Computer science is a vast field, but textbooks typically break it down into several core areas to provide a foundational understanding. Here are the basic parts of computer science you'd commonly find in regular textbooks:

---

1. Introduction to Computer Science & Programming 👨‍💻

This foundational section covers what computer science is, its history, and its various subfields. It usually introduces basic programming concepts, algorithms, and problem-solving techniques. You'll often start with a high-level overview of how computers work and basic programming constructs in a beginner-friendly language.

---

2. Data Structures and Algorithms 📊

This is a cornerstone of computer science. It focuses on:

- **Data Structures:** How data is organized and stored efficiently (e.g., arrays, linked lists, stacks, queues, trees, graphs, hash tables).
- **Algorithms:** Step-by-step procedures for solving computational problems (e.g., sorting, searching, graph traversal, dynamic programming). Analysis of algorithm efficiency (time and space complexity) is a major part here.

---

3. Computer Organization and Architecture 🖥️

This area delves into the hardware aspects of computers, covering:

- **Digital Logic:** Boolean algebra, logic gates, and circuits.
- **Processor Design:** CPU architecture, instruction sets, pipelining, memory hierarchy (cache).
- **Memory Systems:** RAM, ROM, virtual memory.
- **Input/Output (I/O) Systems:** How computers interact with external

devices.

---

4. Operating Systems (OS) ⚙️

Operating systems manage a computer's hardware and software resources. This section explores:

- **Process Management:** How the OS handles multiple running programs (processes), scheduling, and synchronization.
- **Memory Management:** Allocating and deallocating memory to processes.
- **File Systems:** Organizing and managing data on storage devices.
- **I/O Management:** Handling communication with peripheral devices.
- **Security and Protection:** Mechanisms to ensure system integrity and user privacy.

---

5. Discrete Mathematics 🧱

While not strictly a "computer science" topic, discrete mathematics is fundamental to understanding many CS concepts. It typically includes:

- **Logic:** Propositional and predicate logic.
- **Set Theory:** Collections of objects.
- **Relations and Functions:** Mapping between sets.
- **Graph Theory:** Nodes and edges, used in networks and algorithms.
- **Combinatorics and Probability:** Counting techniques and likelihood.

---

6. Databases 🗄️

This section focuses on the organization, storage, and retrieval of data. Key topics include:

- **Database Models:** Relational, NoSQL.
- **Database Design:** Entity-Relationship (ER) diagrams, normalization.
- **Query Languages:** SQL (Structured Query Language) for managing and retrieving data.
- **Transaction Management:** Ensuring data integrity.

---

7. Networking and Communications 🌐

This area covers how computers connect and communicate with each other:

- **Network Topologies:** How networks are physically and logically arranged.
- **Network Protocols:** Rules for communication (e.g., TCP/IP, HTTP).
- **Network Devices:** Routers, switches, hubs.
- **Security:** Cryptography, firewalls, network attacks.

---

8. Software Engineering 🏗️

This discipline focuses on the systematic approach to designing, developing, deploying, and maintaining software. Topics often include:

- **Software Development Life Cycle (SDLC):** Phases from requirements to maintenance.
- **Requirements Engineering:** Gathering and documenting user needs.
- **Design Patterns:** Reusable solutions to common software design problems.
- **Testing and Quality Assurance:** Ensuring software reliability.
- **Version Control:** Managing changes to code (e.g., Git).

