

GAC024 Discrete Mathematics

GAC Mathematics

What this module is, and how it is marked

A repeated set member is counted once, a binary carry may exceed a fixed width, and the fewest-edge route may not have the smallest weight. Discrete mathematics makes those rules explicit.

GAC024 covers sets, counting systems, binary logic, algorithms and networks. Your centre's current brief determines assessment tasks, tools, weights and deadlines. These original practice sheets do not establish official marking rules or a university credit decision.

State the universe, representation width, allowed inputs or graph assumptions before solving. Show enough working for another reader to reproduce the result and distinguish a mathematical model from its real implementation.

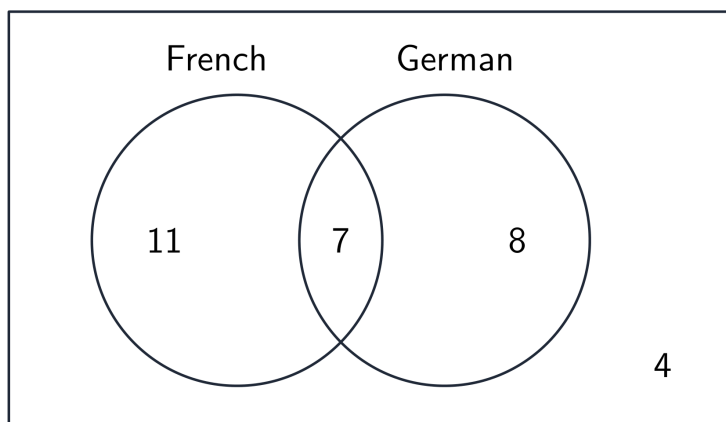
Sets, relations and functions

- A **set** is a collection of distinct objects. Order and repetition do not matter.
- **Union** $A \cup B$ is everything in either; **intersection** $A \cap B$ is what is in both; the **set complement** is everything in the stated universe but outside the set.
- A **subset** has all its elements inside another set.
- A **relation** pairs elements of two sets. A **function** is a relation where each input in its stated domain has exactly one output. Different inputs may share an output; an inverse relation is a function only when outputs uniquely identify their inputs.
- A **Venn diagram** turns a set problem into a picture, and can show the disjoint regions and their counts. Check that those regions add to the supplied universe total.

The **inclusion-exclusion principle** subtracts the twice-counted overlap once: $|A \cup B| = |A| + |B| - |A \cap B|$.

| **Worked example.** subtract an overlap only once

universe: 30 learners



Known: 30 learners, 18 study French, 15 German, and 7 both. The overlap is included in both subject totals.

$$|F \cup G| = |F| + |G| - |F \cap G| = 18 + 15 - 7 = 26$$

$$N_{\text{neither}} = |U| - |F \cup G| = 30 - 26 = 4$$

French-only is $18 - 7 = 11$ and German-only $15 - 7 = 8$. The four disjoint regions sum to 30.

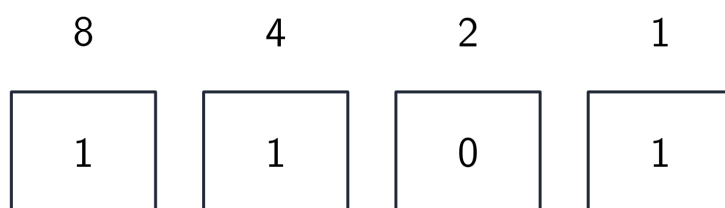
Practice sheet 4.1 includes progressively harder problems and independently checked solutions.

Counting systems

- A positional **number base** b uses digits from zero to b minus one and place weights b^i . **Decimal** uses ten, **binary** two, **hexadecimal** sixteen.
- Every digit's value is its **place value** : in binary the places are 1, 2, 4, 8, 16 and so on.
- Hexadecimal is shorthand for binary: one hex digit is exactly four bits, so conversion can group a stated-width binary pattern into four-bit blocks. Leading zeros preserve width while leaving the unsigned value unchanged.

For n unsigned bits, values run from zero to $2^n - 1$. Distinguish an unrestricted sum from a stored fixed-width result; a wraparound rule, if explicitly given, keeps the low n bits.

| **Worked example.** place weights determine the decimal value



place weights above binary digits

Known numeral 1101_2 . Use weights from right to left: 1, 2, 4 and 8.

$$V = \sum d_i 2^i$$

$$V = 1(8) + 1(4) + 0(2) + 1(1) = 13$$

The same value is D in hexadecimal. Leading zeros would not change this nonnegative value but can record an intended width.

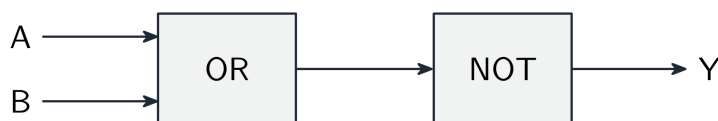
Practice sheet **4.2** includes progressively harder problems and independently checked solutions.

