To apply any algorithm properly, it is very important that you select a proper data structure. It is because a particular operation performed on a data structure may take more time as compared to the same operation performed on another data structure.

**Example** – To access the $i^{th}$ element in a set by using an array, it may take a constant time but by using a linked list, the time required to perform the same operation may become a polynomial.

Therefore, the selection of a data structure must be done considering the architecture and the type of operations to be performed.

The following data structures are commonly used in parallel programming –

- Linked List
- Arrays
- Hypercube Network

**Linked List**

A linked list is a data structure having zero or more nodes connected by pointers. Nodes may or may not occupy consecutive memory locations. Each node has two or three parts – one **data part** that stores the data and the other two are **link fields** that store the address of the previous or next node. The first node’s address is stored in an external pointer called **head**. The last node, known as **tail**, generally does not contain any address.

There are three types of linked lists –

- Singly Linked List
- Doubly Linked List
- Circular Linked List

**Singly Linked List**

A node of a singly linked list contains data and the address of the next node. An external pointer called **head** stores the address of the first node.

![Singly Linked List Diagram](image)

**Doubly Linked List**

A node of a doubly linked list contains data and the address of both the previous and the next node. An external pointer called **head** stores the address of the first node and the external pointer called **tail** stores the address of the last node.

![Doubly Linked List Diagram](image)

**Circular Linked List**
A circular linked list is very similar to the singly linked list except the fact that the last node saved the address of the first node.

![Circular Linked List Diagram]

### Arrays

An array is a data structure where we can store similar types of data. It can be one-dimensional or multi-dimensional. Arrays can be created statically or dynamically.

- **In statically declared arrays**, dimension and size of the arrays are known at the time of compilation.
- **In dynamically declared arrays**, dimension and size of the array are known at runtime.

For shared memory programming, arrays can be used as a common memory and for data parallel programming, they can be used by partitioning into sub-arrays.

### Hypercube Network

Hypercube architecture is helpful for those parallel algorithms where each task has to communicate with other tasks. Hypercube topology can easily embed other topologies such as ring and mesh. It is also known as n-cubes, where \( n \) is the number of dimensions. A hypercube can be constructed recursively.

![Hypercube Diagrams]

**Figure (a) 1-cube**

**Figure (b) 2-cube**

**Figure (c) 3-cube**
Figure (d) 4-cube