2023 Mathematics *Standards of Learning*
Understanding the Standards – Discrete Mathematics

The following standards outline the content of a one-year course in Discrete Mathematics. If a one-semester course is desired, the standards with a dagger (†) would apply.

Discrete Mathematics may be described as the study of mathematical properties of sets and systems that have a countable (discrete) number of elements. With the advent of modern technology, discrete (discontinuous) models have become as important as continuous models. In this course, the main focus is problem solving in a discrete setting. Techniques that are not considered in the current traditional courses of algebra, geometry, and calculus will be utilized. As students solve problems, they will analyze and determine whether a solution exists (existence problems), investigate how many solutions exist (counting problems), and focus on finding the best solution (optimization problems). Connections will be made to other disciplines. The importance of discrete mathematics has been influenced by technology.

Technology tools will be used to assist in teaching and learning. Graphing technologies facilitate visualizing, analyzing, and understanding algebraic and statistical behaviors and provide a powerful tool for solving and verifying solutions.

Logical Reasoning

DM.LR.1† The student will use reasoning to develop and apply logical arguments.

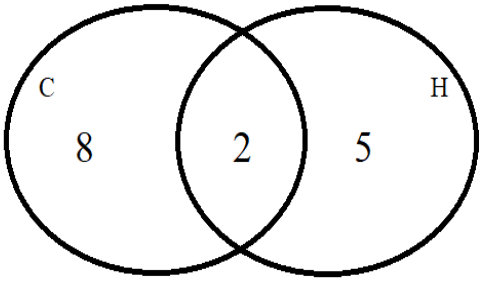
Students will demonstrate the following Knowledge and Skills:

a) Use Venn diagrams to codify and solve logic problems.
b) Express logical statements in symbolic form.
c) Represent a conditional statement as its converse, inverse, and contrapositive.
d) Describe how symbolic logic can be used to map the processes of computer applications.
e) Construct a truth table to display all possible input combinations and their outputs.
f) Identify the rules of inference and model basic logical statements including De Morgan's Law.
g) Apply logical reasoning to model contextual situations and make decisions.

DM.LR.1† The student will use reasoning to develop and apply logical arguments. *Additional Content Background and Instructional Guidance:*
• Exploration of the representation of conditional statements using Venn diagrams may assist in deepening student understanding. • Venn diagrams can be used to codify and solve logic problems or to illustrate results of probability experiments. For example, the Venn diagram below shows the results of a survey of students to determine who likes comedy movies (C) and/or horror movies (H). Eight students like comedy movies, but not horror movies; five students like horror movies, but not comedy movies; and two students like both comedy movies and horror movies.

DM.LR.1† The student will use reasoning to develop and apply logical arguments.

Additional Content Background and Instructional Guidance:



- Symbolic notation is used to represent logical arguments, including the use of →, ↔, ~, ∴, ∧, and ∨. The symbol ∴ is read as "therefore."
- Logical arguments that are valid may not be true. Truth and validity are not synonymous.
- Validity requires only logical consistency between the statements, but it does not imply true statements.
- A conditional statement is an if-then statement, where some conditions must be satisfied to reach some conclusion. A conditional statement has a hypothesis, which is the *if* part of the statement. These conditions must be met to reach the conclusion, the *then* part of the statement. The symbolic representation is $p \rightarrow q$.
- The inverse of a conditional statement retains order of hypothesis and conclusion while negating both the hypothesis and the conclusion. The symbolic representation is $\sim p \rightarrow \sim q$.
- The converse interchanges the order of the hypothesis and conclusion. The symbolic representation is $q \rightarrow p$.
- The contrapositive interchanges the order of the hypothesis and conclusion and negates the hypothesis and conclusion. The symbolic representation is $\sim q \rightarrow \sim p$.
- When a conditional ($p \rightarrow q$) and its converse ($q \rightarrow p$) are true, the statements can be written as a biconditional:
    - *p iff q*
    - *p if and only if q*
    - $p \leftrightarrow q$
- Logical arguments consist of a set of premises or hypotheses and a conclusion.
- For example, the following argument is valid, but not true. *If you are a happy person, then you like animals. If you like animals, then you like dogs. Therefore, if you are a happy person, then you like dogs.*
- A counterexample of a statement confirms the hypothesis but negates the conclusion.
- Truth tables can be used to determine if the structure of a logical argument is valid. An argument is valid when regardless of what statements are substituted for the premises, if the resulting premises are all true, then the conclusion is also true. If the argument does not have a valid form, it is said to be invalid.

DM.LR.1† The student will use reasoning to develop and apply logical arguments.

Additional Content Background and Instructional Guidance:

- To test an argument for validity, the following steps can be used –
  - Identify the premises and conclusion of the argument and represent them with symbols.
  - Make a truth table with a column for each premise and column for the conclusion.
  - Find the rows in which all of the premises are true. These are called critical rows.
  - If the conclusion is true for each critical row, then the argument is valid. If there are any critical rows in which the conclusion is false, then the argument is invalid. For example –

    Diane was on the red team or Diane was on the blue team.

    Diane was not on the red team.

    Therefore, Diane was on the blue team.

    $p \vee q$

    $\sim p$

    $\therefore q$

p	q	$p \vee q$	$\sim p$	q
T	T	T	F	T
T	F	T	F	F
F	T	T	T	T
F	F	F	T	F

    There is one critical row in which both premises are true. Because the conclusion in this row is also true, this is a valid argument.

- De Morgan's laws are used to determine the negation of a conjunction in and the negation of a disjunction. The negation of a conjunction is logically equivalent to a disjunction in which each component is negated.

$$\sim(p \vee \sim q) \equiv \sim p \vee \sim q$$

  As with other compound statements, a truth table can be used to demonstrate the logical equivalence of De Morgan's laws.
- Rules of inference may include Modus Ponens, Modus Tollens, hypothetical syllogism, disjunctive syllogism, addition, simplification, conjunction, and resolution.

