QuickNotes

FREE STUDY NOTES · PDF

UGC NET Paper 2

Computer Science and Applications (087)
Complete revision notes — all 10 units

10 units | Tables & formulas | Key revision points

01 Discrete Structures and Optimization
02 Computer System Architecture
03 Programming Languages and Computer Graphics
04 Database Management Systems
05 System Software and Operating System
06 Software Engineering
07 Data Structures and Algorithms
08 Theory of Computation and Compilers
09 Data Communication and Computer Networks
10 Artificial Intelligence

getquicknotes.com

Edition: November 2026

CONTENTS

What's inside

Ten units of focused revision notes. Each unit ends with a box of key revision points.

01 **Discrete Structures and Optimization** — Sets, relations, functions, counting, logic, graph theory, Boolean algebra and optimization techniques. 3

02 **Computer System Architecture** — Number systems, digital logic, CPU organisation, addressing modes, pipelining, memory hierarchy, I/O and parallel processing. 5

03 **Programming Languages and Computer Graphics** — Paradigms, OOP concepts, binding, scope and parameter passing; graphics algorithms, transformations, clipping and colour models. 7

04 **Database Management Systems** — Architecture, ER model, keys, relational algebra, SQL, normalization, transactions, concurrency, recovery, indexing and NoSQL. 9

05 **System Software and Operating System** — Assemblers, linkers, loaders; processes, scheduling, synchronization, deadlocks, memory management, disks and files. 11

06 **Software Engineering** — Process models, requirements, design (cohesion and coupling), testing, metrics and estimation, maintenance and quality. 13

07 **Data Structures and Algorithms** — Linear and non-linear data structures, hashing, sorting and searching, graph algorithms, design paradigms and complexity classes. 15

08 **Theory of Computation and Compilers** — Automata and formal languages, the Chomsky hierarchy, decidability, and the phases and parsers of a compiler. 17

09 **Data Communication and Computer Networks** — OSI and TCP/IP models, transmission, error and flow control, medium access, IP addressing and routing, transport protocols and security. 19

10 **Artificial Intelligence** — Intelligent agents, search, game playing, knowledge representation, reasoning under uncertainty, machine learning and neural networks. 21

How to use these notes

- Read one unit at a time, then test yourself with its key revision points.
- Make your own short notes on topics you find difficult.
- Units follow the main areas of the official syllabus — the exact unit list can change between sessions.
- Always check the latest syllabus and notices on the official UGC NET portal: **ugcnet.nta.ac.in**.

UNIT 01

Discrete Structures and Optimization

Sets, relations, functions, counting, logic, graph theory, Boolean algebra and optimization techniques.

> Discrete mathematics studies countable structures — sets, logic, graphs — that underpin every algorithm and data structure.

Sets, relations and functions

- A set with n elements has 2^n subsets.
- **Inclusion–exclusion:** $|A \cup B| = |A| + |B| - |A \cap B|$.
- **De Morgan:** $(A \cup B)' = A' \cap B'$ and $(A \cap B)' = A' \cup B'$.

Relation property	Meaning (for all a, b, c in A)
Reflexive	$(a, a) \in R$
Symmetric	$(a, b) \in R \Rightarrow (b, a) \in R$
Antisymmetric	(a, b) and $(b, a) \in R \Rightarrow a = b$
Transitive	(a, b) and $(b, c) \in R \Rightarrow (a, c) \in R$

Equivalence relation = reflexive + symmetric + transitive (partitions the set). **Partial order** = reflexive + antisymmetric + transitive (drawn as a Hasse diagram).

Functions: injective (one-to-one), surjective (onto), bijective (both — has an inverse).

Counting

Concept	Formula
Permutations	${}^nP_r = n! \div (n - r)!$
Combinations	${}^nC_r = n! \div (r!\,(n - r)!)$
Pigeonhole principle	$n + 1$ objects in n boxes $\Rightarrow$ some box has ≥ 2

Mathematical logic

- Connectives: $\neg$, $\wedge$, $\vee$, $\rightarrow$ (false only when T $\rightarrow$ F), $\leftrightarrow$.
- **Tautology** — always true; **contradiction** — always false.
- $p \rightarrow q \equiv \neg p \vee q$; its **contrapositive** $\neg q \rightarrow \neg p$ is equivalent.
- Quantifiers: $\neg \forall x\, P(x) \equiv \exists x\, \neg P(x)$.

