A blue and white logo

AI-generated content may be incorrect.

University of Information Technology

Faculty of Computing

2024-2025 Academic Year

Project Report



Discrete Mathematics

CST-3202

Tracking the Pandemic with Mathematics



Group-2

Semester-III Section-B

|  |  |
| --- | --- |
| **Project Title** | **Tracking the Pandemic with Mathematics** |
| **Subject** | **Discrete Mathematics (CST - 3202)** |
| **Section** | **Section-B** |
| **Team** | **Group-2** |

**Committee:**

|  |  |
| --- | --- |
| **Member 1 :** | **- - - - - - - - - - - - -**  **(Assoc. Prof. Daw Phyu Phyu Aung)** |
| **Member 2 :** | **- - - - - - - - - - - - -**  **(Lecturer Daw Aye Nyein San)** |
| **Supervisor:** | **- - - - - - - - - - - - -**  **(Prof. Dr. Khin Myo Myo Min)** |

**(Prof. Dr. Win Win Myo)**

**Head of Faculty of Computing**

**Team Members**

1. Min Khant Hein ( TNT - 2148 ) ( \*Leader )
2. Thinzar Lin Htet ( TNT - 2234 )
3. Aung Chan Myat Paing ( TNT - 2291 )
4. Bhone Min Thant ( TNT - 2170 )

**Abstract**

Discrete mathematics plays a fundamental role in solving real-world problems across various domains. This thesis explores the application of basic discrete structures, such as sets, relations, and matrices, in two significant areas: epidemic modeling and social network analysis.

The first part of the thesis focuses on tracking the spread of measles using matrices. The SIR model (Susceptible-Infected-Recovered) is used to represent disease transmission, with matrix multiplication applied to predict infection trends over time. The infection rate (β) and recovery rate (γ) govern the transitions between states, enabling an analytical approach to understanding outbreaks. By leveraging this mathematical framework, we can evaluate the impact of different vaccination strategies and intervention policies.

The second part examines friend suggestion algorithms in social networks. Platforms like Facebook use set operations to recommend new connections. In this thesis, we apply set union, intersection, and difference to identify potential friends based on mutual connections. By filtering out existing friends and analyzing friends-of-friends, the system suggests relevant users while improving network engagement.

Both applications demonstrate the practical significance of discrete mathematics in diverse fields. This thesis highlights its impact on public health through epidemiological modeling and on technology through social network algorithms. These examples illustrate how mathematical structures provide powerful tools for real-world decision-making and problem-solving.

**Acknowledgements**

We would like to express our heartfelt appreciation to all those who contributed to the completion of this project. Their support, guidance, and encouragement were invaluable throughout this journey.

We are profoundly grateful to Professor **Dr. Saw Sanda Aye**, the esteemed Principal of the University of Information Technology (UIT), for her unwavering support and visionary leadership. Her dedication to fostering an environment that encourages innovation, academic excellence, and creativity has played a pivotal role in our ability to pursue and successfully complete this project. The constant encouragement and commitment to advancing technological education and student development have been an enduring source of inspiration throughout our journey.

We would like to extend our sincere appreciation to Professor **Dr. Win Win Myo** for her exceptional mentorship and insightful feedback. Her unwavering support and expert guidance have been pivotal in shaping the direction of this work.

We would also like to extend our sincere thanks to our Lecturer **Daw Aye Nyein San** for her invaluable contributions and thorough oversight. Her dedication to fostering an environment of learning and discovery has been instrumental in the successful completion of this project.

Lastly, we acknowledge the support and encouragement of our peers and family, whose continuous belief in our efforts has motivated us to strive for excellence.

**Table of** **Contents**

[**Chapter(1) Introduction** 1](#_Toc190274955)

[**Chapter(2) Basic Structures: Sets, Functions, Sequences and Matrices** 2](#_Toc190274957)

[2.1 SETS 2](#_Toc190274958)

[2.1.1 Definition of set 2](#_Toc190274959)

[2.1.2 Characteristics of a Set 2](#_Toc190274960)

[2.1.3 Key Properties of a Set 2](#_Toc190274961)

[2.1.4 Set Notation 3](#_Toc190274962)

[2.1.5 Set Operation 4](#_Toc190274963)

[2.1.6 Type of Sets 5](#_Toc190274964)

[2.2 Functions 9](#_Toc190274965)

[2.2.1 Definition of a Function 9](#_Toc190274966)

[2.2.2 One-to-One and Onto Functions 9](#_Toc190274967)

[2.2.3 Bijective Functions 9](#_Toc190274968)

[2.2.4 Composition of Functions 9](#_Toc190274969)

[2.2.5 Floor and Ceiling Functions 9](#_Toc190274970)

[2.3 Sequences and Summations 10](#_Toc190274971)

[2.3.1 Definition of a Sequence 10](#_Toc190274972)

[2.3.2 Arithmetic Progression A sequence with a common difference d: 10](#_Toc190274973)

