University of Engineering and Management

Institute of Engineering & Management, Salt Lake Campus
Institute of Engineering & Management, New Town Campus
University of Engineering & Management, Jaipur

Syllabus for B.Tech Admission Batch 2023

Subject Name: Discrete Mathematics
Credit: 3
Lecture Hours: 36
Subject Code: PCCCS401

Course objective:

(1) To introduce the concepts of sets, relations, and functions and to perform the operations associated with sets, functions, and relations.
(2) To introduce the concepts of mathematical logic, generating functions and recurrence relations and their implementation.
(3) To introduce the concepts of groups, fields etc and their real-life applications in cryptography;
(4) To introduce concepts of Graph Theory and network flow problems

Course Outcomes:

PCCCS401.1: Ability to apply mathematical logic to solve Problems.

PCCCS401.2: Able to use fundamental mathematical concepts such as sets relations, functions, congruence etc to solve problems in cryptography.

PCCCS401.3: Able to formulate problems and solve recurrence relations.

PCCCS401.4: Able to model and solve real world problems using graphs and trees.

CO-PO mapping:

CO. NO.	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
PCCCS401. 1	2	3	3	3	1	1	-	2	2	3	2	-
PCCCS401. 2	2	2	3	3	2	1	2	1	2	-	1	3
PCCCS401. 3	2	2	3	3	3	1	1	1	2	2	1	3
PCCCS401. 4	1	3	3	3	3	1	3	-	2	1	2	1

Study material **Coursera** **Linkedin** **NPTEL**

Module number	Topic	Sub-topics	Mapping with Industry and International Academia	Lecture Hours	Corresponding Lab Assignment
1	**Sets, Relation and Function, Principles of mathematical induction and cryptography**	Operations and Laws of Sets, Cartesian Products, Binary Relation, Partial Ordering Relation, Equivalence Relation, Image of a Set, Sum and Product of Functions, Bijective functions, Inverse and Composite Function, Size of a Set, The Well-Ordering Principle, The Division algorithm, Prime Numbers, The Greatest Common Divisor, Euclidean Algorithm, The Fundamental Theorem of Arithmetic. Coprimality (or Euler's totient function), Chinese Remainder Theorem.	***International Academia:*** (*https://ocw.mit.edu/courses/18-310-principles-of-discrete-applied-mathematics-fall-2013/pages/syllabus/*) ***AICTE-prescribed syllabus:*** (*https://www.aicte-india.org/sites/default/files/Model_Curriculum/AICTE%20-%20UG%20CSE.pdf)* ***Industry Mapping: https://www.sagemath.org/ ,***	10	(1) Implement Euclidean algorithm using C/ Python; (2) Implement RSA algorithm using C/ Python; (3) Implement Fermat's little theorem / Primality checking using C / Python. (4) Check if any two given number is co-prime using Python / C.

			MATLAB		

2	**Basic counting techniques, Propositional Logic, Proof Techniques**	**Basic counting techniques:** Inclusion and exclusion principle, Pigeon-hole principle, permutation and combination. **Propositional Logic:** Syntax, Semantics, Validity and Satisfiability, Basic Connectives and Truth Tables, Logical Equivalence: The Laws of Logic, Logical Implication, Rules of Inference, The use of Quantifiers. **Proof Techniques:** Some Terminology, Proof Methods and Strategies, Forward Proof, Proof by Contradiction, Proof by Contraposition, Proof of Necessity and Sufficiency.	***International Standards*** : (*https://ocw.mit.edu/courses/18-310-principles-of-discrete-applied-mathematics-fall-2013/pages/syllabus/*) ***AICTE prescribed syllabus:*** *(https://www.aicte-india.org/sites/default/files/Model_Curriculum/AICTE%20-%20UG%20CSE.pdf*) ***Industry Mapping:*** ***https://www.sagemath.org/ , MATLAB***	12	(1) Constructing n-SAT/3-SAT solver using C/ Python; (2) Constructing propositional logic examples using Python;

		Boolean Algebra: Identities of Boolean Algebra, Duality, Representation of Boolean Function, Disjunctive and Conjunctive Normal Form.			

3	**Algebraic Structures and Morphism:**	Algebraic Structures with one Binary Operation, Semi Groups, Monoids, Groups, Permutation Groups, Normal Subgroups, Ring, Field, Vector spaces, Inner-product spaces	***International Standards:*** (*https://ocw.mit.edu/courses/18-310-principles-of-discrete-applied-mathematics-fall-2013/pages/syllabus/*) ***AICTE prescribed syllabus:*** (*https://www.aicte-india.org/sites/default/files/Model_Curriculum/AICTE%20-%20UG%20CSE.pdf*) ***Industry Mapping:*** ***https://www.sagem***	8	(1) Conversion of *First Order Logic* statements to *Conjunctive Normal Form using Python/ SAGEMATH;* (2) Conversion of *First Order Logic* statements to *Disjunctive Normal Form using Python/ SAGEMATH;*

			ath.org/ , MATLAB		
4	**Graphs and Trees**	Graphs and their properties, Degree, Connectivity, Path, Cycle, Sub Graph, Isomorphism, Eulerian and Hamiltonian Walks, Graph Coloring, Planar Graphs, Matching, Trees	***International Standards :*** (*https://ocw.mit.edu/courses/18-310-principles-of-discrete-applied-mathematics-fall-2013/pages/syllabus/*) ***AICTE prescribed syllabus:*** (*https://www.aicte-india.org/sites/default/files/Model_Curriculum/AICTE%20-%20UG%20CSE.pdf*) ***Industry Mapping:*** ***https://www.sagemath.org/, MATLAB***	6	(1) Implementation of maximum flow problem using Python; (2) Checking a graph is Hamiltonian using Python / SAGEMATH.

Textbooks: (1) Discrete Mathematics and Application by Kenneth Rosen, 8th Edition.

(2) ***Introductory Discrete Mathematics by V. K. Balakrishnan, Prentice Hall***

**Submitted by Dr. Prithwineel Paul, Dr. Bivas Bhaumick Dept. of CSE, IEM, Kolkata, Salt Lake Campus*

STUDY MATERIAL

Course objective:

(1) To introduce the concepts of sets, relations, and functions. To perform the operations associated with sets, functions, and relations.

