Computer Science Essentials Cheatsheet

A concise reference for fundamental computer science concepts, algorithms, data structures, and programming paradigms. Ideal for students and professionals alike.

Core Concepts

Data Structures

Arrays	Contiguous memory blocks; efficient for random access (O(1) lookup by index).
Linked Lists	Nodes containing data and a pointer to the next node; efficient for insertion/deletion (O(1) if location is known).
Stacks	LIFO (Last-In, First-Out) data structure. Push (add), Pop (remove) operations.
Queues	FIFO (First-In, First-Out) data structure. Enqueue (add), Dequeue (remove) operations.
Hash Tables	Key-value pairs; efficient for search, insertion, and deletion (average O(1) time complexity).
Trees	Hierarchical data structure; Binary Search Trees (BSTs) allow for efficient searching, insertion and deletion (O(log n) average).
Graphs	Nodes connected by edges; used to represent relationships between objects. Can be directed or undirected.

Algorithms

Sorting Algorithms	Examples include Bubble Sort, Insertion Sort, Merge Sort, Quick Sort. Different algorithms have different time complexities and suitability for various data sets.
Searching Algorithms	Linear Search (O(n)), Binary Search (O(log n) - requires sorted data).
Graph Algorithms	Examples: Breadth-First Search (BFS), Depth-First Search (DFS), Dijkstra's Algorithm (shortest path), Prim's Algorithm (minimum spanning tree).
Dynamic Programming	Optimizing by breaking problems into overlapping subproblems and storing solutions to avoid redundant computations.
Greedy Algorithms	Making locally optimal choices at each step with the hope of finding a global optimum (not always guaranteed).

Time Complexity (Big O Notation)

O(1): Constant time (e.g., accessing an element in an array by index).
O(log n): Logarithmic time (e.g., binary search).
O(n): Linear time (e.g., linear search).
O(n log n): (e.g., merge sort, quicksort).
O(n^2): Quadratic time (e.g., bubble sort, insertion sort).
O(2^n): Exponential time (e.g., brute-force search).
O(n!): Factorial time (e.g., traveling salesman problem brute-force).

Programming Paradigms

Imperative Programming

Focuses on *how* to achieve a result by explicitly changing the program's state through commands (statements).

Example: C, Java

Declarative Programming

Focuses on *what* result is desired, without specifying the exact steps. Examples include functional and logic programming.

Example: Haskell, SQL

Object-Oriented Programming (OOP)

Encapsulation	Bundling data (attributes) and methods that operate on that data within a class.
Inheritance	Creating new classes (derived classes) from existing classes (base classes), inheriting their properties and behaviors.
Polymorphism	The ability of an object to take on many forms. Achieved through method overriding and interfaces.
Abstraction	Hiding complex implementation details and exposing only essential information to the user.
Examples	Java, C++, Python

Functional Programming

Immutability	Data cannot be changed after it is created. Creates predictable state.
Pure Functions	Functions that always return the same output for the same input and have no side effects.
First-Class Functions	Functions can be treated as values, passed as arguments, and returned from other functions.
Examples	Haskell, Lisp, Scala, JavaScript (with functional libraries).

Computer Architecture

CPU Components

ALU (Arithmetic Logic Unit)	Performs arithmetic and logical operations.
Control Unit	Fetches instructions from memory and decodes them, coordinating the activities of the CPU.
Registers	Small, fast storage locations within the CPU used to hold data and instructions that are being actively processed.
Cache Memory	Small, fast memory that stores frequently accessed data, reducing the time needed to retrieve it from main memory (RAM).

Memory Hierarchy

CPU Registers -> Cache Memory (L1, L2, L3) -> RAM (Main Memory) -> Solid State Drive (SSD) / Hard Disk Drive (HDD).

Speed and cost decrease as you move down the hierarchy, while capacity increases.

Input/Output (I/O)

Input Devices	Keyboard, Mouse, Scanner, Microphone
Output Devices	Monitor, Printer, Speakers
I/O Controllers	Manage data transfer between the CPU and I/O devices.

Operating Systems (OS)

Manages hardware resources, provides services to applications (e.g., memory management, file system, process scheduling).

Examples: Windows, macOS, Linux.

Networking Fundamentals

OSI Model

A conceptual model that standardizes the communication functions of a telecommunication or computing system without regard to its underlying internal structure and technology.

Layers:
7. Application
6. Presentation
5. Session
4. Transport
3. Network
2. Data Link
1. Physical

TCP/IP Model

A practical model for network communication used on the Internet.

Layers:
4. Application
3. Transport
2. Internet
1. Network Access (Data Link + Physical)

Common Protocols

HTTP/HTTPS	Hypertext Transfer Protocol (Secure). Used for web browsing.
TCP	Transmission Control Protocol. Provides reliable, ordered delivery of data.
UDP	User Datagram Protocol. Provides fast, connectionless delivery of data (unreliable).
IP	Internet Protocol. Provides addressing and routing of data packets.
DNS	Domain Name System. Translates domain names (e.g., google.com) to IP addresses.

Network Devices

Routers	Forward data packets between networks.
Switches	Connect devices within a network.
Firewalls	Protect networks from unauthorized access.