[2.3.3 Geometric Progression A sequence with a common ratio *r*: 10](#_Toc190274974)

[2.3.4 Recurrence Relations 10](#_Toc190274975)

[2.3.5 Fibonacci Sequence 10](#_Toc190274976)

[2.3.6 Summations 10](#_Toc190274977)

[2.4 Matrices 11](#_Toc190274978)

[2.4.1 Definition of a matrix 11](#_Toc190274979)

[2.4.2 Matrix Arithmetic 11](#_Toc190274980)

[2.4.3 Symmetric Matrices 13](#_Toc190274981)

[2.4.4 Join and meet matrices 13](#_Toc190274982)

[2.4.5 Transition Matrix 15](#_Toc190274983)

[**Chapter(3) Set Operations in Friend Suggestion Algorithms** 16](#_Toc190274985)

[3.1 Introduction 16](#_Toc190274986)

[3.2 Fundamentals of Set Operations 17](#_Toc190274987)

[3.2.1 Set Representation in Programming 17](#_Toc190274988)

[3.2.2 Set Operations & Their Role in Friend Suggestion 17](#_Toc190274989)

[3.3 Mathematical Model for Friend Suggestion 18](#_Toc190274990)

[3.3.1 Graph Representation of Social Networks 18](#_Toc190274991)

[3.3.2 Dictionary-Based Representation & Set Operations 18](#_Toc190274992)

[3.3.3 Algorithm for Friend Suggestions 19](#_Toc190274993)

[3.4 Implementation in Python 20](#_Toc190274994)

[3.5 Case Study: Social Media Friend Suggestion in Practice 21](#_Toc190274995)

[3.6 Advantages & Challenges 21](#_Toc190274996)

[3.6.1 Advantages 21](#_Toc190274997)

[3.6.2 Challenges 21](#_Toc190274998)

[3.7 Summary 22](#_Toc190274999)

[**Chapter(4) Tracking the Pandemic with Mathematics** 23](#_Toc190275000)

[4.1 Introduction 23](#_Toc190275001)

[4.2 The SIR Model for Measles 24](#_Toc190275002)

[4.2.1 Understanding the SIR Model 24](#_Toc190275003)

[4.2.2 The Role of the Total Population Size (N) 25](#_Toc190275004)

[4.2.3 Measles vs. Other Diseases 26](#_Toc190275005)

[4.3 Matrix Representation of the SIR Model 27](#_Toc190275006)

[4.3.1 3.1 Defining the State Vector and Transition Matrix 27](#_Toc190275007)

[4.3.2 Example: Measles Outbreak in an Unvaccinated Population 27](#_Toc190275008)

[4.4 Numerical Simulation of Measles Spread 28](#_Toc190275009)

[4.5 Analysis and Interpretation 30](#_Toc190275010)

[4.5.1 Understanding Measles Spread in an Unvaccinated Population 30](#_Toc190275011)

[4.5.2 Impact of Vaccination 30](#_Toc190275012)

[4.5.3 Limitations of the Model 30](#_Toc190275013)

[4.6 Summary 31](#_Toc190275014)

[**Chapter(5) Conclusion** 32](#_Toc190275016)

[**References** 33](#_Toc190275017)



Introduction

Discrete mathematics serves as a foundation for various computational and analytical problems, playing a crucial role in fields such as epidemiology, computer science, and network analysis. This thesis explores the **applications of basic discrete structures**, including **sets, relations, and matrices**, in two real-world problems: **tracking the spread of measles** and **friend suggestion algorithms in social networks**. These topics demonstrate how discrete mathematical concepts are used to model and solve practical problems.

The first application explores **friend suggestion algorithms** in social networks. Platforms like Facebook and LinkedIn use **set operations** to recommend new connections based on **mutual friends** and **common interests**. In this thesis, we utilize **set union, intersection, and difference** to analyze friend relationships and suggest new connections to users. By filtering out existing friends and identifying **friends-of-friends**, the system recommends relevant users while ensuring efficient network growth. This showcases how **discrete structures** are applied in social media to enhance user engagement and connectivity.

The second application focuses on **mathematical modeling of measles outbreaks** using the **SIR (Susceptible-Infected-Recovered) model**. Measles, a highly contagious viral disease, spreads rapidly in populations without vaccination. By representing the dynamics of disease transmission through **matrices**, we analyze how the number of susceptible, infected, and recovered individuals change over time. The infection rate (β) and recovery rate (γ) define the transition process, and through **matrix multiplication**, we can estimate the future spread of measles and evaluate the effectiveness of vaccination strategies. This mathematical approach helps in predicting outbreaks and making informed public health decisions.

Both applications highlight the **practical significance of discrete mathematics** beyond theoretical study. In epidemiology, mathematical models contribute to **disease control and prevention**, while in computer science, set theory helps in **data structuring and recommendation algorithms**. Through these examples, this thesis demonstrates how **basic mathematical structures** provide powerful tools for **analyzing real-world problems and making data-driven decisions**.



