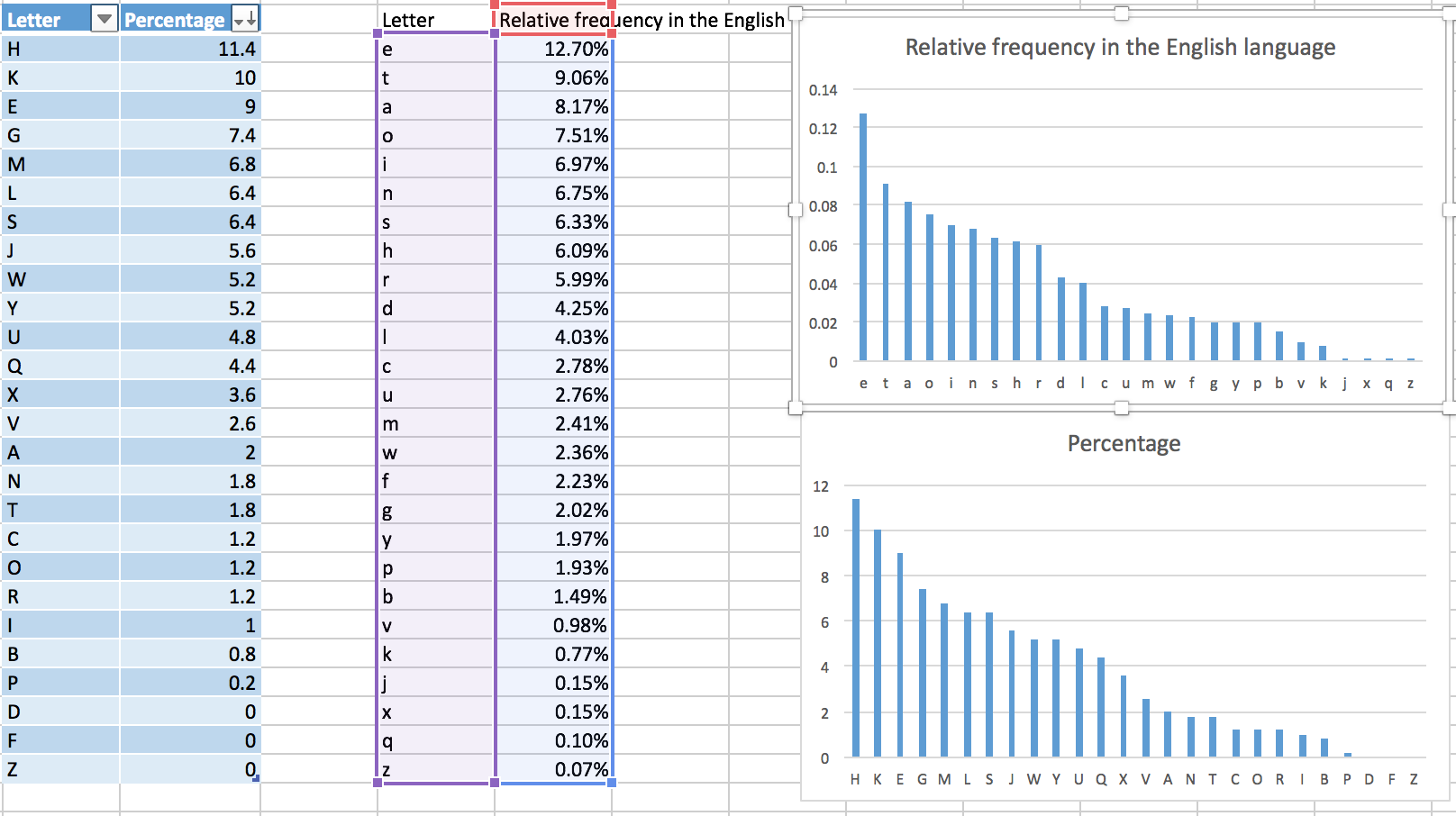


(Fig. 1) Number of Occurrences of Letters



(Fig. 2) Percentage of occurances

Next, I compared to side by side with the English natural frequency occurances (Fig 3). From there I found the letter “E“ and common digrams and trigrams like “the”. From there I found that there was a common 5 letter combination that kept on repeating which later I found out it was the phrase “COMMA”. Now I knew that punctuation was spelt out. There I found common trigrams which was a candidate for “DOT” as I already found “O” and “T”. From there I found words like “their” and “gather”. With trial and error I finally completed the decryption.