(2) To introduce the concepts of mathematical logicTo introduce generating functions and recurrence relations.

(3) To introduce the concepts of groups, fields etc and their real-life applications in cryptography;

(4) To use Graph Theory for solving problems and solving network flow problems

MODULE 1

A set is an unordered collection of objects, called elements or members of the set. A set is said to contain its elements. We write a ∈ A to denote that a is an element of the set A. The notation a ∈ A denotes that a is not an element of the set A.

Two sets are equal if and only if they have the same elements. Therefore, if A and B are sets, then A and B are equal if and only if ∀x(x ∈ A ↔ x ∈ B). We write A = B if A and B are equal sets.

The set A is a subset of B if and only if every element of A is also an element of B. We use the notation A ⊆ B to indicate that A is a subset of the set B.

For every set S, (i) ∅ ⊆ S and (ii) S ⊆ S.

Let A and B be sets. The union of the sets A and B, denoted by A ∪ B, is the set that contains those elements that are either in A or in B, or in both.

Let A and B be sets. The intersection of the sets A and B, denoted by A ∩ B, is the set containing those elements in both A and B.

Two sets are called disjoint if their intersection is the empty set.

Let A and B be sets. The difference of A and B, denoted by A − B, is the set containing those elements that are in A but not in B. The difference of A and B is also called the complement of B with respect to A.

Let U be the universal set. The complement of the set A, denoted by $\bar{A}$, is the complement of A with respect to U. Therefore, the complement of the set A is U − A.

Showing Two Sets are Equal To show that two sets A and B are equal, show that A ⊆ B and B ⊆ A.

A ∩ U = A ;A ∪∅= A Identity laws
A ∪ U = U Domination laws A ∩∅=∅
A ∪ A = A Idempotent laws A ∩ A = A
($\bar{\bar{A}}$) = A
A ∪ B = B ∪ A Commutative laws A ∩ B = B ∩ A
A ∪ (B ∪ C) = (A ∪ B) ∪ C Associative laws A ∩ (B ∩ C) = (A ∩ B) ∩ C
A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C) Distributive laws A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)
$\overline{A \cap B}$ **= A ∪ B De Morgan's laws A ∪ B = A ∩ B**
A ∪ (A ∩ B) = A Absorption laws A ∩ (A ∪ B) = A
A ∪ A = U Complement laws A ∩ A = ∅

Let S be a set. If there are exactly n distinct elements in S where n is a nonnegative integer, we say that S is a finite set and that n is the cardinality of S. The cardinality of S is denoted by |S|.

A set is said to be infinite if it is not finite.

Given a set S, the power set of S is the set of all subsets of the set S. The power set of S is denoted by P(S).

Let A and B be sets. The Cartesian product of A and B, denoted by A × B, is the set of all ordered pairs (a, b), where a ∈ A and b ∈ B. Hence, A × B = {(a, b) | a ∈ A ∧ b ∈ B}.

The Cartesian product of the sets A1, A2,...,An, denoted by A1 × A2 ×⋯× An, is the set of ordered n-tuples (a1, a2,...,an), where ai belongs to Ai for i = 1, 2,...,n. In other words, A1 × A2 ×⋯× An = {(a1, a2,...,an) | ai ∈ Ai for i = 1, 2,...,n}.

Let A and B be sets. A binary relation from A to B is a subset of A × B.

In other words, a binary relation from A to B is a set R of ordered pairs where the first element of each ordered pair comes from A and the second element comes from B. We use the notation aRb to denote that (a, b) ∈ R and a R b to denote that (a, b) /∈ R. Moreover, when (a, b) belongs to R, a is said to be related to b by R.

A relation on a set A is a relation from A to A.

A relation R on a set A is called reflexive if (a, a) ∈ R for every element a ∈ A.

A relation R on a set A is called symmetric if(b, a) ∈ R whenever(a, b) ∈ R, for all a, b ∈ A. A relation R on a set A such that for all a, b ∈ A, if (a, b) ∈ R and (b, a) ∈ R, then a = b is called antisymmetric.

A relation R on a set A is called transitive if whenever (a, b) ∈ R and (b, c) ∈ R, then (a, c) ∈ R, for all a, b, c ∈ A.

Let A and B be nonempty sets. A function f from A to B is an assignment of exactly one element of B to each element of A. We write f (a) = b if b is the unique element of B assigned by the function f to the element a of A. If f is a function from A to B, we write f : A → B.

If f is a function from A to B, we say that A is the domain of f and B is the codomain of f. If f (a) = b, we say that b is the image of a and a is a preimage of b. The range, or image, of f is the set of all images of elements of A. Also, if f is a function from A to B, we say that f maps A to B.

Let f be a function from A to B and let S be a subset of A. The image of S under the function f is the subset of B that consists of the images of the elements of S. We denote the image of S by f (S), so f (S) = {t | ∃ s ∈S (t = f (s))}. We also use the shorthand {f (s) | s ∈ S} to denote this set.

A function f is said to be one-to-one, or an injunction, if and only if f (a) = f (b)implies that a = b for all a and b in the domain of f. A function is said to be injective if it is one-to-one.

A function f from A to B is called onto, or a surjection, if and only if for every element b ∈ B there is an element a ∈ A with f (a) = b. A function f is called surjective if it is onto.

The function f is a one-to-one correspondence, or a bijection, if it is both one-to-one and onto. We also say that such a function is bijective.

Let f be a one-to-one correspondence from the set A to the set B. The inverse function of f is the function that assigns to an element b belonging to B the unique element a in A such that f (a) = b. The inverse function of f is denoted by f −1. Hence, f −1(b) = a when f (a) = b.

Let g be a function from the set A to the set B and let f be a function from the set B to the set C. The composition of the functions f and g, denoted for all a ∈ A by f ◦ g, is defined by (f ◦ g)(a) = f (g(a)).