Basic Structures: Sets, Functions, Sequences and Matrices

* 1. SETS
     1. Definition of set

A **set** is an unordered collection of distinct objects. Each element in a set is unique, meaning no duplicates are allowed. Sets are generally denoted using uppercase letters, while their elements are represented using lowercase letters.

* + 1. Characteristics of a Set

 **Distinct Elements** – A set cannot contain duplicate elements. Each element appears only once within the set.

 **Unordered** – The arrangement of elements in a set does not matter. For example, the set {1,2,3} is considered the same as {3,2,1}.

 **Well-defined** – A set must have clearly specified elements. There should be no ambiguity in what belongs to the set and what does not.

* + 1. Key Properties of a Set

A set is usually represented using curly brackets {} and contains elements separated by commas. Below are some examples of sets:

**A = {1, 2, 3, 4, 5}** → This represents a set containing five distinct numbers.

**B = {a, e, i, o, u}** → This set contains the vowels of the English alphabet.

**C = {apple, banana, cherry}** → This set consists of three different types of fruits.

**D = {0, 1}** → This set represents the Boolean values, which are often used in computing and logic.

In each case, the elements are clearly defined, do not repeat, and their order does not change the identity of the set.

* + 1. Set Notation
       1. Set Builder Notation

In set builder notation, a set is described using a condition that its elements must satisfy. The notation uses a variable and a condition separated by a vertical bar | or a slash /, which means "such that".

For example:

C = { x | x is a positive integer less than 50 }→This means C contains all positive integers less than 50.

* + - 1. Tabular (Roster) Notation

In tabular notation, the elements of a set are explicitly listed within curly brackets .

For example:

→ This set contains five numbers.

* + - 1. Interval Notation

Interval notation is commonly used to describe continuous sets of numbers, particularly real numbers. Different types of intervals include:

**Closed interval [a, b]:** Includes all numbers between a and b, including the endpoints.

**Left-closed, right-open interval [a, b):** Includes a but not b.

**Left-open, right-closed interval (a, b]:** Includes b but not a.

**Open interval (a, b**): Excludes both a and b.

* + 1. Set Operation
       1. Union of Sets (A ∪ B)

The **union** of two sets, A and B, is a set that contains all elements that belong to either A, B, or both. The union operation ensures that no element is repeated in the resulting set.

Mathematically, the union of A and B is represented as:

* + - 1. Intersection of Sets (A ∩ B)

The **intersection** of two sets, A and B, is a set that contains only those elements that are present in both A and B.

Mathematically, the intersection of A and B is represented as:

* + - 1. Difference of Sets (A − B)

The **difference** between two sets, A and B, is the set that contains all the elements that are in A but not in B.

Mathematically, the difference of A and B is represented as:

* + - 1. Symmetric Difference of Sets (A Δ B)

The **symmetric difference** of two sets, A and B, is the set of elements that belong to either A or B, but not both.

Mathematically, the symmetric difference is given by:

A Δ B = (A−B) ∪ (B−A)

* + - 1. Cartesian Product of Sets (A × B)

The **Cartesian product** of two sets, A and B, is the set of all ordered pairs (a, b), where the first element is from A and the second element is from B.

Mathematically, the Cartesian product is defined as:

* + 1. Type of Sets
       1. Empty Set (Null Set)

An empty set, also known as a null set, is a set that contains no elements. It is a subset of every set and is denoted by:

Key properties of an empty set:

The union of any set with an empty set result in the original set:

The intersection of any set with an empty set result in an empty set: 0

* + - 1. Singleton Set

A singleton set, also called a unit set, is a set that contains exactly one element. It is considered a finite set.

Example:

* + - 1. Finite Set

A finite set is a set that contains a countable number of elements. The number of elements in a finite set is a specific, finite number.

Example:

The cardinality of a set is the number of elements in it.

For the above set:

* + - 1. Infinite Set

An infinite set is a set that contains an uncountable number of elements. There are two types of infinite sets:

Countably Infinite – The set has an infinite number of elements, but they can be listed in a sequence.

Example: The set of natural numbers

Uncountably Infinite – The set has an infinite number of elements that cannot be listed in a sequence.

Example: The set of real numbers between 0 and 1

* + - 1. Equal Sets

Two sets are equal if they contain exactly the same elements, regardless of their order.

Example:

Since both sets contain the same elements, we conclude that:

* + - 1. Subset

A set A is a subset of set B if every element of A is also an element of B. A set is always a subset of itself. This is denoted as:

Example:

Since all elements of A are in B, we write: A⊆B

* + - 1. Superset

A set A is a superset of set B if every element of B is contained in A. A set is always a superset of itself.

This is denoted as:

A ⊇ B

Example: A = {1,2,3,4}, B = {1,2}

Since all elements of B are in A, we write: A ⊇ B

* + - 1. Proper Subset and Proper Superset

A proper subset is a subset that does not include all elements of the other set. If A is a proper subset of B, then:

(A is a proper subset of B)

Example:

Here, A is a proper subset of B because B contains an extra element (3).

