Project 1
Implementing the Encryption and Decryption Algorithm using Vignere Cipher

Assigned: September 14, 2015                                  Due: October 14, 2015, 4 PM

Goal of the Project
In this project, you will develop the code to encrypt and decrypt using Vignere Cipher. Given a plaintext
and a string of characters as key, you will implement the Vignere Cipher encryption algorithm to compute
the ciphertext. Similarly, given the ciphertext and a string of characters as key, you will implement the
Vignere Cipher decryption algorithm to generate back the plaintext.

Vignere Cipher
In this project, you will implement the Vignere Cipher, a stream cipher: so, you will encrypt one character
at a time. Assume for simplicity, all characters to be encrypted and decrypted are only uppercase
characters (A-Z). The Vignere Cipher takes a character-string (all uppercase characters: A-Z) as the key.

Encryption algorithm
A key string is written parallel along with the plaintext. If the end of the key is reached, the key string is
continued by repeating the key. This is continued till the last plaintext character is reached.

If you encounter a non-uppercase plaintext character, do not use the character from the key string, just
skip that plaintext character and proceed to the next plaintext character. In other words, the non-uppercase
plaintext character appears as it is in the ciphertext too.

For every character index i, $C[i] = P[i] + K[i]$

Decryption algorithm
A key string is written parallel along with the ciphertext. If the end of the key is reached, the key string is
continued by repeating the key. This is continued till the last ciphertext character is reached.

If you encounter a non-uppercase ciphertext character, do not use the character from the key string, just
skip that ciphertext character and proceed to the next ciphertext character. In other words, the non-
uppercase ciphertext character appears as it is in the plaintext too.

For every character index i, $P[i] = C[i] - K[i]$

Example on the working of Vignere Cipher

Key: ZEBRA

Plaintext: NATARAJAN MEGHANATHAN
Ciphertext: MEURZNBE MDKI RNZX I RN

| Plaintext | N | A | T | A | R | A | J | A | N | M | E | G | H | A | N | A | T | H | A | N |
| KeyString | Z | E | B | R | A | Z | E | B | R | A | Z | E | B | R | A | Z | E | B | R | A |
| Ciphertext| M | E | U | R | R | Z | N | B | E | M | D | K | I | R | N | Z | X | I | R | N |
Plaintext and Key to be Used:

The Plaintext is your first name followed by your last name (with a blank space in between) as my name in the example shown in the previous page.
Key to be used: the city where you reside (e.g., BRANDON)

What is required for you to do?

You need to implement the encryption algorithm and decryption algorithm for Vignere Cipher as described above, as two separate programs (one program for encryption and another for decryption).

Your encryption algorithm should input the plaintext (your full name, with a blank space in between your first name and last name) as a string of uppercase characters and a one-word character string as the key (the city you reside). Implement the encryption algorithm as described above to generate the ciphertext. Your encryption algorithm program should output the ciphertext.

Your decryption algorithm should input the ciphertext generated from the encryption algorithm as a string of uppercase characters and the one-word character string as the key (the city you reside). Implement the decryption algorithm as described above to get back the plaintext. Your decryption algorithm program should output the plaintext.

What to submit?

Hardcopy (submit in class on the due date/time) and Softcopy (email me: natarajan.meghanathan@jsums.edu)
1) The code for the encryption algorithm
2) A sample screenshot showing the encryption algorithm taking the input plaintext and the one-word character string as the key to generate the ciphertext.
3) The code for the decryption algorithm
4) A sample screenshot showing the decryption algorithm taking the ciphertext generated from the encryption algorithm and the one-word character string as the key to get back the plaintext.

Video
A Desktop recorded video (displaying your code, in full or in parts) with your explanation on the different sections of the code (starting from the execution in the main function) and walk through the code explaining how it will be executed starting from the input phase to the output phase (both for the encryption and decryption algorithms). You should show the execution of the programs for the input assigned to you.
Note that even though I am not specifying a minimum time for your video, your video explanation is expected to last at least for 8-10 minutes and should cover all of the above required explanations. Note that the contents of the desktop/programs captured through your video should be clearly readable.

Upload the video to your JSU email account using Google Drive and email me (natarajan.meghanathan@jsums.edu) the link to download it.

You could try using one of the desktop recording software (or anything of your choice):
CamStudio: http://sourceforge.net/projects/camstudio/files/legacy/
CSC 323 Algorithm Design and Analysis
Spring 2015
Instructor: Dr. Natarajan Meghanathan

Project 6
Theorem Proving Project: Bloom Filters, Breadth First Search and Asymptotic Complexity

Submission:
(1) Hardcopy report of your proofs for each task
(2) Desktop recorded videos of the explanation of the proofs, uploaded through Google Drive to my email address: natarajan.meghanathan@jsms.edu

Due: April 9, 2015: 1 PM
Max. Points: 100

In this project, you will prove things as required below in each of the tasks and record your proof in a video (desktop recording). You will submit a hardcopy report of your proofs for each task and submit the video through Google Drive. Try to submit just a single video that includes a recording of your proof for all the three tasks (instead of three separate videos).