Let a and b be integers, not both zero. The largest integer d such that d | a and d | b is called the greatest common divisor of a and b. The greatest common divisor of a and b is denoted by gcd(a, b).

The integers a and b are relatively prime if their greatest common divisor is 1.

The least common multiple of the positive integers a and b is the smallest positive integer that is divisible by both a and b. The least common multiple of a and b is denoted by lcm(a, b).

Let a = bq + r, where a, b, q, and r are integers. Then gcd(a, b) = gcd(b, r).

ALGORITHM 1 The Euclidean Algorithm.
procedure gcd(a, b: positive integers)
x := a

y := b
while y not equal to 0
r := x mod y
x := y
y := r
return x{gcd(a, b) is x}

If a and b are positive integers, then there exist integers s and t such that gcd(a, b) = sa + tb.

THE CHINESE REMAINDER THEOREM Let m1, m2,...,mn be pairwise relatively prime positive integers greater than one and a1, a2,...,an arbitrary integers. Then the system
x ≡ a1 (mod m_1),
x ≡ a2 (mod m_2),
. . .
x ≡ an (mod m_n)
has a unique solution modulo m = m_1m_2 ⋯ m_n. (That is, there is a solution x with 0 ≤ x.

FERMAT'S LITTLE THEOREM If p is prime and a is an integer not divisible by p, then a^{p-1} ≡ 1 (mod p). Furthermore, for every integer a we have a^{p} ≡ a (mod p).

A cryptosystem is a five-tuple (P, C, K, E, D), where P is the set of plaintext strings, C is the set of ciphertext strings, K is the keyspace (the set of all possible keys), E is the set of encryption functions, andD is the set of decryption functions.We denote by Ek the encryption function in E corresponding to the key k and Dk the decryption function in D that decrypts ciphertext that was encrypted using Ek, that is Dk(Ek(p)) = p, for all plaintext strings p.

All classical ciphers, including shift ciphers and affine ciphers, are examples of private key cryptosystems. In a private key cryptosystem, once you know an encryption key, you can quickly find the decryption key. So, knowing how to encrypt messages using a particular key allows you to decrypt messages that were encrypted using this key.

To avoid the need for keys to be shared by every pair of parties that wish to communicate securely, in the 1970s cryptologists introduced the concept of public key cryptosystems. When such cryptosystems are used, knowing how to send an encrypted message does not help decrypt messages. In such a system, everyone can have a publicly known encryption key. Only the decryption keys are kept secret, and only the intended recipient of a message can decrypt it, because, as far as it is currently known, knowledge of the encryption key does not let someone recover the plaintext message without an extraordinary amount of work (such as billions of years of computer time).

The RSA Cryptosystem: In 1976, three researchers at the Massachusetts Institute of Technology—Ronald Rivest, Adi Shamir, and Leonard Adelman—introduced to the world a public key cryptosystem, known as the RSA system, from the initials of its inventors. As often happens with cryptographic discoveries, the RSA system had been discovered several years earlier in secret government research in the United Kingdom. Clifford Cocks, working in secrecy at the United Kingdom's Government Communications Headquarters (GCHQ), had discovered this cryptosystem in 1973. However, his invention was unknown to the outside world until the late 1990s, when he was allowed to share classified GCHQ documents from the early 1970s. (An excellent account of this earlier discovery, as well as the work of Rivest, Shamir, and Adleman, can be found in [Si99].)

In the RSA cryptosystem, each individual has an encryption key (n, e) where n = pq, the modulus is the product of two large primes p and q, say with 200 digits each, and an exponent e that is relatively prime to (p − 1)(q − 1). To produce a usable key, two large primes must be found. This can be done quickly on a computer using probabilistic primality tests, referred to earlier in this section. However, the product of these primes n = pq, with approximately 400 digits, cannot, as far as is currently known, be factored in a reasonable length of time. As we will see, this is an important reason why decryption cannot, as far as is currently known, be done quickly without a separate decryption key.

RSA Encryption: To encrypt messages using a particular key (n, e), we first translate a plaintext message M into sequences of integers. To do this, we first translate each plaintext letter into a two-digit number, using the same translation we employed for shift ciphers, with one key difference. That is, we include an initial zero for the letters A through J, so that A is translated into 00, B into 01, ... , and J into 09. Then, we concatenate these two-digit numbers into strings of digits. Next, we divide this string into equally sized blocks of 2N digits, where 2N is the largest even number such that the number 2525 ... 25 with 2N digits does not exceed n. (When necessary, we pad the plaintext message with dummy Xs to make the last block the same size as all other blocks.)
After these steps, we have translated the plaintext message M into a sequence of integers m1, m2,...,mk for some integer k. Encryption proceeds by transforming each block mi to a ciphertext block ci. This is done using the function $C = M^e \bmod n$**. (To perform the encryption, we use an algorithm for fast modular exponentiation, such as Algorithm 5 in Section 4.2.) We leave the encrypted message as blocks of numbers and send these to the intended recipient. Because the RSA cryptosystem encrypts blocks of characters into blocks of characters, it is a block cipher.**

RSA Decryption: The plaintext message can be quickly recovered from a ciphertext message when the decryption key d, an inverse of e modulo (p − 1)(q − 1), is known. [Such an inverse exists because gcd(e, (p − 1)(q − 1)) = 1.] To see this, note that if de ≡ 1 (mod (p − 1)(q − 1)), there is an integer k such that de = 1 + k(p − 1)(q − 1). It follows that $C^d \equiv (M^e)^d = M^{de} = M^{1+k(p-1)(q-1)} \pmod n$**.**

By Fermat's little theorem [assuming that gcd(M, p) = gcd(M, q) = 1, it follows that $M^{p-1} \equiv 1\ (mod\ p)$**and** $M^{q-1} \equiv 1\ (mod\ q)$**. Consequently,** $C^d \equiv M^{(Mp-1)\,k(q-1)} \equiv M \cdot 1 = M\ (mod\ p)$ **and** $C^d \equiv M^{(Mq-1)\,k(p-1)} \equiv M \cdot 1 = M\ (mod\ q)$**. Because gcd(p, q) = 1, it follows by the Chinese remainder theorem that** $C^d \equiv M\ (mod\ pq)$.