Similarly, a proper superset is a set that contains all elements of another set, plus at least one more element.  
Example:

* + - 1. Universal Set

The universal set is the set that contains all possible elements under consideration in a particular discussion. It is usually denoted by U.

Example: If we are considering numbers between 1 and 10, then the universal set might be:

* + - 1. Power Set

The power set of a set A is the set of all possible subsets of A, including the empty set and A itself. The number of elements in the power set is given by: 2^n where n is the number of elements in A.

Example: If , then:

The number of elements in P(A) is:

* + - 1. Disjoint Sets

Two sets are disjoint if they have no common elements. Their intersection is an empty set.

Example:

Since A and B do not share any elements, we write: A∩B = ∅

* + - 1. Complement of a Set

The complement of a set A (denoted as A') is the difference between the universal set and A.

Example:

The complement of A is:

* + - 1. Computer Representation of Sets

**1. Bit Representation of Sets**  
Given a universal set , subsets can be represented using bit strings.

The set of even numbers is represented as .

The set is represented as .

**2. Set Operations Using Bit Strings**

**Union**: The union of and {1,3,5,7,9} is represented as:

Resulting in the set .

**Intersection**: The intersection is given by:

Resulting in the set {1, 3, 5}.

* + - 1. Multisets

A multiset allows elements to appear multiple times. It is represented as:

where denotes the multiplicity of element ​.

**Multiset Operations**  
Given and :

* **Union**:
* **Intersection**:
* **Difference**:
* **Sum**:
  1. **Functions**
     1. **Definition of a Function**

A function assigns each element in to exactly one element in .

* + 1. One-to-One and Onto Functions

**One-to-One (Injection):** A function is injective if different inputs have different outputs.

**Onto (Surjection):** A function is surjective if every element in *B* has a preimage in *A*.

* + 1. **Bijective Functions**

A function is bijective if it is both injective and surjective. The inverse function exists for bijective functions.

* + 1. Composition of Functions

The composition of two functions and is defined as:

* + 1. Floor and Ceiling Functions
* **Floor Function:** is the greatest integer .
* **Ceiling Function:** is the smallest integer .
  1. Sequences and Summations
     1. **Definition of a Sequence**

A sequence is a function from a subset of integers to a set *S*, written as ​.

* + 1. **Arithmetic Progression**  
       A sequence with a common difference d:
    2. **Geometric Progression**  
       A sequence with a common ratio *r*:
    3. **Recurrence Relations**

A sequence where each term depends on previous terms.

Example:

* + 1. Fibonacci Sequence

Defined by:

* + 1. Summations

The sum of terms from ​ to ​ is represented as:

Example:

* 1. Matrices
     1. Definition of a matrix

A matrix is a rectangular array of numbers, symbols, or expressions arranged in rows and columns. It is commonly used in mathematics, physics, engineering, and computer science to represent data, transformations, and systems of equations.

A matrix with m rows and n columns is called an m × n (read as "m by n").

For example, a 2 x 3 matrix looks like this:

* + 1. Matrix Arithmetic

Matrix arithmetic is a fundamental concept in linear algebra that involves various operations such as addition, subtraction, scalar multiplication, matrix multiplication, and transposition

* + - 1. Matrix Addition

Matrix addition is defined for matrices of the same dimensions. Given two matrices and of order, their sum is computed by adding corresponding elements:

For example:

* + - 1. Matrix Subtraction

Matrix subtraction follows the same principle as addition but involves subtracting corresponding elements:

For example:

* + - 1. Scalar Multiplication

A matrix can be multiplied by a scalar (a constant value) by multiplying each element by that scalar:

For example:

* + - 1. Matrix Multiplication

Matrix multiplication is only defined when the number of columns of the first matrix matches the number of rows of the second matrix. Given matrices of order and of order, their product is an m x p matrix where:

For example:

* + - 1. Matrix Transposition

The transpose of a matrix A, denoted as AT, is obtained by interchanging its rows and columns:

For example:

* + - 1. Powers of Matrices

The power of a matrix refers to multiplying a matrix by itself a certain number of times. For a square matrix A and a positive integer n, the nth power of A is denoted as An, and it is calculated by multiplying the matrix *A* by itself n times:

Example:

If A is a 2 x 2 matrix:

Then:

* + 1. Symmetric Matrices

A symmetric matrix is a square matrix that is equal to its transpose, meaning A=AT. For a matrix A, this implies that ​ for all elements.

Key Properties:

* Square: It must be a square matrix (n× n).
* Symmetry: Elements are mirrored across the diagonal.
* Real Eigenvalues: Symmetric matrices have only real eigenvalues.