Graph theory

Fact	Statement
Handshaking lemma	Sum of degrees = 2 $\|E\|$; number of odd-degree vertices is even
Complete graph K_n	$n(n-1) \div 2$ edges
Tree	Connected, acyclic; n vertices $\Rightarrow$ n − 1 edges
Euler circuit	Connected and every vertex has even degree
Euler path	Connected with exactly 0 or 2 odd-degree vertices
Planar graph (Euler)	V − E + F = 2

Boolean algebra

Laws: identity, complement, idempotent (A + A = A), absorption (A + AB = A), De Morgan ((A + B)′ = A′B′). Simplify with Karnaugh maps.

Optimization

- **Linear programming:** maximise/minimise a linear objective subject to linear constraints; the optimum lies at a corner of the feasible region; solved by the **simplex method**.
- **Transportation problem:** initial solution by North-West Corner, Least Cost or Vogel's approximation.
- **Assignment problem:** Hungarian method.
- **PERT/CPM:** project scheduling; the **critical path** is the longest path, with zero slack.

★ Key revision points

- Equivalence = R + S + T; partial order = R + A + T.
- K_n has $n(n-1)/2$ edges; a tree has $n-1$ edges; V − E + F = 2.
- Euler circuit: all degrees even.
- Assignment = Hungarian method; critical path has zero slack.

UNIT 02

Computer System Architecture

Number systems, digital logic, CPU organisation, addressing modes, pipelining, memory hierarchy, I/O and parallel processing.

Computer architecture describes how a computer is organised — from logic gates up to processors, memory and input/output.

Number systems

- Bases: binary (2), octal (8), decimal (10), hexadecimal (16).
- **2's complement** of a number = 1's complement + 1. With n bits the range is **-2^{n-1} to $2^{n-1} - 1$** (8 bits: −128 to 127).

Digital logic

- **NAND** and **NOR** are universal gates — any circuit can be built from either alone.
- **Combinational circuits** (output depends only on inputs): adders, multiplexers, decoders, encoders.
- **Sequential circuits** (have memory): flip-flops — SR, JK, D, T — registers and counters.

CPU and instructions

Instruction cycle: fetch → decode → execute (→ store). The CPU contains the ALU, control unit and registers (PC, IR, MAR, MDR, accumulator).

Addressing mode	Where the operand is
Immediate	In the instruction itself
Direct	At the address given in the instruction
Indirect	At the address stored at the given address
Register	In a register
Indexed	Base address + index register
Relative	PC + offset (used for branches)

RISC	CISC
Few, simple, fixed-length instructions	Many, complex, variable-length instructions
Mostly one cycle per instruction; suits pipelining	Several cycles per instruction
Load/store architecture, many registers	Memory operands in many instructions

Pipelining

- Overlaps the stages of several instructions. Ideal speed-up with k stages ≈ **k**.
- For n instructions in a k-stage pipeline: time = (k + n − 1) cycles.
- **Hazards:** structural (resource conflict), data (dependence between instructions), control (branches).

Memory hierarchy

Registers → cache → main memory (RAM) → secondary storage: faster and costlier at the top, larger and cheaper at the bottom.

- **Cache mapping:** direct, fully associative, set-associative.
- **Hit ratio** h = hits ÷ total accesses. Average access time ≈ $h \times T_{cache} + (1 - h) \times T_{memory}$ (simple model).
- Locality of reference — temporal and spatial — is why caches work.

Input/output and parallelism

- I/O methods: programmed I/O, interrupt-driven I/O, **DMA** (device transfers data directly to memory).
- **Flynn's taxonomy:** SISD, SIMD (vector/GPU), MISD, MIMD (multiprocessors).