DM.LR.2† The student will apply logic and proof techniques in the construction of a sound argument.

Students will demonstrate the following Knowledge and Skills:

a) Apply informal logical reasoning to contextual problems (e.g., predicting the behavior of software, solving puzzles).
b) Outline the basic structure of a proof technique (e.g., direct proof, proof by contradiction, induction).
c) Deduce the best type of proof for a given problem.
d) Use the rules of inference to construct direct proofs and proofs by contradiction.
e) Construct induction proofs involving summations and inequalities.
f) Use a truth table to prove the logical equivalence of statements.

DM.LR.2† The student will apply logic and proof techniques in the construction of a sound argument.

Additional Content Background and Instructional Guidance:

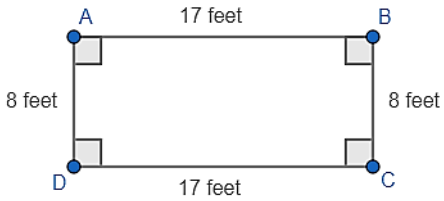
- Proof is a justification that is logically valid and based on initial assumptions, definitions, postulates, theorems, and/or properties.
- Inductive reasoning, deductive reasoning, and proof are critical in establishing general claims.
- A counterexample can be used to show an argument is false.
- Proving a statement by contradiction involves negating the conclusion and finding a logic error.
- Formal proofs utilize symbols of formal logic to determine the validity of a logical argument.
- Inductive reasoning, deductive reasoning, and proofs are critical in establishing general claims.
- Inductive reasoning is the method of drawing conclusions from a limited set of observations.
- Deductive reasoning is the method that uses logic to draw conclusions based on definitions, postulates, and theorems. Valid forms of deductive reasoning include the law of syllogism, the law of contrapositive, the law of detachment, and the identification of a counterexample.
- Proof is a justification that is logically valid and based on initial assumptions, definitions, postulates, theorems, and/or properties.
- The law of detachment states that if $p \rightarrow q$ is true and p is true, then q is true. For example, if two angles are vertical, then they are congruent. $\angle A$ and $\angle B$ are vertical, therefore $\angle A \cong \angle B$.
- The law of syllogism states that if $p \rightarrow q$ is true and $q \rightarrow r$ is true, then $p \rightarrow r$ is true. For example, if two angles are vertical, then they are congruent. If two angles are congruent,

DM.LR.2† The student will apply logic and proof techniques in the construction of a sound argument.

Additional Content Background and Instructional Guidance:

then they have the same measure. Thus, if two angles are vertical, then they have the same measure.

- The law of contrapositive states that if $p \rightarrow q$ is true and $\sim q$ is true, then $\sim p$ is true. For example, if two angles are vertical, then they are congruent. $\angle A \cong \angle B$, therefore $\angle A$ and $\angle B$ are not vertical.
- A counterexample is used to show an argument is false. For example, the argument "*All rectangles are squares,"* is proven false with the following counterexample since quadrilateral *ABDC* is a rectangle but not a square.



- Mathematical induction is a method of proof that depends on a recursive process.
- Mathematical induction allows reasoning from specific true values of the variable to general values of the variable.

DM.LR.3† The student will apply Boolean algebra to represent and analyze the function of logical gates and circuits.

Students will demonstrate the following Knowledge and Skills:

a) Explain basic properties of Boolean algebra: duality, complements, and standard forms.
b) Represent verbal statements as Boolean expressions.
c) Apply Boolean algebra to prove identities and simplify expressions.
d) Generate truth tables that encode the truth and falsity of two or more statements.
e) Explain the operation of discrete logic gates.
f) Describe the relationship between Boolean algebra and electronic circuits.
g) Analyze a combinational network using Boolean expressions.
h) Design simple combinational networks that use NAND (AND followed by NOT), NOR (OR followed by NOT), and XOR (exclusive-OR) gates.

DM.LR.3† The student will apply Boolean algebra to represent and analyze the function of logical gates and circuits.

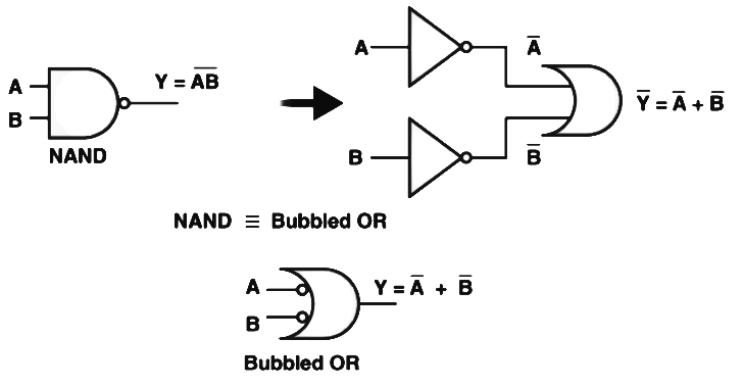
Additional Content Background and Instructional Guidance:

- De Morgan's theorems are used to solve the expressions of Boolean Algebra. This theorem explains that the complements of the products of all the terms are equal to the sums of the complements of each and every term. Likewise, the complements of the sums of all the terms are equal to the products of the complements of each and every term. Duality is equivalent to writing a negative logic of the given Boolean relation.
- Two theorems are useful in Boolean Algebra:
    - Theorem 1:

$$\overline{A.B} = \overline{A} + \overline{B}$$

NAND = Bubbled OR

The left of this theorem represents the NAND gate having inputs of A and B. The right of this theorem represents the OR gate that has inverted inputs. The OR gate is known as the Bubbled OR.