Example:

* + 1. Join and meet matrices

In the context of matrices, particularly zero-one matrices, the join and meet operations are defined using Boolean algebra. These operations are based on logical OR (for join) and logical AND (for meet). Zero-one matrices consist of only the values 0 and 1, where 1 represents true and 0 represents false. These operations are useful in various applications such as in Boolean algebra, logic circuits, and discrete mathematics.

* + - 1. Definition Of Join

The join of two zero-one matrices A and B, denoted by A ∨ B, is a matrix where each element in position (i, j) is the result of the logical OR operation applied to the corresponding elements of A and B. Mathematically, the join is defined as:

* If either or (or both), then .
* If both and , then .
  + - 1. Definition Of Meet

The meet of two zero-one matrices A and B, denoted by A ∧ B, is a matrix where each element in position (i, j) is the result of the logical AND operation applied to the corresponding elements of A and B. Mathematically, the meet is defined as:

* If both and , then .
* If either or , then .
  + - 1. Example Of Join And Meets

Consider the zero-one matrices:

Join of A and B ( A ∨ B):

Meet of A and B (A ∧ B):

* + 1. Transition Matrix
* **Definition of a Transition Matrix**  
  A transition matrix is a square matrix used to represent the transitions of a Markov chain, where each element in the matrix represents the probability of moving from one state to another.
* **Structure of a Transition Matrix**  
  Each row represents a current state, and each column represents a possible future state. The sum of elements in any row must equal 1.
* **Markov Chains**  
  A system that transitions between states with probabilities that depend only on the current state, not the past states.
* **Example of a Transition Matrix**  
  For a 2-state system with states *A* and *B*, the transition matrix *P* might look like:

Where:

* The probability of transitioning from state *A* to state *A* is 0.8.
* The probability of transitioning from state *A* to state *B* is 0.2.
* The probability of transitioning from state *B* to state *A* is 0.4.
* The probability of transitioning from state *B* to state *B* is 0.6.



Set Operations in Friend Suggestion Algorithms

* 1. Introduction

In today's digital age, social media platforms like Facebook, Twitter, and LinkedIn have revolutionized how we connect with one another. Friend suggestion algorithms are a core feature of these platforms, helping users discover new people to connect with based on their existing social circles. These algorithms aim to enhance social connectivity by recommending individuals with shared connections, interests, or activities. To achieve this, social media platforms leverage mathematical concepts such as set operations.

Set operations—such as union, intersection, and difference—are used to analyze and manipulate the relationships between users. Through the application of these operations, friend suggestion systems efficiently process user data, identifying potential friends based on common connections, mutual acquaintances, or even excluding already existing relationships.

This chapter explores how set operations are applied in the context of social media friend suggestion systems, helping to better understand their practical utility.

* 1. Fundamentals of Set Operations
     1. Set Representation in Programming

In programming, sets are used to store collections of unique items, making them ideal for representing relationships like friendships. For example, a social media user's friends can be represented as a set, where each element in the set is a friend.

In Python, sets are created using curly braces and automatically enforce uniqueness-i.e., no duplicate friends. Here’s an example:

friends\_of\_alice = {'bob', 'charlie', 'david'}

This code snippet creates a set representing Alice's friends. In the context of a social media platform, this set can be used to compute various set operations.

* + 1. Set Operations & Their Role in Friend Suggestion
       1. Union ( ) – Gathering All Possible Connections

The union operation combines two sets, resulting in a new set that contains all unique elements from both sets. In the context of friend suggestions, it can be used to gather all friends of a user’s friends. This helps identify potential connections that the user may not have yet met but shares mutual connections with.

Example: If Alice’s friends are {'bob', 'charlie', 'david'} and Bob’s friends are {'alice', 'emma', 'frank'}, the union operation helps identify all unique friends in Alice’s and Bob’s circles.

* + - 1. Intersection ( ) – Finding Mutual Friends

The intersection operation finds the common elements between two sets. This is useful in identifying mutual friends between two users, which can be valuable in suggesting friends. If two users share mutual connections, they may be more likely to be interested in connecting with each other.

Example: If Alice’s friends are {'bob', 'charlie', 'david'} and Charlie’s friends are {'alice', 'george', 'helen'}, the intersection operation reveals that Alice and Charlie share one mutual friend—'bob'.

* + - 1. Difference ( − ) – Excluding Existing Friends

The difference operation removes elements from one set that are present in another. This operation can be used to exclude users who are already friends from the friend suggestion list. It ensures that the algorithm suggests only new connections that the user has not yet made.

Example: If Alice’s friends are {'bob', 'charlie', 'david'} and Bob’s friends are {'alice', 'emma', 'frank'}, the difference operation helps ensure that the algorithm doesn’t suggest Alice as a friend to Bob.

* 1. Mathematical Model for Friend Suggestion

In the previous sections, we discussed the role of set operations in the context of social media algorithms. In this section, we will dive into a more formal **mathematical model** for friend suggestions. Specifically, we’ll use **graph theory** and **set operations** to illustrate how these suggestions are generated.