(Fig. 3) cypher text sorted next to the letter statistics of English.

# Problem #2

Cypher Text:

HSOLGRRELZDBWMXOLIYKVDRVQVKIUVKOVZHGFUTAMDOHQPQDDBWNFUPGVAOUCHOIBDDWAKHXAPTAAZVACVAKBHVAOUCOVQOITAKQFKIHVZDCOHQPQDOUZCFWUVKOOIKOTOILTZVXMGAVEVGWTLOUXOFVGRVZBODWOUXCXOFUZRHFOHQPQDTHIVLIDVVIXOVYQVDWQTOUILWVAVUKGBDLQDOQEHSFVQAZVFXOHSLIHVZUATTBVQCHCMTULCTWYSENDLLDDNQOOCHVZUSLXPCHLRTHLSWYVMPOCZVQKVCVTAVZHDDWILZROUILXERIXOREYDIGGWEGCXTKUTPGOUCHMOHVWZEMDVIHOHQPQDCXILFWVROHQPQDDBONDBOMIHYWQZLZIROUDVHPIMEIIHDCDHOHQPQDVLZRZVGRDCFCOUCLOQAETLVBICEHSPYCVQTHDWOUPLOUDVVGOIUVLCFDLCGIOLVQRWPGTHLICZWVAVOUCHMOCQHZOUDVHPKOYCKATHVASFKDQERHVKVLBDLPYDHUILZOLCOHQPLDCPOIFGTDCUTHHILHMOFUPCRGRHTLTQPHOCYDSFEXKVCVDBTVLCZEBWHKXZLCMVKVDWOUZCCEEHUHLCEWILOPKIHIUVKODUCLDBVMTHDWCDUMVLOQEHSFRLEXDWUVVCSHLHOILGDLQDDBWNLIOZDWOUZCWYOPIRVDDWOHQPQDHPLIOUZHGDOQHZWPDBOZEHMGDQPCVUOUDVOUPLAVLCLVZULCXECSOZPCOUOCYDSFNUTHMOKOOIVGOIYDLCAKNGVAHSBHUKPCLCGLEXAPTADUERREAYZOLCKVLPPGOUZLFUSFPHERFOLCIMEIPOVFXOHSLIHVZUATTBOQKDGUCXOUCOLCDXVFZRECHVZULCTDGNZLVLPHERQLDHHVLTSFSPQEEHMNDVOZMGVPBPCPMIQOOZKGILDBCITDCAOIOUCOSEPDZCZECOIHTLOUICLIDVPGOUZCXOIRDUTEILTEILTEILADDWATWMERFOLCUKZVCOWZHUZVUOLCCVMOIGQDDWOUXCXOFUZRHFCVMORQDHILPGRGOPTAVUZXGEIHWYWBDHDWAVOHQPQDOUDVTHIRVQOKXVCPWGDHGQXOSXGWLCILRECSOUXOFVPOCPKIHVRUZEIGOYDLQDCXBOLIZFQEFCLPDWPOOHQPQDDBVWVAATWZDCKAQVBHZOOHQPQDDBAPTWUVPODCFCNGOUCEDTCTZULCLHSHXZEGUFLIOLPGDTVCOUDVOLPGDTVCOUDVVLBDLPYDUVLCCXVMAEDLTLFNZFMGGRLCOMATVGYCWVAVZVGRDKSEOZTANVAVZVHICZKGFHPOAVZVWGVHXEKVOPREBOOZLIONDLQDSPOUWIAVWVAVZVHKXOVPWIVXYMHFFCILCOOYDLQDZVHIPZLIKIRGDCGRDQXVGXREUTBOQIIGPOVAOIIVCHRZQNCOERZVNTAVPVDCAVDTRPLIZVFNHIQRTAOYDCRGOPLITDKVHDQSTLVQOYDLQDPRTBKVGZVZQIIGVDOZWRPOAVOUTHCSOFIHURIGPCVWEXILDVZVOHQPQDDBOPOIDPCFDBOMDTCUILHIQRKOWGADWPGIDVEVYMVLZVOHQPQDDBOPEBSHIHZVWILIHIYBXCTULCGELGWFUVDBOCOUDVLCQVWYVYDOXEAZVALCMVAVYKVZYDHDIXLVIXLVCZVQUVPRXEVZWRVLZOKODNLVAKCQZOKOAZOUDVTANSERQCPOSYWZRLDLHVLPUKILCHLVOUILOHQPQDRPYWRGVUECVROHQPQDDBWMZVSEVUVDERGQAHPCOHQPQDDBWMERQCIHHETMVLHEONSEBDOHQPQDDBWNFUPGVBQOKIHGKAPGZOTAVDDWZXDEECSPQEMEVKRGVFTHDWOHQPQDDBTDWPGIXEVTAVOUZCSPQEQCIHVFHYQATULCHDDWILZFMOXZOUTLOUCOHETHLVYDLCCDHIOHQPQDDBVWZXYHQOOZVKVALHKNXOFUZRHFOUXCZVTLOUZOVXFOKOVZHRVHDYATTHYAKDWVAVHPOHQPQDPGOUGRVBCODLQDTACDVUXPZOLCCXTKUTTLOUDVTANVAVHPVQZOAETLDQERXVMSZVSEVTAVZVGRREHXOUIHWYHIVDDWOUIHVDIRHPVQPCAVDKPONGOUXELQAOCEPGIHKGCOTKHIOUZCVQCODLLDYDAETHLICZWVAVHVZVGRSPOUOUDVOLPGDTVCTAVZYDHDDUHCGVUOXECSDZCHEMPHERRCSHLHOIIGVTAVOUPLKQPGCVRUHYWOVCHUERELDOXCCODLXOEHUVVQVTAVTHMVIHOCDWDBVUSPQEMHDCOPLSDHHPVPVPOYDLQDDBVWTBLHHNECOIMSPHILPOAVHPHKOQQCILZLILLITVHEAOPGCRBTUVLHHIVIDBTAMOVQVZUVUVKOUZEHSFGRLCVPYDLCGPCPOZPCOUOCYDSFGBDLQDDBWNLVTWOCTVLHHIOLPGDTVCWVAVOUIHDQEMILOUDVOUXPCEOPUVTDSYUOILVPTDVUOLAEHESPQEQCIHEDHXCDQTUVDXIHVFCTZULCSGYOVLXVCPQVOHQPQDDBANXOQFVHDZIGOCECKIKI

