**Linear Data Structures (Revision)**

Abstract data types, such as stacks and queues, allow more sophisticated organisation and manipulation of data, and provide methods for adding, removing, and traversing data.

**Stacks**

A stack is an abstract data type that holds an ordered, linear sequence of items. In contrast to a queue, a stack is a last in, first out (LIFO) structure.

In a stack data structure, you can only access the element on the top of the stack. The element that was added last will be the one to be removed first. Therefore, to implement a stack, you need to maintain a pointer to the top of the stack (the last element to be added).

**The main stack operations are:**

|  |  |
| --- | --- |
| push(data) | adds an element to the top of the stack |
| pop() | removes an element from the top of the stack |
| peek() | returns a copy of the element on the top of the stack without removing it |
| is\_empty() | checks whether a stack is empty |
| is\_full() | checks whether a stack is at maximum capacity when stored in a static (fixed-size) structure |

**Queues**

A queue is an abstract data type that holds an ordered, linear sequence of items. You can describe it as a first in, first out (FIFO) structure. New elements are added to the back or rear of the queue. When an element is removed, the remaining elements do not move up to take the empty space.

A priority queue is one where each element in the queue has a priority. When new elements are added to the queue, they are inserted ahead of those of lower priority and behind elements of equal priority.

**The main operations on a queue are:**

|  |  |
| --- | --- |
| enqueue(data) | add an element to the queue |
| dequeue() | return an element from the front of the queue |
| is\_empty() | check whether the queue is empty |
| is\_full() | check whether the queue is at maximum capacity (if the size of the queue is constrained) |

**Linked Lists**

A linked list is a dynamic data structure, which means that the size of the list can change at run time. Each element in a linked list is called a node. Each node stores:

* The data relating to the element
* A pointer to the next node

There is also a separate pointer that indicates the first element in the list (the head of the list). This has a null value when the list is empty. The next node pointer of the last element in the list always points to a null value to mark the end of the list. An additional pointer can be used to indicate the last element of the list (the tail of the list). This is useful when implementing a queue using a linked list so that the end of the list can be accessed without needing to traverse the list.

**Exam Questions**

1. **What is a Linked List and What are its types?**

A linked list is a linear data structure (like arrays) where each element is a separate object. Each element (that is node) of a list is comprised of two items – the data and a reference to the next node.

Types of Linked List:

* Singly Linked List
* Doubly Linked List
* Circular Linked List

1. **Which data structures are used for BFS and DFS of a graph?**

* Queue is used for BFS
* Stack is used for DFS. DFS can also be implemented using recursion (Note that recursion also uses function call stack).

1. **Can doubly linked be implemented using a single pointer variable in every node?**

A doubly linked list can be implemented using a single pointer.

1. **Which Data Structure Should be used for implementing LRU cache?**

* Queue which is implemented using a doubly linked list. The maximum size of the queue will be equal to the total number of frames available (cache size). The most recently used pages will be near the rear end and the least recent pages will be near the front end.
* A Hash with page number as key and address of the corresponding queue node as value

1. **How to check if a given Binary Tree is BST or not?**

If in order traversal of a binary tree is sorted, then the binary tree is BST. The idea is to simply do in order traversal and while traversing keeps track of previous key value. If current key value is greater, then continue, else return false.

1. **What is a Queue, how it is different from the stack and how is it implemented?**

Queue is a linear structure that follows the order is First In First Out (FIFO) to access elements. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added. Both Queues and Stacks can be implemented using Arrays and Linked Lists.

1. **What is Stack and where it can be used?**

Stack is a linear data structure in which the order LIFO(Last In First Out) or FILO(First In Last Out) for accessing elements

Applications of Stack:

* Infix to Postfix Conversion using Stack
* Evaluation of Postfix Expression
* Reverse a String using Stack
* Implement two stacks in an array
* Check for balanced parentheses in an expression

1. **What are linear and non-linear data Structures?**

* Linear: A data structure is said to be linear if its elements form a sequence or a linear list. Examples: Array. Linked List, Stacks and Queues
* Non-Linear: A data structure is said to be non-linear if the traversal of nodes is nonlinear in nature. Example: Graph and Trees.