VIETNAM GENERAL CONFEDERATION OF LABOR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

FINAL REPORT
DISCRETE STRUCTURES

Instructor: **NGUYEN QUOC BINH**

Executor: **VO NHAT HAO – 522H0090**

DANG THANH NHAN – 522H0006

Class : **22H50202 – 22H50201**

Course : **26**

HO CHI MINH CITY , YEAR 2024

VIETNAM GENERAL CONFEDERATION OF LABOR
TON DUC THANG UNIVERSITY
FACULTY OF INFORMATION TECHNOLOGY

FINAL REPORT

DISCRETE STRUCTURES

Instructor: **NGUYEN QUOC BINH**

Executor: **VO NHAT HAO – 522H0090**

DANG THANH NHAN – 522H0006

Class : **22H50202 – 22H50201**

Course : **26**

HO CHI MINH CITY, YEAR 2024

THANK YOU

- We are deeply grateful to Mr. Nguyen Quoc Binh for her constant support and enthusiastic direction throughout our investigation and final report.
- We also like to thank Ton Duc Thang University's Faculty of Information Technology for providing us with an enriching academic environment. The faculty's willingness to share vital expertise and reference materials has not only aided our research endeavor, but has also improved our overall educational experience at the university.
- As we wrap up our study project, we reflect on the vital lessons and insights learned from our educators. Regardless of our limitations and areas for improvement, we are willing to learn and grow. We really seek further assistance to improve our work and appreciate the critical input from our professors and classmates. With their continuous assistance, we are determined to improve our research talents in future initiatives.
- We wish all of our teachers and friends ongoing health and happiness, as their support and care have been invaluable to us on our path.

WE THANK YOU!

THE PROJECT IS COMPLETED
AT TON DUC THANG UNIVERSITY

I hereby declare that this is my own project product and is guided by Mr. Nguyen Quoc Binh. The research content and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments, and evaluation were collected by the author from different sources and clearly stated in the reference section.

In addition, the project also uses a number of comments, assessments as well as data from other authors and other organizations, all with citations and source notes.

If any fraud is discovered, I will take full responsibility for the content of my project. Ton Duc Thang University is not involved in copyright violations caused by me during the implementation process (if any).

Ho Chi Minh City, 25 May 2024

Author

(sign and write full name)

Vo Nhat Hao

Dang Thanh Nhan

INSTRUCTOR VERIFICATION AND EVALUATION SECTION

Confirmation from the instructor

Ho Chi Minh City, day month year

(sign and write full name)

The teacher's evaluation part marks the test

Ho Chi Minh City, day month year

(sign and write full name)

SUMMARY

I. Euclid’s Algorithm and Bezout’s Identity

- Utilize Euclid's algorithm to calculate the gcd and lcm of numbers 2024 and 1000 + m, where m is the last three digits of your student ID.
- Find five integer solution pairs (x, y) that satisfy the linear equation derived from the gcd result.

II. Recurrence Relation

- Solve the recurrence relation $a_n = 8. a_{n-1} - 15. a_{n-2}$ with initial conditions $a_0 = 5$ and $a_1 = m$.

III. Set Operations

- Create a set of characters from your case-insensitive, non-diacritical full name.
- Perform operations like union, intersection, non-symmetric difference, and symmetric difference between this set and another predefined set.

IV. Relations

- Analyze a binary relation defined on integers involving divisibility by m.
- Determine if this relation is reflexive, symmetric, antisymmetric, and transitive.

V. Kruskal’s Algorithm

- Propose a method for circuit checking in Kruskal's algorithm to ensure that no cycles are formed while selecting edges.
- Provide an example to illustrate this method.

VI. Eulerian Circuit

- Determine whether a provided graph has an Eulerian circuit or path.
- Discuss Hierholzer’s algorithm for finding an Eulerian circuit and apply it if applicable.

VII. Map Coloring

- Model a provided map by a graph.
- Apply graph coloring to color the map with a minimum number of colors, using specific conditions based on the last four digits of your student ID.

VIII. Finding an Inverse Modulo n

- Study and describe the process of finding an inverse modulo n using the extended Euclidean algorithm.
- Implement a Python program to perform this calculation and verify its correctness with examples.

IX. RSA Cryptosystem

- Conduct research on the RSA cryptosystem, including its mathematical foundations.
- Develop and test a Python program for RSA encryption and decryption.
- Analyze the efficiency and security of your implementation, discussing potential threats and providing improvement recommendations.

TABLE OF CONTENTS

CHAPTER – 1: EUCLID'S ALGORITHM AND BEZOUT'S IDENTITY

a) **Using Euclid's algorithm to calculate gcd(2024, 1000 + m) and lcm(2024, 1000 + m), where m is the last 3 digits of your student ID.**

(Student code 522H0006 is the smallest of the group's 3 student codes, we have gcd(2024, 1006) and lcm(2024, 1006)).

- gcd(2024, 1006) → gcd(1006, 12) => 2024 = 1006 × 2 + 12

  → gcd(12, 10) => 1006 = 12 × 83 + 10

  → gcd(10, 2) => 12 = 10 × 1 + 2

  → gcd(2, 0) => 10 = 2 × 5 + 0

➔ **Thus gcd(2024, 1006) = 2**

- lcm(2024, 1006)

  2024 = $2^3 \times 11 \times 23$

  1006 = 2×503

➔ lcm(2024, 1006) = $2^3 \times 11 \times 23 \times 503$ = 1018072

- **Another way:** lcm (a, b) = $\frac{(a)(b)}{\text{gcd }(a,b)}$

  $\Longrightarrow$ lcm(2024, 1006) = $\frac{2024.1006}{2} = 1018072$

b) **Apply above result(s) in to find 5 integer solution pairs (x,y) of this equation:**

2024x + (1000 + m)y = gcd(2024, 1000 + m)

We have: 2024x + 1006y = 2

$2 = 12 - 10 \times 1 = 12 + 10 \times (-1)$

$= 12 + (1006 - 12 \times 83) \times (-1) = 1006 \times (-1) + 12 \times 84$

$= 1006 \times (-1) + (2024 - 1006 \times 2) \times 84 = 2024 \times 84 + 1006 \times (-169)$

➔ Thus: $2 = 2024 \times 84 + 1006 \times (-169)$

➔ x = 84, y = -169, d = 2

$(x + \frac{kb}{d}, y - \frac{ka}{d})$, where k is any integer.

⇔ $\left(84+\frac{1006k}{2},\ -169-\frac{2024k}{2}\right)$

- k = 0: $\left(84+\frac{1006\times 0}{2},\ -169-\frac{2024\times 0}{2}\right)=(84,-169)$
- k = 1: $\left(84+\frac{1006\times 1}{2},\ -169-\frac{2024\times 1}{2}\right)=(587,-1181)$
- k = −1: $\left(84+\frac{1006\times(-1)}{2},\ -169-\frac{2024\times(-1)}{2}\right)=(-419,843)$
- k = 2: $\left(84+\frac{1006\times 2}{2},\ -169-\frac{2024\times 2}{2}\right)=(1090,-2193)$
- k = −2: $\left(84+\frac{1006\times(-2)}{2},\ -169-\frac{2024\times(-2)}{2}\right)=(-922,1855)$

➔ The 5 pairs of integer solutions (x,y) of the equation are:

$(84,-169)$, $(587,-1181)$, $(-419,843)$, $(1090,-2193)$, $(-922,1855)$.

CHAPTER – 2: RECURRENCE RELATION

Solve this recurrence relation.

$$a_n = 8.\ a_{n-1} - 15.\ a_{n-2}$$

with a_0 = 5 and a_1 = m, where m is the last 2 digits of your student ID. We have 522H0006 then a_1 = 6.

- Hence: $t^2 - 8t + 15 = 0$

  $\Delta = b^2 - 4ac$

  $= (-8)^2 - 4.1.15$

  $= (-8)^2 - 4.1.15$

  $= 4$

  - $t_1 = \frac{-b+\sqrt{\Delta}}{2a} = \frac{8+\sqrt{4}}{2.1} = 5$
  - $t_2 = \frac{-b-\sqrt{\Delta}}{2a} = \frac{8-\sqrt{4}}{2.1} = 3$

- Explicit formula:

  $a_n = C.t_1^n + D.t_2^n$

  $a_n = C.5^n + D.3^n$

- We have:

  $a_0 = C + D = 5$

  $a_1 = 5C + 3D = 6$

  $$\begin{cases} C = -\dfrac{9}{2} \\ D = \dfrac{19}{2} \end{cases}$$

- Thus: $a_n = -\frac{9}{2}.5^n + \frac{19}{2}.3^n = -\frac{45^n}{2} + \frac{57^n}{2}, \forall\, n \geq 0$

CHAPTER – 3: SET

a) **Create a set Γ of characters from your case-insensitive non-diacritical full name.**

Student code 522H0006 has full name Dang Thanh Nhan then $\Gamma = \{A, D, N, G, T, H\}$

b) **Find the union, intersect, non-symmetric difference, and symmetric difference of Γ and Δ, where Γ and Δ are from question 3a.**

$$\Gamma = \{A, D, G, H, N, T\}$$

$$\Delta = \{A, C, D, G, H, N, O, T, U\}$$

- Union: $\Gamma \cup \Delta = \{A, D, N, G, T, H, C, O, U\}$
- Intersection: $\Gamma \cap \Delta = \{A, D, N, G, T, H\}$
- Non-symmetric difference: $\Delta - \Gamma = \{C, O, U\}$
- Symmetric difference: $\Gamma \oplus \Delta = \{C, O, U\}$

CHAPTER – 4: RELATIONS

Student ID is 5220006 then the valid binary relation is: $\forall a, b \in$ N (aRb↔06|$(a.b)$)

- **Reflexive:**
  - A relation R is reflexive if every element is related to itself: $\forall\ a \in N, aRa$
  - For R to be reflexive: $aRa \leftrightarrow 6 \mid (a.a)$
  - This means 6 must divide a^2 for all $a \in N$.
  - However, 6 is not a factor of a^2 for all a. For example, if a = 1, a^2 = 1 which is not divisible by 6.
  - Thus, R is not reflexive.
- **Symmetric:**
  - A relation R is symmetric if $\forall\ a, b \in N, aRb \Longrightarrow bRa$.
  - For R to be symmetric: $aRb \leftrightarrow 6 \mid (a.b) \Longrightarrow bRa \leftrightarrow 6 \mid (b.a)$
  - Since multiplication is commutative $(a \cdot b = b \cdot a)$, if 6 divides $a \cdot b$, it also divides $b \cdot a$
  - Thus, R is symmetric.
- **Anti $-$ symmetric**:
  - A relation R is anti-symmetric if $\forall\ a, b \in N, (aRb \wedge bRa) \Longrightarrow a = b$.
  - For R to be anti-symmetric: $aRb \leftrightarrow 6 \mid (a.b)$ and $bRa \leftrightarrow 6 \mid (b.a) \Longrightarrow$ a = b
  - However, this is not necessarily true. For instance, if $a = 1$ and $b = 6$, then $a \cdot b = 6$ which is divisible by 6, and so is $b \cdot a$. But $1 \neq 6$
  - Thus, R is not anti-symmetric.
- **Transitive:**
  - A relation R is transitive if $\forall\ a, b, c \in N, (aRb \wedge bRc) \Longrightarrow aRc$.
  - For R to be transitive:
    $aRb \leftrightarrow 6 \mid (a.b)$ and $bRc \leftrightarrow 6 \mid (b.c) \Longrightarrow$ aRc $\leftrightarrow 6 \mid (a \cdot c)$
  - However, transitivity does not necessarily hold in this case.
  - For example, let $a = 1$, $b = 6$, and $c = 1$.

