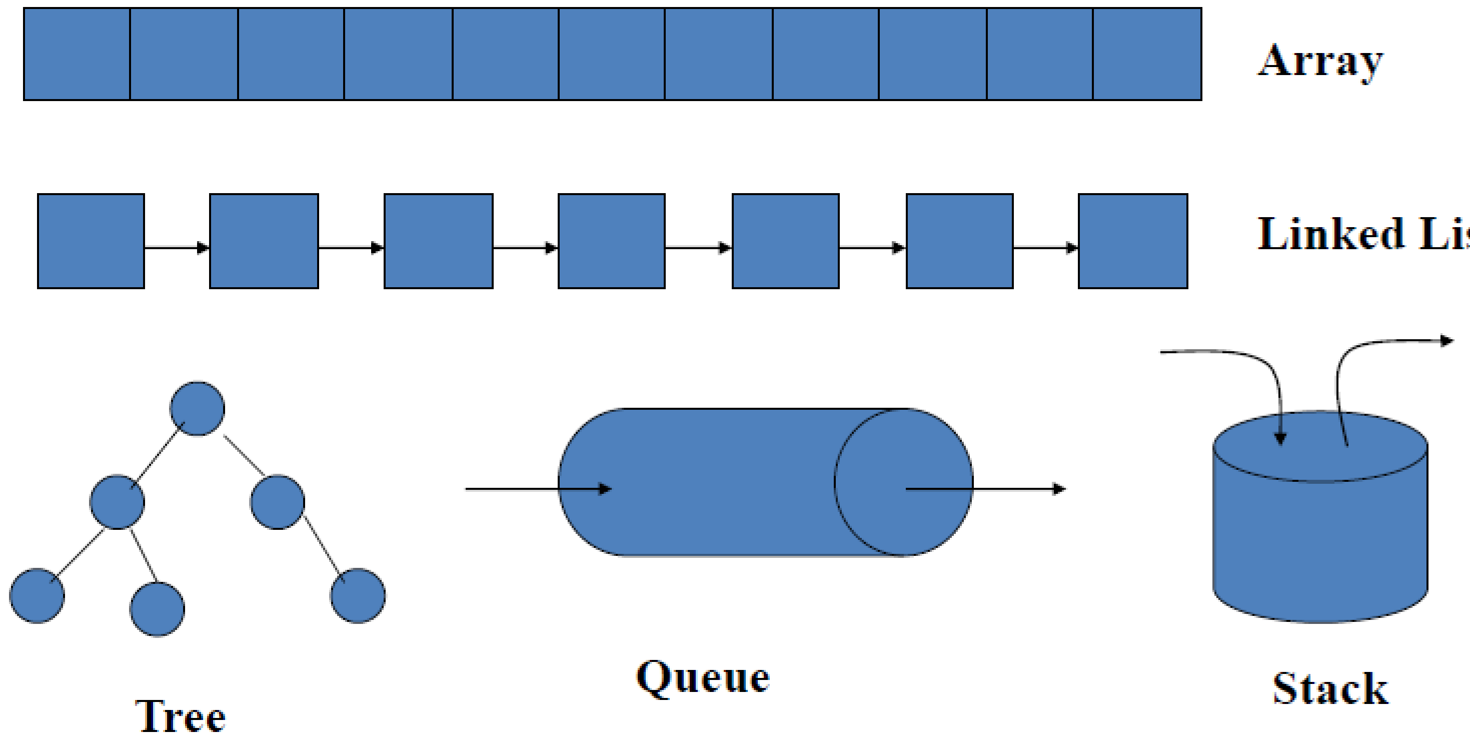
BML MUNJAL UNIVERSITY™

Data Structures & Algorithms

Selection of appropriate data structure for a given problem

Choosing the Right Data Structure to solve problems

Programs = Data Structures + Algorithms



Based on requirements, always pick the right data structure for the job.

Array

We can use an array in the following use cases.

- Need access to the elements using the index.
- Know the size of the array before defining the memory.
- Speed when iterating through all the elements in the sequence.
- The array takes less memory compare than a linked list.

Matrix

We can use Matrix in the following use cases.

- Matrix arithmetic in graphic processing algorithms.
- Represent the graph.

Singly Linked List

We can use the linked list in the following use cases.

- When the developer needs constant time for insertion and deletion.
- When the data dynamically grows.
- Do not access random elements from the linked list.
- Insert the element in any position of the list.

Doubly Linked List

We can use a doubly linked list in the following uses cases.

- Easier to delete the node from the doubly linked list.
- It can be iterated in reverse order without recursion implementation.
- Insert or remove from double-linked lists faster.

Stack

We can use the stack in the following use cases.

- Expression evaluation and syntax parsing.
- Runtime memory management.
- Recursive function.

Queue

We can use Queue in the following use cases.

- Use a queue when the developer wants an order.
- Processed in First In First Out order.
- If the developer wants to add or remove both ends, they can use the queue or a double-ended queue.

Binary Search Tree

We can use Binary Search Tree in the following use cases.

- Binary Search Trees are memory-efficient.
- Use when the data need to be sorted.
- Search can be done for a range of values.
- Height balancing helps to reduce the running time.

Heap

We can use Heap in the following use cases.

- Implement Priority Queue.
- Whenever the developer wants quick access to the largest (or smallest) item.
- Good for selection algorithms (finding the min or max).
- Operations tend to be faster than for a binary tree.
- Heap sort sorting methods being in-place and with no quadratic worst-case scenarios.

AVL tree

We can use AVL Tree in the following use cases.

- When the developer wants to control the tree height outside -1 to 1 range.
- Fast looking element.

B tree

We can use B-Tree and it’s variations in the following use cases.

- File systems.
- Database operations.

Graph

We can use Graph in the following use cases.

- Networks have many uses in the practical side of graph theory.
- Finding the shortest path between the cities.
- Find the optimized route between the cities.

Hashing

We can use a Hash table in the following use cases.

- Constant time operation.
- Hashing is used so that searching a database can be done more efficiently.

Example problem

Suggest choice of appropriate data structure to use for the following problem

An international cellphone company provides service on 7 different frequencies. They wish to set up business in Haryana and have fixed the locations of 100 towers for their new service. The company has to ensure that two towers broadcasting on the same frequency are at least 100 km apart, so that there is no interference of signals. Describe an algorithm which will answer the question “Is it feasible to set up towers at the given locations and provide service on 7 different frequencies?”. Your algorithm should say “feasible” if it is feasible, otherwise output the minimum number of frequencies needed to utilize all 100 towers.

To model this problem what data structure you should use?

Graphs