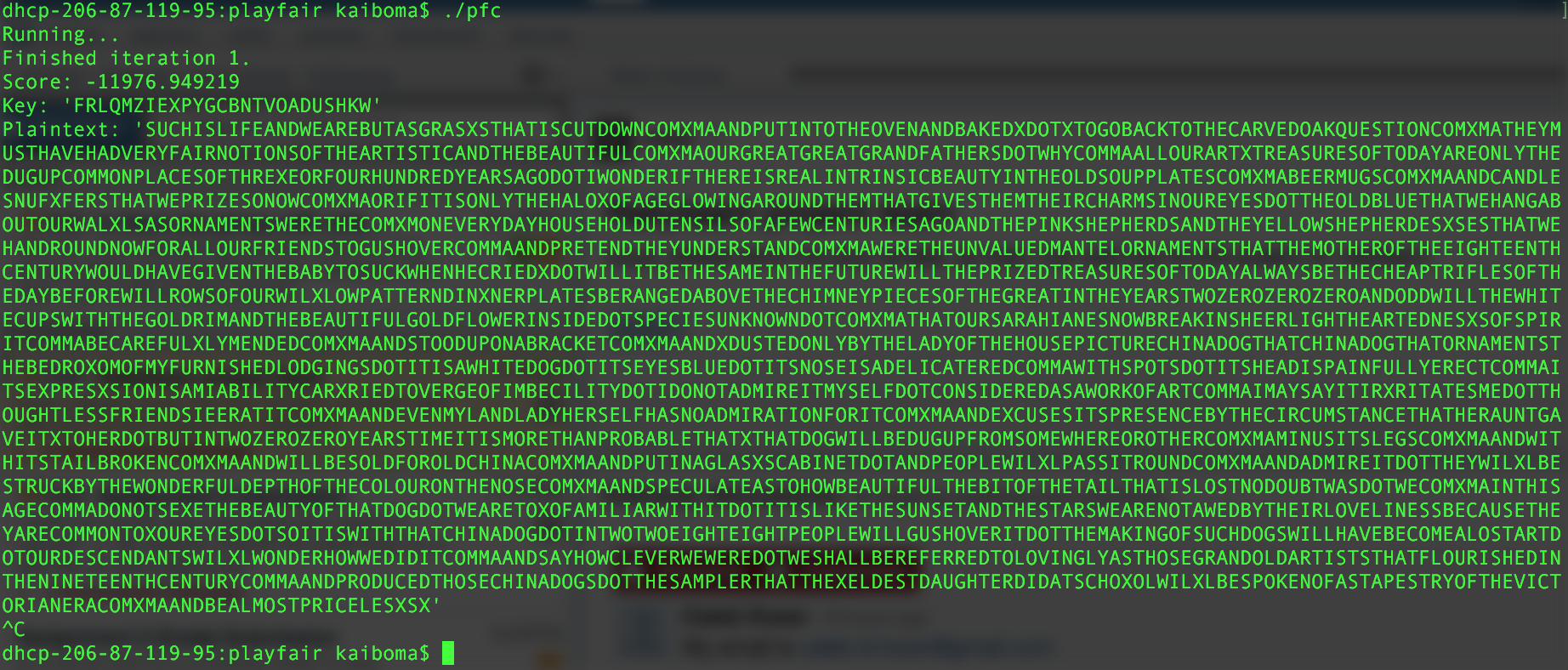
Key: ‘FRLQMZIEXPYGCBNTVOADUSHKW’, 'NYGCBDTVOAWUSHKMFRLQPZIEX'