- We have:

$$6 \mid (a \cdot c) \rightarrow 6 \mid (1 \cdot 6) \text{ (true)}$$

$$6 \mid (b \cdot c) \rightarrow 6 \mid (6 \cdot 1) \text{ (true)}$$

- But:

$$6 \mid (a \cdot c) \rightarrow 6 \mid (1 \cdot 1) \text{ (false)}$$

- Thus, R is not transitive.

➢ **Summary:**

- R is not reflexive.
- R is symmetric.
- R is not anti-symmetric.
- R is not transitive.

CHAPTER – 5: KRUSKAL'S ALGORITHM

- Kruskal's approach is commonly used to determine the Minimum Spanning Tree (MST) of a graph. One important stage in Kruskal's approach is to avoid adding edges that form a cycle (circuit). To accomplish this, we employ a data structure known as the Union-Find or Disjoint Set Union (DSU). This data structure can efficiently handle two operations:
  1. Determine an element's subset.
  2. Union: Combines two subsets into a single set.
- Solution for Circuit Checking with Union-Find:
  1. Set up the Union-Find Structure: Each vertex starts in their own set.
  2. Sort the edges in the graph by weight in non-decreasing order.
  3. Process each edge by checking if its vertices belong to various subsets using the Find operation.
     - If the edges are in distinct subsets, add them to the MST and use the Union procedure to unify them.
     - To avoid a cycle, disregard edges within the same subgroup.

Example:

Let's consider a simple undirected graph with 4 vertices and 5 edges.

Vertices: {A, B, C, D}

Edges with weights:

- (A, B, 1)
- (B, C, 4)
- (A, C, 3)
- (C, D, 2)
- (B, D, 5)

Steps of Kruskal's Algorithm with Circuit-Checking

1. Initialize the Union-Find Structure: Each vertex is its own set: {A}, {B}, {C}, {D}

2. Sort the Edges by Weight: Sorted edges: (A, B, 1), (C, D, 2), (A, C, 3), (B, C, 4), (B, D, 5)
3. Process Each Edge:

- Edge (A, B, 1):
  - Find(A) ≠ Find(B), so add (A, B) to MST.
  - Union(A, B): {A, B}, {C}, {D}
- Edge (C, D, 2):
  - Find(C) ≠ Find(D), so add (C, D) to MST.
  - Union(C, D): {A, B}, {C, D}
- Edge (A, C, 3):
  - Find(A) ≠ Find(C), so add (A, C) to MST.
  - Union(A, C): {A, B, C, D}
- Edge (B, C, 4):
  - Find(B) = Find(C), so skip this edge to avoid a cycle.
- Edge (B, D, 5):
  - Find(B) = Find(D), so skip this edge to avoid a cycle.

Final MST:

The MST includes the edges: (A, B, 1), (C, D, 2), and (A, C, 3).

Union-Find Operations

Initialization:

Each vertex is its own parent and has a rank of 0.

- Parents: A → A, B → B, C → C, D → D
- Ranks: A: 0, B: 0, C: 0, D: 0

Processing Edges:

1. Edge (A, B, 1):
   - Find(A) = A, Find(B) = B
   - Union(A, B)
     - Parents: A → A, B → A, C → C, D → D

- Ranks: A: 1, B: 0, C: 0, D: 0

2. Edge (C, D, 2):

- Find(C) = C, Find(D) = D
- Union(C, D)

- Parents: A → A, B → A, C → C, D → C
- Ranks: A: 1, B: 0, C: 1, D: 0

3. Edge (A, C, 3):

- Find(A) = A, Find(C) = C
- Union(A, C)

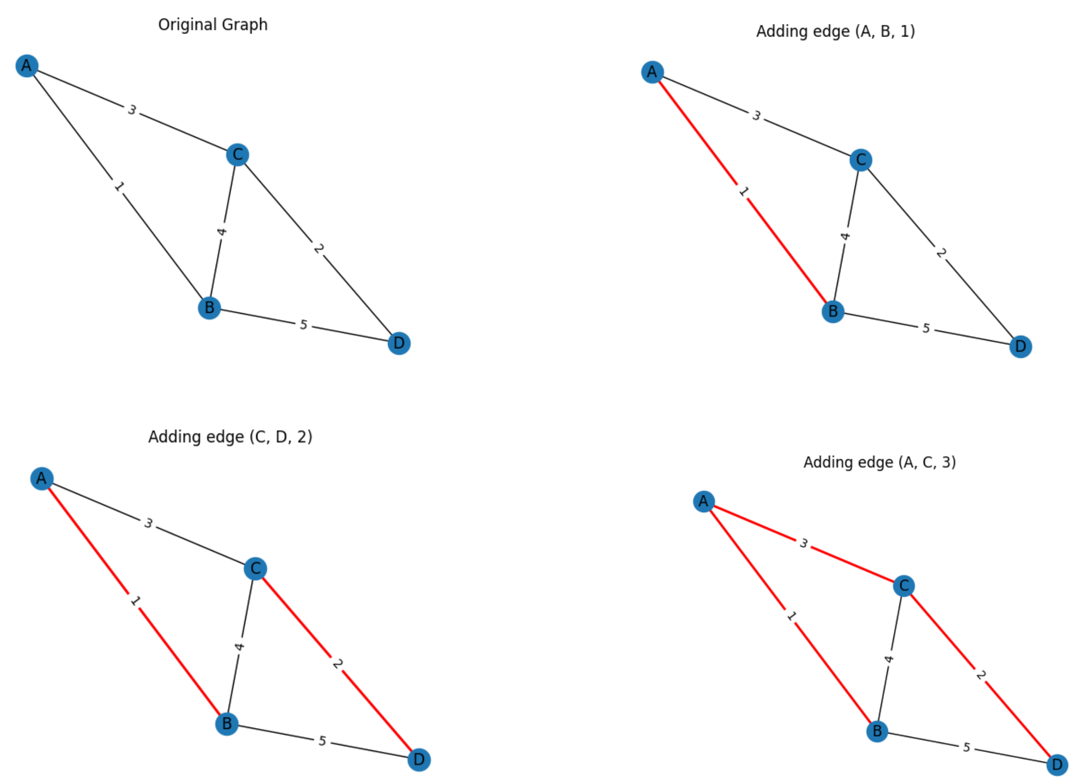
- Parents: A → A, B → A, C → A, D → C
- Ranks: A: 2, B: 0, C: 1, D: 0

4. Edge (B, C, 4):

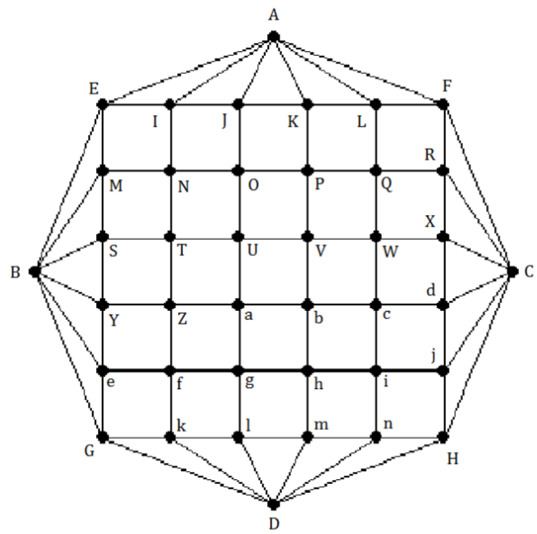
- Find(B) = A, Find(C) = A (same set, skip edge)

5. Edge (B, D, 5):

- Find(B) = A, Find(D) = A (same set, skip edge)



CHAPTER – 6: EULERIAN CIRCUIT



a) Does the following graph have an Eulerian circuit or Eulerian path? Why?

- To determine if a graph has an Eulerian circuit or an Eulerian path, consider the degree (number of connected edges) of each vertex. Here are the main properties to consider:
  - Eulerian Circuit: A connected graph with an even degree has at least one Euler circuit.
  - An Eulerian path exists when a graph is linked and has two vertices of odd degree. Any such path must begin at one of the odd-degree vertices and conclude at the other.
- Analyze the given graph.
  - Every vertex is connected.
  - {A, B, C, D} has 6 degrees.
  - {E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, I, j, k, l, m, n} have 4 degrees.

➢ Based on the aforementioned theorem, this graph is connected, with all vertices having an even degree. So, the presented graph contains an Eulerian Circuit.

b) Study and present your knowledge about Hierholzer's algorithm to find an Eulerian circuit.

- Hierholzer's algorithm identifies Eulerian circuits in graphs. It is vital to highlight that the algorithm is only relevant if the network has all vertices of even degree and is connected. Hierholzer's algorithm works as follows:
- Conditions for the Eulerian Circuit.
  - All non-zero vertices are connected.
  - Each vertex has an even degree.
- Steps for Hierholzer's Algorithm:

1. Initialization:
   - Select any starting vertex v.
   - Traverse through the graph by following edges that haven't been marked yet until you return to the starting vertex v, forming a circuit R_1.
2. Check for Completion:
   - Examine if every edge in the graph is included in the current circuit R_i.
   - If yes, R_i is a complete Eulerian circuit, and the process stops.
3. Identify a Vertex with Unused Edges:
   - Look for a vertex v_i in the current circuit R_i that has connections (edges) which haven't been used in any of the circuits formed so far.
   - Select one such unused edge ei connected to v_i.
4. Build a New Circuit:
   - Starting from vi, form a new circuit Q_i by following the unused edge e_i and continue traversing unused edges until you return to v_i.
   - Mark all edges used in Q_i to avoid reusing them.
5. Integrate the New Circuit:
   - Insert the newly formed circuit $Q_i$into the existing circuit R_i at the vertex v_i.