★ Key revision points

- n-bit 2's complement range: -2^{n-1} to $2^{n-1} - 1$.
- NAND and NOR are universal.
- k-stage pipeline: n instructions take k + n − 1 cycles.
- DMA moves data without the CPU; SIMD = one instruction, many data.

UNIT 03

Programming Languages and Computer Graphics

Paradigms, OOP concepts, binding, scope and parameter passing; graphics algorithms, transformations, clipping and colour models.

> This unit has two halves: how programming languages work, and how computers draw images.

Programming paradigms

Paradigm	Idea	Examples
Imperative / procedural	Sequence of statements changing state	C, Pascal
Object-oriented	Objects combining data and behaviour	Java, C++, Python
Functional	Pure functions, no side effects	Haskell, Lisp
Logic	Facts and rules; queries	Prolog

Object-oriented concepts

- **Encapsulation** — data and methods bundled; access controlled.
- **Inheritance** — a class reuses another's members.
- **Polymorphism** — compile-time (overloading) and run-time (overriding with dynamic binding).
- **Abstraction** — expose what an object does, hide how.

Names, binding and parameters

- **Static (lexical) scope** is resolved from program text; **dynamic scope** from the call sequence.
- C storage classes: auto, register, static, extern.
- **Call by value** passes a copy; **call by reference** passes the address, so changes affect the caller.

Computer graphics: basics

- **Raster** graphics store pixels; **vector** graphics store geometric primitives and scale without loss.
- **DDA** line drawing uses floating-point increments; **Bresenham's** algorithm uses only integer arithmetic (faster).
- **Midpoint circle** algorithm uses 8-way symmetry.

2D transformations (homogeneous coordinates, 3 × 3 matrices)

Transformation	Effect
Translation	$x' = x + tx$, $y' = y + ty$
Scaling	$x' = sx \cdot x$, $y' = sy \cdot y$
Rotation (by θ)	$x' = x \cos\theta - y \sin\theta$, $y' = x \sin\theta + y \cos\theta$

Homogeneous coordinates let translation be written as a matrix, so a sequence of transformations becomes one matrix product. Matrix multiplication is not commutative — order matters.

Clipping, projection and colour

- **Cohen–Sutherland** line clipping uses 4-bit region codes; **Sutherland–Hodgman** clips polygons.
- **Parallel projection** keeps parallel lines parallel; **perspective projection** makes distant objects smaller.
- Colour models: **RGB** (additive — screens), **CMY/CMYK** (subtractive — printing), **HSV** (hue, saturation, value).

★ Key revision points

- Overloading = compile-time; overriding = run-time polymorphism.
- Bresenham uses integers only.
- Homogeneous coordinates turn translation into matrix multiplication.
- Cohen–Sutherland = lines (region codes); Sutherland–Hodgman = polygons.

UNIT 04

Database Management Systems

Architecture, ER model, keys, relational algebra, SQL, normalization, transactions, concurrency, recovery, indexing and NoSQL.

> A DBMS stores data so that many users can share it reliably, securely and efficiently.

Architecture

- **Three-schema architecture:** external (user views), conceptual (logical structure), internal (physical storage).
- **Logical data independence** — change the conceptual schema without changing views; **physical data independence** — change storage without changing the conceptual schema.

ER model and keys

- Entities, attributes (simple, composite, multivalued, derived), relationships (1:1, 1:N, M:N).
- **Super key** — any attribute set that identifies a tuple. **Candidate key** — minimal super key. **Primary key** — the chosen candidate key. **Foreign key** — refers to a primary key in another table.

Relational algebra and SQL

Operator	Meaning	SQL idea
σ (select)	Choose rows	WHERE
π (project)	Choose columns	SELECT column list
⋈ (join)	Combine related rows	JOIN ... ON
∪, −, ×	Union, difference, Cartesian product	UNION, EXCEPT, CROSS JOIN

SQL sub-languages: **DDL** (CREATE, ALTER, DROP), **DML** (SELECT, INSERT, UPDATE, DELETE), **DCL** (GRANT, REVOKE), **TCL** (COMMIT, ROLLBACK).