DM.LR.3† The student will apply Boolean algebra to represent and analyze the function of logical gates and circuits.

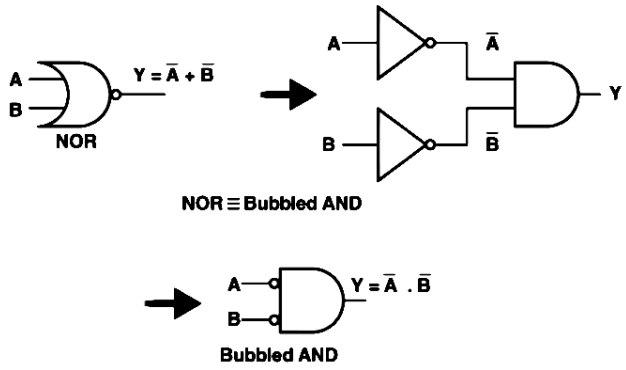
Additional Content Background and Instructional Guidance:

- Theorem 2:

$$\overline{A + B} = \overline{A}.\overline{B}$$

NOR = Bubbled AND

The left of this theorem represents the NOR gate having inputs of A and B. The right of this theorem represents the AND gate that has inverted inputs. The AND gate is known as the Bubbled AND.



- Boolean logic is a system using variables with only two values: *True* and *False*.
- **Relational Operators:**

Symbol	Operation	Example	Description
=	Equal	$x = y$	*True* if *x* is equal to *y*.
>	Greater than	$x > y$	*True* if *x* is greater than *y*.
<	Less than	$x < y$	*True* if *x* is less than *y*.
>=	Greater than or equal to	$x >= y$	*True* if *x* is greater than or equal to *y*.
<=	Less than or equal to	$x <= y$	*True* if *x* is less than or equal to *y*.
!=	Not equal to	$x != y$	*True* if *x* is not equal to *y*.

- The "if" statement is a fundamental control structure that allows branches in the flow of control.
- *If* is used to create a single selection control structure. The *If* command is followed by a condition which can be determined to be *true* or *false* and is also referred to as a Boolean test. This differs from a Boolean variable, which is available in many programming languages but not on most graphing calculators and stores *true* or *false* as its data. *If* the condition is *true*, control goes to the next command and it is executed. *If* the condition is *false*, the next command, and ONLY the next command, is skipped and control will go to the following command.

DM.LR.3† The student will apply Boolean algebra to represent and analyze the function of logical gates and circuits.

Additional Content Background and Instructional Guidance:

- *If-Then* executes a group of commands rather than a single command. The *Then* command immediately follows *If* to mark the beginning of the group and the *End* command marks the end of the group.
- The use of "*if*" statements, "*if/then/else*" statements, case statements, and Boolean logic can increase the complexity of algorithms.
- Use selection to determine which parts of an algorithm are executed based on a condition being *true* or *false*.
- Implement nested conditionals as an alternative to the use of complex Boolean expressions.
- "Chained conditional" refers to a sequence of *if/else* conditional statements.
- A logic gate is a digital circuit with a single output whose value depends upon the logical relationship between the input(s) and output.
- The logic gates have only one output for a single input or a number of inputs. To understand the operation of a logic gate, truth tables can be prepared by stating all the combinations of inputs together with the output.
- There are three types of logic gates –
    - Basic Gates (OR, AND, and NOT): A basic gate behaves the same as the corresponding logic operator.
    - Universal Gates (NAND and NOR): The universal gate can implement any Boolean expression on its own and does not require any other gate for implementation.
    - Derived Gates (XOR and XNOR [exclusive-OR]): The derived gate is made for specific applications such as half adders, full adders, and subtractors.

DM.LR.4 The student will use mathematical induction to prove formulas and mathematical statements.

Students will demonstrate the following Knowledge and Skills:

a) Compare and contrast inductive and deductive reasoning.
b) Explain the relationship between weak and strong induction.
c) Construct induction proofs involving a divisibility argument.
d) Prove the Binomial Theorem through mathematical induction.

DM.LR.4 The student will use mathematical induction to prove formulas and mathematical statements.

Additional Content Background and Instructional Guidance:

- Inductive reasoning is the method of drawing conclusions from a limited set of observations. Patterns are established and conclusions are drawn based on those patterns. Inductive reasoning moves from the specific to the general.
- Deductive reasoning is the method that uses logic to draw conclusions based on definitions, postulates, and theorems. This type of reasoning is used most often in direct proofs and moves from the general to the specific.
- Mathematical induction proves each of an infinite sequence of logical statements is true.
- The inductive hypothesis of strong induction assumes all cases preceding k are true, whereas weak induction only assumes case k is true.
- To prove divisibility by induction –
  - Show that the base case (where $n = 1$) is divisible by the given value.
  - Assume that the case of $n = k$ is divisible by the given value.
  - Use this assumption to prove that the case where $n = k + 1$ is divisible by the given value.
  - Conclude that by induction, the divisibility is true for all values of n.
- Proof of the Binomial Theorem through mathematical induction can require the use of Pascal's identity in the form –

$$\binom{n}{r-1} + \binom{n}{r} = \binom{n+1}{r}, \qquad \text{for} \quad 0 < r \leq n$$

To prove that –

$$(a+b)^n = a^n + \binom{n}{1}a^{n-1}b + \binom{n}{2}a^{n-2}b^2 + \cdots + \binom{n}{r}a^{n-r}b^r + \cdots + \binom{n}{n-1}ab^{n-1} + > b^n.$$

The result is true for $n = 1$ and $n = 2$. Let k be a positive integer with $k \geq 2$ for which the statement is true. So –

$$(a+b)^k = a^k + \binom{k}{1}a^{k-1}b + \binom{k}{2}a^{k-2}b^2 + \cdots + \binom{k}{r}a^{k-r}b^r + \cdots + \binom{k}{k-1}ab^{k-1} + b^k.$$

DM.LR.4 The student will use mathematical induction to prove formulas and mathematical statements.