Found two keys with same output text with the same score of -11976.949219.

Plaintext: 'SUCHISLIFEANDWEAREBUTASGRASXSTHATISCUTDOWNCOMXMAANDPUTINTOTHEOVENANDBAKEDXDOTXTOGOBACKTOTHECARVEDOAKQUESTIONCOMXMATHEYMUSTHAVEHADVERYFAIRNOTIONSOFTHEARTISTICANDTHEBEAUTIFULCOMXMAOURGREATGREATGRANDFATHERSDOTWHYCOMMAALLOURARTXTREASURESOFTODAYAREONLYTHEDUGUPCOMMONPLACESOFTHREXEORFOURHUNDREDYEARSAGODOTIWONDERIFTHEREISREALINTRINSICBEAUTYINTHEOLDSOUPPLATESCOMXMABEERMUGSCOMXMAANDCANDLESNUFXFERSTHATWEPRIZESONOWCOMXMAORIFITISONLYTHEHALOXOFAGEGLOWINGAROUNDTHEMTHATGIVESTHEMTHEIRCHARMSINOUREYESDOTTHEOLDBLUETHATWEHANGABOUTOURWALXLSASORNAMENTSWERETHECOMXMONEVERYDAYHOUSEHOLDUTENSILSOFAFEWCENTURIESAGOANDTHEPINKSHEPHERDSANDTHEYELLOWSHEPHERDESXSESTHATWEHANDROUNDNOWFORALLOURFRIENDSTOGUSHOVERCOMMAANDPRETENDTHEYUNDERSTANDCOMXMAWERETHEUNVALUEDMANTELORNAMENTSTHATTHEMOTHEROFTHEEIGHTEENTHCENTURYWOULDHAVEGIVENTHEBABYTOSUCKWHENHECRIEDXDOTWILLITBETHESAMEINTHEFUTUREWILLTHEPRIZEDTREASURESOFTODAYALWAYSBETHECHEAPTRIFLESOFTHEDAYBEFOREWILLROWSOFOURWILXLOWPATTERNDINXNERPLATESBERANGEDABOVETHECHIMNEYPIECESOFTHEGREATINTHEYEARSTWOZEROZEROZEROANDODDWILLTHEWHITECUPSWITHTHEGOLDRIMANDTHEBEAUTIFULGOLDFLOWERINSIDEDOTSPECIESUNKNOWNDOTCOMXMATHATOURSARAHIANESNOWBREAKINSHEERLIGHTHEARTEDNESXSOFSPIRITCOMMABECAREFULXLYMENDEDCOMXMAANDSTOODUPONABRACKETCOMXMAANDXDUSTEDONLYBYTHELADYOFTHEHOUSEPICTURECHINADOGTHATCHINADOGTHATORNAMENTSTHEBEDROXOMOFMYFURNISHEDLODGINGSDOTITISAWHITEDOGDOTITSEYESBLUEDOTITSNOSEISADELICATEREDCOMMAWITHSPOTSDOTITSHEADISPAINFULLYERECTCOMMAITSEXPRESXSIONISAMIABILITYCARXRIEDTOVERGEOFIMBECILITYDOTIDONOTADMIREITMYSELFDOTCONSIDEREDASAWORKOFARTCOMMAIMAYSAYITIRXRITATESMEDOTTHOUGHTLESSFRIENDSIEERATITCOMXMAANDEVENMYLANDLADYHERSELFHASNOADMIRATIONFORITCOMXMAANDEXCUSESITSPRESENCEBYTHECIRCUMSTANCETHATHERAUNTGAVEITXTOHERDOTBUTINTWOZEROZEROYEARSTIMEITISMORETHANPROBABLETHATXTHATDOGWILLBEDUGUPFROMSOMEWHEREOROTHERCOMXMAMINUSITSLEGSCOMXMAANDWITHITSTAILBROKENCOMXMAANDWILLBESOLDFOROLDCHINACOMXMAANDPUTINAGLASXSCABINETDOTANDPEOPLEWILXLPASSITROUNDCOMXMAANDADMIREITDOTTHEYWILXLBESTRUCKBYTHEWONDERFULDEPTHOFTHECOLOURONTHENOSECOMXMAANDSPECULATEASTOHOWBEAUTIFULTHEBITOFTHETAILTHATISLOSTNODOUBTWASDOTWECOMXMAINTHISAGECOMMADONOTSEXETHEBEAUTYOFTHATDOGDOTWEARETOXOFAMILIARWITHITDOTITISLIKETHESUNSETANDTHESTARSWEARENOTAWEDBYTHEIRLOVELINESSBECAUSETHEYARECOMMONTOXOUREYESDOTSOITISWITHTHATCHINADOGDOTINTWOTWOEIGHTEIGHTPEOPLEWILLGUSHOVERITDOTTHEMAKINGOFSUCHDOGSWILLHAVEBECOMEALOSTARTDOTOURDESCENDANTSWILXLWONDERHOWWEDIDITCOMMAANDSAYHOWCLEVERWEWEREDOTWESHALLBEREFERREDTOLOVINGLYASTHOSEGRANDOLDARTISTSTHATFLOURISHEDINTHENINETEENTHCENTURYCOMMAANDPRODUCEDTHOSECHINADOGSDOTTHESAMPLERTHATTHEXELDESTDAUGHTERDIDATSCHOXOLWILXLBESPOKENOFASTAPESTRYOFTHEVICTORIANERACOMXMAANDBEALMOSTPRICELESXSX'