Questions(Bloom's level 4, 5 and 6):

(1) Justify the following statement: A × B = ∅ if and only if A = ∅ or B = ∅.
(2) Justify the following statement: (A1 ∪ A2) × B = (A1 × B) ∪ (A2 × B)
(3) Compare and contrast linear and exponential functions, highlighting their key characteristics.
(4) Evaluate the properties of power sets. How does the cardinality of a power set relate to the cardinality of the original set?
(5) Investigate the concept of subsets and proper subsets. How does the relationship between sets change when considering proper subsets?
(6) Explore the concept of partitioning a set. How does partitioning relate to equivalence relations, and what implications does it have for set theory?
(7) Analyze the intersection and union operations on sets. Provide examples to illustrate the properties of these operations and how they relate to real-world scenarios.
(8) Investigate the conditions under which two sets are considered equal. Provide examples and non-examples to demonstrate your understanding.
(9) Investigate the concept of inverse functions in the context of bijective functions. How is the inverse of a bijective function related to its original function?
(10) Examine the graphical representation of a bijective function. How can you visually determine if a function is bijective by looking at its graph?
(11) Analyze the composition of two bijective functions. What can be concluded about the composition in terms of injectivity and surjectivity?
(12) Analyze the definition of the GCD of two numbers. Explain its fundamental properties and how it is related to the numbers involved.
(13) Calculate the value of t and s where 5 = g.c.d (475, 120) = 475 t + 120 s.
(14) Calculate the remainder when 11^{40} **is divided by 8.**
(15) How many reflexive relations are there on a set with n elements?
(16) Determine whether the relation R on the set of all real numbers is reflexive, symmetric, antisymmetric, and/or transitive, where (x, y) ∈ R if and only if a) x + y = 0. b) x = ±y. c) x − y is a rational number. d) x = 2y. e) xy ≥ 0. f) xy = 0. g) x = 1. h) x = 1 or y = 1.

MODULE 2

Let p be a proposition. The negation of p, denoted by ¬p (also denoted by p), is the statement "It is not the case that p." The proposition ¬p is read "not p." The truth value of the negation of p, ¬p, is the opposite of the truth value of p.

Let p and q be propositions. The conjunction of p and q, denoted by p ∧ q, is the proposition "p and q." The conjunction p ∧ q is true when both p and q are true and is false otherwise.

Let p and q be propositions. The disjunction of p and q, denoted by p ∨ q, is the proposition "p or q." The disjunction p ∨ q is false when both p and q are false and is true otherwise.

Let p and q be propositions. The exclusive or of p and q, denoted by p ⊕ q, is the proposition that is true when exactly one of p and q is true and is false otherwise.

Let p and q be propositions. The conditional statement p → q is the proposition "if p, then q." The conditional statement p → q is false when p is true and q is false, and true otherwise. In the conditional statement p → q, p is called the hypothesis (or antecedent or premise) and q is called the conclusion (or consequence).

CONVERSE, CONTRAPOSITIVE, AND INVERSE We can form some new conditional statements starting with a conditional statement p → q. In particular, there are three related conditional statements that occur so often that they have special names. The proposition q → p is called the converse of p → q. The contrapositive of p → q is the proposition ¬q → ¬p. The proposition ¬p → ¬q is called the inverse of p → q. We will see that of these three conditional statements formed from p → q, only the contrapositive always has the same truth value as p → q.

Let p and q be propositions. The biconditional statement p ↔ q is the proposition "p if and only if q." The biconditional statement p ↔ q is true when p and q have the same truth values, and is false otherwise. Biconditional statements are also called bi-implications.

A compound proposition that is always true, no matter what the truth values of the propositional variables that occur in it, is called a tautology. A compound proposition that is always false is called a contradiction. A compound proposition that is neither a tautology nor a contradiction is called a contingency.

$p \rightarrow q \equiv \neg p \vee q$
$p \rightarrow q \equiv \neg q \rightarrow \neg p$
$p \vee q \equiv \neg p \rightarrow q$
$p \wedge q \equiv \neg(p \rightarrow \neg q)$

$\neg(p \rightarrow q) \equiv p \wedge \neg q$
$(p \rightarrow q) \wedge (p \rightarrow r) \equiv p \rightarrow (q \wedge r)$
$(p \rightarrow r) \wedge (q \rightarrow r) \equiv (p \vee q) \rightarrow r$
$(p \rightarrow q) \vee (p \rightarrow r) \equiv p \rightarrow (q \vee r)$
$(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$

A statement of the form P (x1, x2,...,xn) is the value of the propositional function P at the n-tuple (x1, x2,...,xn), and P is also called an n-place predicate or a n-ary predicate.

The universal quantification of P (x) is the statement "P (x) for all values of x in the domain." The notation $\forall x P(x)$ denotes the universal quantification of P (x). Here $\forall$ is called the universal quantifier. We read $\forall x P(x)$ as "for all xP (x)" or "for every xP (x)." An element for which P (x) is false is called a counterexample of $\forall x P(x)$.

The existential quantification of P (x) is the proposition "There exists an element x in the domain such that P (x)." We use the notation $\exists x P(x)$ for the existential quantification of P (x). Here $\exists$ is called the existential quantifier.

De Morgan's Laws for Quantifiers.
$\neg \forall x P(x) \equiv \exists x \neg P(x)$.
$\neg \exists x Q(x) \equiv \forall x \neg Q(x)$.

Rules of inference:
$(p \wedge (p \rightarrow q)) \rightarrow q$
$(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$
$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$
$((p \vee q) \wedge \neg p) \rightarrow q$
$p \rightarrow (p \vee q)$
$(p \wedge q) \rightarrow p$
$((p) \wedge (q)) \rightarrow (p \wedge q)$
$((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$

(The Pigeonhole Principle, simple version.) If k + 1 or more pigeons are distributed among k pigeonholes, then at least one pigeonhole contains two or more Pigeons.

