tags: - cocubes - technical - data-structures

Data Structures

Array

An array stores same-type elements in contiguous memory.

Operation	Typical time	Why
Index access	O(1)	address is calculated directly
Search unsorted	O(n)	may inspect every element
Insert/delete middle	O(n)	shift elements
Append dynamic array	amortised O(1)	resize happens occasionally

Use an array for fast indexed access and a fixed/known sequence.

Linked list

Each node stores data and links to next, and possibly previous, nodes.

Operation	Time
Access kth element	O(n)
Insert at known node/head	O(1)
Search	O(n)

Use a linked list when frequent insertion/deletion matters more than random access. A doubly linked list stores both next and previous links but uses more memory.

Stack

Stack = Last In, First Out.

- Main operations: push, pop, peek/top.
- All are O(1) in normal implementations.
- Uses: function call stack, undo, expression evaluation, balanced parentheses, DFS.

Queue

Queue = First In, First Out.

- Main operations: enqueue at rear, dequeue at front.
- Uses: scheduling, printer queue, BFS, buffering.
- A circular queue reuses empty positions in a fixed-size array.

Deque

Deque = double-ended queue. Insert/delete at both ends. It can behave as stack or queue.

Trees

A tree is a connected acyclic graph. For a tree with n nodes:

- number of edges = n − 1;
- a full binary tree has either 0 or 2 children per internal node;
- at level L, with root at level 0, maximum nodes = 2^L;
- maximum nodes through level h = 2^(h+1) − 1.

Binary search tree

For every node:

- values in left subtree are smaller;
- values in right subtree are larger.

Inorder traversal of a BST gives keys in sorted order.

Operation	Balanced BST	Worst skewed BST
Search/insert/delete	O(log n)	O(n)

Heap

A binary heap is a complete binary tree commonly stored in an array.

- Min-heap: parent <= children; minimum at root.
- Max-heap: parent >= children; maximum at root.
- Insert/remove root: O(log n).
- Build heap from n items: O(n).
- Use: priority queue, heap sort, scheduling.

Hash table

A hash function maps a key to an index/bucket.

- Average search/insert/delete: O(1).
- Worst case: O(n), for many collisions.
- Collision handling: chaining or open addressing (linear/quadratic probing/double hashing).
- Load factor = elements/table slots. High load factor increases collisions.

Use a hash table for membership, frequency counting and key-to-value lookup.

Graph

Graph = vertices/nodes plus edges.

Representation	Space	Good for
Adjacency matrix	$O(V^2)$	dense graph, O(1) edge lookup
Adjacency list	O(V+E)	sparse graph, iterate neighbours

- BFS uses a queue and finds shortest path in an unweighted graph.
- DFS uses recursion/stack; useful for connectivity, cycles and traversal.

Linked-list cycle detection

Floyd's tortoise-and-hare method advances one pointer by one node and another by two nodes. If a cycle exists, they eventually meet; if the fast pointer reaches null/end, no cycle exists. Time O(n), extra space O(1).

Choose the structure quickly

Requirement	Structure
Last item first	Stack
First arrival first	Queue
Fast arbitrary index	Array
Fast key lookup/frequency	Hash table
Repeated minimum/maximum	Heap/priority queue
Hierarchy	Tree
General connections/routes	Graph

[[03_Technical/Algorithms|Algorithms]] · [[03_Technical/Technical_Quick_Recall|Quick recall]]