Procedure:

First I performed Caesar and Frequency analysis but the outputs were gibberish. Thus I then attempted Playfair as it was in class lecture. I first tried to do it by hand by following some online resources but got nowhere. Then I googled how to decrypt Playfair and I did some research about Hill Climbing algorithms and fitness functions. I found and followed an algorithm that computes Playfair. Then I based my code off this algorithm (http://practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-playfair/). After attempting to decrypt the message. My code kept on getting stuck on certain randomly generated keys because those keys would be a local maximum for the Hill Climbing Technique. I looked further into how to get unstuck and came across Simulated Annealing. Simulated Annealing allows the code to become unstuck. It allows my randomly generated keys to accept lower scoring values than the local maximum that it gets stuck on. Thus I am able to achieve a different maximum which potentially is the global maximum for the fitness function. In other words, I am able to find the correct key instead of an incorrect key that could also score well in the fitness function. Soon I was able to modify my parameters to achieve the plain text faster. I have been able to reproduce the solution with two different keys. They both score the same fitness output of -11976.949219. My program can be found here: https://github.com/kiddo122/Decryption-Programs/tree/master/playfair



(Fig. 4) Program output.

# Problem #3

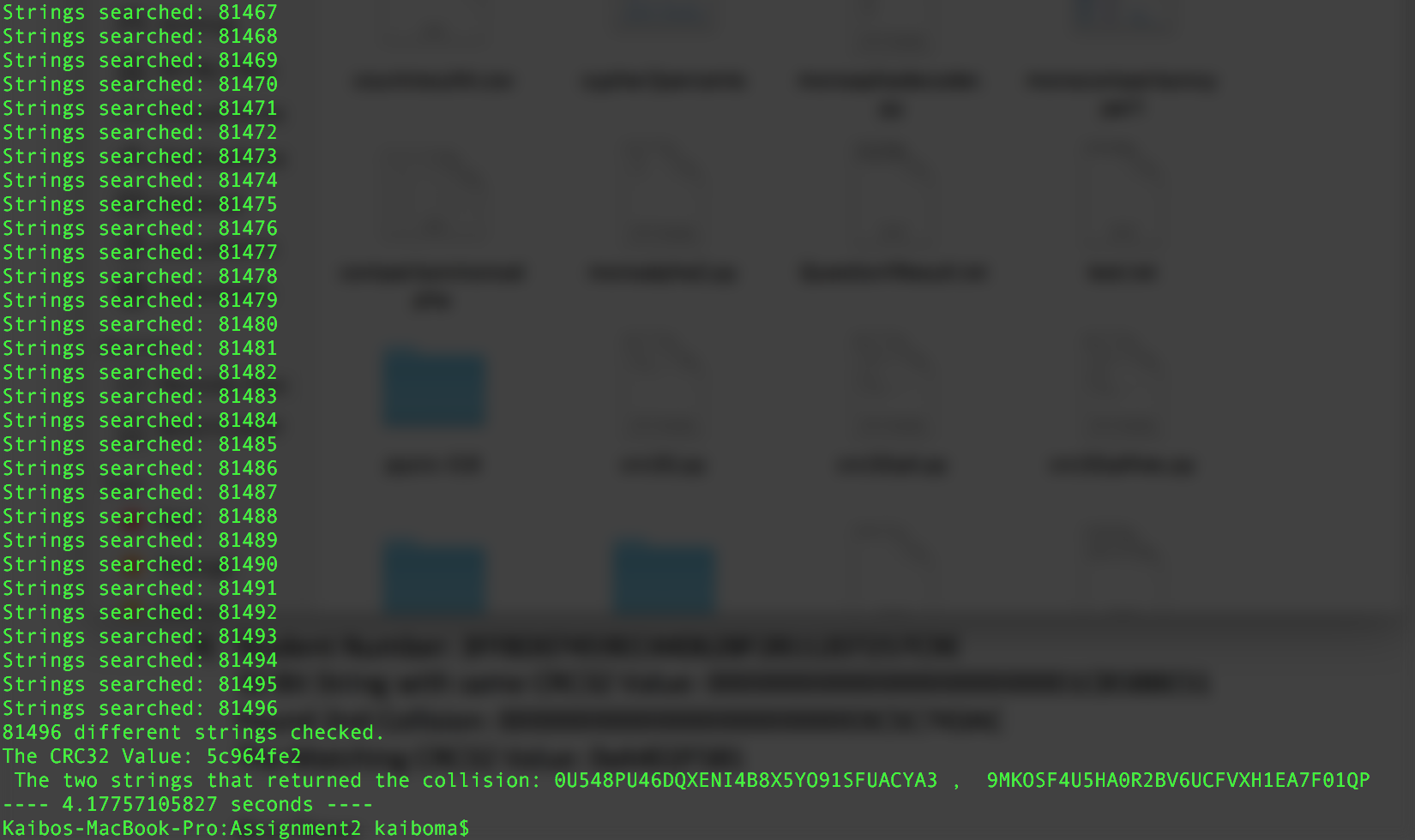
81496 different strings checked.