(The Pigeonhole Principle.) If n or more pigeons are distributed among k > 0 pigeonholes, then at least one pigeonhole contains at least $\lceil n/k \rceil$ pigeons.

1. **Construct a truth table for each of these compound propositions. a) $p \rightarrow \neg p$ b) $p \leftrightarrow \neg p$ c) $p \oplus (p \vee q)$ d) $(p \wedge q) \rightarrow (p \vee q)$ e) $(q \rightarrow \neg p) \leftrightarrow (p \leftrightarrow q)$ f) $(p \leftrightarrow q) \oplus (p \leftrightarrow \neg q)$**
2. **. Construct a truth table for $((p \rightarrow q) \rightarrow r) \rightarrow s$.**

(3) Construct a truth table for $(p \leftrightarrow q) \leftrightarrow (r \leftrightarrow s)$.

(4) Justify that $\neg(p \vee (\neg p \wedge q))$ and $\neg p \wedge \neg q$ are logically equivalent.

(5) Justify that $(p \wedge q) \rightarrow (p \vee q)$ is a tautology.

(6) Show that $p \leftrightarrow q$ and $\neg p \leftrightarrow \neg q$ are logically equivalent.

(7) Show that $\rightarrow q$ and $\neg q \rightarrow \neg p$ are logically equivalent.

(8) Show that $\neg p \leftrightarrow q$ and $p \leftrightarrow \neg q$ are logically equivalent.

(9) Show that $\neg(p \oplus q)$ and $p \leftrightarrow q$ are logically equivalent.

(10) Show that the following statements are logically equivalent. $\neg(p \leftrightarrow q)$ and $\neg p \leftrightarrow q$

(11) $(p \rightarrow q) \wedge (p \rightarrow r)$ and $p \rightarrow (q \wedge r)$

(12) $(p \rightarrow r) \wedge (q \rightarrow r)$ and $(p \vee q) \rightarrow r$

(13) $(p \rightarrow q) \vee (p \rightarrow r)$ and $p \rightarrow (q \vee r)$

(14) $(p \rightarrow r) \vee (q \rightarrow r)$ and $(p \wedge q) \rightarrow r$

(15) $\neg p \rightarrow (q \rightarrow r)$ and $q \rightarrow (p \vee r)$

(16) Show that $\forall x P(x) \vee \forall x Q(x)$ and $\forall x(P(x) \vee Q(x))$ are not logically equivalent.

(17) Show that $\exists x P(x) \wedge \exists x Q(x)$ and $\exists x(P(x) \wedge Q(x))$ are not logically equivalent.

(18) Show that $\forall x P(x) \wedge \forall x Q(x)$ and $\forall x(P(x) \wedge Q(x))$ are logically equivalent.

(19) Show that $\exists x P(x) \vee \exists x Q(x)$ and $\exists x(P(x) \vee Q(x))$ are logically equivalent.

(20) Express the negations of each of these statements so that all negation symbols immediately precede predicates. a) $\forall x \exists y \forall z T(x, y, z)$ b) $\forall x \exists y P(x, y) \vee \forall x \exists y Q(x, y)$ c) $\forall x \exists y(P(x, y) \wedge \exists z R(x, y, z))$ d) $\forall x \exists y(P(x, y) \rightarrow Q(x, y))$.

(21) Show that the premises $(p \wedge q) \vee r$ and $r \rightarrow s$ imply the conclusion $p \vee s$.

(22) Show that the premises "If you send me an e-mail message, then I will finish writing the program," "If you do not send me an e-mail message, then I will go to sleep early," and "If I go to sleep early, then I will wake up feeling refreshed" lead to the conclusion "If I do not finish writing the program, then I will wake up feeling refreshed."

(23) Every student in a discrete mathematics class is either a computer science or a mathematics major or is a joint major in these two subjects. How many students are in the class if there are 38 computer science majors (including joint majors), 23 mathematics majors (including joint majors), and 7 joint majors?

(24) How many different functions are there from a set with 10 elements to sets with the following numbers of elements? a) 2 b) 3 c) 4 d) 5

(25) How many one-to-one functions are there from a set with five elements to sets with the following number of elements? a) 4 b) 5 c) 6 d) 7

(26) How many functions are there from the set {1, 2,...,n}, where n is a positive integer, to the set {0, 1}?

(27) How many functions are there from the set {1, 2,...,n}, where n is a positive integer, to the set {0, 1} a) that are one-to-one? b) that assign 0 to both 1 and n? c) that assign 1 to exactly one of the positive integers less than n?

(28) In a discrete mathematics class every student is a major in computer science or mathematics or both. The number of students having computer science as a major (possibly along with mathematics) is 25; the number of students having mathematics as a major (possibly along with computer science) is 13; and the number of students majoring in both computer science and mathematics is 8. How many students are in the class?

(29) A total of 1232 students have taken a course in Spanish, 879 have taken a course in French, and 114 have taken a course in Russian. Further, 103 have taken courses in both Spanish and French, 23 have taken courses in both Spanish and Russian, and 14 have taken courses in both French and Russian. If 2092 students have taken a course in at least one of Spanish French and Russian, how many students have taken a course in all 3 languages.

(30) Prove that if seven distinct numbers are selected from {1, 2, . . . , 11}, then some two of these numbers sum to 12.

(31) Prove that if 11 integers are selected from among {1, 2, . . . , 20}, then the selection includes integers a and b such that $a - b = 2$.

MODULE 3

Semigroups:

A semigroup is a set equipped with an associative binary operation. In simpler terms, it is a mathematical structure consisting of a set of elements and an operation that combines any two elements to produce another element from the set. The key property is associativity, meaning that the way elements are grouped in the operation does not affect the result. Formally, for a semigroup (S, ∗), where S is a set and ∗ is a binary operation, it satisfies the associative law:

(a∗ b)∗ c=a∗ (b∗ c)for all a,b,c∈S(a∗ b)∗ c=a∗ (b∗ c)for all a,b,c∈S

Monoids:

A monoid is an extension of a semigroup with the addition of an identity element. Specifically, a monoid (M, ∗) consists of a set M, a binary operation ∗ , and an identity element e such that for all elements a in M:

1. **Associativity: (a∗ b)∗ c=a∗ (b∗ c)(a∗ b)∗ c=a∗ (b∗ c) for all a,b,c∈Ma,b,c∈M.**
2. **Identity Element: There exists an element e in M such that a∗ e=e∗ a=aa∗ e=e∗ a=a for all a∈Ma∈M.**

In summary, while both semigroups and monoids involve sets and binary operations, monoids include the additional concept of an identity element.