---

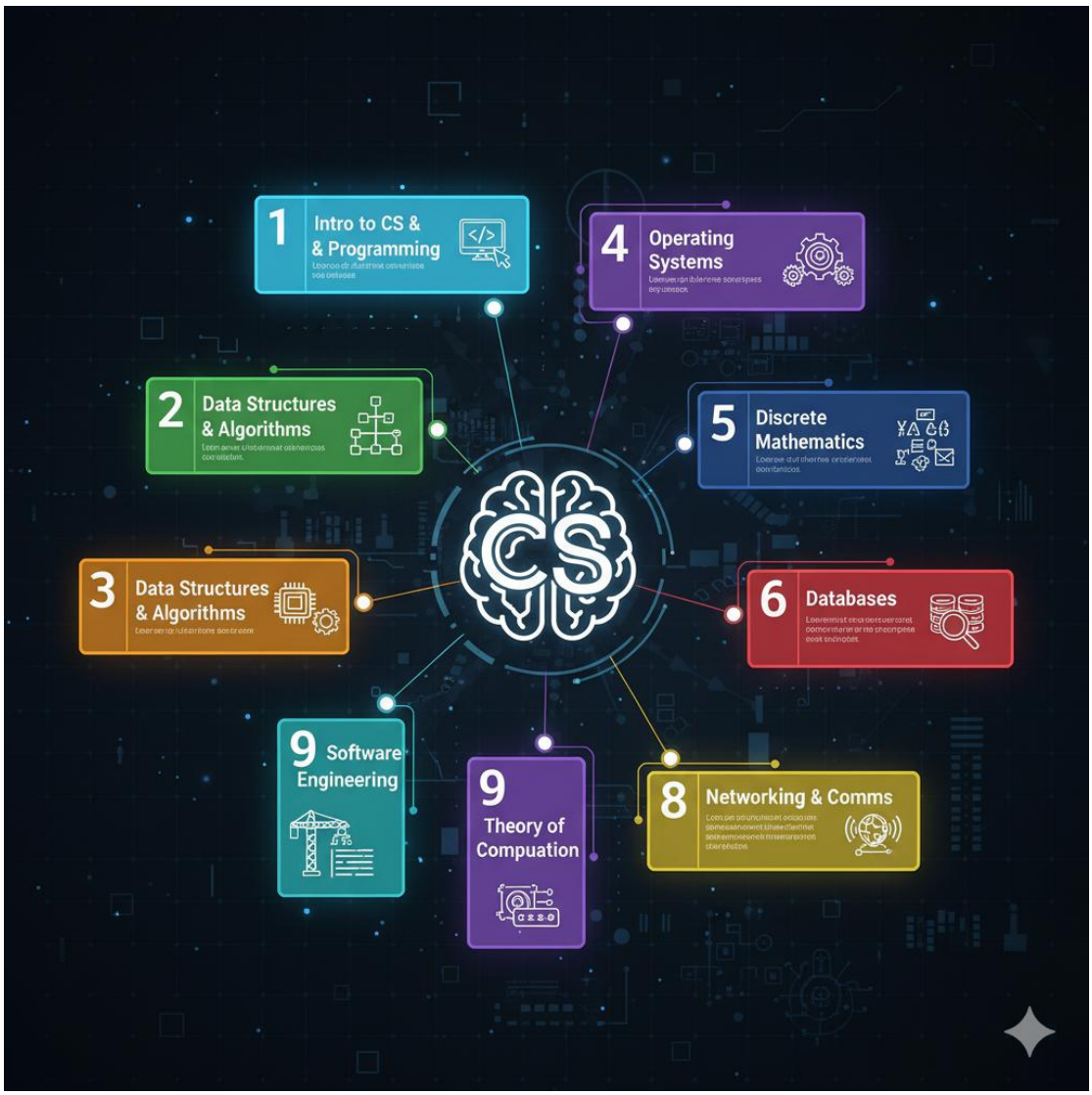
9. Theory of Computation (Formal Languages and Automata) 🧠

This more theoretical branch explores the fundamental capabilities and limitations of computation:

- **Automata Theory:** Abstract machines (e.g., finite automata, pushdown automata, Turing machines).
- **Formal Languages:** Grammars and languages these machines can process.
- **Computability Theory:** What problems can and cannot be solved by computers.
- **Complexity Theory:** The resources (time, space) required to solve computational problems.

These core areas often build upon each other, providing a comprehensive understanding of how computers work, how to program them, and the theoretical underpinnings of computation.

Here's an image that visualizes these main pillars of computer science:


1
Intro to CS &
& Programming
4
Operating
Systems
2
Data Structures
& Algorithms
5
Discrete
Mathematics
CS
3
Data Structures
& Algorithms
6
Databases
9
Software
Engineering
9
Theory of
Compuation
8
Networking & Comms