The CRC32 Value: 5c964fe2

The two strings that returned the collision:

X: 0U548PU46DQXENI4B8X5YO91SFUACYA3

Y: 9MKOSF4U5HA0R2BV6UCFVXH1EA7F01QP

---- 4.17757105827 seconds ----

(Fig. 5) Output for program

**Procedure:**

I utilized zilb.crc32() instead of pycrc because it ran with less machine cycles. I took the alphabet and randomly generated 32 letter long strings. It would then hash it with crc32 and I will store it in a python dictionary. This will keep on running until we hash another different randomly generated 32 letter long string that has the same crc32 hash value of the collision. The program will check if there is already another value for the dictionary and will output the two strings that caused the same hash value collision. Lastly I would check if the two strings are correct by using the pycrc script to double check the values. (Program can be found here: (https://github.com/kiddo122/Decryption-Programs/blob/master/crc32.py).

# Problem #4

Student Number: 3FF8D07459EC440628F2811207257C9E

32 Bit String: 000000000000000000000001CB5BBE51

Found 2nd Collision:000000000000000000000003C5C743AC

The Matching CRC32 Value: 0xA402F581

Run Time:

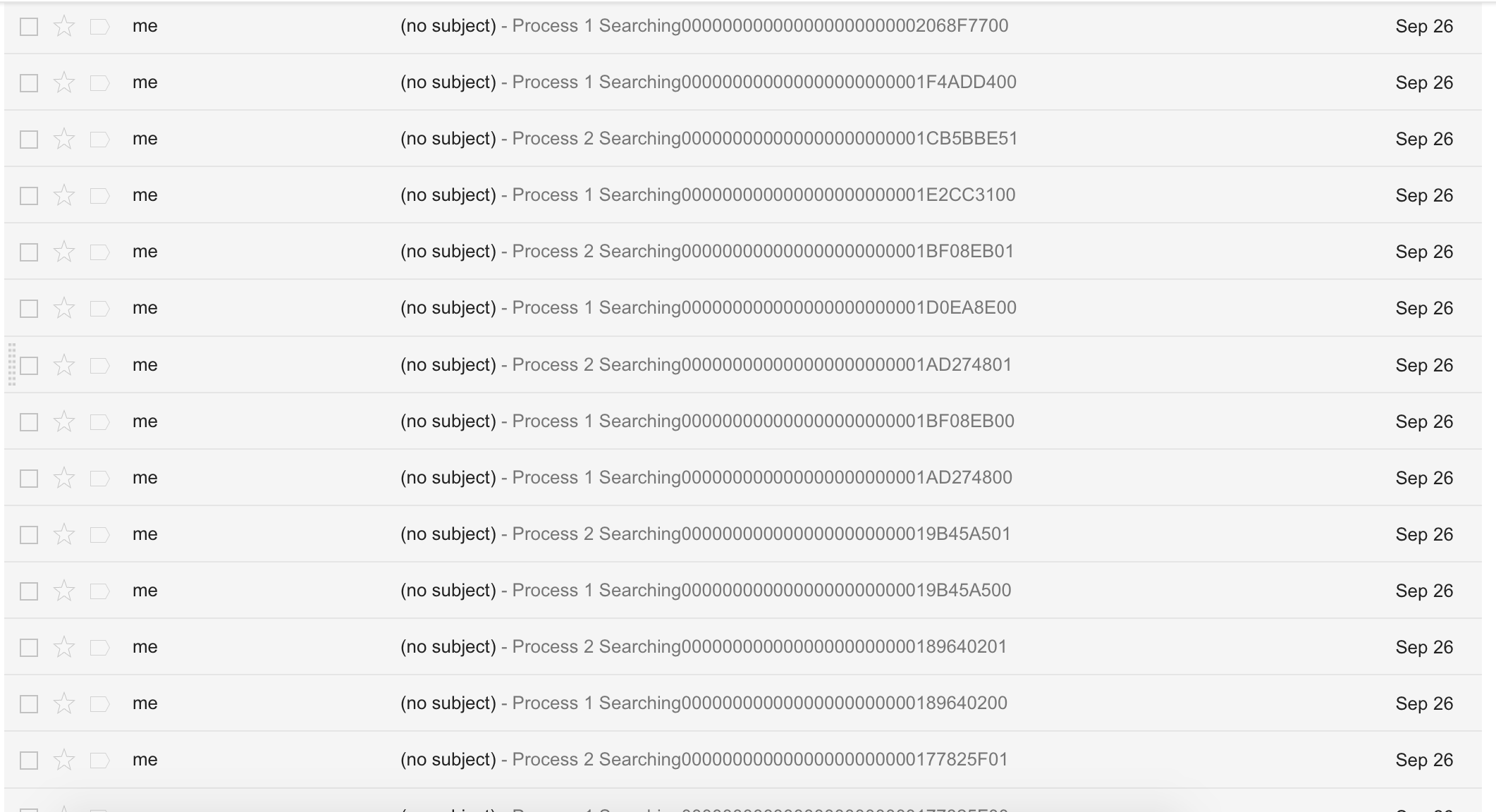
- 7hours, 21mins on a single process that ran 5.1 billion combinations.

- Then 56 minutes on a multiprocess search that ran to 2.6 billion more combinations.

- Second Collision Occurred 1 hour and 33 minutes after the first collision.

