Algorithms are simply steps in a process that we take to perform a desired action, usually with computers.

Algorithms allow us to use built-in and custom data structures to perform actions on that data.

Data Structures + Algorithms = Programs

Certain algorithms allow us to improve our time complexity, possibly from O(n2) to O(logn).

This is super useful for big companies as they work with huge volumes of data.

RECURSION:

This is more of a concept than an algorithm. It is a highly popular topic in interviews. Recursion is when you define something in terms of itself. In programming, this can be a function that calls itself.

From Bing GPT, recursion is a concept that occurs when the definition of a process depends on a simpler version of itself. It is the repeated application of a recursive procedure or definition. In maths or computer science, recursion is often used where a function being defined is applied within its own definition. This technique allows the representation of data whose exact size is unknown to the programmer.

A class, objects, or functions, exhibit recursive behaviour when it can be defined b two properties:

* A simple base case/s – a terminating scenario that does not use recursion to produce an answer. If this is left out or never reached, then the function will continue calling itself until a stack overflow (function calls are stored in memory, eventually this memory runs out) occurs.
* A recursive step – a set of rules that reduces all successive cases towards the base case.

For example, the Fibonacci sequence is a classic example of recursion:

* Fib(0) = 0 as base case 1,
* Fib(1) = 1 as base case 1,
* For all integers n>1, Fib(n) = Fib(n-1) + Fib(n-2)

Recursive functions such as the solution to the Fibonacci sequence can have an O(2n) complexity, which is exponential time complexity and is utterly shit. It is terrible.

This can be improved to linear time complexity via the application of dynamic programming and memorization.

So, why use recursion? Especially when any problem that can be solved with recursion can also be solved with normal loops.

Recursion can improve readability, but there are trade-offs, names increased memory usage via a large call stack.

Recursion can be good for data structures where the depth is unknown, such that the programmer does not know how many loops to cover. A common usage is Tree or Graph traversal.