Additional Content Background and Instructional Guidance:

Consider the expansion –

$$
\begin{aligned}
&(a+b)^{k+1}\\
&=(a+b)(a+b)^k\\
&=(a+b)\left(a^k+\binom{k}{1}a^{k-1}b+\binom{k}{2}a^{k-2}b^2+\cdots+\binom{k}{r}a^{k-r}b^r+\cdots+\binom{k}{k-1}ab^{k-1}+b^k\right)\\
&=a^{k+1}+\left[1+\binom{k}{1}\right]a^kb+\left[\binom{k}{1}+\binom{k}{2}\right]a^{k-1}b^2+\cdots\\
&\cdots+\left[\binom{k}{r-1}+\binom{k}{r}\right]a^{k-r+1}b^r+\cdots+\left[\binom{k}{k-1}+1\right]ab^k+b^{k+1}.
\end{aligned}
$$

Using Pascal's identity, it will follow that –

$$(a+b)^{k+1}=a^{k+1}+\binom{k+1}{1}a^kb+\cdots+\binom{k+1}{r}a^{k-r+1}b^r+\cdots+\binom{k+1}{k}ab^k+b^{k+1}.$$

Therefore, the result is true for $k + 1$. By induction, the result is true for all integers n.

Set and Number Theory

DM.SNT.1† The student will identify and use the properties of sets and set operations.

Students will demonstrate the following Knowledge and Skills:

a) Compare and contrast sets, relations, and functions.
b) Express relationships between sets using Venn diagrams.
c) Describe a set using set-builder notation.
d) Construct new sets using the set operations intersection, union, difference, and complement.
e) Identify the laws of set theory (e.g., associative, commutative, distributive, De Morgan's Law).
f) Use the principle of inclusion and exclusion to determine the size of a set.
g) Use the properties of set operations to prove set equality.

DM.SNT.1† The student will identify and use the properties of sets and set operations.

Additional Content Background and Instructional Guidance:

- A set is a collection of items, names, objects, or numbers. A capital letter is typically used to denote a set, with items, names, objects, or numbers enclosed by { }.
- Sets can be represented visually using Venn diagrams. Each individual set is represented by a circle and the universal set is represented by a rectangle that encloses all of the sets being considered.
- n(A) denotes the number of elements in a set A.
- Set builder notation is another way of defining sets. This form is divided into three parts. In the first part a variable is identified. In the second part the symbol "|" which means "such that" is used to introduce the conditions that apply to the variable. In the third part, the conditions of the variable follow (e.g., $\{x \mid x \text{ is odd}\}$).
- A universal set U is the set of all elements under consideration for a particular situation. The universal set could be the set of all real numbers, the set of all positive integers, etc. For a universal set U and sets A and B with $A \subseteq U$ and $B \subseteq U$, the following are true –
    - The union of A and B is the set of all elements x in U such that $x \in A$ or $x \in B$. The notation is $A \cup B$.
    - The intersection of A and B is the set of all elements x in U such that $x \in A$ and $x \in B$. The notation is $A \cap B$.
    - The complement of A is the set of all elements x in U such that $x \notin A$. The notation is A^C
- For subsets A, B, and C of a universal set U, the following set identities are true –
    - Associative Property

      $(A \cap B) \cap C = A \cap (B \cap C)$

      $(A \cup B) \cup C = A \cup (B \cup C)$

DM.SNT.1† The student will identify and use the properties of sets and set operations.

Additional Content Background and Instructional Guidance:

- Commutative Property

  $A \cap B = B \cap A$

  $A \cup B = B \cup A$

- Distributive Property

  $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

  $A \cup (B \cap C) = (A \cup B) \cup (A \cup C)$

- Complement of a Complement Property: $(A^C)^C$

- De Morgan's Laws for Sets

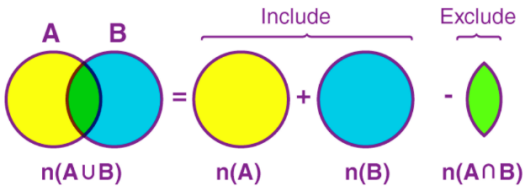
  $(A \cap B)^C = (A)^C \cup (B)^C$

  $(A \cup B)^C = (A)^C \cap (B)^C$

- When there are no elements in a set, the set is referred to as the empty set or null set. The empty set is denoted by { } or ∅.
- The cardinality of a set is defined as the number of elements in a mathematical set that can be finite or infinite.
- The Principle of Inclusion and Exclusion is an approach which derives the method of finding the number of elements in the union of two finite sets. This is used to solve combinations and probability problems when it is necessary to find a counting method, which makes sure that an object is not counted twice. Consider two finite sets, *A* and *B*:

$$n(A \cup B) = n(A) + n(B) - n(A \cap B)$$

n(*A*) denotes the cardinality of set *A*, and *n*(*B*) denotes the cardinality of set *B* and *n*(*A* ∩ *B*) denotes the cardinality of (*A* ∩ *B*). *A* and *B* are included and excluded from their common elements. A visual representation is as follows –



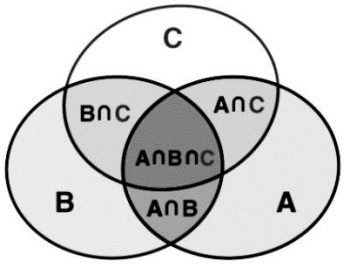
If there are three sets *A*, *B*, and *C*, then:

$$n(A \cup B \cup C) = n(A) + n(B) + n(C) - n(A \cap B) - n(A \cap C) - n(B \cap C) + n(A \cap B \cap C)$$

DM.SNT.1† The student will identify and use the properties of sets and set operations.