* + 1. Graph Representation of Social Networks

Social networks are often represented as **graphs**, where:

* **Nodes (vertices)** represent **users**.
* **Edges** represent **friendships** or **connections** between users.

Consider the following example where users are connected through friendships:

Alice -- Bob -- Charlie

* + 1. Dictionary-Based Representation & Set Operations

In practice, friendships are stored in **dictionaries**. Each key in the dictionary is a **user**, and the corresponding value is a **set** of that user’s friends. For example:

friends\_dict = {

'alice': {'bob', 'charlie', 'david'},

'bob': {'alice', 'emma', 'frank'},

'charlie': {'alice', 'george', 'helen'},

# More users...

}

With this structure, we can efficiently perform **set operations** to manipulate the friendships and suggest new connections. The key operations include:

* **Union ( )**: Combines two sets, resulting in a set of all unique elements.
* **Intersection ( )**: Returns the elements common to both sets.
* **Difference ( )**: Removes the elements of one set from another.
  + 1. Algorithm for Friend Suggestions

Now, let’s break down the algorithm for suggesting new friends based on these operations.

1. **Identify Direct Friends**: First, we identify a user’s **direct friends** by accessing their friend set in the dictionary.
2. **Gather Second-Degree Connections Using Union**: Next, we find friends of the user’s direct friends. This is done by taking the **union** of the sets of friends.
3. **Remove Existing Friends Using Difference**: We remove the user’s **direct friends** from the set of second-degree connections using the **difference** operation, ensuring that no one is suggested who is already a friend.
4. **Rank Suggestions Using Intersection**: To rank the suggestions, we compute the **mutual friends** between the user and each second-degree connection. This is achieved through the **intersection** of their friend sets. The more mutual friends, the higher the ranking.
   1. Implementation in Python

A Python implementation of a friend suggestion algorithm using sets:

# Function for Mutual Friend Searching

def mutual\_friends(user1, user2, friends\_dict):

# '&' for Intersection

    return friends\_dict.get(user1, set()) & friends\_dict.get(user2, set())

# Function for Suggesting Friends

def suggest\_friends(user, friends\_dict):

    friends = friends\_dict.get(user, set())

    suggestions = set()

    for friend in friends:

# '|' for Union

        suggestions |= friends\_dict.get(friend, set())

    suggestions -= friends

    suggestions.discard(user)

    return suggestions

# Example friend connections

friends = {

    'alice': {'bob', 'charlie', 'david'},

    'bob': {'alice', 'emma', 'frank', 'charlie', 'alice'},

    'charlie': {'alice', 'george', 'helen', 'david', 'emma'},

    'david': {'alice', 'ian'},

    'emma': {'bob', 'jack'},

    'frank': {'bob', 'helen', 'charlie'},

    'george': {'charlie'},

    'helen': {'charlie', 'frank', 'bob'},

    'ian': {'david'},

    'jack': {'emma'}

}

# Main Loop

while True:

    user = input("Enter a user's name to get friend suggestions (or type 'q' to quit): ").lower()

    time.sleep(0.5)

    if user == 'q':

        break

    if user in friends:

        # Get Suggestions

        suggestions = suggest\_friends(user, friends)

        # Get Mutual Friends

        if suggestions:

            suggestion\_list = []

            for suggestion in suggestions:

                mutuals = mutual\_friends(user, suggestion, friends)

                suggestion\_list.append(f"{suggestion.capitalize()} (Mutual: {len(mutuals)})")

            print(f"Friend suggestions for {user.capitalize()}:\n{', \n'.join(suggestion\_list)}\n")

        else:

            print(f"No friend suggestions for {user.capitalize()}.")

    else:

        print("User not found. Please try again.")

This algorithm efficiently identifies friend recommendations using set operations.

You can also download the code via GitHub: <https://github.com/maVy14/Friend-Suggestion>

* 1. Case Study: Social Media Friend Suggestion in Practice

Many social media platforms implement variations of this algorithm:

* **Facebook**: Uses mutual friends and machine learning for personalized suggestions.
* **LinkedIn**: Employs professional network analysis and shared connections.
* **Twitter**: Suggests followers based on engagement and common interests.
  1. ****Advantages & Challenges****
     1. ****Advantages****
* **Efficiency**: Set operations provide quick computations.
* **Scalability**: Can handle millions of users.
* **Optimized Storage**: Sets prevent duplication of friend connections.
  + 1. ****Challenges****
* **Privacy Concerns**: Recommending friends based on indirect connections may raise privacy issues.
* **Fake Accounts**: Difficulty in filtering out spam or fake profiles.
* **Computational Load**: Processing large-scale networks requires optimized algorithms.
  1. ****Summary****

Set operations play a crucial role in designing friend suggestion algorithms, enabling social media platforms to enhance user engagement efficiently. By leveraging discrete mathematics, particularly set theory and graph structures, these systems can provide accurate and scalable recommendations. Understanding these concepts allows for the development of more sophisticated and privacy-conscious social networking applications.