Normalization

Normal form	Condition
1NF	All values atomic (no repeating groups)
2NF	1NF and no partial dependency on part of a candidate key
3NF	2NF and no transitive dependency of non-key attributes
BCNF	For every dependency X → Y, X is a super key

Transactions and concurrency

- **ACID:** Atomicity, Consistency, Isolation, Durability.
- A schedule is **conflict-serializable** if its precedence graph has no cycle.
- **Two-phase locking (2PL):** a growing phase then a shrinking phase; guarantees serializability (but deadlocks are possible).
- **Timestamp ordering** orders transactions by their start time.
- **Recovery:** write-ahead log, undo/redo, checkpoints.

Indexing and beyond

- **B+ tree** indexes keep all records at the leaf level, with leaves linked for range queries.
- **NoSQL** types: key-value, document, column-family, graph.
- **OLTP** handles day-to-day transactions; **OLAP** analyses historical data in a data warehouse.

★ Key revision points

- Candidate key = minimal super key.
- 2NF removes partial, 3NF removes transitive dependencies; BCNF: every determinant is a super key.
- Conflict-serializable ⇔ acyclic precedence graph.
- 2PL ensures serializability, not freedom from deadlock.

UNIT 05

System Software and Operating System

Assemblers, linkers, loaders; processes, scheduling, synchronization, deadlocks, memory management, disks and files.

> System software manages the computer itself; the operating system sits between users' programs and the hardware.

System software

- **Assembler** converts assembly to machine code — often in two passes (first pass builds the symbol table).
- **Linker** combines object files and resolves references; **loader** places the program in memory.
- **Compiler** translates the whole program before running; **interpreter** translates and runs line by line.
- **Macro processor** expands macros before assembly or compilation.

Processes and scheduling

Process states: new → ready → running → waiting → terminated. Each process has a **Process Control Block (PCB)**.

Algorithm	Key point
FCFS	Simple; convoy effect
SJF	Minimum average waiting time (non-preemptive)
SRTF	Preemptive version of SJF
Round Robin	Time quantum; good response time
Priority	Risk of starvation — solved by **aging**

Synchronization

- A **critical section** needs mutual exclusion, progress and bounded waiting.
- **Semaphores** use wait (P) and signal (V); a binary semaphore works like a mutex.
- Classic problems: producer–consumer, readers–writers, dining philosophers.

Deadlocks

Four necessary (Coffman) conditions: **mutual exclusion**, **hold and wait**, **no preemption**, **circular wait**. Handling: prevention, **avoidance (Banker's algorithm)**, detection and recovery.

Memory management

- **Paging** — fixed-size pages and frames; no external fragmentation. **Segmentation** — variable-size logical segments.
- **Virtual memory** with demand paging; too much paging causes **thrashing**.
- Page replacement: **FIFO** (can show Belady's anomaly), **LRU**, **Optimal** (replace the page used furthest in future — theoretical best).

Disks and files

- Disk scheduling: FCFS, SSTF (shortest seek first), SCAN (elevator), C-SCAN, LOOK, C-LOOK.
- File allocation: contiguous, linked, indexed.

★ Key revision points

- SJF gives minimum average waiting time; aging prevents starvation.
- Four Coffman conditions; Banker's algorithm = avoidance.
- Belady's anomaly occurs with FIFO.
- Two-pass assembler: pass 1 builds the symbol table.

UNIT 06

Software Engineering

Process models, requirements, design (cohesion and coupling), testing, metrics and estimation, maintenance and quality.

> Software engineering applies a disciplined, measurable approach to building and maintaining software.

Process models

Model	Key feature	Best when
Waterfall	Linear phases, one after another	Requirements are clear and stable
Prototype	Build a quick model to clarify needs	Requirements are unclear
Incremental	Deliver in working increments	Early partial delivery is useful
Spiral (Boehm)	Iterative and **risk-driven**	Large, high-risk projects
V-model	Each development phase has a matching test phase	Verification is critical
Agile (Scrum, XP)	Short iterations, customer collaboration	Requirements change often

Requirements

Captured in the **Software Requirements Specification (SRS)**. **Functional** requirements say what the system does; **non-functional** requirements cover performance, security, usability and reliability.

