Problem 1: Cryptography: An Old Russian Puzzle

(A) Bob wants to send a gift to Alice through the mail. But postmen are corrupt—they open each package and steal expensive contents before sending it on. Bob decides to lock the package before sending it, using a padlock that has only one key. But this does not quite solve the problem—he cannot send the key to Alice securely!

Devise a way for Bob to send the gift securely without using any fancy locks, so that only Alice can open it. You can assume that Alice and Bob can talk on phone, Alice also has access to a padlock that has only one key, and the package can be sent back and forth multiple times. Furthermore, the package can be locked using more than one lock.

(B) Imagine using a scheme similar to your solution for part A to encrypt data. Say Bob wants to send an encrypted message to Alice. He has access to an encryption function $E_B$ that can only be decrypted by him. Alice also has access to an encryption function $E_A$ that can only be decrypted by her. Explain how the message is delivered. What properties should the functions $E_A$ and $E_B$ satisfy for your scheme to work?

(C) Among the encryption schemes studied in class, suggest one that can be used by Alice and Bob in part B above, such that they do not need to exchange any keys, or even disclose their public keys.
Problem 2: Cryptography/ECC : McEliece Cryptosystem

The McEliece Cryptosystem is one of the earliest public-key systems. It is based on linear error-correcting codes. Like the Merkle-Hellman Cryptosystem it uses an easy special case of an NP-complete problem which is disguised as a general case.

Recall that a linear code can be defined by a \( k \times n \) binary generator matrix \( G \), where \( k \) is the size of the plaintext, and \( n \) is the size of the codewords. A plaintext \( x \) is converted into a codeword \( y \) using \( y = xG \). The rows of \( G \) form the basis of the code \( G \), and the minimum Hamming distance between any two codewords generated by \( G \) is called the distance of \( G \). A code \( k \times n \) code with distance \( d \) is called an \([n, k, d]\) code.

The NP-complete problem used by McEliece is the following. Given a \( k \times n \) generator \( G \) and a \( n \)-bit word \( w \), find the nearest codeword (Hamming distance) that can be generated by \( G \). This is exactly what error correcting codes are supposed to find. Fortunately, for certain error correcting codes, solving this problem is easy. The trick of McEliece is therefore to use one of these easy codes and transform it to look like the general case.

McEliece suggested using Goppa codes as the “easy” codes. Goppa codes have the form \([n, k, d] = [2^m, 2^m - mt, 2t + 1]\) (McEliece suggested taking \( m = 10 \) and \( t = 50 \) giving \([1024, 524, 101]\)). Given a word \( w \) of length \( n \), and a Goppa code \( G \), there is a polynomial-time (in \( n \)) algorithm for finding the nearest codeword assuming the number of errors is \( t \) or less.

Encryption in McEliece’s system is based on an \( k \times n \) Goppa code \( G \), a \( n \times n \) permutation matrix \( P \), and an \( k \times k \) invertible binary-matrix \( S \). These three matrices are all part of the private key \((G, S, P)\). The public key is the matrix \( G' = SG \). This is itself an \( k \times n \) linear code, but it is not a Goppa code. A \( k \)-bit word \( x \) is encrypted using \( w = xG' + e \) where \( e \) is a random \( n \)-bit vector of weight \( t \) (i.e., it contains \( t \) 1’s). The hope is that given that \( G' \) is not a Goppa code, it is hard to get \( x \) back from \( w \). No one has yet broken this method (i.e. found a polynomial-time solution) or proved that it is secure.

A. Describe how to decode using McEliece’s method in polynomial time given the private key.

B. Given unbounded memory and unbounded preprocessing time for a given public-key, explain how you can set things up so that you can quickly decode messages \( w \). As a function of \( n \) and \( t \), how much memory would this take (try to minimize this, but don’t get carried away by trying to compress the data)?