Tracking the Pandemic with Mathematics

* 1. Introduction

Mathematical models are crucial for predicting and controlling infectious diseases. One of the most effective ways to model outbreaks is through the **SIR model**, which classifies individuals into three groups: **Susceptible (S), Infected (I), and Recovered (R)**.

In this chapter, we apply **matrix representations** of the SIR model to study the spread of **measles**—a highly contagious viral disease. Unlike COVID-19, measles has a much **higher basic reproduction number (R₀)**, meaning it spreads more rapidly in unvaccinated populations. The **SIR model, implemented using matrices**, allows us to simulate measles outbreaks and analyze the impact of **vaccination and immunity**.

**Objectives of this Chapter**

* Introduce the **SIR model** in the context of **measles outbreaks**.
* Develop a **matrix representation** of measles transmission.
* Use the **transition matrix** to simulate outbreak progression.
* Analyze how vaccination influences the spread of measles.
  1. The SIR Model for Measles
     1. Understanding the SIR Model

The **SIR model** is a foundational framework in epidemiology used to describe the spread of infectious diseases within a population. The model divides the population into three distinct groups or **compartments**:

* Susceptible (S): These are individuals who have not been infected by the disease and are at risk of contracting it upon exposure.
* Infected (I): Individuals in this group are currently infected with the disease and can spread the infection to susceptible individuals.
* Recovered (R): This group consists of individuals who have been infected and recovered from the disease. They are assumed to have gained immunity, which prevents them from being infected again.

The dynamic transitions between these compartments are governed by two critical rates:

* Infection rate (β): This is the probability of transmission per contact between an infected and a susceptible individual. A higher infection rate means that the disease spreads more rapidly within the population.
* Recovery rate (γ): This represents the rate at which infected individuals recover and transition into the immune (recovered) state. This rate is inversely proportional to the duration of infection, i.e., a higher recovery rate means a shorter duration of infectiousness.

The core set of equations governing the SIR model are:

where *N* is the total population size, *S*, *I*, and *R* are the number of susceptible, infected, and recovered individuals, respectively, and *t* is time.

* + 1. The Role of the Total Population Size (N)

While the above equations describe the fundamental dynamics of the disease, they do not take into account the size of the total population. To account for the structure and scale of the population, the equations are typically normalized by dividing by the total population size, *N*, which is the sum of the susceptible, infected, and recovered individuals at any given time:

This adjustment ensures that the equations reflect the relative proportions of susceptible, infected, and recovered individuals in the population, rather than the absolute numbers. Incorporating *N* into the model allows it to scale appropriately, making it applicable to any population size.

The updated system of equations, accounting for the total population *N*, is as follows:

Here, the term ​ represents the fraction of the population that is susceptible to the disease, and ​ represents the fraction that is infected. Dividing by *N* ensures that the disease's spread is modeled based on the relative proportions of the population, rather than absolute numbers, which makes the model more general and scalable.

By normalizing the equations with respect to *N*, the model now accounts for the population size, allowing for more accurate predictions and comparisons across different population scenarios. The total population *N* is assumed to remain constant over time, meaning the sum of *S*, *I*, and *R* will always equal *N*.

* + 1. Measles vs. Other Diseases

The SIR model is particularly relevant for understanding the dynamics of **measles**, a highly contagious viral disease. Measles is unique compared to many other infectious diseases, which makes the SIR model an effective tool for its study. A few reasons for this are:

1. **High Contagion (R₀)**: Measles has an **R₀** (basic reproduction number) of 12 to 18, which means that, in a fully susceptible population, each infected individual can spread the virus to 12 to 18 others. This is significantly higher than other diseases like **COVID-19**, whose R₀ is typically between 2 and 3. The higher the R₀, the more rapidly the disease spreads, making it essential to model and predict the disease's course accurately.
2. **Lifetime Immunity**: After recovering from measles, individuals develop **lifelong immunity**, meaning they do not become susceptible to the disease again. This simplifies the SIR model's representation of the recovered group, as they remain in this compartment permanently without the need for periodic boosters or re-exposure.
3. **Immediate Infectiousness**: Unlike some diseases with a latency period (where individuals do not immediately transmit the disease after being infected), measles has **no latency period**. Infected individuals are immediately infectious to others, which simplifies the modeling process by removing the need to account for delayed transmission.
4. **Vaccination Integration**: Vaccination plays a crucial role in preventing measles and can be easily incorporated into the SIR model. The introduction of a **vaccination rate** *v* effectively reduces the number of susceptible individuals by inoculating them against the disease. This inclusion helps model how vaccination efforts can reduce the spread of the disease, contributing to herd immunity in the population.

In summary, the SIR model is particularly well-suited for analyzing diseases like measles due to its simplicity and the disease's characteristics, such as high contagiousness, lifelong immunity, and no latency period. The ability to integrate vaccination strategies into the model also enhances its relevance for public health planning and disease control.

