Computer Science Fundamentals Cheat Sheet

educative

Hardware and software fundamentals

Hardware *is anything physically connected to a computer.*

Central Processing Unit (CPU)
A physical object that takes data from the main memory, processes it, and returns the updated data to the main memory.

Control unit (CU)
A subunit of the CPU that controls data flow from and into the main memory.

Arithmetic and logic unit (ALU)
Subunit of the CPU that is responsible for processing arithmetic and logic operations.

Input units
Take data from the world or an input device and convert it into streams of bytes.

Output units
Take processed data from the CPU and render it in a human-understandable way.

Storage units
Where data is stored after being retrieved and processed. The physical memory space.

Memory
The main memory and RAM (physical spaces in the computer) and secondary storage (e.g. hard drives).

Software *is a collection of programs and procedures that perform tasks on a computer.*

Machine language
The only language the computer can process: a stream of ones and zeros (binary). A low-level language.
-
Assembly language
A human-readable language that translates binary into assembly instruction, which must be translated into machine language for the computer. A low-level language.

High-level programming languages
Allow the writing of human-readable programs without large amounts of low-level instructions (i.e. assembly language instructions).

Assembler
A utility program that translates an assembly language program into machine language.

Compiler
A program that translates human-readable source code into machine-readable target code in a low-level language. Once the translation is complete, the target code is passed to the target machine for execution.

Interpreter
A program that translates human-readable source code into machine-readable target code in a low-level language command by command while the source code is being executed.

Operating system
Software that supports a computer's basic functions, manages computer hardware and software resources, and provides common services for computer programs

User applications
Software written for the end-user that's designed to carry out a task unrelated to the operation of the computer system.

Data structure fundamentals

Data structures: *Formats for the organization, management, and storage of data that enable efficient access and modification.*

Array
A collection of items of the same variable type that are stored sequentially in memory. Best suited for retrieving data in a constant time (using index) but don't provide fast data insertion or deletion.

Linked list
A linear sequence of nodes linked together. In a singly linked list, each node contains a value and a pointer to the next node in the list. Linked lists provide faster data insertion and deletion but slower data retrieval compared to arrays.

Tree
A non-linear data structure often used to represent hierarchical data.

Stack
A linear structure with last-in, first-out (LIFO) order. Imagine a stack of plates. The last plate placed on top of the stack is the first taken out.

Queue
A linear structure with first-in, first-out (FIFO) order. Imagine lining up for a roller coaster. The first people who line up leave the line for the ride first.

Graph
An abstract notation that represents the connection between all pairs of objects.

Hash table
A structure implemented by storing elements in an array and identifying them through a key. A hash function takes in a key and returns an index for which the value is stored.

Heap
An advanced tree-based data structure used primarily for sorting and implementing priority queues.

Algorithm fundamentals

Algorithm: *A series of well-defined instructions that tell a computer what to do to solve a problem. Algorithms are applied to data structures.*

Complexity and correctness concepts

Asymptotic time complexity
A platform- and input-independent analysis that computes the exact running time of an algorithm. It tells us how a program performs as the size of input grows regardless of the underlying machine. Big O is used to represent the upper bound, Big Omega is used to represent the lower bound, and Big Theta is used to represent the tight bound of running time.

Time complexity of recursive algorithms
Can be computed using the substitution method, Master's theorem, or recursion tree.

Asymptotic space complexity
An analysis of how much memory an algorithm takes.

Correctness proof techniques
Used to prove that a given algorithm is correct and will always produce the intended output. The most common and widely used technique is loop invariant, which is based on mathematical induction.

Design techniques

Brute force
Requires going through all possibilities to find a solution to a problem. The least efficient method and one that mostly doesn't provide the desired solution in a feasible time.

Divide and conquer
Breaks a problem into smaller subtasks that are then solved using recursion and eventually reassembled. Recursion is the practice in which a function calls itself directly or indirectly. Examples include merge sort and quicksort.

Dynamic programming
Similar to divide and conquer. Divides a big problem into small subtasks and combines their solutions. Unlike divide and conquer, a subtask may overlap with other subtasks. To reduce running time, results of each subtask are saved in memory, a process called memoization.

Greedy
A solution for each subtask is attempted using the best available local solution, called local optima. This approach yields optimal results only when local optima leads to the global optima, the best possible global solution.

Other techniques
Approximation algorithms find a near-optimal solution when finding an optimal solution is either time-consuming or not feasible. Other techniques include randomized algorithms and linear programming.

Key categories

Sorting and searching algorithms
Put elements of a list in order, or check for or retrieve an element from any data structure where it's stored. Sorting examples: mergesort, quicksort, bubble sort, selection sort, and insertion sort. Searching examples: linear search and binary search.

Graph algorithms
Solve problems of representing graphs as networks. A graph is an abstract notation that represents the connection between all pairs of objects.

Shortest path algorithms
Find the shortest path in a graph. Many sorting algorithms exist. An algorithm is selected based on the type of data, its size, and the user application.