Additional Content Background and Instructional Guidance:

A visual representation is as follows –



DM.SNT.2† The student will apply the formulas of combinatorics.

Students will demonstrate the following Knowledge and Skills:

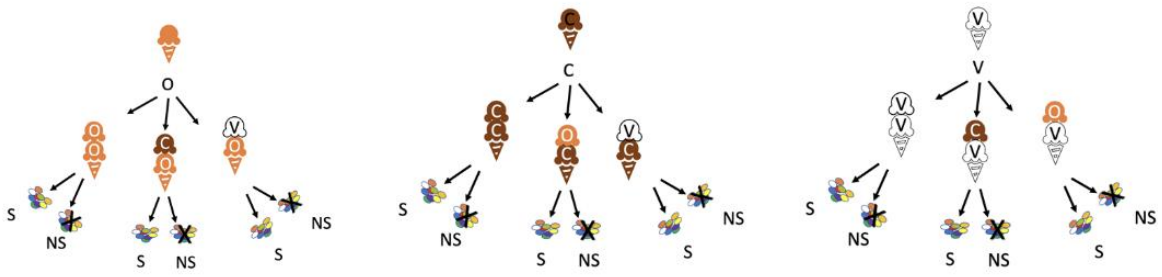
a) Create a tree diagram to represent relationships between independent events.
b) Use the Fundamental (Basic) Counting Principle to determine the number of possible outcomes of an event.
c) Determine the number of combinations possible when subsets of *r* elements are selected from a set of *n* elements without regard to order.
d) Determine the number of permutations possible when *r* objects selected from *n* objects are ordered.
e) Use the pigeonhole principle to solve packing problems to facilitate proofs.
f) Construct a proof by induction using principles of combinatorics.

DM.SNT.2† The student will apply the formulas of combinatorics.

Additional Content Background and Instructional Guidance:

- Patterns may be generalized when determining the sample space. The Fundamental (Basic) Counting Principle is a computational procedure to determine the total number of possible outcomes when there are multiple choices or events. It is the product of the number of outcomes for each choice or event that can be chosen individually. For example, the possible final outcomes or outfits of four shirts (green, yellow, blue, red), two pairs of shorts (tan or black), and three types of shoes (sneakers, sandals, flip flops) is $4 \times 2 \times 3 = 24$ outfits.
- Exploring the use of tree diagrams for modeling combinations helps students develop the Fundamental Counting Principle. For the ice cream combinations problem below, applying the Fundamental Counting Principle shows there are $3 \times 3 \times 2 = 18$ outcomes.

You order a double scoop of ice cream.
Choices
Scoop 1: orange, chocolate, or vanilla
Scoop 2: orange, chocolate, or vanilla
Sprinkles
No Sprinkles
Key:
O → orange
C → chocolate
V → vanilla
S → sprinkles
NS → no sprinkles



DM.SNT.2† The student will apply the formulas of combinatorics.

Additional Content Background and Instructional Guidance:

- Combinatorics is the branch of mathematics that addresses the number of ways objects can be arranged or combined.
- If n and r are positive integers and $n \geq r$,

$$n \text{ P } r = \frac{n!}{(n-r)!} \quad \text{and} \quad n \text{ C } r = \frac{n!}{r!\,(n-r)!} \ .$$

- The pigeonhole principle states that if $n + 1$ objects are put into n boxes, then at least one box contains two or more objects.

DM.SNT.3 The student will use Pascal's Triangle to analyze numerical patterns and relationships.

Students will demonstrate the following Knowledge and Skills:

a) Construct Pascal's Triangle.
b) Expand binomials having positive integral exponents, using the Binomial Theorem and Pascal's Triangle.
c) Compare the binomial coefficient to the calculation of combinations.
d) Identify the Fibonacci numbers within Pascal's Triangle.

DM.SNT.3 The student will use Pascal's Triangle to analyze numerical patterns and relationships.

Additional Content Background and Instructional Guidance:

- Pascal's Triangle is a triangular array of binomial coefficients. The numbers in Pascal's Triangle are placed in such a way that each number is the sum of two numbers just above the number.
- Generally, Pascal's Triangle can be used to find the coefficients of binomial expansion.
- The Binomial Theorem provides a formula for calculating the product $(a + b)^n$ for any positive integer *n.*
- A visual representation of how Pascal's Triangle is used for Binomial Expansion is as follows –

Exponent	Pascal's Triangle	Binomial Expansion
0	1	$(a+b)^0 = 1$
1	1 1	$(a+b)^1 = 1a + 1b$
2	1 2 1	$(a+b)^2 = 1a^2 + 2ab + 1b^2$
3	1 3 3 1	$(a+b)^3 = 1a^3 + 3a^2b + 3ab^2 + 1b^3$

- Fibonacci numbers can be derived by summing the elements on the rising diagonal lines in Pascal's Triangle.

Graph Theory

DM.GT.1† The student will represent problems using vertex-edge graphs. The concepts of degree, connectedness, paths, planarity, and directed graphs will be analyzed.

Students will demonstrate the following Knowledge and Skills:

a) Illustrate the basic terminology of graph theory (e.g., vertex, edge, graph, degree of a vertex).
b) Use graphs to map situations in which the vertices represent objects, and edges represent a particular relationship between objects.
c) Identify and describe degree and connectedness.
d) Determine whether a graph is planar or nonplanar.
e) Analyze the relationship between faces, edges, and vertices using Euler's formula ($F = E - V + 2$).
f) Use directed graphs (digraphs) to represent situations with restrictions in traversal possibilities.
g) Determine when graphs are trees.