Binary applications

- **Binary arithmetic** adds like decimal, carrying at 2 instead of at 10.
- A **bit** is one binary digit; a **byte** is eight.
- **Boolean algebra** works on true and false with AND, OR and NOT.
- A **truth table** lists every Boolean input combination and output. Matching every row proves equivalence for the same finite Boolean inputs; it does not prove physical circuit timing or real-system security.
- **Logic gates** implement stated operations, and a **logic circuit** connects them. Trace the abstract logic according to its connections and input conventions.

Use inclusive OR and explicit brackets. De Morgan gives $\neg(A \wedge B) = (\neg A) \vee (\neg B)$. Bitwise NOT inverts only the stated width, not an unspecified infinite representation.

| **Worked example.** an OR output is inverted by NOT



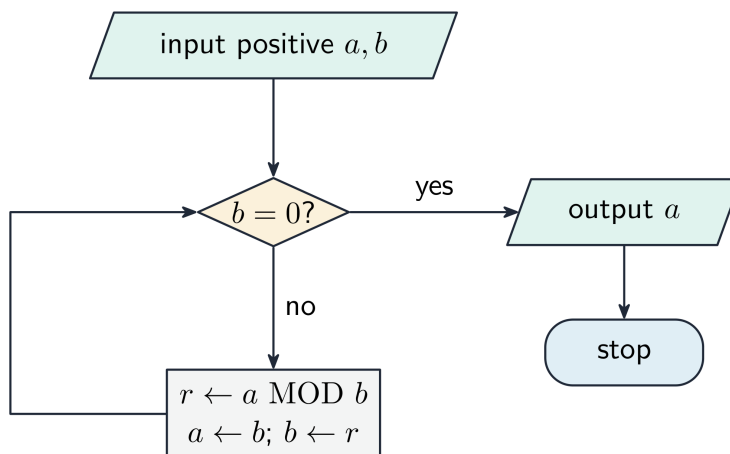
Known: $Y = \neg(A \vee B)$. Inclusive OR is false only when both inputs are false; NOT reverses that result. In row order $(A, B) = (0, 0), (0, 1), (1, 0), (1, 1)$, the output column is 1, 0, 0, 0. De Morgan gives equivalent expression $(\neg A) \wedge (\neg B)$.

Practice sheet 4.3 includes progressively harder problems and independently checked solutions.

Algorithms

- An **algorithm** describes unambiguous steps for a task. A procedure solving the stated finite task must terminate and give the required result for its allowed inputs.
- A **flowchart** draws it: a decision is a diamond, a process a rectangle.
- **Pseudocode** represents its steps without requiring a particular implementation language. State assignment, loop bounds and index conventions before tracing.
- **Tracing** an algorithm – a table with one column per variable and one row per step – records its actual updates. A trace checks the chosen input; a claim for all allowed inputs also needs a correctness argument.
- **Efficiency** matters: a linear search can stop early but may inspect all n items. Binary search repeatedly discards half of an ordered search range; its logarithmic comparison count requires the sorted-data and bound conventions.

Worked example. repeat a remainder step until the second number is zero



Known: start with positive integers a equal to 10 and b equal to 6. While b is nonzero, compute r as $a \text{ MOD } b$, then set a to b and b to r . Pairs after complete iterations are $(6, 4), (4, 2), (2, 0)$, giving output 2. Temporary r preserves the remainder before a and b change. Each nonzero remainder is smaller than the previous positive b , supporting termination.

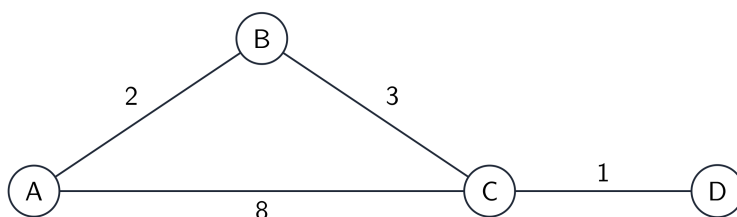
Practice sheet 4.4 includes progressively harder problems and independently checked

solutions.

Graphs and networks

- A **graph** is a set of **vertices** joined by **edges**. It models anything with connections: roads, friendships, dependencies.
- For a simple undirected graph with no loops or repeated edges, the **degree** counts incident edges. Every edge contributes two to the total degree sum.
- A **tree** is a connected graph with no cycles, and a finite tree with n vertices has n minus 1 edges. Some hierarchical models use trees, but actual systems can also contain cross-links or cycles.
- A **shortest path** problem asks for the cheapest route between two vertices, by total weight under the stated constraints, rather than by the number of edges alone. A minimum spanning tree instead connects every vertex without cycles and minimises total included edge weight.

Worked example. compare total route weight, not the number of edges



Known edge weights are $AB = 2$, $BC = 3$, $AC = 8$ and $CD = 1$. The path A-C-D has weight 9, while A-B-C-D has weight 6. Therefore the three-edge path is shorter by weight despite having more edges. The minimum spanning tree for this small network uses AB, BC and CD with total 6; the agreement of totals here does not make the tasks identical.

Practice sheet 4.5 includes progressively harder problems and independently checked solutions.