Design: cohesion and coupling

Cohesion (aim: high)

How closely the parts of one module belong together. Best: **functional**. Worst: **coincidental**.

Coupling (aim: low)

How much modules depend on each other. Best: **data** coupling. Worst: **content** coupling.

Testing

- Levels: **unit → integration → system → acceptance**.
- **Black-box** (based on specification): equivalence partitioning, boundary value analysis.
- **White-box** (based on code): statement, branch and path coverage.

- **Cyclomatic complexity** V(G) = E − N + 2P (edges, nodes, connected components) = number of independent paths.
- **Verification**: are we building the product right? **Validation**: are we building the right product?

Metrics and estimation

- Size: lines of code (LOC), **function points**.
- **COCOMO** (Boehm): effort from size; project types organic, semi-detached, embedded.

Maintenance and quality

- Maintenance types: **corrective** (fix bugs), **adaptive** (new environment), **perfective** (improve features), **preventive** (avoid future problems).
- **CMM levels:** 1 Initial, 2 Repeatable, 3 Defined, 4 Managed, 5 Optimizing.

★ Key revision points

- Spiral = risk-driven (Boehm).
- High cohesion, low coupling.
- V(G) = E − N + 2P.
- CMM: Initial → Repeatable → Defined → Managed → Optimizing.

UNIT 07

Data Structures and Algorithms

Linear and non-linear data structures, hashing, sorting and searching, graph algorithms, design paradigms and complexity classes.

> Data structures organise data; algorithms process it. The right choice decides how fast a program runs.

Linear structures

- **Stack** — LIFO; push/pop in O(1); used for recursion and expression evaluation.
- **Queue** — FIFO; variants: circular queue, deque, priority queue.
- **Linked list** — singly, doubly, circular; O(1) insertion when the position is known.

Trees and heaps

- A binary tree has at most 2^i nodes at level i and at most $2^{h+1} - 1$ nodes in height h (root at level 0).
- Traversals: **inorder** (left, root, right — gives sorted order in a BST), preorder, postorder, level order.
- **AVL tree**: balance factor of every node is −1, 0 or +1; search O(log n).
- **Heap**: complete binary tree; max-heap or min-heap; build in O(n).

Hashing

Maps keys to table slots. Collisions are handled by **chaining** or **open addressing** (linear probing, quadratic probing, double hashing).

Searching and sorting

Algorithm	Best	Average	Worst
Linear search	O(1)	O(n)	O(n)
Binary search (sorted)	O(1)	O(log n)	O(log n)
Bubble / insertion sort	O(n)	$O(n^2)$	$O(n^2)$
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Merge sort	O(n log n)	O(n log n)	O(n log n)
Quick sort	O(n log n)	O(n log n)	$O(n^2)$
Heap sort	O(n log n)	O(n log n)	O(n log n)

Graph algorithms

Algorithm	Purpose	Note
BFS / DFS	Traversal	BFS uses a queue; DFS uses a stack (or recursion)
Dijkstra	Single-source shortest path	No negative edge weights
Bellman–Ford	Single-source shortest path	Handles negative weights; detects negative cycles
Floyd–Warshall	All-pairs shortest path	$O(V^3)$
Kruskal / Prim	Minimum spanning tree	Both greedy

Design paradigms and complexity

- **Divide and conquer** (merge sort, quick sort), **greedy** (Kruskal, Huffman coding), **dynamic programming** (0/1 knapsack, longest common subsequence), **backtracking** (N-queens).
- **P** — solvable in polynomial time. **NP** — solutions verifiable in polynomial time. **NP-complete** — in NP and NP-hard; SAT was the first proved NP-complete (Cook).

★ Key revision points

- Merge and heap sort are O(n log n) in all cases; quick sort worst case is $O(n^2)$.
- Dijkstra fails with negative edges; Bellman–Ford handles them.
- Inorder traversal of a BST gives sorted order.
- 0/1 knapsack = dynamic programming; fractional knapsack = greedy.