C. Given a \([1024, 524, 101]\) Goppa code, assume you want to correct up to 10 bit-errors as well as use it for encryption. How do you change the encryption algorithm? How does this affect security?

D. What are some advantages or disadvantages of the McEliece cryptosystem as compared to RSA? Don’t worry about time, but you can assume that using a \([1024, 524, 101]\) Goppa code gives about the same security as a 766-bit RSA modulus. (I’m thinking of one advantage and three disadvantages, but yours don’t have to match mine.)

Problem 3: String Matching: Too Many Repetitions

We say that some string \( s \) is \( k \)-repeated in \( S \) if there are \( k \) distinct positions \( i_1, i_2, \ldots, i_k \) such that \( s = S[i_j : i_j + |s| - 1] \) for each \( 1 \leq j \leq k \). Let \( K_S(s) \) be the largest value \( k \) such that \( s \) is \( k \)-repeated in \( S \).

Given a string \( S \), give an efficient algorithm to output a substring \( s \) that maximizes the quantity \(|s| \times K_S(s)|

Please make this as clear as possible. It should not take more than a paragraph.
Problem 4: Partitioning via Integer Programming

Here’s an IP formulation of the $s - t$ minimum cut problem (that is, find the cheapest set of edges whose removal disconnects $s$ from $t$). Our input is an undirected graph $G = (V, E)$ with costs $c : E \rightarrow \mathbb{N}$. ($G$ has no self loops). Below, our variables are $\{d_{sv} | v \in V\}$ and $\{x_e | e \in E\}$. For edge $e = \{u, v\}$, we let $x_e$ and $x_{\{u,v\}}$ denote the same variable.

$$\begin{align*}
\text{min} & \sum_e c(e)x_e \\
d_{ss} &= 0 \\
d_{st} &\geq 1 \\
d_{sv} &\leq d_{su} + x_{\{u,v\}} \quad \text{for all } \{u, v\} \in E \\
0 &\leq x_e \leq 1 \quad \text{for all } e \in E \\
x_e &\in \mathbb{Z} \quad \text{for all } e \in E
\end{align*}$$

(i) Assume $c(e) > 0$ for each edge $e$. Show that the above IP formulation is correct. Specifically, prove that there is a (simple, easy to compute in $O(|E|)$ time) mapping between optimal solutions to the IP given above and minimum $s - t$ cuts in $G$. Give an $O(|E|)$ time algorithm to compute a minimum $s - t$ cut in $G$ given an optimal solution $(x, d)$ to the IP above. (Hint: We’d like variable $d_{sv}$ to equal the shortest path distance from $s$ to $v$ in $G$ with edge lengths $x_e$. The IP doesn’t quite enforce that. However, note that $d$ doesn’t appear in the objective function, so given any $(x, d)$ you might be able to modify $d$ without changing the value of the solution.)

(ii) The ideas inherent in the IP formulation above can be extended to yield IP encodings for various partitioning and clustering problems. For this part of the problem, write down an IP formulation for the following graph partitioning problem (GPP): Given undirected graph $G = (V, E)$, edge costs $c(e)$, and terminals $t_1, t_2, \ldots, t_k \in V$, find the minimum cost set of edges $A$ to remove from $E$ so that in the remaining graph $G' = (V, E \setminus A)$ no two terminals are connected by a path. Your formulation should have size polynomial in the size of $G$. (FYI, one application of this problem is to a particular image segmentation problem, when you have several regions of the image and $k$ points which you know belong to distinct regions, and you wish to find the region boundaries).

Prove your formulation is correct. Give a high level description of your IP: list the variables and what they are supposed to represent, give a one sentence description of what each (non-trivial) constraint represents, and give some brief intuition of why it should be correct. Assume $c(e) > 0$ for each edge $e$. Specifically, provide an efficient algorithm which, when given an optimal IP solution, constructs an optimal GPP solution. (Ideally, this algorithm should be very simple). Prove its correctness.