DM.GT.1† The student will represent problems using vertex-edge graphs. The concepts of degree, connectedness, paths, planarity, and directed graphs will be analyzed.

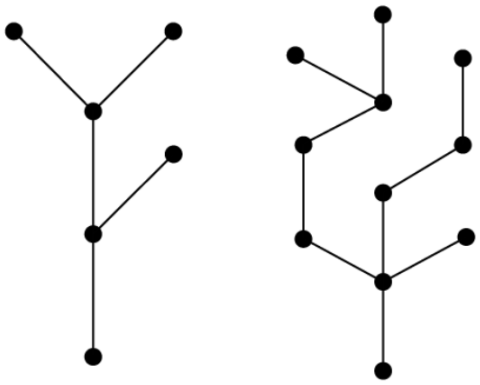
Additional Content Background and Instructional Guidance:

- A tournament is a digraph that results from giving directions to the edges of a complete graph.
- Euler's formula can be written as $F = E - V + 2$, where F is equal to the number of faces, V is equal to the number of vertices, and E is equal to the number of edges. Euler's formula states that for many solid shapes the number of faces plus the number of vertices minus the number of vertices is equal to 2.
- A graph is defined by a finite set of points, called vertices, and a finite set of segments, called edges, that connect the vertices. Each edge is associated with either one or two vertices.
  - An edge that is associated with just one vertex is called a loop.
  - Two distinct edges that are associated with the same set of vertices are parallel edges.
  - A simple graph is a graph that does not have any loops or parallel edges.
  - Adjacent vertices are connected by an edge.
  - In a connected graph, every pair of vertices is adjacent.
- Graphs can be used to solve problems including food chains and number of paths.
- When a connected graph can be drawn without any edges crossing, it is planar. When a connected graph cannot be drawn without edges crossing, it is nonplanar.
- Trees
  - Two vertices of a graph, v_1 and v_2, are connected *iff* there is a walk from to v_1 to v_2.
  - A graph is connected *iff* every pair of vertices in the graph is connected.
  - A trivial circuit is a circuit that consists of only a single vertex.

DM.GT.1† The student will represent problems using vertex-edge graphs. The concepts of degree, connectedness, paths, planarity, and directed graphs will be analyzed.

Additional Content Background and Instructional Guidance:

- A graph is a tree *iff* it is connected and has only trivial circuits. The graphs shown below are trees, as every pair of vertices is connected and the only circuits are trivial circuits.



DM.GT.2† The student will solve problems through analysis and application of circuits, cycles, Euler paths, Euler circuits, Hamilton paths, and Hamilton circuits. Optimal solutions will be determined using existing algorithms and student-created algorithms.

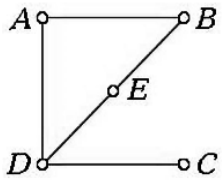
Students will demonstrate the following Knowledge and Skills:

a) Determine whether a graph has an Euler circuit or path, and determine the circuit or path, if it exists.
b) Determine whether a graph has a Hamilton circuit or path, and determine the circuit or path, if it exists.
c) Count the number of Hamilton circuits for a complete graph with n vertices.
d) Use an Euler circuit algorithm to solve optimization problems.

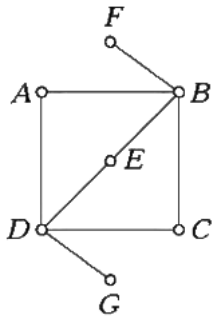
DM.GT.2† The student will solve problems through analysis and application of circuits, cycles, Euler paths, Euler circuits, Hamilton paths, and Hamilton circuits. Optimal solutions will be determined using existing algorithms and student-created algorithms.

Additional Content Background and Instructional Guidance:

- If *G* is a connected graph and all its valences are even, then *G* has an Euler circuit.
- There are $\frac{(n-1)!}{2}$ Hamilton circuits.
- Pairs of routes (circuits) correspond to the same Hamilton circuit because one route can be obtained from the other by traversing the vertices in reverse order.
- The figure below shows a graph that has no Euler circuits, but does have Euler paths (for example *C*, *D*, *E*, *B*, *A*, *D*). The figure has no Hamilton circuits because at some point one travels back to C and there are no other routes; however, the figure has Hamilton paths (*A*, *B*, *E*, *D*, *C*). The illustration demonstrates that there can be a Hamilton path, but no Hamilton circuit.



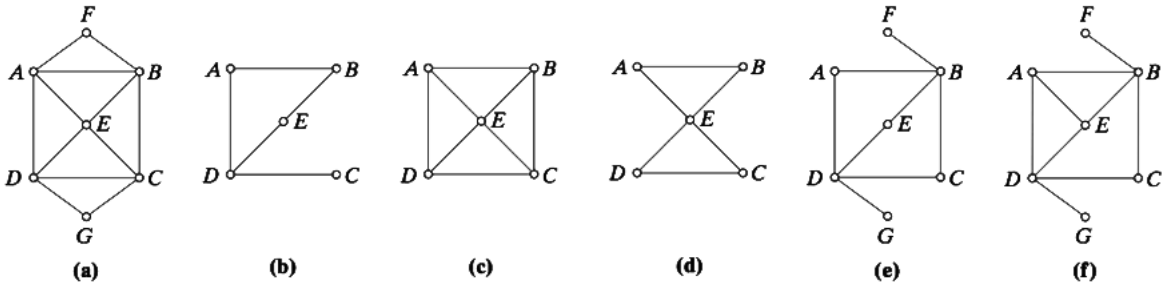
- The figure below shows a graph that has no Euler circuits but has Euler paths (*F* and *G* are the two odd vertices). The figure has neither Hamilton circuits nor Hamilton paths.