UNIT 08

Theory of Computation and Compilers

Automata and formal languages, the Chomsky hierarchy, decidability, and the phases and parsers of a compiler.

> Theory of computation asks what can be computed and how efficiently; compiler design applies that theory to translating programs.

Chomsky hierarchy

Type	Grammar / language	Recognised by
Type 0	Unrestricted (recursively enumerable)	Turing machine
Type 1	Context-sensitive	Linear bounded automaton
Type 2	Context-free	Pushdown automaton
Type 3	Regular	Finite automaton

Finite automata and regular languages

- **DFA and NFA are equivalent** in power (subset construction converts an NFA to a DFA, possibly with 2^n states).
- Regular expressions describe exactly the regular languages.
- Regular languages are closed under union, concatenation, Kleene star, intersection and complement.
- The **pumping lemma** is used to prove a language is *not* regular (e.g. $a^n b^n$).

Context-free languages

- Recognised by pushdown automata (a finite automaton with a stack).
- Closed under union, concatenation and star — **not** under intersection or complement.
- A grammar is **ambiguous** if some string has two parse trees.

Turing machines and decidability

A Turing machine is the model of general computation. The **halting problem is undecidable** — no algorithm can decide for every program and input whether it halts.

Phases of a compiler

1. Lexical analysis (tokens)
2. Syntax analysis (parse tree)
3. Semantic analysis (types)
4. Intermediate code (three-address code)
5. Code optimization
6. Code generation

The **symbol table** is shared by all phases. Lexical analysers are built from regular expressions / finite automata (e.g. Lex); parsers from context-free grammars (e.g. YACC).

Parsing

Top-down	Bottom-up
Builds the tree from the root; leftmost derivation	Builds from the leaves; rightmost derivation in reverse
Recursive descent, LL(1)	Shift-reduce: LR(0), SLR, **LALR**, CLR
Grammar must have no left recursion and be left-factored	Power: LR(0) < SLR < LALR < CLR; YACC uses LALR

★ Key revision points

- Type 0 → TM, Type 1 → LBA, Type 2 → PDA, Type 3 → FA.
- DFA = NFA in power; CFLs are not closed under intersection or complement.
- The halting problem is undecidable.
- LL(1) cannot handle left recursion; CLR is the most powerful LR parser.

UNIT 09

Data Communication and Computer Networks

OSI and TCP/IP models, transmission, error and flow control, medium access, IP addressing and routing, transport protocols and security.

Computer networks let devices exchange data reliably using layered protocols.

Reference models

OSI layer	Role	Example / device
7 Application	Services for users	HTTP, SMTP, DNS
6 Presentation	Format, encryption, compression	TLS, JPEG
5 Session	Dialogue control	Sessions, checkpoints
4 Transport	End-to-end delivery	TCP, UDP
3 Network	Routing, logical addressing	IP; router
2 Data link	Framing, MAC addressing, error control	Ethernet; switch
1 Physical	Bits on the medium	Cables; hub, repeater

The **TCP/IP model** has 4 layers: application, transport, internet, network access (link).

Transmission

- Modes: simplex, half-duplex, full-duplex.
- Media: guided (twisted pair, coaxial, optical fibre) and unguided (radio, microwave, infrared).
- Multiplexing: FDM, TDM, WDM (optical). Switching: circuit vs packet.

Error and flow control

- **Detection:** parity, checksum, **CRC**. **Correction:** Hamming code.
- **Flow control:** stop-and-wait; sliding window — **Go-Back-N** and **Selective Repeat**.

Medium access

- **Pure ALOHA** maximum throughput ≈ 18.4%; **slotted ALOHA** ≈ 36.8%.
- **CSMA/CD** — wired Ethernet; **CSMA/CA** — Wi-Fi.

