**INTRODUCTION TO DATA STRUCTURES**

What is data structure?

Data structure is a method of organizing a large amount of data more efficiently so that any operation on that any operation on that data becomes easy.

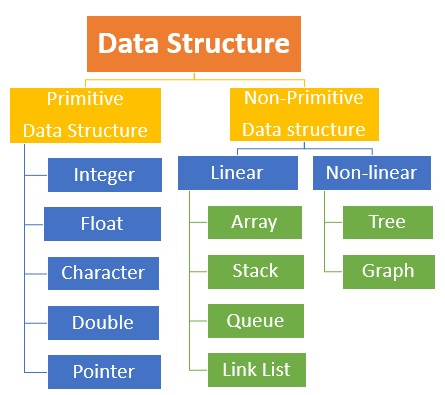
Why to Learn Data Structure and Algorithms?

1.Data Search

2.processor speed

3.multiple requests

Data structure classification



Why we use linked list?

* If we want to store the value in a memory, we need a memory manager that manages the memory for every variable.

**1.Linked list**

When we want to work with an unknown number of data values, we use a linked list data structure to organize that data.

Types of Linked list

1. Single Linked list

Single linked list is a sequence of elements in which every element has link to its next element in the sequence.

Graphical Representation



1. Double Linked list

Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence.

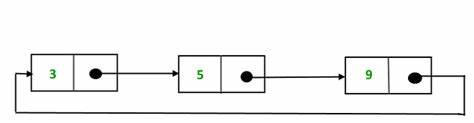
Graphical Representation



1. Circular Linked List

Circular Linked List is a type of linked list where list are linked in such a manner to form a circle i.e. **Last Node points the First Node**.

Graphical Representation



Operations on Linked list

1. **Create ( ):** It is used to create the node. ( **Refer the above creating a node concept**)
2. **Insertion( ):** it is used to insert the node at

Start ( Insertion at beginning )

Middle (Insertion after specified node)

End (Insertion at end of list)

1. **Deletion( ):** it is used to delete the node at

Start ( Deletion at beginning )

Middle (Deletion after specified node)

End (Deletion at end of list)

4. **Display( ):** it is used to display the nodes in the SLL

**5.Search( ):** it used to search particular

Advantages

* **Dynamic data structure:** A linked list is a dynamic arrangement so it can grow and shrink at runtime by allocating and [deallocating memory](https://www.geeksforgeeks.org/how-to-deallocate-memory-without-using-free-in-c/). So there is no need to give the initial size of the linked list.
* **No memory wastage:**  size of the linked list increase or decrease at run time so there is no memory wastage and there is no need to pre-allocate the In the Linked list, efficient memory utilization can be achieved since the memory.

Disadvantages

* **Memory usage:** More memory is required in the linked list as compared to an array. Because in a linked list, a [pointer](https://www.geeksforgeeks.org/pointers-in-c-and-c-set-1-introduction-arithmetic-and-array/) is also required to store the address of the next element and it requires extra memory for itself.
* **Traversal:** In a [Linked list traversal](https://www.geeksforgeeks.org/recursive-insertion-and-traversal-linked-list/) is more time-consuming as compared to an array. Direct access to an element is not possible in a linked list as in an array by index. For example, for accessing a node at position n, one has to traverse all the nodes before it.

Application

1. Implementation of [stacks](https://www.geeksforgeeks.org/stack-data-structure/) and [queues](https://www.geeksforgeeks.org/queue-data-structure/)
2. Implementation of graphs: [Adjacency list representation of graphs](https://www.geeksforgeeks.org/graph-and-its-representations/) is the most popular which uses a linked list to store adjacent vertices

**2.Stack**

A Stack is a linear data structure that follows the **LIFO (Last-In-First-Out)** principle. Stack has one end, whereas the Queue has two ends (**front and rear**).

operations

* **push():** When we insert an element in a stack then the operation is known as a push. If the stack is full then the overflow condition occurs.
* **pop():** When we delete an element from the stack, the operation is known as a pop. If the stack is empty means that no element exists in the stack, this state is known as an underflow state.
* **isEmpty():** It determines whether the stack is empty or not.
* **isFull():** It determines whether the stack is full or not.'
* **peek():** It returns the element at the given position.

Advantages

* **Simplicity:**Stacks are a simple and easy-to-understand data structure, making them suitable for a wide range of applications.
* **Efficiency:**Push and pop operations on a stack can be performed in constant time **(O(1))**, providing efficient access to data.

Disadvantages

* **Limited access:** Elements in a stack can only be accessed from the top, making it difficult to retrieve or modify elements in the middle of the stack.
* **Potential for overflow:**If more elements are pushed onto a stack than it can hold, an overflow error will occur, resulting in a loss of data.

Application

* **Function calls:** Stacks are used to keep track of the return addresses of function calls, allowing the program to return to the correct location after a function has finished executing.
* **Recursion:**Stacks are used to store the local variables and return addresses of recursive function calls, allowing the program to keep track of the current state of the recursion.

**3.Queue**

A queue is a linear data structure that follows the FIFO (First In First Out) principle. Elements are added at the rear and removed from the front.

**Operations**:

* **Enqueue**
* **Dequeue**
* **Peek**

**Space Complexity**: O(n)

**Advantages:**

* Simple and efficient for adding/removing elements in FIFO order.

**Disadvantages:**

* Limited access to elements (only the front and rear elements can be accessed directly).

**Applications:**

1. **Task Scheduling**: Queues can be used to schedule tasks based on priority or the order in which they were received.
2. **Resource Allocation:** Queues can be used to manage and allocate resources, such as printers or CPU processing time.

**4.Trees**

A collection of elements with Last In First Out (LIFO) access.

Advantages

* **Efficient searching**
* **Flexible size**
* **Easy to traverse**.
* **Easy to maintain**
* **Natural organization**
* **Fast insertion and deletion**

Disadvantages

* **Memory overhead**
* **Imbalanced trees**
* **Complexity**
* **Limited flexibility**
* **Inefficient for certain operations**

Application

* **File Systems**
* **XML Parsing**
* **Database Indexing**
* **Compiler Design**
* **Artificial Intelligence**

**5.Graphs**

A tree is a hierarchical data structure consisting of nodes, with a single node designated as the root. Each node has zero or more child nodes.

Advantages

* **Representing complex data**
* **Efficient data processing**
* **Network analysis**
* **Pathfinding**
* **Visualization**
* **Machine** **learning**

Disadvantages

* **Limited representation**
* **Difficulty in interpretation**
* **Scalability issue**s
* **Data quality issues**
* **Lack of standardization**
* **Privacy concerns**

Application

* **Social media analysis**
* **Network monitoring**
* **Financial trading**
* **Internet of Things (IoT) management**
* **Autonomous vehicles**