DM.GT.2† The student will solve problems through analysis and application of circuits, cycles, Euler paths, Euler circuits, Hamilton paths, and Hamilton circuits. Optimal solutions will be determined using existing algorithms and student-created algorithms.

Additional Content Background and Instructional Guidance:

- The table below summarizes whether Euler and/or Hamilton Circuits and/or Paths exist based on the given figures.



	Euler circuit	Euler path	Hamilton circuit	Hamilton path
Figure (a)	Yes	No	Yes	Yes
Figure (b)	No	Yes	No	Yes
Figure (c)	No	No	Yes	Yes
Figure (d)	Yes	No	No	Yes
Figure (e)	No	Yes	No	No
Figure (f)	No	No	No	No

- A multigraph is connected if there is a path between every pair of vertices.

DM.GT.3† The student will apply graphs to conflict-resolution problems, such as graph coloring, scheduling, matching, and optimization.

Students will demonstrate the following Knowledge and Skills:

a) Model projects consisting of several subtasks, using a graph.
b) Use graphs to resolve conflicts that arise in scheduling.
c) Use graph coloring to determine the chromatic number of a graph.

DM.GT.3† The student will apply graphs to conflict-resolution problems, such as graph coloring, scheduling, matching, and optimization.

Additional Content Background and Instructional Guidance:

- Every planar graph has a chromatic number that is less than or equal to four (the four-color-map theorem).
- A graph can be colored with two colors if and only if it contains no cycle of odd length.
- The chromatic number of a graph cannot exceed one more than the maximum number of degrees of the vertices of the graph.

DM.GT.4 The student will recognize and apply algorithms to solve configuration, conflict-resolution, and sorting problems.

Students will demonstrate the following Knowledge and Skills:

a) Recognize algorithms such as nearest neighbor, brute force, and cheapest link as they apply to graphs.
b) Use Kruskal's algorithm to determine the shortest spanning tree of a connected graph.
c) Use Prim's algorithm to determine the shortest spanning tree of a connected graph.
d) Use Dijkstra's algorithm to determine the shortest spanning tree of a connected graph.

DM.GT.4 The student will recognize and apply algorithms to solve configuration, conflict-resolution, and sorting problems.

Additional Content Background and Instructional Guidance:

- A spanning tree of a connected graph *G* is a tree that is a subgraph of *G* and contains every vertex of *G*.
- Brute force is used in algorithmic problem-solving that checks every possible solution until the correct one is found. Brute force algorithms function by searching each element sequentially until the desired result is found or all options are exhausted.
- The cheapest link algorithm is different from the nearest neighbor algorithm in that the nearest neighbor finds the shortest path between two nodes, while the cheapest link algorithm finds the least expensive path between all the nodes in the graph.
- A spanning tree is a subset of a graph that includes all the graph's vertices and some of the edges of the original graph, intending to have no cycles.
    - A spanning tree is not necessarily unique because it is possible for there to be multiple spanning trees for a given graph. However, a given graph will always have at least one spanning tree.
    - The edges in a spanning tree are called branch edges, while the edges not in the spanning tree are called cycle edges.
    - This type of graph helps find the minimum number of edges required to connect all vertices in a graph.
    - This type of graph creates minimally secured networks with redundant paths.
- A minimum spanning tree is a subset of the edges of a connected, edge-weighted graph that connects all the vertices together without any cycles and with the minimum possible total edge weight. It is a way of finding the most economical way to connect a set of vertices.
    - A minimum spanning tree has precisely $n - 1$ edges, where n is the number of vertices in the graph.
- Kruskal's algorithm is used to discover the shortest path between two points in a connected weighted graph. This algorithm converts a given graph into the forest, considering each node as a separate tree.

DM.GT.4 The student will recognize and apply algorithms to solve configuration, conflict-resolution, and sorting problems.
Additional Content Background and Instructional Guidance:
● Prim's algorithm starts with a single node and moves through several adjacent nodes to explore all the connected edges along the way. ● Dijkstra's algorithm finds the shortest path between a given node (source node) and all other nodes in a graph. This algorithm uses the weights of the edges to find the path that minimizes the total distance (weight) between the source node and all other nodes.

DM.GT.5 The student will use algorithms to schedule tasks to determine a minimum project time.

Students will demonstrate the following Knowledge and Skills:

a) Specify in a digraph the order in which tests are to be performed.
b) Identify the critical path to determine the earliest completion time (minimum project time).
c) Use the list-processing algorithm to determine an optimal schedule.
d) Create and test scheduling algorithms.

DM.GT.5 The student will use algorithms to schedule tasks to determine a minimum project time. *Additional Content Background and Instructional Guidance:*
● An order requirement diagraph is a directional graph that shows a specific ordering of tasks needed to complete a job. ● The critical path is the longest path in an order requirement diagraph representing the shortest completion time. ● Critical path scheduling sometimes yields optimal solutions. ● The list processing algorithm works by coordinating the scheduling of the tasks, taking account of a priority list, and then coordinating this list with the demands imposed by the task analysis digraph. ● To create an optimal schedule, begin by putting the first task on a longest path (breaking ties with the task of lowest number) at the start of the list. Then, remove this task and edges that come out of it from the task analysis digraph and repeat the process by finding a new task to add to the list that is at the start of a longest path. ● Four factors impact the final schedule – time of the tasks, number of processors, order-requirement diagram, and the order of the tasks on the priority list.

Computational Methods

DM.CM.1† The student will describe and apply sorting and searching algorithms used in processing and communicating information.