Network layer

- **IPv4** is 32-bit; classful ranges by first octet: A 1–126, B 128–191, C 192–223 (127 is loopback). **IPv6** is 128-bit.
- Subnetting splits a network; CIDR notation e.g. /24.
- Routing: **distance vector** (RIP), **link state** (OSPF), path vector (BGP, between autonomous systems).

Transport and application

TCP	UDP
Connection-oriented (3-way handshake), reliable, ordered	Connectionless, no delivery guarantee
Flow and congestion control	Low overhead — streaming, DNS queries

Well-known ports: FTP 20/21, SSH 22, Telnet 23, SMTP 25, DNS 53, HTTP 80, HTTPS 443.

Network security

- **Symmetric** encryption (one shared key): DES, AES. **Asymmetric** (public/private keys): RSA.
- **Digital signature** — signed with the sender's private key, verified with the public key.
- Firewalls filter traffic; VPNs create encrypted tunnels.

★ Key revision points

- Hub = layer 1, switch = layer 2, router = layer 3.
- Pure ALOHA 18.4%, slotted 36.8%.
- RIP = distance vector; OSPF = link state.
- HTTP 80, HTTPS 443, DNS 53, SMTP 25.

UNIT 10

Artificial Intelligence

Intelligent agents, search, game playing, knowledge representation, reasoning under uncertainty, machine learning and neural networks.

> Artificial intelligence builds systems that perceive, reason, learn and act to achieve goals.

Foundations

- **Turing test** (Alan Turing, 1950) — a machine is intelligent if a human cannot tell it from a person in conversation.
- **Agents:** simple reflex, model-based, goal-based, utility-based, learning. Specified by **PEAS** — Performance, Environment, Actuators, Sensors.

Search

Strategy	Type	Note
Breadth-first	Uninformed	Complete; optimal when step costs are equal
Depth-first	Uninformed	Low memory; not optimal
Uniform cost	Uninformed	Expands lowest path cost first
Iterative deepening	Uninformed	DFS memory with BFS completeness
Greedy best-first	Informed	Uses heuristic h(n) only
A*	Informed	f(n) = g(n) + h(n); optimal if h is admissible

Game playing: minimax for two-player games; **alpha–beta pruning** skips branches that cannot affect the result.

Knowledge representation and reasoning

- Propositional and **first-order logic**; inference by **resolution**.
- Semantic networks, frames, production rules.
- **Expert systems** = knowledge base + inference engine; **forward chaining** (data-driven) and **backward chaining** (goal-driven).

Uncertainty

- **Bayes' theorem:** $P(A \mid B) = P(B \mid A) \times P(A) \div P(B)$. Bayesian networks model dependencies between variables.
- **Fuzzy logic** allows degrees of truth between 0 and 1 (Lotfi Zadeh).

Machine learning

Type	Learns from	Examples
Supervised	Labelled data	Classification, regression
Unsupervised	Unlabelled data	Clustering (k-means)
Reinforcement	Rewards and penalties	Game-playing, robotics

Neural networks and evolutionary methods

- A **perceptron** is a single artificial neuron; a single layer cannot learn XOR.
- Multi-layer networks are trained by **backpropagation**; activation functions include sigmoid, tanh and ReLU.
- **Genetic algorithms**: selection, crossover and mutation over a population.

★ Key revision points

- A* is optimal with an admissible heuristic; f = g + h.
- Alpha–beta pruning speeds up minimax without changing the result.
- Forward chaining = data-driven; backward chaining = goal-driven.
- A single-layer perceptron cannot learn XOR.

QuickNotes

Learn Quickly. Prepare Smartly.

More free study notes

Free PDFs for UGC NET Paper 1, every Paper 2 subject, IBPS RRB and SSC CHSL — no sign-up.

www.getquicknotes.com

Free for personal study — not for sale. You may share this PDF unchanged and free of charge.

QuickNotes is an independent educational platform. It is not affiliated with, endorsed by or connected to the University Grants Commission (UGC), the National Testing Agency (NTA) or any government examination authority. These are original revision notes; always refer to the official notification for exam dates, eligibility, pattern and syllabus.

Found a mistake? Email support@getquicknotes.com · Edition: November 2026