- This involves reconfiguring Ri to include the path of Q_i, resulting in a new circuit $R_{(i+1)}$.

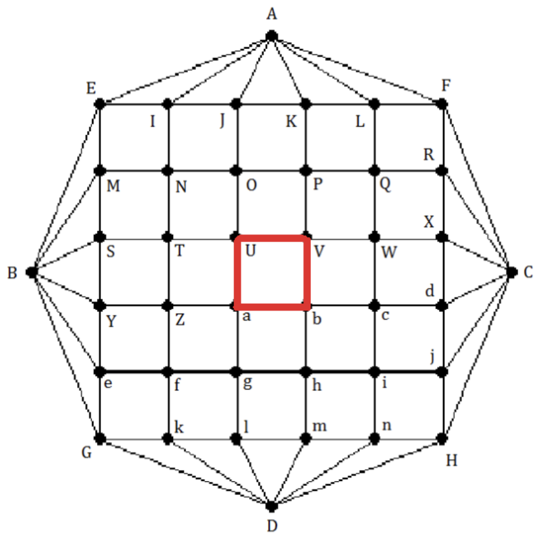
6. Repeat:
   - Increment the circuit counter i and return to step 2 to continue the process with the updated circuit $R_{(i+1)}$.

- By repeating these steps, the algorithm systematically builds a single Eulerian circuit by integrating smaller circuits, ensuring that every edge in the graph is used exactly once, which is the defining property of an Eulerian circuit.

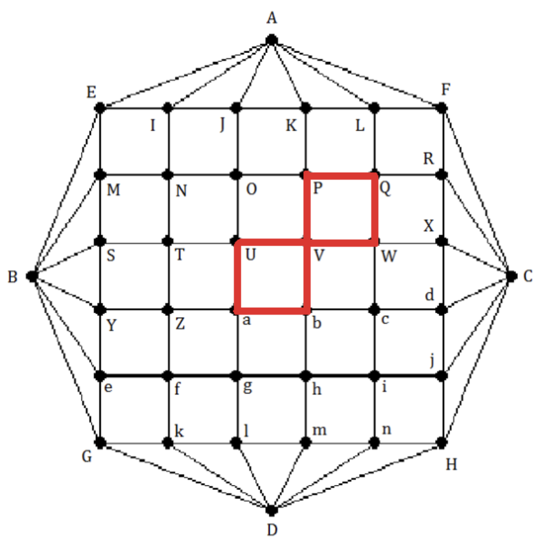
c) If the graph has an Eulerian circuit, use Hierholzer's algorithm to find an Eulerian circuit of that graph when the initial circuit R1 is:

Student ID 5220006, $\overline{abcd}$ % 4 = 2. Then, R1 is UVbaU.

- R_1 is UVbaU

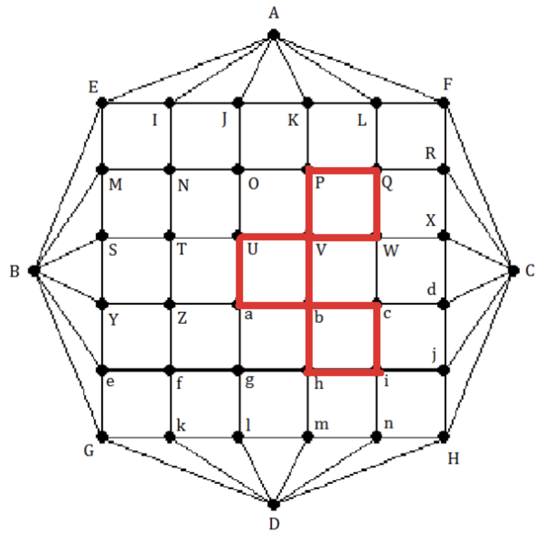


- Step1: Q_1= VPQWV



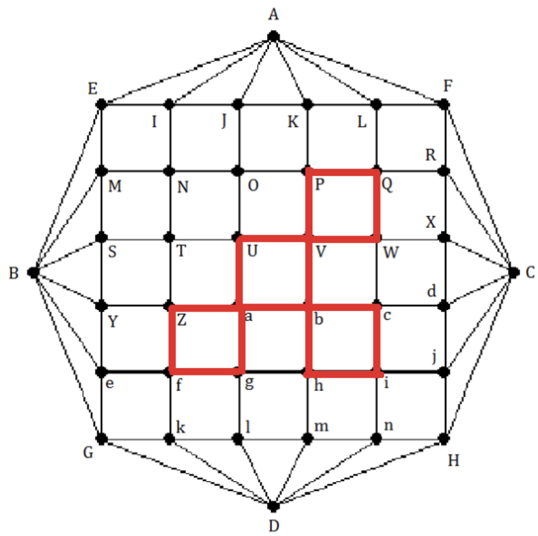
➢ R_1= UVPQWVbaU

- Step2: Q_2= bcihb



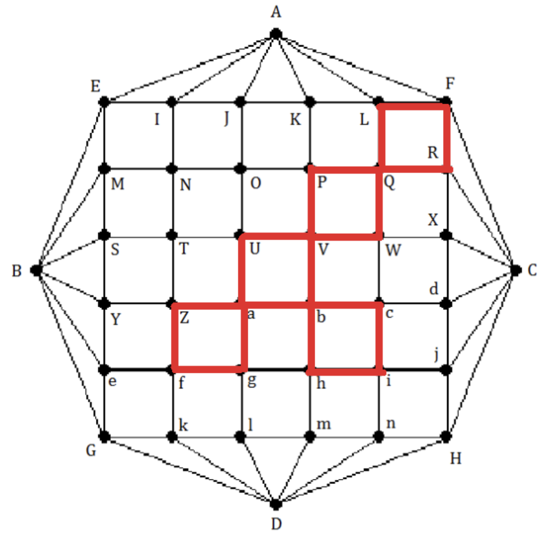
  - ➢ R_2= UVPQWVbcihbaU

- Step3: Q_3= aZfga



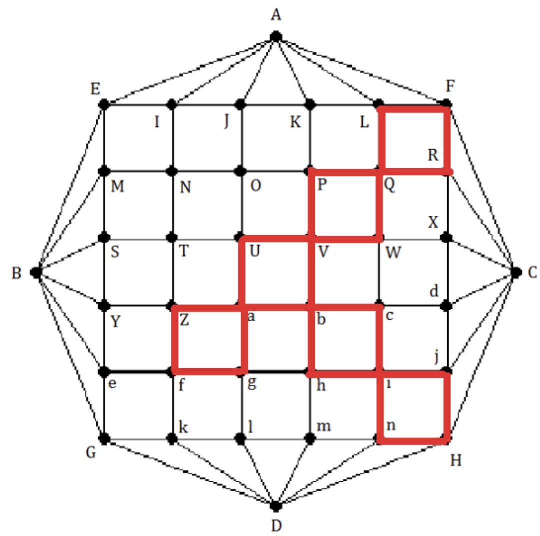
  - ➢ R_3= UVPQWVbcihbaZfgaU

- Step4: Q_4= QRFLQ



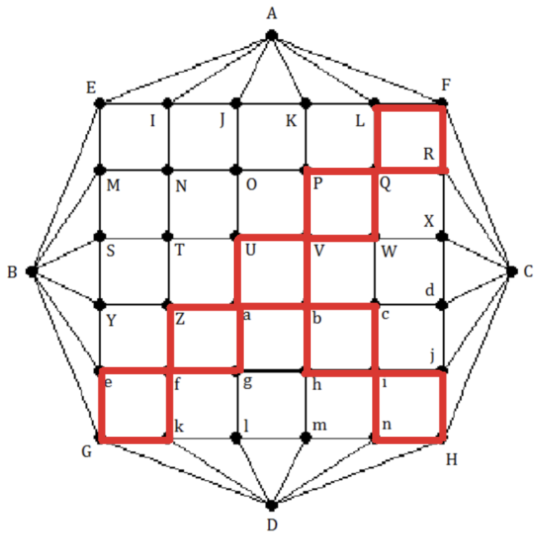
➢ R_4= UVPQRFLQWVbcihbaZfgaU

- Step5: Q_5= ijHni



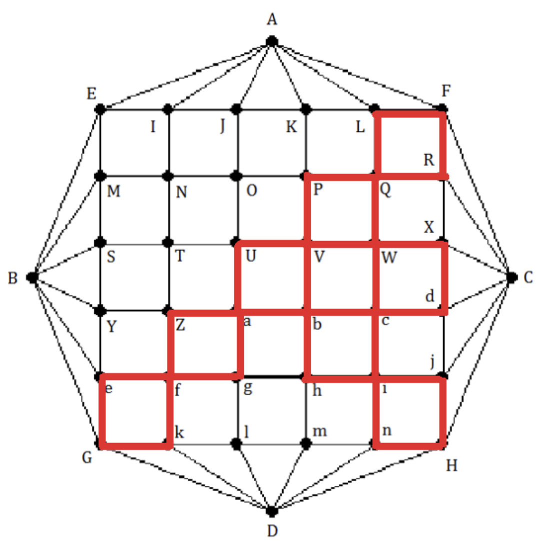
➢ R_5= UVPQRFLQWVbcijHnihbaZfgaU

- Step6: Q_6= fkGef



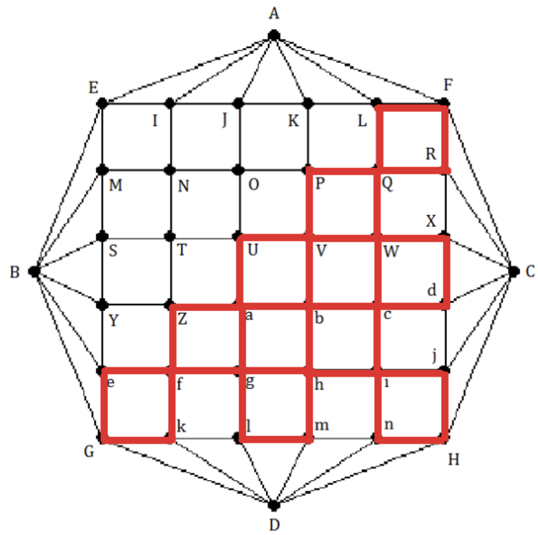
➢ R_6= UVPQRFLQWVbcijHnihbaZfkGefgaU

- Step7: Q_7= WXdcW



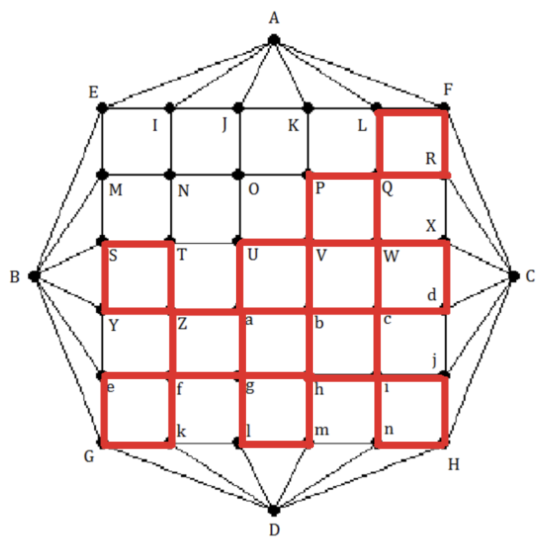
➢ R_7= UVPQRFLQWXdcWVbcijHnihbaZfkGefgaU

