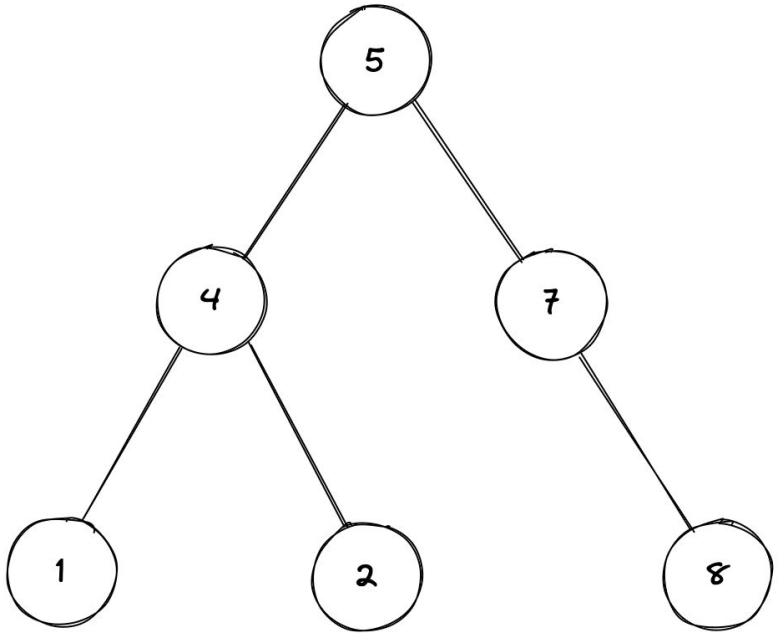
skill academy

Introduction to Data Structures and Algorithms



By Pranjal Nadhani

Data Structures

skill academy

What is a data structure?

Wikipedia defines a data structure as a data organization, management, and storage format that is usually chosen for efficient access to data.

A data structure is a way of storing data such that it can be read, written to or deleted in the most efficient manner, depending on the use case.

Every data structure has a specific set of pros and cons and they're chosen according to the requirements of the problem statement.

Algorithms

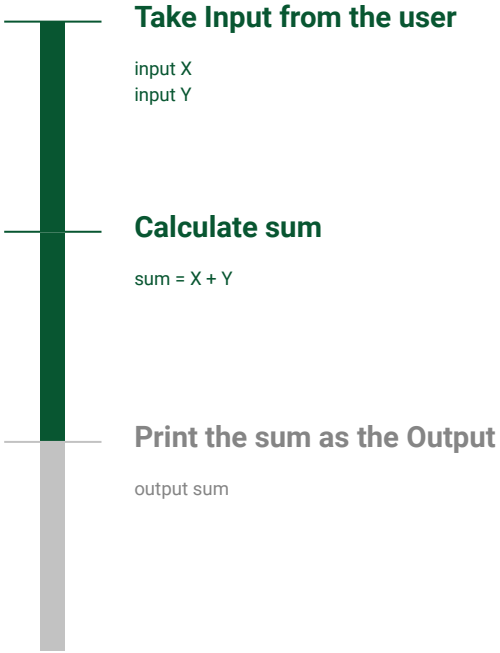
skill academy

What is an algorithm?

Wikipedia defines an algorithm as a finite sequence of instructions, typically used to solve a class of specific problems or to perform a computation.

An algorithm is a set of steps that are executed to solve a problem in the most efficient manner. We take into account how much time an algorithm takes, through the concept of **Time Complexity** (TC) and how much space it requires for execution, through the concept of **Space Complexity** (SC)

Just like data structures, every algorithm has a specific set of pros and cons and they're chosen according to the requirements of the problem statement.



Why do we need Data Structures and Algorithms?

skill academy

The concept of data structures makes us think in terms of organising our data in terms of simplicity and efficiency. This way, we can come up with more elegant solutions to the problems we face.

Similarly, the concept of algorithms makes us think in terms of solving a problem in an efficient manner.

Consider the example of finding the meaning of a word in the dictionary. We unknowingly use the algorithm of binary search to find the word we're looking for. To facilitate this, the makers of the dictionary list down the words in a sorted order, which is the most efficient organisation of data (words).

Arrays

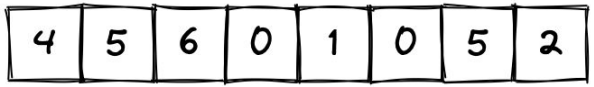
skill academy

Arrays are a linear data structure consisting of a collection of similar items. For e.g., an array of numbers would contain a collection of numbers.

We can store multiple items using Arrays. This means that we do not need to keep separate variables for multiple data items.

We can traverse the Array to read through all the items present in it, which is a good way of reading the data thus stored.

An Array requires a contiguous allocation of memory. It allows random access to the data inside it.



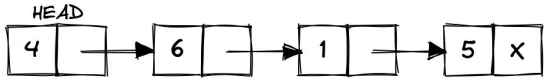
Linked Lists

skill academy

A **Linked List** is a linear data structure just like Arrays, but they're not required to be contiguously allocated in the memory. It consists of **Nodes** which hold the data and the reference to the next node in the memory.

Just like Arrays, we can store multiple items in a Linked List. The first node in the Linked List is called a **HEAD** node. It is the starting point

Linked Lists do not allow random access to the data inside it, hence the only way to access data at a particular node is to go through all the nodes from the HEAD node to that node.