Groups:

In the realm of abstract algebra, a group is a fundamental mathematical structure that captures the concept of symmetry and transformations. Formally, a group (G, ∗) consists of a set G and a binary operation ∗ satisfying the following properties:

1. **Closure: For any elements a,ba,b in G, a∗ ba∗ b is also in G.**
2. **Associativity: (a∗ b)∗ c=a∗ (b∗ c)(a∗ b)∗ c=a∗ (b∗ c) for all a,b,ca,b,c in G.**
3. **Identity Element: There exists an element ee in G such that a∗ e=e∗ a=aa∗ e=e∗ a=a for all aa in G.**
4. **Inverse Element: For every element aa in G, there exists an element a−1a−1 in G such that a∗ a−1=a−1∗ a=ea∗ a−1=a−1∗ a=e, where ee is the identity element.**

Permutation Groups:

Permutation groups are a specific type of group that focuses on the study of rearrangements or permutations of a set. Formally, a permutation group on a set XX is a group consisting of all possible bijective mappings (permutations) from XX to itself, where the group operation is function composition.

1. **Definition: Let X be a set, and let $S(X)$ denote the set of all bijections (permutations) from X to itself. A permutation group on X is a subgroup of $S(X)$ under function composition.**
2. **Group Structure: The elements of a permutation group are the various ways the elements of X can be rearranged. The group operation is the composition of permutations, reflecting how one permutation followed by another results in a new permutation.**
3. **Example: Consider the set $X=\{1,2,3\}$. The permutation group on X, denoted as S_3, includes all possible ways to permute the elements of X. For instance, one permutation in S_3 could be $(1\ 2\ 3)$, representing the cyclic permutation that sends 1 to 2, 2 to 3, and 3 to 1.**

Normal Subgroups:

In group theory, normal subgroups are a specific type of subgroup that plays a significant role in understanding the structure and properties of groups. A subgroup N of a group G is considered normal if, for every element g in G, the conjugate gNg^{-1} is still within the subgroup N.

1. **Definition: Let G be a group and N be a subgroup of G. The subgroup N is normal in G, denoted as $N \trianglelefteq G$, if $gNg^{-1}=N$ for all g in G.**
2. **Conjugation: The concept of normal subgroups is closely tied to conjugation. If N is normal in G, the conjugation gNg^{-1} essentially means transforming the elements of N by g, and the result remains in N.**
3. **Quotient Groups: Normal subgroups are crucial in defining quotient groups. The quotient group G/N is formed by considering cosets of N and has a natural group structure.**
4. **Significance: Normal subgroups are essential for understanding the internal structure of groups. They provide a natural way to decompose a group into simpler components, facilitating the analysis of group properties and behaviors.**
5. **Examples: In the symmetric group S_n, the alternating group A_n is a normal subgroup. Additionally, the kernel of a group homomorphism is always a normal subgroup.**

MODULE 4

A graph G = (V , E) consists of V , a nonempty set of vertices (or nodes) and E, a set of edges. Each edge has either one or two vertices associated with it, called its endpoints. An edge is said to connect its endpoints.

A directed graph (or digraph) (V , E) consists of a nonempty set of vertices V and a set of directed edges (or arcs) E. Each directed edge is associated with an ordered pair of vertices. The directed edge associated with the ordered pair (u, v) is said to start at u and end at v.

Two vertices u and v in an undirected graph G are called adjacent (or neighbors) in G if u and v are endpoints of an edge e of G. Such an edge e is called incident with the vertices u and v and e is said to connect u and v.

The set of all neighbors of a vertex v of G = (V , E), denoted by N (v), is called the neighborhood of v. If A is a subset of V , we denote by N (A) the set of all vertices in G that are adjacent to at least one vertex in A.

The degree of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex. The degree of the vertex v is denoted by deg(v).

THE HANDSHAKING THEOREM Let G = (V , E) be an undirected graph with m edges. Then 2m = $\Sigma_{v \in V}$**deg(v). (Note that this applies even if multiple edges and loops are present.)**

An undirected graph has an even number of vertices of odd degree.

When (u, v) is an edge of the graph G with directed edges, u is said to be adjacent to v and v is said to be adjacent from u. The vertex u is called the initial vertex of (u, v), and v is called the terminal or end vertex of (u, v). The initial vertex and terminal vertex of a loop are the same.

In a graph with directed edges the in-degree of a vertex v, denoted by deg−(v), is the number of edges with v as their terminal vertex. The out-degree of v, denoted by deg+(v), is the number of edges with v as their initial vertex. (Note that a loop at a vertex contributes 1 to both the in-degree and the out-degree of this vertex.)

Let G = (V , E) be a graph with directed edges. Then $\Sigma_{v \in V}$ **deg−(v) =** $\Sigma_{v \in V}$ **deg+(v) = |E|.**

A simple graph G is called bipartite if its vertex set V can be partitioned into two disjoint sets V1 and V2 such that every edge in the graph connects a vertex in V1 and a vertex in V2 (so that no edge in G connects either two vertices in V1 or two vertices in V2). When this condition holds, we call the pair (V1, V2) a bipartition of the vertex set V of G.

Complete Bipartite Graphs A complete bipartite graph $K_{m,n}$ is a graph that has its vertex set partitioned into two subsets of m and n vertices, respectively with an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.

HALL'S MARRIAGE THEOREM The bipartite graph G = (V , E) with bipartition (V1, V2) has a complete matching from V1 to V2 if and only if $|N(A)| \geq |A|$ for all subsets A of V1.