- Step8: Q_8= hmlgh



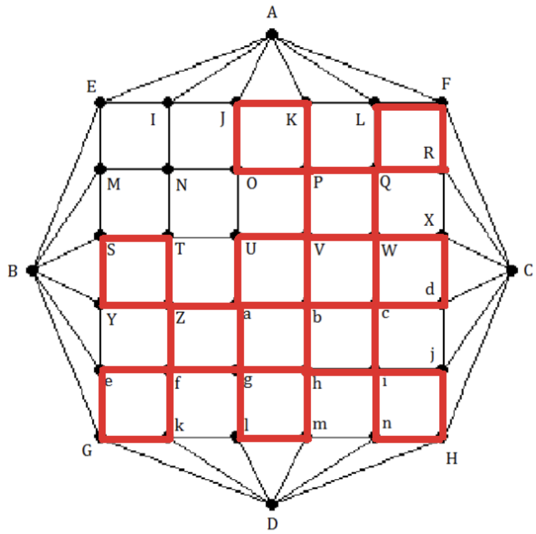
➢ R_8= UVPQRFLQWXdcWVbcijHnihmlghbaZfkGefgaU

- Step9: Q_9= ZYSTZ



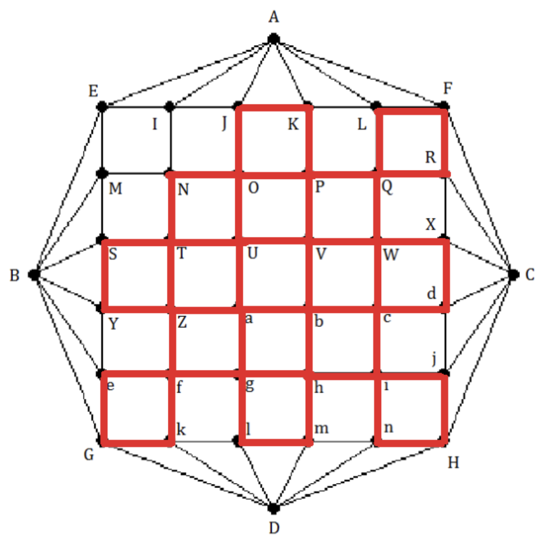
➢ R_9= UVPQRFLQWXdcWVbcijHnihmlghbaZYSTZfkGefgaU

- Step10: Q_{10}= POJKP



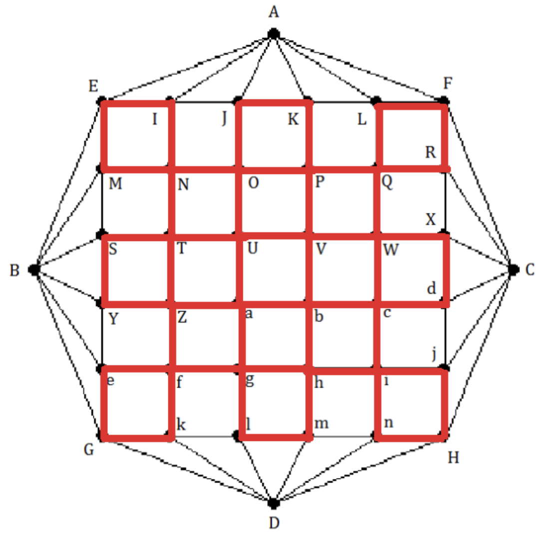
➢ R_{10}= UVPOJKPQRFLQWXdcWVbcijHnihmlghbaZYSTZfkGefgaU

- Step11: Q_{11}= ONTUO



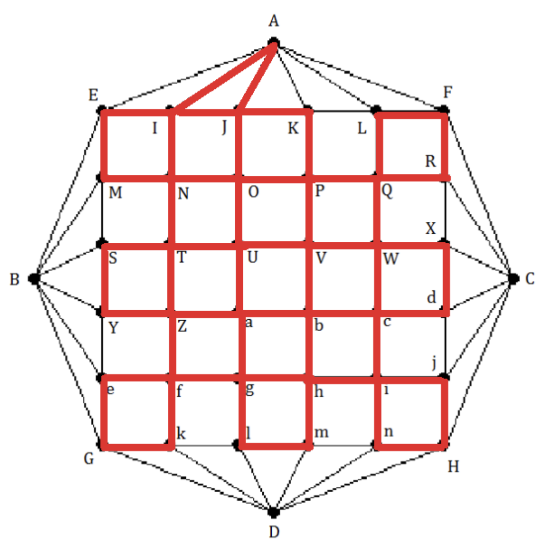
➢ R_{11}=UVPONTUOJKPQRFLQWXdcWVbcijHnihmlghbaZYSTZfkGefgaU

- Step12: Q_{12}= NIEMN



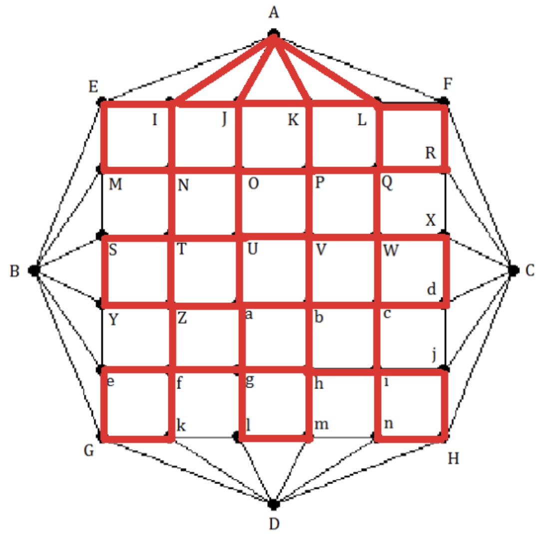
➢ R_{12}=UVPONIEMNTUOJKPQRFLQWXdcWVbcijHnihmlghbaZYSTZfkGefgaU

- Step13: Q_{13}= IAJI



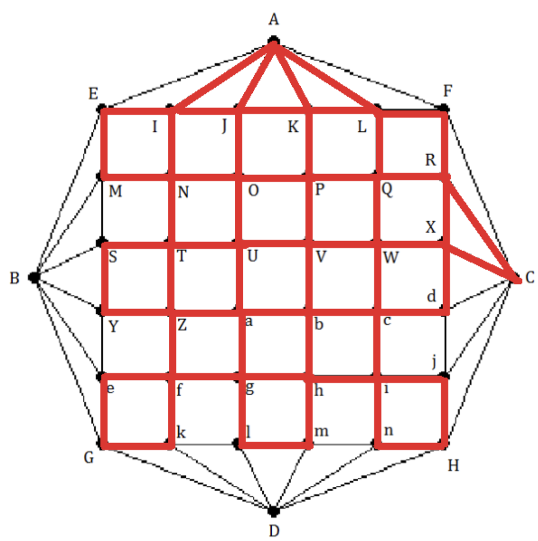
➢ R_{13}=UVPONIAJIEMNTUOJKPQRFLQWXdcWVbcijHnihmlghbaZYSTZfkGefgaU

- Step14: Q_{14}= LAKL



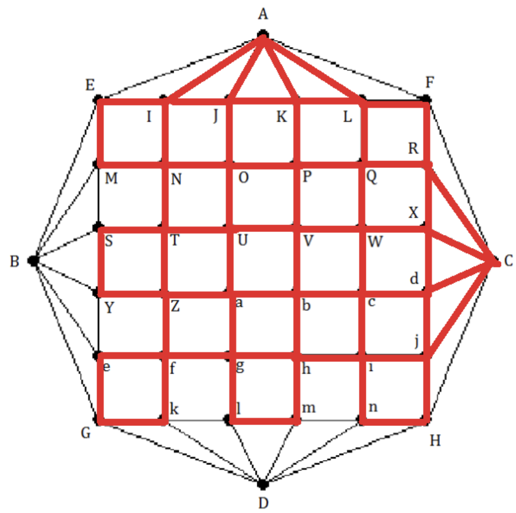
➢ R_{14}=UVPONIAJIEMNTUOJKPQRFLAKLQWXdcWVbcijHnihmlghbaZYSTZfkGefgaU

- Step15: Q_{15}= RCXR



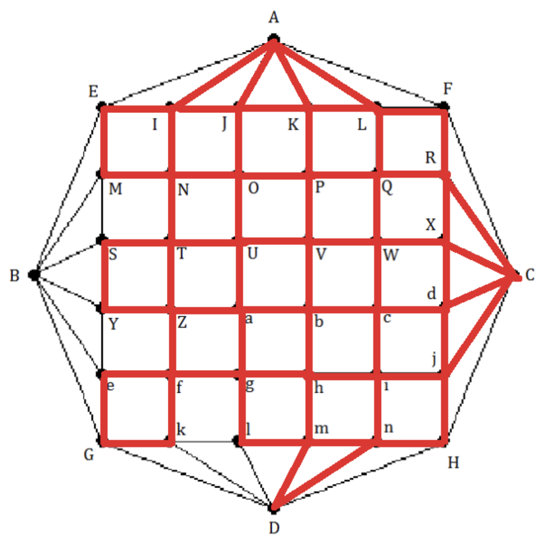
➢ R_1=UVPONIAJIEMNTUOJKPQRCXRFLAKLQWXdcWVbcijHnihmlghbaZYSTZfkGefgaU