* 1. Matrix Representation of the SIR Model
     1. 3.1 Defining the State Vector and Transition Matrix

At any time **t**, the state of the population is represented as:

The transition between states is modeled using a **transition matrix (A)**:

Multiplying this matrix by the **current state vector** gives the **next state**:

* + 1. Example: Measles Outbreak in an Unvaccinated Population

Consider a **school outbreak scenario** with 1,000 students:

* **990 Susceptible (S)**
* **10 Infected (I)**
* **0 Recovered (R)**

Given:

* **β = 0.2** (20% transmission probability).
* **γ = 0.1** (10% recovery per step).

**Setting Up the Initial State Vector**

The **transition matrix A** at **Day 1** is computed as:

* 1. Numerical Simulation of Measles Spread

Using the equation:

Computing each row:

1. **New Susceptible (S(1)):**
2. **New Infected (I(1)):**
3. **New Recovered (R(1)):**

Thus, after **one day**, the new state is:

* 988 Susceptible
* 11 Infected (increase from 10)
* 1 Recovered

This is our prediction of measles spread in 10 days:

A graph showing the number of measles

AI-generated content may be incorrect.

Figure 1

This is prediction for 100 days:

A graph of different colored lines

AI-generated content may be incorrect.

Figure 2

**Simulating the Full Outbreak**

Iterating this process for multiple days shows:

* **Infected population initially grows** before peaking.
* **Eventually, herd immunity is reached** as S decreases.
* **Vaccination drastically reduces peak infections.**
  1. Analysis and Interpretation
     1. Understanding Measles Spread in an Unvaccinated Population
* Without vaccination, measles infects **a large majority of the population**.
* The **peak infection period** occurs within a few weeks.
* Recovery eventually leads to **herd immunity**, but **at the cost of mass infections**.
  + 1. Impact of Vaccination

If **90% of children are vaccinated**, then **S(0) = 100** instead of 990.

Recomputing with a **lower S**:

* **Much lower peak infections.**
* **Outbreak dies out faster.**
* **Fewer complications and deaths.**
  + 1. Limitations of the Model
* Assumes **homogeneous mixing** (real-world contacts vary).
* **No birth/death rates** (important for long-term measles control).
* **Fixed parameters (β and γ)**, though they vary with interventions.
  1. Summary

This chapter explored the use of the **SIR model** in tracking and predicting measles outbreaks through **matrices** and the **transition matrix approach**. By categorizing individuals into **Susceptible, Infected, and Recovered (SIR)** groups, we demonstrated how measles spreads over time and how factors like vaccination influence its trajectory. The transition matrix allowed us to simulate population changes efficiently, providing insights into infection dynamics.

One key takeaway is the **importance of vaccination**, which significantly reduces the number of susceptible individuals and helps control outbreaks. The model also highlights how mathematical tools aid in **public health decision-making**, such as planning vaccination strategies and predicting outbreak risks.

Future extensions could refine the model by incorporating **exposed individuals (SEIR model)**, real-world **social networks**, and **machine learning** for real-time adjustments. These enhancements would improve accuracy and make the model even more useful for managing infectious diseases like measles.



Conclusion

This thesis has explored the fundamental structures of **discrete mathematics** and their real-world applications, focusing on **matrices and set operations**. Through these mathematical tools, we analyzed two important case studies: **tracking the spread of measles using the SIR model** and **friend suggestion mechanisms in social networks**.

The **SIR model**, represented using **transition matrices**, provided a structured approach to understanding how measles spreads within a population. By simulating infection dynamics over time, we demonstrated how **vaccination** plays a crucial role in controlling outbreaks. The model highlighted how mathematical predictions can support **public health decisions**, such as optimizing vaccination strategies to prevent large-scale epidemics.

In the **friend suggestion system**, we applied **set operations**, specifically **union and intersection**, to identify mutual connections and recommend new friends. This showed how fundamental concepts of **set theory** can be used in real-world applications like social networking, enhancing user engagement by suggesting relevant connections.

Overall, this thesis illustrates the **practical power of discrete mathematics** in solving real-world problems. Whether in **epidemiology** or **technology**, the use of **matrices and set operations** provides valuable insights and efficient solutions. Future research could explore more advanced mathematical models to improve accuracy and applicability in these domains.

References

* Ponnusamy, K., et al. (2019). *Mathematical Modelling of the Impact of Age-Structured Vaccination on Measles Transmission Dynamics*. PMC7321055. Retrieved from <https://pmc.ncbi.nlm.nih.gov/articles/PMC7321055/>
* Tom Rocks Maths. (n.d.). *Mathematical Concepts and Tutorials*. Retrieved from [Tom Rocks Maths YouTube Channel](https://www.youtube.com/channel/UCO7D4Qy0x07WxeO2rT0trXQ)
* Various websites. (n.d.). *Google Search Results*. Retrieved from Google.com.