A subgraph of a graph G = (V , E) is a graph H = (W, F), where $W \subseteq V$ and $F \subseteq E$. A subgraph H of G is a proper subgraph of G if H = G.

Let G = (V , E) be a simple graph. The subgraph induced by a subset W of the vertex set V is the graph (W, F), where the edge set F contains an edge in E if and only if both endpoints of this edge are in W.

The union of two simple graphs G1 = (V1, E1) and G2 = (V2, E2) is the simple graph with vertex set $V1 \cup V2$ and edge set $E1 \cup E2$. The union of G1 and G2 is denoted by $G1 \cup G2$.

Suppose that G = (V , E) is a simple graph where |V | = n. Suppose that the vertices of G are listed arbitrarily as v1, v2,..., vn. The adjacency matrix A (or AG) of G, with respect to this listing of the vertices, is the n x n zero–one matrix with 1 as its (i, j)th entry when vi and vj are adjacent, and 0 as its (i, j)th entry when they are not adjacent. In other words, if its adjacency matrix is A = [aij], then
aij = 1 if {vi, vj } is an edge of G,
= 0 otherwise.

Another common way to represent graphs is to use incidence matrices. Let G = (V , E) be an undirected graph. Suppose that v1, v2,..., vn are the vertices and e1, e2,...,em are the edges of G. Then the incidence matrix with respect to this ordering of V and E is the n × m matrix M = [mij], where
mij = 1 when edge ej is incident with vi,
= 0 otherwise.

The simple graphs G1 = (V1, E1) and G2 = (V2, E2) are isomorphic if there exists a one-to-one and onto function f from V1 to V2 with the property that a and b are adjacent in G1 if and only if f (a) and f (b) are adjacent in G2, for all a and b in V1. Such a function f is called an isomorphism. ∗ Two simple graphs that are not isomorphic are called nonisomorphic.

Let n be a nonnegative integer and G an undirected graph. A path of length n from u to v in G is a sequence of n edges e1,...,en of G for which there exists a sequence $x0 = u, x1,...,xn-1, xn = v$ of vertices such that ei has, fori = 1,...,n, the endpoints $xi-1$ and xi. When the graph is simple, we denote this path by its vertex sequence x0, x1,...,xn (because listing these vertices uniquely determines the path). The path is a circuit if it begins and ends at

the same vertex, that is, if u = v, and has length greater than zero. The path or circuit is said to pass through the vertices x1, x2,...,xn−1 or traverse the edges e1, e2,...,en. A path or circuit is simple if it does not contain the same edge more than once.

Let n be a nonnegative integer and G a directed graph. A path of length n from u to v in G is a sequence of edges e1, e2,...,en of G such that e1 is associated with (x0, x1), e2 is associated with (x1, x2), and so on, with en associated with (xn−1, xn), where x0 = u and xn = v. When there are no multiple edges in the directed graph, this path is denoted by its vertex sequence x0, x1, x2,...,xn. A path of length greater than zero that begins and ends at the same vertex is called a circuit or cycle. A path or circuit is called simple if it does not contain the same edge more than once.

An undirected graph is called connected if there is a path between every pair of distinct vertices of the graph. An undirected graph that is not connected is called disconnected. We say that we disconnect a graph when we remove vertices or edges, or both, to produce a disconnected subgraph.

There is a simple path between every pair of distinct vertices of a connected undirected graph.

A connected component of a graph G is a connected subgraph of G that is not a proper subgraph of another connected subgraph of G. That is, a connected component of a graph G is a maximal connected subgraph of G. A graph G that is not connected has two or more connected components that are disjoint and have G as their union.

EDGE CONNECTIVITY We can also measure the connectivity of a connected graph G = (V , E) in terms of the minimum number of edges that we can remove to disconnect it. If a graph has a cut edge, then we need only remove it to disconnect G. If G does not have a cut edge, we look for the smallest set of edges that can be removed to disconnect it. A set of edges E is called an edge cut of G if the subgraph G − E is disconnected.
The edge connectivity of a graph G, denoted by λ(G), is the minimum number of edges in an edge cut of G. This defines λ(G) for all connected graphs with more than one vertex because it is always possible to disconnect such a graph by removing all edges incident to one of its vertices. Note that λ(G) = 0 if G is not connected. We also specify that λ(G) = 0 if G is a graph consisting of a single vertex.

A directed graph is strongly connected if there is a path from a to b and from b to a whenever a and b are vertices in the graph.

A directed graph is weakly connected if there is a path between every two vertices in the underlying undirected graph.

Let G be a graph with adjacency matrix A with respect to the ordering v1, v2,..., vn of the vertices of the graph (with directed or undirected edges, with multiple edges and loops

allowed). The number of different paths of length r from vi to vj , where r is a positive integer, equals the (i, j)th entry of Ar.

An Euler circuit in a graph G is a simple circuit containing every edge of G. An Euler path in G is a simple path containing every edge of G.

A connected multigraph with at least two vertices has an Euler circuit if and only if each of its vertices has even degree.

A connected multigraph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree.

A simple path in a graph G that passes through every vertex exactly once is called a Hamilton path, and a simple circuit in a graph G that passes through every vertex exactly once is called a Hamilton circuit. That is, the simple path x0, x1,...,xn−1, xn in the graph G = (V , E) is a Hamilton path if V = {x0, x1,...,xn−1, xn} and xi = xj for 0 ≤ i 0) is a Hamilton circuit if x0, x1,...,xn−1, xn is a Hamilton path.

ALGORITHM 1 Dijkstra's Algorithm.
procedure Dijkstra(G: weighted connected simple graph, with all weights positive)
{G has vertices a = v0, v1,..., vn = z and lengths w(vi, vj) where w(vi, vj) = ∞ if {vi, vj } is not an edge in G}
for i := 1 to n
L(vi) := ∞
L(a) := 0
S := ∅
{the labels are now initialized so that the label of a is 0 and all other labels are ∞, and S is the empty set}
while z ∈ S
u := a vertex not in S with L(u) minimal
S := S ∪ {u}
for all vertices v not in S if L(u) + w(u, v) < L(v) then L(v) := L(u) + w(u, v)
{this adds a vertex to S with minimal label and updates the
labels of vertices not in S}
return L(z) {L(z) = length of a shortest path from a to z}

Dijkstra's algorithm finds the length of a shortest path between two vertices in a connected simple undirected weighted graph.