- Step16: Q_{16}= jCdj



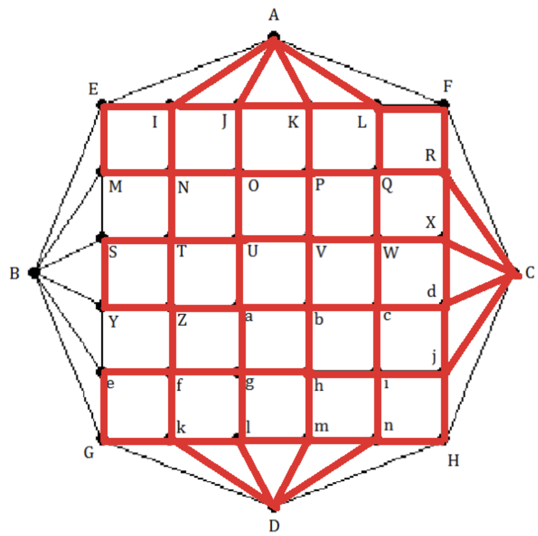
  - R_{16}=UVPONIAJIEMNTUOJKPQRCXRFLAKLQWXdcWVbcijCdjHnih mlghbaZYSTZfkGefgaU

- Step17: Q_{17}= nDmn



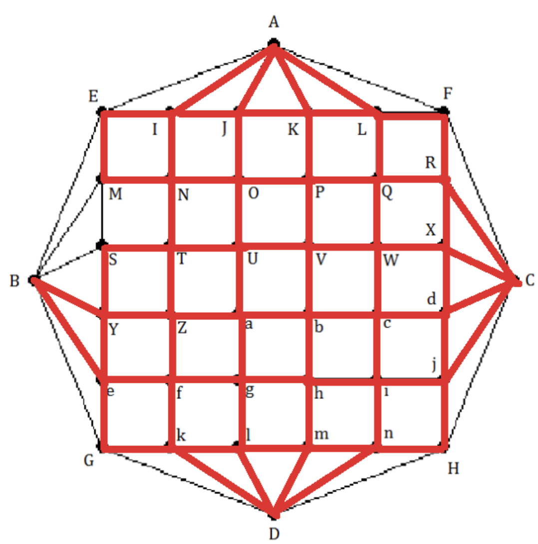
  - R_{17}=UVPONIAJIEMNTUOJKPQRCXRFLAKLQWXdcWVbcijCdjHnD mnihmlghbaZYSTZfkGefgaU

- Step18: Q_{18}= kDlk



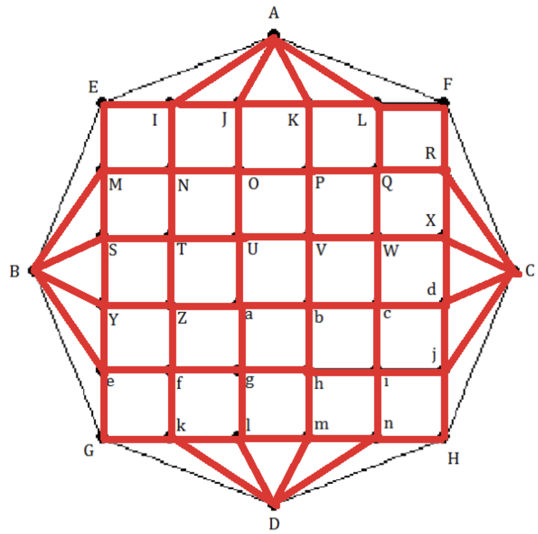
> R_{18}=UVPONIAJIEMNTUOJKPQRCXRFLAKLQWXdcWVbcijCdjHnD mnihmlghbaZYSTZfkDlkGefgaU

- Step19: Q_{19}= eBYe



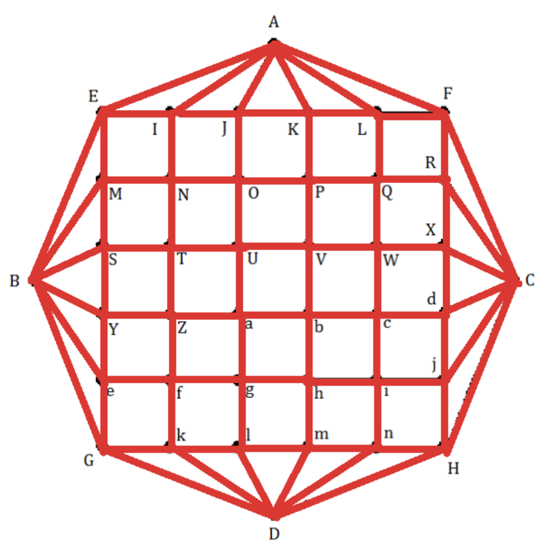
> R_{19}=UVPONIAJIEMNTUOJKPQRCXRFLAKLQWXdcWVbcijCdjHnD mnihmlghbaZYSTZfkDlkGeBYefgaU

- Step20: Q_{20}= MBSM



  - R_{20}=UVPONIAJIEMBSMNTUOJKPQRCXRFLAKLQWXdcWVbcijCdjHnDmnihmlghbaZYSTZfkDlkGeBYefgaU

- Step21: Q_{21}= AFCHDGBEA

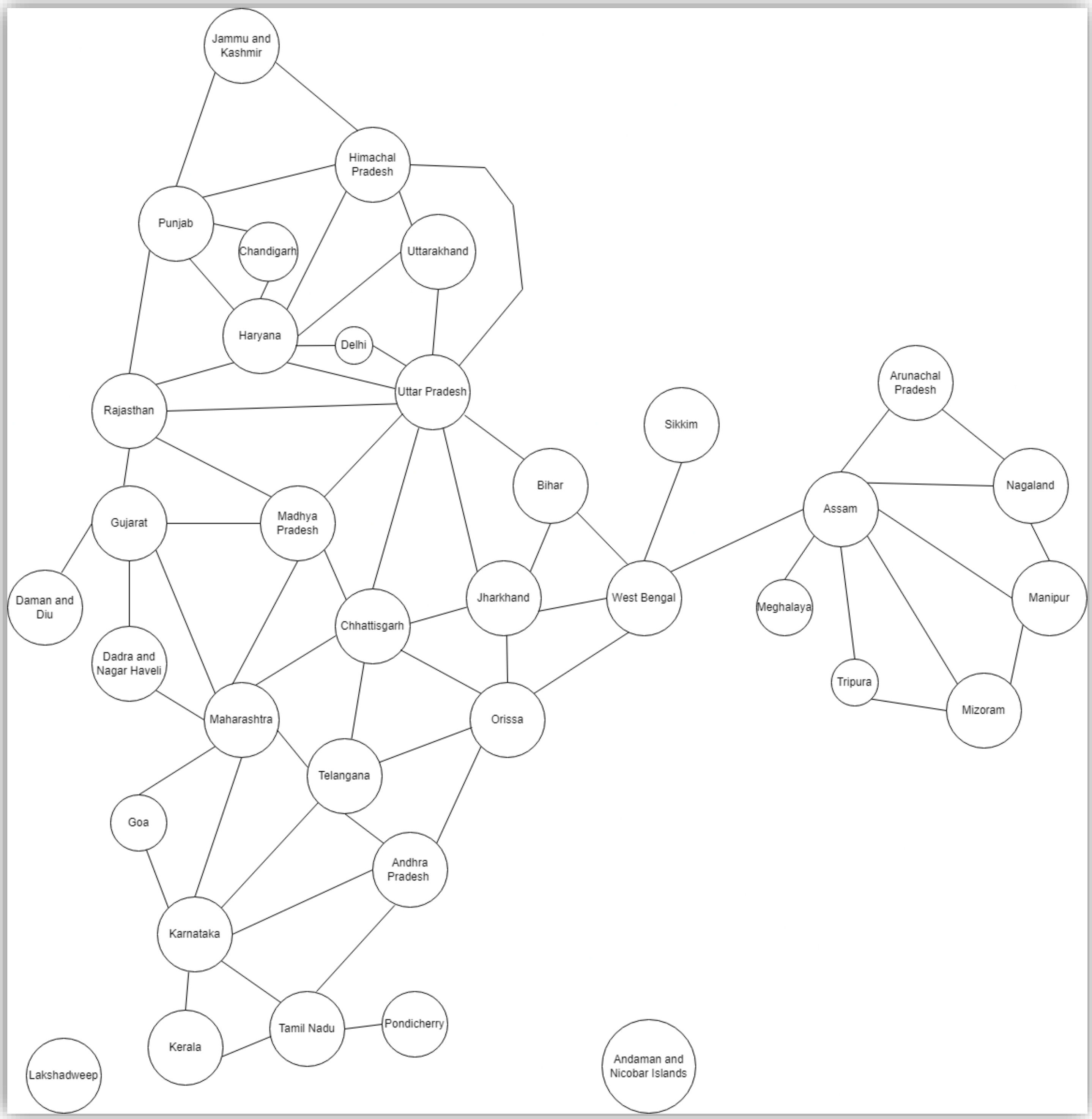


  - R_{21}=UVPONIAFCHDGBEAJIEMBSMNTUOJKPQRCXRFLAKLQWXdcWVbcijCdjHnDmnihmlghbaZYSTZfkDlkGeBYefgaU

- Now R_{21} contains every edge of the graph. So, the Eulerian circuit of that graph:
UVPONIAFCHDGBEAJIEMBSMNTUOJKPQRCXRFLAKLQWXdcWV
bcijCdjHnDmnihmlghbaZYSTZfkDlkGeBYefgaU