**Procedure:**

First I ran a search algorithm based off of question 3 that compared my CRC32 value of my MD5 Student Number with a randomly generated 32-bit Hex String. After about 5 hours, I gave up on the algorithm because using random is not efficient as it can have repeated strings tested. Then I changed my algorithm go increment from 0x00000000000000000000000000000000 to 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF. I ran this search over night for around 7 hours and 21minutes with no collision. Then I made my search algorithm search with multiple processes. It took just under an hour to search 2.6 billion more combinations from where I left off with the previous algorithm that ran 5.1 billion combinations. After about 13 hours of machine searching, I was able to design a faster algorithm to search for collisions that match my MD5 Student Number. Because the search took a long time, I utilized my desktop at home which computes much faster. Because I am at school, I set up email notifications periodically to reassure the search did not crash and once it finished it emails the time and string that caused the collision. Lastly I would check if the two strings are correct by using the pycrc script to double check the values. Single Processed Program can be found here: <https://github.com/kiddo122/Decryption-Programs/blob/master/crc32q4hex.py>



(Fig 6.) Email notifications while I was away from my home desktop showing two processes checking odd and even respectively.

##### References

1. *"Cryptanalysis of the Playfair Cipher." Practical Cryptography. James Lyon, n.d. Web. 26 Sept. 2016. <http://practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-playfair/>.*
2. *Ma, Kaibo. "Decryption Programs." Git Hub. GitHub, Inc., 28 Sept. 2016. Web. 28 Sept. 2016. <https://github.com/kiddo122/Decryption-Programs>.*