Task 1 (40 points): Bloom filters are considered an application of hash tables. Bloom filters are used for proactive password cracking - to identify whether a password entered by a user at the time of registration is vulnerable for cracking or not. To do so, the Bloom filter maintains the hash values for a potentially vulnerable list of passwords (commonly used passwords, words from dictionary, etc) and if the hash value of the user entered password matches to those in the Bloom filter, the user password is not accepted as part of the registration process and the user is forced to choose a password whose hash value does not match to the one in the Bloom filter.

For the purpose of this project, you could envision a Bloom filter that calculates the hash value for a word as the sum of the ascii values of the characters in the word divided by a prime number; the size of the Bloom filter (hash table) is the magnitude of the divisor prime number. You can assume either a closed or open hash table.

A false positive scenario is one wherein the user entered password is not vulnerable; but a Bloom filter flags the password as vulnerable. A false negative scenario is one wherein the user entered password is vulnerable; but a Bloom filter does not flag the password as vulnerable.

Given the above description of Bloom filters, prove that the Bloom filters cannot have a false negative scenario, but could have a false positive scenario. You could show your proof with examples and record a video.

Task 2 (30 points): Prove that when Breadth First Search (BFS) is conducted on a graph starting from a particular vertex \( s \), we are guaranteed to find the minimum hop path (paths with the minimum number of intermediate edges) from \( s \) to every other vertex in the graph.

Task 3 (30 points): In Module 1, we proved in class that if \( t_1(n) \in O(g_1(n)) \) and \( t_2(n) \in O(g_2(n)) \), then: \( t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\}) \). Review this proof and on similar lines, prove the following and record a video explaining your proof.

If \( t_1(n) \in \Omega(g_1(n)) \) and \( t_2(n) \in \Omega(g_2(n)) \), then prove that: \( t_1(n) + t_2(n) \in \Omega(\min\{g_1(n), g_2(n)\}) \)
In this project, you will primarily use the Gephi tool to analyze and visualize networks. Gephi is a complete stand-alone application (available at: http://gephi.github.io/) that you can download and install on your computer. You will analyze two networks: Your Facebook network of friends and a classical network (assigned to each of you) that has been commonly used for research in Network Science.

**Facebook network analysis:** As illustrated in one of my demo videos, go to your Facebook account, use netvizz to download your connections as a GDF file. Load the GDF file in Gephi and analyze it as demonstrated in the demo video.

**Classical network analysis:** The following networks are assigned for each student. If you get the .gml files for your network, you could directly load them to Gephi and analyze. If you get the topology information as nodes and edges, copy and paste them appropriately in Excel, delete the unnecessary columns; insert the required columns titles and save them as .csv files (look at the demo videos for how I do these). Then, go to the Data laboratory of your project in Gephi, upload the nodes and edges .csv files, visualize and analyze the networks.

Data for a majority of the networks assigned in this project are available at: http://www-personal.umich.edu/~mejn/netdata/

For the Soccer World Cup 98 network data, visit: http://vlado.fmf.uni-lj.si/pub/networks/data/sport/football.net

<table>
<thead>
<tr>
<th>Student name</th>
<th>Network assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Susmita Atluri</td>
<td>Word adjacencies</td>
</tr>
<tr>
<td>Bharath Gajjela</td>
<td>Dolphins' Social Network in NZ</td>
</tr>
<tr>
<td>Pratik Jannela</td>
<td>American College Football Network</td>
</tr>
<tr>
<td>Gulam M. Khan</td>
<td>Karate Network</td>
</tr>
<tr>
<td>Joel Maddirala</td>
<td>Books about US Politics</td>
</tr>
<tr>
<td>Ankamma Ravi</td>
<td>Les Miserables</td>
</tr>
<tr>
<td>Yuxiao Zhou</td>
<td>Soccer World Cup 98</td>
</tr>
<tr>
<td>Alain Rafiki</td>
<td>Word adjacencies</td>
</tr>
</tbody>
</table>

**Network Metrics:** For each of the above two network analysis, you will determine the following and show the appropriate visualization:

1. Degree distribution (node degree vs. probability of node degree) and a plot of the same in Excel
2. A suitable network layout depicting the tradeoff and/or correlation between node degree and local cluster coefficient
3. A suitable network layout depicting the tradeoff and/or correlation between the authority scores and the PageRank values
4. A suitable network layout depicting the tradeoff and/or correlation between Closeness Centrality and Betweenness Centrality values
(5) A suitable network layout depicting the tradeoff and/or correlation between the Hub and Authority scores.
(6) A suitable network layout depicting the different communities of nodes in your network and the Eigenvector Centrality values of the nodes.
(7) What is your average path length, network diameter and modularity score?

Submission:
Report and discussion: Compile a report for the network metrics evaluated for your Facebook network and the classical network assigned to you. Include screenshots for all the figures and layouts.
Video(s): Record video(s) demonstrating your analysis of the Facebook network and the classical network. If the demonstration runs for a longer time, you could record separate videos (one for each of the two networks) and upload them to GoogleDrive sent to my email address: natarajan.meghanathan@jsums.edu.