CHAPTER – 7: MAP COLORING

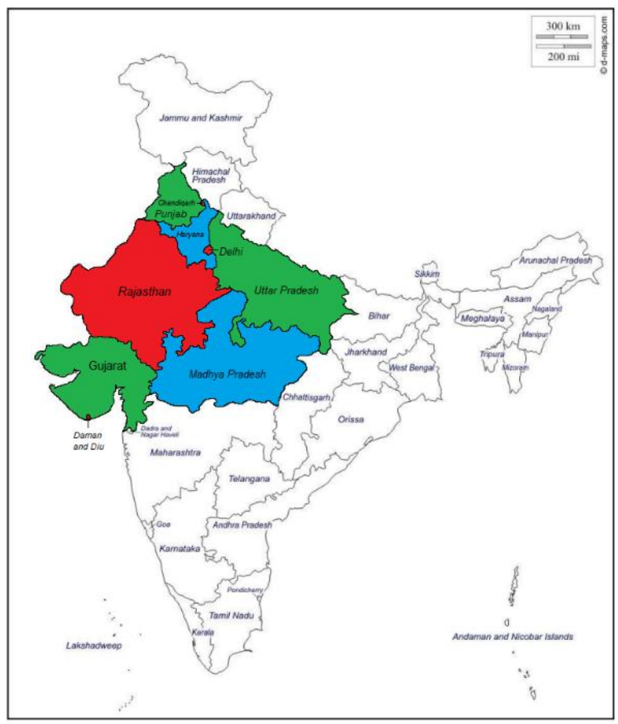
a) **Modeling this map by a graph.**



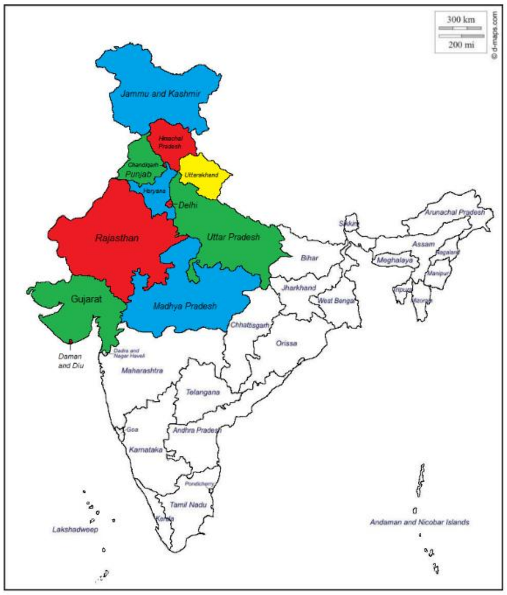
b) Color the map (graph) with a minimum number of colors. Present your solution step by step.

- StudentID 522H0006 has $\overline{abcd}$ = 0006
- We have: 6 % 4 = 2 then start from Rajasthan.
- Hence:
  - Color 1: Red
  - Color 2: Green
  - Color 3: Blue
  - Color 4: Yellow

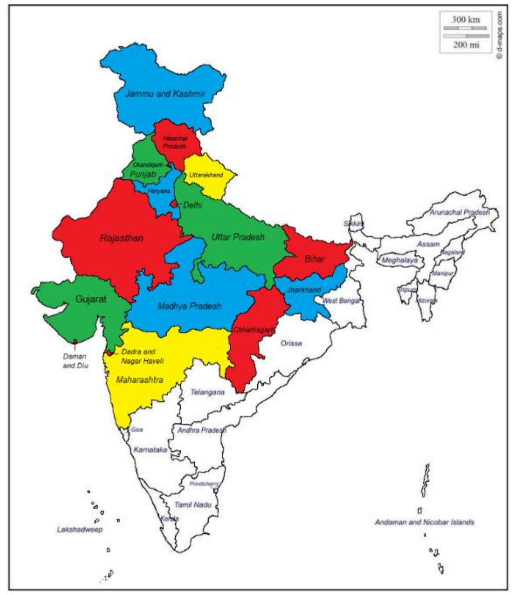
Step 1: Start coloring from Rajasthan.



Step 2: Next color the neighbors of Rajasthan.



Step 3: Next is to color the northern part of India based on the colors of Punjab, Haryana, Uttar Pradesh.



Step 4: Then colorize Bihar, Jharkhand, Chhattisgarh, Maharashtra.

Sep 5: Next, color the southern part of India based on the colors of Indian cities and color the 2 islands.

Step 6: Finally, the eastern part of India is completed.

CHAPTER – 8: FINDING AN INVERSE MODULO N

8.1 Theorical research

8.1.1 Inverse modulo

- In modular arithmetic, the inverse modulo is a crucial concept that allows us to perform various mathematical operations efficiently. Given a number a and a modulo n, the inverse modulo of a modulo n is find number x that satisfies the equation ax ≡ 1 (mod n). This means that the remainder of ax when divided by n equals 1.
- The inverse modulo only exists if a and n are coprime, that is, if their greatest common divisor is 1. If a and n are not coprime, then there is no inverse modulo. The inverse modulo is an important concept in number theory and has various applications in cryptography, computer science, and other fields.
- Finding the inverse modulo n using the extended Euclidean algorithm is an important concept in number theory that has various applications in cryptography, computer science, and other fields. In this context, the extended Euclidean algorithm is used to calculate the greatest common divisor (GCD) of two numbers a and b, as well as to find two integers x and y such that ax + by = GCD(a, b).

8.1.2 The Extended Euclidean algorithm

- To find the inverse modulo n using the extended Euclidean algorithm, we need to find two integers x and y such that ax + by = 1, where a is the number to find the inverse and b is the modulo. Once we have found x, it is the inverse of a modulo b.

8.1.3 Example

Let's find the inverse of 19 modulo 47. We can use the extended Euclidean algorithm to find x and y such that 19x +47y = 1. Starting with a = 19 and b = 47, we can use the following steps:

- Step1: Divide 47 by 19 to get a quotient of 2 and a remainder of 9.
  - This means 47 = 2*19+9.
- Step2: Divide 19 by 9 to get a quotient of 2 and a remainder of 1.

- This means 19 = 2*(9)+1.

Now we can work backwards to express 1 as a linear combination of 19 and 47:

1 = 19 – 2*(9) = 19 – 2*(47 – 2*(19)) = 5*(19) – 2*(47)

So x = 5 and y = –2, which means the inverse of 19 modulo 47 is 5.

8.2 Implement extended Euclidean algorithm

To implement the extended Euclidean algorithm in Python, we can define a function that takes two integers a and b as arguments and returns the greatest common divisor (GCD) of a and b, as well as x and y such that ax + by = GCD(a, b).

We can then define another function that takes two integers a and n as arguments and uses the extended Euclidean algorithm to find the inverse of a modulo.

Here is the Python code to implement the extended Euclidean algorithm to find the inverse modulo:

```
def extended_euclid(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        gcd, x, y = extended_euclid(b % a, a)
        return (gcd, y - (b // a) * x, x)

def inverse_modulo(a, b):
    gcd, x, y = extended_euclid(a, b)
    if gcd == 1:
        return f"inverse_modulo({a}, {b}) is: {x%b}"
    else:
        return f"The GCD of {a} and {b} is not 1, so {a} does not have a modular inverse modulo {b}.
```

- The function **'extended_euclid(a, b)'** implements the extended Euclidean algorithm to calculate the greatest common divisor (GCD) of two numbers 'a' and 'b', as well as to find two integers 'x' and 'y' such that ax + by = GCD(a, b).
- Then, the function **'inverse_modulo(a, b)'** uses the **'extended_euclid()'** function to find the GCD and two integers 'x' and 'y' such that ax + by = GCD(a, b). If the GCD is not equal to 1, it means that 'a' does not have a modular inverse

modulo 'b'. Otherwise, the function returns the value of 'x' modulo 'b', which is the inverse modulo of 'a' modulo b.

8.3 Test the implemented program

- **Test case:**

```
# TestCase
print(inverse_modulo(20, 23))

print(inverse_modulo(15, 20))
```

- **Result:**

```
nhandang@Nhans-MacBook-Air SourceCode % python3 Question8.py
inverse_modulo(20, 23) is: 15
The GCD of 25 and 30 is not 1, so 25 does not have a modular inverse modulo 30.
nhandang@Nhans-MacBook-Air SourceCode %
```

- **Explain:**
  - **Test case 1: Find inverse_modulo (20,23)**
    - First, I find the greatest common divisor (GCD) of 20 and 23 using the Euclidean algorithm:

      23 = 1 * 20 + 3

      20 = 6 * 3 + 2

      3 = 1 * 2 + 1
    - The last non-zero remainder is 1, so the GCD(20, 23) = 1. Since the GCD is 1, we know that 20 has a modular inverse modulo 23.
    - Next, we work backwards to express the GCD as a linear combination of 20 and 23:

      1 = 3 - 1 * 2

      1 = 3 - 1 * (20 - 6 * 3)

      1 = -1 * 20 + 7 * 3

      1 = -1 * 20 + 7 * (23 - 1 * 20)

      1 = -8 * 20 + 7 * 23

- From this, we can see that -8 is a solution to the equation 20x ≡ 1 (mod 23). To find the smallest positive solution, we can add 23 to -8 until we get a positive number: -8 + 23 = 15. So, 15 is the modular inverse of 20 modulo 23.
  - Therefore, **inverse_modulo (20, 23) = 15**

- **Test case 2: Find inverse_modulo (15, 20)**
  - First, we need to find the greatest common divisor (gcd) of 15 and 20 using the Euclidean algorithm:

    20 = 1 * 15 + 5

    15 = 3 * 5 + 0
  - The last non-zero remainder is 5, so gcd(15,20) = 5. Since the gcd is not equal to 1, we know that 15 does not have a modular inverse modulo 20.
    - Therefore, **inverse_modulo(15,20)** does not exist.

CHAPTER – 9: RSA CRYPTOSYSTEM

9.1 Theory research

- RSA is a public-key cryptosystem commonly used for secure data transmission. It was created by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977 and is named after their initials. The security of RSA is dependent on the difficulty of factoring big integers into prime factors, which is currently thought to be an intractable task for traditional computers.
- The following mathematical ideas form the foundation of the RSA cryptosystem:

- **Generation of keys:**
  - **Step1**: Choose two large prime numbers of the same length: p, q
  - **Step2**: Compute n and ϕ (n):
    - n = p * q
    - ϕ (n) = (p–1) * (q–1).
  - **Step3**: Choose the **Public Exponent** e:
    - Choose e, which has a range greater than 1 and smaller than ϕ (n). And coprime with ϕ (n). We often utilize the value of e as (2*k+1) for numbers like 3, 17, 19, 65537.
  - **Step4:** Compute the **Private Exponent** d:
    - Determine d such that d × e ≡ 1 mod ϕ (n)
      - ➢ The public key consists of (n, e), and the private key consists of (n, d). The primes p and q should be kept secret.

- **Encryption algorithm:** Given a known plaintext x (0 ≤ x < n), divide x into character blocks. Then calculate the ciphertext C = x^e mod n.
- **Decryption algorithm:** Given a known ciphertext c and the private key (n, d) the plaintext x can be calculated as x = c^d (mod n).
- **Security of the RSA algorithm:** The security of the RSA algorithm relies on the difficulty of factoring the large number n n into its prime factors p and q. If an

attacker could factor n, they could compute ϕ (n) and subsequently determine d from e. This problem, known as integer factorization, is considered computationally infeasible for sufficiently large n (e.g., 2048-bit keys).

