Binary search needs sorted data so it can cut the range in half each time. If the data is not sorted, the mid check tells you nothing and the search can jump away from the target.

A hash table handles collisions by chaining or probing. This matters because if collisions stack up, the table slows down a lot.

A min-heap restores order by bubbling the new item up or pushing the removed root down. This keeps the root as the smallest.

BFS checks all nodes layer by layer, so the first time it reaches a node is the shortest path in steps. This works only because all edges have equal weight.

Quicksort is fast when pivots split the array well. It is slow when the pivot is always bad and makes one huge side.

A linked list inserts by changing two pointers, so it does not shift data like an array. It is slower to index but easy to insert.

DP saves answers to subproblems so you do not compute them again. Overlapping subproblems make this pay off.

DFS uses a stack to go deep before going wide. The stack order forces it to follow one long path first.

A trie stores each word by its letters in a tree path. This helps with prefix lookups because shared parts are stored once.

Balanced BSTs keep height small so searches stay fast. When they tilt to one side, they act like linked lists and slow down.