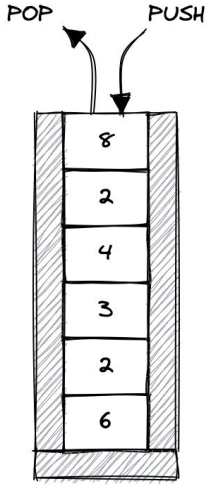
Stacks

skill academy

A **Stack** is a linear data structure which has a special property - we can only insert or delete items from one side. It can be represented either as an Array or a Linked List.

Inserting an item is termed as a **PUSH** operation while deleting an item is termed as a **POP** operation. The order of insertion or deletion of items follow **LIFO** or **Last In First Out**. The latest item thus pushed into the Stack is called the **TOP** of the Stack.

We cannot traverse the items in a Stack due to its unique behaviour. Hence, it is used in very specific use cases where such a behaviour is required.



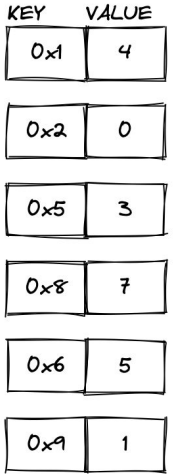
HashMap

skill academy

A **HashMap** is a data structure which follows a **key-value pair** system to store data. Each data item is assigned to a unique key and is referred to in this context as a value.

A new data item can be inserted into a HashMap by assigning it to a unique key. Setting a new data item on an existing key pair should either replace the existing value, or concatenate itself in the form of a list, depending on how the HashMap is implemented.

We cannot traverse a HashMap as it is not a linear data structure, unlike the ones we have discussed. We can **get** a value if we know the key, **set** a value to a key, or check if a key **exists** in the HashMap.

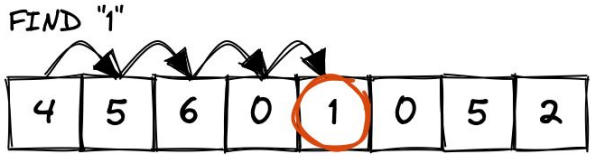


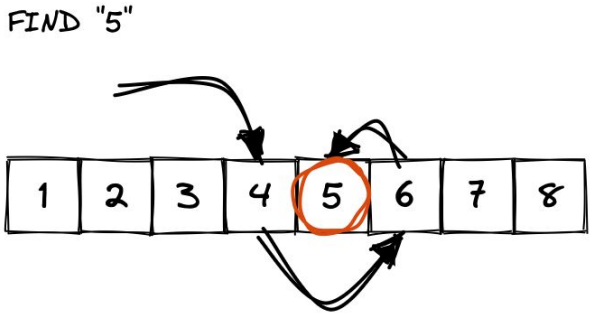
Searching

skill academy

Searching is an algorithm which, as the name suggests, is about finding a data item in a collection of data items. Searching can be done on linear as well as non-linear data structures, via different methods.

The most common use case of searching is in linear data structures, where we use **linear search** techniques on an Array or a Linked List, or **binary search** techniques on a sorted Array.



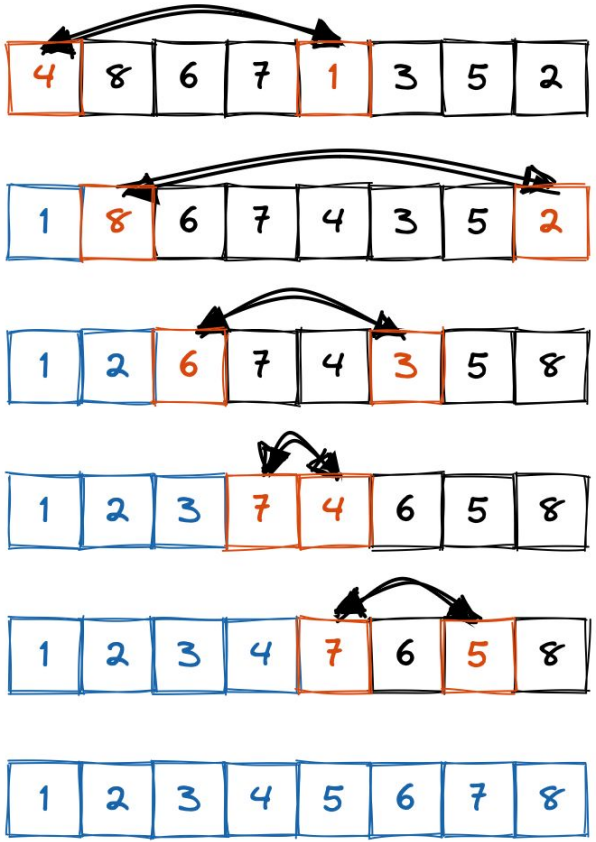


Sorting

skill academy

Sorting is an algorithm which, as the name suggests, is about re-arranging a collection of data items in a particular order. Sorting can be done on linear as well as non-linear data structures, via different methods.

There are a lot of sorting algorithms available with ranging Time and Space Complexity, and we choose the best one according to our needs. The slowest sorting algorithm is **Bubble Sort** and the fastest sorting algorithm is **Quick Sort**, and in some cases, **Counting Sort**.



Dynamic Programming

skill academy

Dynamic Programming is a way of breaking down a complex problem into simpler sub-problems, and building up the solution by solving and combining the solutions to the sub-problems.

Using Dynamic Programming, we solve each sub-problem just once, and then save the solution of the same in a table. This way, we can avoid the work of re-calculating the solution every time we encounter the same sub-problem.