- **Example:**
  - **Step 1: Initialize Parameters**

    + Choose two prime numbers: p=17 and q=11.

    + Calculate $N = p \times q = 17 \times 11 = 187$.

    + Calculate $\phi(N) = (p - 1) \times (q - 1) = 16 \times 10 = 160$.

    + Choose $e = 7$, with e and $\phi(N)$ being coprime.

    + Find d such that $e \times d \equiv 1 \bmod \phi(N)$. Using the extended Euclidean algorithm, we find $d = 23$ because $7 \times 23 \equiv 1 \bmod 160$.

    - The public key is $(N, e) = (187, 7)$.
    - The private key is $(N, d) = (187, 23)$
  - **Step 2: Encrypt the Message**

    - Assume the message is "HELLO":

      + Use the ASCII table to convert the string to a sequence of numbers:

      H = 72, E = 69, L = 76, L = 76, O = 79

    - Since each character's encoded value must be smaller than N, we will encrypt each character individually:

      + Encrypt H: 72^7 mod 187 = 30

      + Encrypt E: 69^7 mod 187 = 86

      + Encrypt L: 76^7 mod 187 = 32

      + Encrypt L: 76^7 mod 187 = 32

      + Encrypt O: 79^7 mod 187 = 139

      ➢ The encrypted sequence is: C = 30, 55, 72, 72, 44.

- **Step 3: Decrypt the Message**

- The recipient uses the private key (N, d) = (187, 23) to decrypt the message:

+ Decrypt 30: 30^{23} mod 187 = 72

+ Decrypt 86: 86^{23}mod 187 = 69

+ Decrypt 32: 32^{23}mod 187 = 76

+ Decrypt 32: 32^{23} mod 187 = 76

+ Decrypt 139: 139^{23} mod 187 = 79

- Convert the numbers back to ASCII characters: 72 = H, 69 = E, 76 = L, 76 = L, 79 = O

➢ Decryption result: “HELLO”.

9.2 Implement a Python program

- We can either implement manually or use Python libraries to support cryptography. In this case, we have alots options: the Crypto library (cryptography, cryptodome) and the rsa module. For this tutorial, I will be using the rsa library because I just need rsa algorithm and it result return big int number while cryptography or cryptodome return base64.To run this code you need install rsa library (python version 3.10.6). Here is the code applied to RSA encryption:

```python
import rsa # import library

# Generation public key and private key
(public_key, private_key) = rsa.newkeys(1024)

#print public key
print("public key: {}\n".format(public_key))
print("private key: {}\n".format(private_key))

# plain text
message = "Hello, I'm doing my final report for the discrete structures course"

#encrypt
encrypted_message = rsa.encrypt(message.encode(), public_key)

print("encrypt text: {}\n".format(encrypted_message))

#decrypt
decrypted_message = rsa.decrypt(encrypted_message, private_key).decode()

# print decrypted message
print("decrypt message: {}".format(decrypted_message))

# message authentication
print('plain text = decrypt text: {}'.format(message == decrypted_message))
```

9.3 Test the implemented program

- **Plain text:**

```python
# plain text
message = "Hello, I'm doing my final report for the discrete structures course"
```

- **Result:**

```
nhandang@Nhans-MacBook-Air SourceCode % python3 Question9.py
public key: PublicKey(24432404718423048551303075944937099456575611715068268714076300720779473309169809325612834272415
0143654291193507857054942730262220212563696665506914409060947779489728443099458006552170902859079536763904972532574443
4872852881276223355940237955290225401614333552342639062327104677860758562498268625486288036107644610944182350560213 48
094381037563023855042189133087059301301918922343339405180961619038755891865006028980479530357225201626734012166392843
1932699376448363565182378712867942410306010520198798435749814686911802663673555994077803156584196720820138515481844 83
5671917377401404848852440119941751612451436739760172 73, 65537)

private key: PrivateKey(24432404718423048551303075944937099456575611715068268714076300720779473309169809325612834272 4
1501436542911935078570549427302622202125636966655069144090609477794897284430994580065521709028590795367639049725325744
4348728528812762233559402379552902254016143335523426390623271046778607585624982686254862880361076446109441823505602 13
4809438103756302385504218913308705930130191892234333940518096161903875589186500602898047953035722520162673401216639 28
4319326993764483635651823787128679424103060105201987984357498146869118026636735559940778031565841967208201385154818 44
835671917377401404848852440119941751612451436739760172 73, 65537, 96913925425351415252403186332084904324753895073762 10
590828501663753043138153689720747535586702179126961798474800878893283512971613680910913171456535666087835506888942228
2605297515942870550167331082361668363947881791273379948070591589825989681015297318556159523616355101017947200333850 96
516977468586100498077967909968423478115275710871266284433037858123902269287622880066395140514188963756870412575829556
0752801973729451989713236774518954576213738599019455136826902998265589983329466499675187375400468695991974178507066 85
97608579528925862426849595065628369964611427149701516453639040188546687498611920577631152729139 3, 2537561756697581277
779185062927720845125807850599687350901334173084315439020878949669812430192474040634450448272394338500349778437032429
146915558413130641175716007000746753687351517421353106108206327642763179118867694859701976429822891898248748707194190
1675027511761280618604729698630729791854091513124170958168405602049117445 21, 96282995493357865095586135952009823768 72
772632346116229089528538972482794820086453146194603499246599416162332663766285931779254672731750787749331412617724892
325368192607708408897526008219001797625159276977748545061770346374505516148650521091397561521146461460916126319023801
838349485718513)

encrypt text: b'\xbf=\x92\xff+\xea!e\xe6\x94\xc1R\xe7|gobS\x10<;\xc1\x137*\xff\xbb\x9d\x9f\xaeI\x94b\xe9\x81\x970\x00
\x138\x8b3C\x1a\x9c#\x80\x88\xdd|\xd0\x1a\xcd\xde\xd3\xbbb\xc8\x08!\x8c\xceq\xddn\x1f\xe4\x9f\x84\x06p\x19\x8d\xc3\xb
f2\xf1\xff\x9a6E\xf4\x05\xd7\x10\x00\xa2\x9cG\xdf\x87\x93\xc6\xcbT\xe8\x02i\xe6\xa0!vb\x02=\xfc\xbd,\xbc\xd5:\x8d\x9d
g"\xd8\xf3\x9e\x83\t\x8c\x0e\xb1\x82S1\xa7/\xda\xbcN\xb1u\x95:\xc9\xcb\x9c[\xfaAq\xa7\xa2lV\r\x84,w\x07\xd4\xdb\x8b\x
f1w\x1d\xef\xb5\x8e7pG\x91\xd1d\xe2\xbb\\\x89\xc1\x828.\xbe\x9f%@\xc4\xac\xda@cC\x83\x83\xe2\xdd\r\xf1!\xf8\xfc\x9a\x
e9\xe4\x17\xdf$\xed3Af\x8a\xfc\xba\xbdv\x8e0r\xb8\xb1=\xef0\x7f\xccs\xbd\x80>>X\n7\x926\xaa\xf5+@\x02\xd1\x10&\x04\x0
f\xf4\x0f\x9c\xd4\xe1\xa3\xd4\xe3f:W\xa9P\xce\x85\x94\xf2}'

decrypt message: Hello, I'm doing my final report for the discrete structures course

plain text = decrypt text: True
```

The terminal show that decrypt message equal original message. The public key and private key store value I mention above. The last line I compare plain text and decrypted text, it returns true meaning successful decrypt.

9.4 Analyze the efficiency and security of the implemented RSA cryptosystem

9.4.1 Efficiency

- RSA encryption is often used in combination with other encryption schemes and for digital signatures that can verify the authenticity and integrity of a message. It is usually not used to encrypt entire messages or files because it is less efficient and more resource-intensive than symmetric key encryption.
- To make things more efficient, a file is usually encrypted using a symmetric key algorithm. Then the symmetric key is encrypted using RSA encryption. In this process, only the person with access to the RSA private key can decrypt the symmetric key. If the symmetric key cannot be accessed, then the original file

cannot be decrypted. This method can be used to secure messages and files without taking up too much time and resources.

- RSA encryption can be used in various systems. It can operate in OpenSSL, wolfCrypt, cryptlib, and other cryptographic libraries. Traditionally, it has been used in TLS and was also the initial algorithm used in PGP encryption. RSA is still seen in a variety of web browsers, email, VPNs, chat, and other communication channels.
- RSA is also commonly used to create secure connections between VPN clients and VPN servers. In protocols such as OpenVPN, TLS can use the RSA algorithm to exchange keys and establish a secure channel.

9.4.2 Security

- Secure communication occurs when two entities communicate without allowing a third party to listen in. For this to be the case, the entities must communicate in a fashion that is not vulnerable to eavesdropping or interception. There are several requirements for secure communication which rsa have, including:
  - **Confidentiality:** The content of the communication should be kept secret from anyone who is not authorized to access it. (Has been encrypt)
  - **Integrity:** The content of the communication should not be altered in transit without being detected. (Encrypt message equal original message)
  - **Authentication:** The identity of the sender and receiver should be verified to ensure that they are who they claim to be. (Has private key)
  - **Non-repudiation:** The sender should not be able to deny sending the message, and the receiver should not be able to deny receiving it. (Ensures that the sender of a message cannot deny sending it and the receiver cannot deny receiving it. In the RSA (Rivest-Shamir-Adleman) algorithm, non-repudiation is achieved using digital signatures. A digital signature is a mathematical scheme that allows a sender to prove the authenticity and integrity of a message to a receiver. When the sender signs a message with their private key, the receiver can verify the signature using the sender's

public key. If the verification process is successful, the receiver can be assured that the message was indeed sent by the sender and has not been tampered with in transit. This makes it difficult for either party to deny their involvement in the communication.

- Security Basis: RSA's security relies on the difficulty of factoring large integers. Security experts and researchers continuously analyze RSA to identify vulnerabilities and enhance the algorithm. Key security analyses include:
    - **Mathematical analysis** of RSA involves analyzing the properties of prime numbers, mathematical functions used in the algorithm, and the properties of modular arithmetic.
    - **Encryption and decryption analysis** of RSA involves analyzing the encryption and decryption processes and the properties of the keys used.
    - **Vulnerability testing** of RSA involves testing the algorithm for potential vulnerabilities and weaknesses.
    - **Detecting potential attacks** involves analyzing the different methods attackers could use to exploit RSA's vulnerabilities.