Students will demonstrate the following Knowledge and Skills:

a) Select and apply a sorting algorithm, such as a bubble sort, merge sort, or network sort.
b) Describe the advantages and disadvantages of various sorting algorithms.
c) Analyze the knapsack and bin-packing problems.
d) Select and apply search algorithms to analyze problems.
e) Determine the average, best-case, and worst-case reasoning for different searches.

DM.CM.1† The student will describe and apply sorting and searching algorithms used in processing and communicating information.
Additional Content Background and Instructional Guidance:
● A routine is a sequence of code intended for the execution of a program. ● Search routines are sequences of code that use loops to find a given target. ● Differentiating between search routines that include linear and binary search. ● Linear search routines are sequential whereas binary search routines are logarithmic. ● Sort algorithms use elements of an array or list as an input and arrange the elements into a meaningful order so that they can be analyzed effectively. ● A bubble sort orders elements of an array by comparing adjacent elements. ● A merge sort combines two sorted lists into a single sorted list. A network sort iteratively exchanges inputs if they are out of order. ● Coding algorithms must account for the number of possible codes within the constraints of the coding system. ● Bin-packing and knapsack packing are optimization techniques. ● A bin-packing problem determines the minimum number of containers of fixed volume (bins) required to hold a set of objects. ● A knapsack problem determines the most valuable set of objects that fit into a container (knapsack) of fixed volume.

DM.CM.2† The student will use recursive processes.

Students will demonstrate the following Knowledge and Skills:

a) Compare and contrast iterative and recursive processes.
b) Use recursive processes to model growth and decay.
c) Use recursive processes to create fractals.
d) Use recursive processes to generate the Fibonacci sequence.
e) Determine if a recursive solution is more efficient than an iterative solution.

DM.CM.2† The student will use recursive processes.

Additional Content Background and Instructional Guidance:

- Recursion is a process that creates new objects from existing objects that were created by the same process.
- Recursion and iteration both repeatedly execute the set of instructions. Recursion happens when a statement in a function calls itself repeatedly. The iteration occurs when a loop repeatedly executes until the controlling condition becomes false.
- The difference between recursion and iteration is that recursion is a process always applied to a function and iteration is applied to the set of instructions which are to be executed repeatedly.
- To understand how recursion operates in generating the Fibonacci sequence, consider $F(n)$, the nth number in the series. $F(n)$ is the sum of the preceding two numbers $F(n-1)$ and $F(n-2)$. If n is 4, $F(4) = F(3) + F(2)$. $F(3)$ and $F(2)$ can be broken down into smaller problems, using the same logic until the the base cases, $F(0)$ and $F(1)$ are reached.
- Iteration can be faster and more efficient than recursion as it is easier to optimize iterative codes because of polynomial time complexity. These are used to iterate over the elements present in data structures like an array, a set, or a map.
- A fractal is a figure whose dimension is not a whole number.
- Fractals are self-similar.

DM.CM.3 The student will identify and apply cryptographic methods.

Students will demonstrate the following Knowledge and Skills:

a) Compare and contrast ciphers and codes.
b) Describe the evolution of cipher systems.
c) Identify the Fundamental Theorem of Arithmetic.
d) Describe how the complexity of prime factorization is used in cryptography.
e) Describe modular arithmetic in context (e.g., clocks, days of the week, measures of time).
f) Analyze the relationship between divisibility and modulus.
g) Determine congruence within modular arithmetic.
h) Perform operations within modular arithmetic.
i) Apply modular arithmetic to problems in context (e.g., cryptography, International Standard Book Number (ISBN), International Bank Account Number (IBAN)).

DM.CM.3 The student will identify and apply cryptographic methods.

Additional Content Background and Instructional Guidance:

- Cryptography is the practice of coding information to ensure only the person that a message was written for can read and process the information.
- A cipher involves the use of mathematical algorithms to scramble information into an unreadable format.
- A code is a method of representing information using symbols or signals to convey its meaning.
- Modular arithmetic is a system of arithmetic for integers, which considers the remainder. In modular arithmetic, numbers "wrap around" upon reaching a given fixed quantity (this given quantity is known as the modulus) to leave a remainder.
- An intuitive usage of modular arithmetic is with a 12-hour clock. If it is 10:00 now, then in 5 hours the clock will show 3:00 instead of 15:00. 3 is the remainder of 15 with a modulus of 12.
- The Fundamental Theorem of Arithmetic states that every positive integer (except the number 1) can be represented in exactly one way apart from rearrangement as a product of one or more primes.

DM.CM.4 The student will analyze the limitations of algorithms and their contextual relationships in computing.

Students will demonstrate the following Knowledge and Skills:

a) Describe maximum complexity of an algorithm using Big O notation.
b) Describe Turing machines and how they are used to test the limits of computation.
c) Describe the halting problem and explain how it characterizes the fundamental limitations of computation and undecidability.
d) Explain the P versus NP problem and defend a justification for equality, inequality, or undecidability.
e) Analyze how the equivalence of P- and NP-class problems might impact society.

DM.CM.4 The student will analyze the limitations of algorithms and their contextual relationships in computing.

Additional Content Background and Instructional Guidance:

- Big O notation is a mathematical notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity.
- Turing machines are a fundamental concept in the theory of computation and play an important role in the field of computer science.
- Turing machines have finite memory; thus, limiting the size of the problem it can solve.
- Turing machines have finite computational power; thus, these machines can only perform simple (basic) operations like moving the tape right or left or changing the state of a cell.
- The halting problem is the problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running, or continue to run forever.
- The P versus NP problem is a major unsolved problem in theoretical computer science. In informal terms, it asks whether every problem whose solution can be quickly verified can also be quickly solved.