A graph is called planar if it can be drawn in the plane without any edges crossing (where a crossing of edges is the intersection of the lines or arcs representing them at a point

other than their common endpoint). Such a drawing is called a planar representation of the graph.

EULER'S FORMULA Let G be a connected planar simple graph with e edges and v vertices. Let r be the number of regions in a planar representation of G. Then r = e − v + 2.

If G is a connected planar simple graph with e edges and v vertices, where v ≥ 3, then e ≤ 3v − 6.

If G is a connected planar simple graph, then G has a vertex of degree not exceeding five.

If G is a connected planar simple graph, then G has a vertex of degree not exceeding five.

A graph is nonplanar if and only if it contains a subgraph homeomorphic to K3,3 or K5.

A coloring of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices are assigned the same color.

The chromatic number of a graph is the least number of colors needed for a coloring of this graph. The chromatic number of a graph G is denoted by χ (G). (Here χ is the Greek letter chi.)

THE FOUR COLOR THEOREM The chromatic number of a planar graph is no greater than four.

A tree is a connected undirected graph with no simple circuits.

An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

A rooted tree is called an m-ary tree if every internal vertex has no more than m children. The tree is called a full m-ary tree if every internal vertex has exactly m children. An m-ary tree with m = 2 is called a binary tree.

A tree with n vertices has n − 1 edges.

A full m-ary tree with i internal vertices contains n = mi + 1 vertices.

Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G.

A simple graph is connected if and only if it has a spanning tree.

ALGORITHM 1 Depth-First Search. procedure
DFS(G: connected graph with vertices v1, v2,..., vn)
T := tree consisting only of the vertex v1
visit(v1)
procedure visit(v: vertex of G)
for each vertex w adjacent to v and not yet in T
add vertex w and edge {v,w} to T
visit(w)

ALGORITHM 2 Breadth-First Search. procedure BFS
(G: connected graph with vertices v1, v2,..., vn)
T := tree consisting only of vertex v1
L := empty list
put v1 in the list L of unprocessed vertices
while L is not empty
remove the first vertex, v, from L
for each neighbor w of v
if w is not in L and not in T
then add w to the end of the list L
add w and edge {v,w} to T

A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

ALGORITHM 1 Prim's Algorithm.
procedure Prim(G: weighted connected undirected graph with n vertices)
T := a minimum-weight edge
for i := 1 to n − 2
e := an edge of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T
T := T with e added
return T {T is a minimum spanning tree of G}

ALGORITHM 2 Kruskal's Algorithm.
procedure Kruskal(G: weighted connected undirected graph with n vertices)
T := empty graph
for i := 1 to n − 1
e := any edge in G with smallest weight that does not form a simple circuit when added to T
T := T with e added
return T {T is a minimum spanning tree of G}

Questions:

1. **Compare and contrast different algorithms used to find a Minimum Spanning Tree, such as Kruskal's algorithm and Prim's algorithm. Analyze the strengths and weaknesses of each.**
2. **Analyze the criteria used for selecting edges in the process of building a Minimum Spanning Tree. How does the choice of edges impact the overall efficiency and structure of the tree?**
3. **Justify the selection of a specific algorithm for finding a Minimum Spanning Tree in a given context. What factors should be considered when choosing between different algorithms?**
4. **Break down the steps involved in Prim's algorithm for finding a Minimum Spanning Tree. How does the algorithm systematically select and add vertices and edges to form the tree?**
5. **Break down the essential properties of a tree in graph theory. How do these properties distinguish a tree from other types of graphs?**
6. **Critique the performance of tree-based search algorithms (e.g., binary search tree). How does the structure of the tree influence the time complexity of search operations, and what challenges may arise in certain scenarios?**
7. **Compare and contrast trees with other types of graphs, such as cycles and forests. Analyze the structural differences and their implications.**
8. **Break down the Four Color Theorem and its connection to planar graphs. How does the theorem relate to coloring the regions of a planar graph with a minimal number of colors?**
9. **Analyze Kuratowski's Theorem and its significance in identifying non-planar graphs. How does the theorem work, and why is it a reliable criterion for determining planarity?**
10. **Compare and contrast planar and non-planar graphs. Analyze the structural differences and discuss the implications of planarity on graph properties.**
11. **Break down the essential properties that define a planar graph. How do these properties distinguish planar graphs from non-planar ones?**
12. **Break down the concept of chromatic number in graph coloring. How does the chromatic number influence the properties and relationships within a graph?**
13. **Conduct a comprehensive evaluation of how various graph properties (e.g., planarity, connectivity) impact the complexity of graph coloring. How do these properties influence the choice and performance of coloring algorithms?**
14. **Analyze the concept of the chromatic polynomial and its role in counting the number of ways a graph can be colored with a given number of colors. How does the chromatic polynomial capture the coloring possibilities in a graph?**
15. **Assess the significance of Hall's Marriage Theorem in the context of bipartite matchings. How does the theorem guarantee the existence of perfect matchings in certain bipartite graphs, and what are its implications?**

(16) Break down the essential properties that define a matching in graph theory. How do these properties contribute to distinguishing matchings from other substructures in a graph?

(17) Break down the fundamental concept of graph isomorphism. What does it mean for two graphs to be isomorphic, and how can you visually or algebraically determine if two graphs are isomorphic?

(18) Break down the challenges and solutions associated with determining isomorphism in special types of graphs, such as trees, planar graphs, or bipartite graphs. How do the properties of these graphs impact the isomorphism testing process?

(19) Synthesize a set of criteria that determine whether a graph has an Eulerian circuit. Consider both necessary and sufficient conditions, and explain how these criteria are interconnected.

(20) Synthesize a set of necessary and sufficient conditions for a graph to have a Hamiltonian cycle. How do these conditions ensure the existence of a cycle that visits each vertex exactly once?