9.5 Discuss the potential security threats and limitations of the RSA cryptosystem

9.5.1 The potential security threats

- **Mathematical Analysis Threats**
    - **Prime Number Factorization:** The security of RSA relies heavily on the difficulty of factoring large prime numbers. Advances in factorization algorithms, such as the General Number Field Sieve (GNFS), could potentially reduce the time required to factorize the large composite numbers used in RSA keys.
    - **Quantum Computing:** Quantum algorithms, particularly Shor's algorithm, pose a significant threat to RSA. Quantum computers could theoretically factorize large integers exponentially faster than classical computers, rendering RSA insecure.

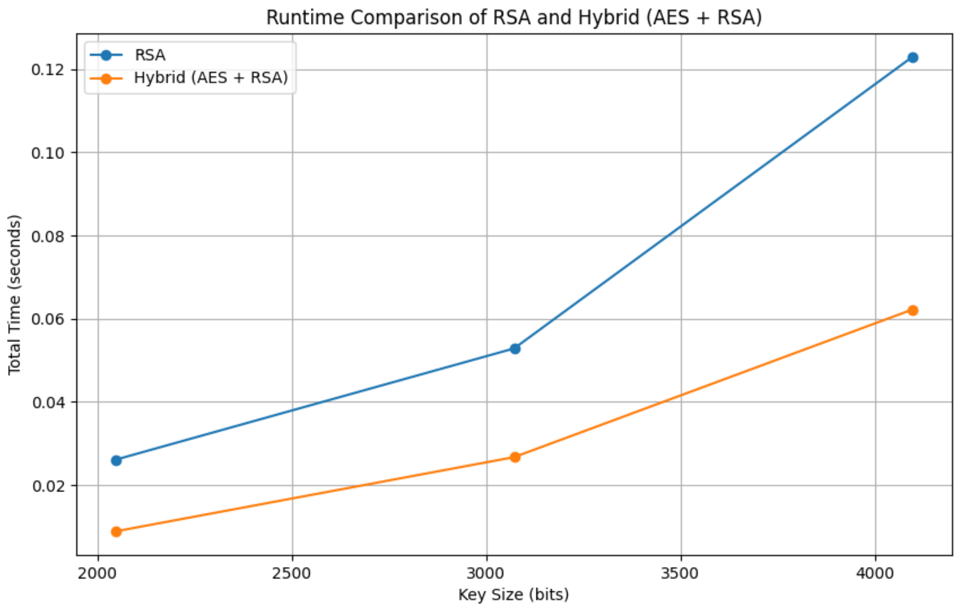
- **Encryption and Decryption Analysis Threats**
  - **Weak Key Generation:** Improper generation of RSA keys, such as using small primes or predictable patterns, can make the system vulnerable to attacks. Ensuring that the primes are large and randomly generated is crucial for security.
  - **Side-Channel Attacks:** These attacks exploit information leaked during the encryption and decryption processes. Timing attacks, power analysis, and electromagnetic leaks can potentially reveal the private key without directly attacking the mathematical structure of RSA.
- **Vulnerability Testing Threats**
  - **Implementation Flaws:** Vulnerabilities can arise from incorrect or insecure implementations of the RSA algorithm. For example, improper padding schemes (e.g., PKCS#1 v1.5) have led to attacks such as the Bleichenbacher attack, which can decrypt messages without the private key.
  - **Backdoors in Software:** Malicious backdoors intentionally placed in software that implements RSA can allow attackers to bypass encryption entirely. Ensuring the integrity and security of cryptographic software is essential.
- **Potential Attack Methods**
  - **Chosen Ciphertext Attacks (CCA):** In these attacks, the attacker can choose a ciphertext and obtain its decryption under an unknown key. Adaptive chosen-ciphertext attacks (CCA2) can be particularly effective against systems that do not employ proper padding schemes.
  - **Timing Attacks:** By measuring the time it takes for certain operations, attackers can gain information about the private key. Variations in timing due to different computations can provide clues to an attacker.
  - **Mathematical Attacks:** Advanced mathematical techniques, such as lattice-based attacks, can be used to solve problems related to RSA's security

assumptions. These attacks become more feasible as computational power increases.

9.5.2 The limitations

- RSA encryption and decryption can be slow, especially for big volumes of data. This can be a concern in real-time applications when speed is important.
- To determine key bit sizes, I measured the execution time of the RSA and combined RSA and AES algorithms on the same piece of text 100 times each. See table below for results. I then used the matplotlib library to graph the findings, which are presented in the table and figure below. (Note that the algorithm's execution time varies depending on computer hardware, Python version, and randomly generated keys.)

Number Bits of key	2048	3072	4096
RSA time	0.026126	0.052841	0.122883
RSA & AES time	0.008894	0.026709	0.062173



- **Comment:**
  - After conduct experiment, I see that RSA runtime longer than RSA mixed AES ((Advanced Encryption Standard) is a symmetric encryption algorithm, which means that the same key is used for both encryption and decryption. It was introduced as a replacement for the outdated DES (Data Encryption Standard) algorithm).
  - Time complete depend on number of bit and length of plain text. Key size: RSA requires large key sizes to ensure security, which can be a problem in certain applications where the available storage space is limited.
  - This experiment does not conclude that the RSA algorithm is inferior to other algorithms in terms of speed and security, but rather shows that the RSA algorithm can increase operational efficiency if combined with other algorithms.

9.6 Recommendations

- RSA remains a popular and secure cryptosystem for large keys. To improve the security of RSA, utilize large keys, carefully implement the algorithm, and update the keys on a regular basis. Furthermore, employing block cipher algorithms like AES to encrypt data and then RSA to encrypt the AES key might improve the security of RSA by making it more resistant to potential weaknesses and attacks.
- Analyzing the security and limitations of the RSA algorithm is crucial for ensuring reliable encryption in real-world applications. Security specialists and researchers should continue to update, research, and develop new approaches to improve RSA's security against potential flaws and threats. By constantly strengthening the security of RSA, we can assure that it stays a dependable and secure means of data encryption.

SELF EVALUTAION

Criteria	Scale	1	2	3	Self-evaluation	Reason
	Score /10	**0 score**	**½ score**	**Full score**		
Question 1	1	Do nothing or wrongly.	Correct gcd and lcm, but incorrect solutions of the Bezout's identity.	Correct calculation, detailed explanation.	1	Correct calculation, detailed explanation.
Question 2	0.5	Do nothing or wrongly.	Correct calculation but wrong result or conclusion.	Correct calculation, detailed explanation.	0.5	Correct calculation, detailed explanation.
Question 3	0.5	Do nothing or wrongly.	Correct Γ but incorrect operations.	Correct calculation, detailed explanation.	0.5	Correct calculation, detailed explanation.
Question 4	0.5	Do nothing or wrongly.	Correct results but incorrect proofs.	Right results, detailed explanation.	0.5	Right results, detailed explanation.

Question 5		1	Do nothing or wrongly.	Reasonable but indetailed proposition. No illustration.	Reasonable detailed proposition with illustration.	1	Reasonable detailed proposition with illustration.
Question 6		1	Do nothing or wrongly.	a-Correct recognition, right explanation. b,c-Good study but incorrect applications.	a-Correct recognition, right explanation. b,c-Good study, right calculation, detailed explanation.	1	a-Correct recognition, right explanation. b,c-Good study, right calculation, detailed explanation.
Question 7		1	Do nothing or wrongly.	Correct modeling but wrong coloring.	Correct modeling but right coloring.	1	Correct modeling but right coloring
Question 8	Theorical research	0.5	Do nothing or wrongly.	Not enough details, no example, no comment.	Correct calculations, detailed explanations.	0.25	Not enough details, no example, no comment.
	Implementation	0.5	Error	Correct but bad performance.	Correct and good performance.	0.5	Correct and good performance.

	Test	0.5	No test	Test without verification.	Test and verification.	0.5	Test and verification.
Question 9	Theorical research	0.5	Do nothing or wrongly.	Not enough details, no example, no comment.	Correct calculations, detailed explanations.	0.5	Correct calculations, detailed explanations
	Implementation	0.5	Error	Correct but bad performance.	Correct and good performance.	0.5	Correct and good performance.
	Test	0.5	No test	Test without verification	Test and verification		Test and verification
	Analysis	0.5	Do nothing or wrongly.	Not enough details, no example, no comment	Correct, detailed explanations	0.25	Not enough details, no example, no comment
	Discussion	0.5	Do nothing or wrongly.	Not enough details, no example, no comment	Correct, detailed explanations	0.5	Correct, detailed explanations
	Recommendatio n	0.5	Do nothing or wrongly.	Not enough details, no example, no comment	Correct, detailed explanations	0.5	Correct, detailed explanations
Total		**10**			Result	9.5	

WORK ASSIGNMENT TABLE

Full Name	StudentID	Preceding activity
Vo Nhat Hao	522H0090	Question 1, 2, 3, 4, 9
Dang Thanh Nhan	522H0006	Question 5, 6, 7, 8

REFERENCES

- Eulerian path and Eulerian circuit: https://math.libretexts.org/Bookshelves/Applied_Mathematics/Book%3A_College_Mathematics_for_Everyday_Life_(Inigo_et_al)/06%3A_Graph_Theory/6.03%3A_Euler_Circuits
- Hierholzers Algorithm: https://www.geeksforgeeks.org/hierholzers-algorithm-directed-graph/
- Extended Euclidean algorithm: Extended Euclidean algorithm - Wikipedia
- Modular multiplicative inverse: https://www.geeksforgeeks.org/multiplicative-inverse-under-modulo-m/
- Euclidean algorithm to find inverse module: modular arithmetic - Euclidean algorithm to find inverse modulo - Mathematics Stack Exchange
- RSA (cryptosystem) algorithm: https://en.wikipedia.org/wiki/RSA_(cryptosystem)
- Nisha, Shireen & Farik, Mohammed. (2017). RSA Public Key Cryptography Algorithm – A Review. International Journal of Scientific & Technology Research. 6. 187-191.
- NaQi, Wei Wei, Jing Zhang, Wei Wang, Jinwei Zhao, Junhuai Li, Peiyi Shen, Xiaoyan Yin, Xiangrong Xiao and Jie Hu, 2013. Analysis and Research of the RSA Algorithm. Information Technology Journal, 12: 1818-1824.
- A Hybrid Cryptographic Model Using AES and RSA: https://pdfs.semanticscholar.org/7b39/171aa64d24de52983b75847c205b55f74ccb.pdf