What is Static Data structure or Dynamic Data Structure?

In Static data structure the size of the structure is fixed.

The content of the data structure can be modified but without changing the memory space allocated to it. Ex: Array

In Dynamic data structure the size of the structure in not fixed and can be modified during the operations performed on it. Ex: Linked List

---

Array:

An array is collection of elements of same data type and stores data elements in a continuous memory location.

The elements of an array are accessed by using an index.

In array, elements are stored in the consecutive location,

array of size N

Range 0 to N-1

What is an array?

Answer: An array is a data structure consisting of a collection of elements, each identified by at least one array index or key.

Can an array be resized at runtime?

Answer: In some programming languages, arrays can be resized dynamically, while in others, such as C, the size is fixed.

What is the time complexity for accessing an element in an array?

Answer: The time complexity for accessing an element in an array is **O(1),** as it can be accessed directly using its index.

What is the difference between an array and a linked list?

Answer: An array is a static data structure, while a linked list is a dynamic data structure. Arrays have a fixed size, and elements are stored consecutively in memory, while linked lists can grow and do not require contiguous memory allocation.

How would you find the smallest and largest element in an array?

Answer: To find the smallest and largest elements in an array, one common approach is to iterate through the array and keep track of the smallest and largest elements encountered so far.

Linked Lst:

Linked List is a linear data structure that stored the element in the form of a node where each node contains a data and a reference(link) to the next node in the list.

Unlike arrays, linked list elements are not stored at a contiguous location; the elements are linked using pointers.

Advantages:

1)Dynamic data structure - The size of the linked list may vary according to the requirements. Linked list does not have a fixed size.

2)Insertion and deletion - Unlike arrays, insertion, and deletion in linked list is easier.

In array, elements are stored in the consecutive location, whereas the elements in the linked list are stored at a random location.

To insert or delete an element in an array, we have to shift the elements for creating the space.

Whereas, in linked list, instead of shifting, we just have to update the address of the pointer of the node.

3)Memory efficient - The size of a linked list can grow or shrink according to the requirements, so memory consumption in linked list is efficient.

4)Implementation - We can implement both stacks and queues using linked list.

Disadvantages of Linked Lists:

Slow random access

More memory overhead

Difficult to debug

Not cache-friendly

Can suffer from memory leaks

Singly Linked List : In this type of linked list, every node stores address or reference of the next node in the list and the last node has next address or reference as NULL.

For example 1->2->3->4->NULL

Doubly Linked List : Here, here are two references associated with each node, One of the reference points to the next node and one to the previous node.

Eg. NULL<-1<->2<->3->NULL

Circular Linked List : Circular linked list is a linked list where all nodes are connected to form a circle.

There is no NULL at the end. A circular linked list can be a singly circular linked list or a doubly circular linked list.

Eg. 1->2->3->1 [The next pointer of the last node is pointing to the first]

The time **complexity** of common operations on a singly-linked list are as follows:

Insertion:

At the beginning: O(1)

At the end: O(n)

At a specific position: O(n)

Deletion:

At the beginning: O(1)

At the end: O(n)

At a specific position: O(n)

Search: O(n)

Traversal: O(n)

Stacks(storing the chairs):

Stacks are dynamic data structures for storing the data that follow the Last In First Out (LIFO) or First In Last Out (FILO). principle.

It contains only one pointer top pointer pointing to the topmost element of the stack.

It is defined as a container in which insertion and deletion can be done from the one end known as the top of the stack.

A Stack is an abstract data type with a pre-defined capacity, which means that it can store the elements of a limited size.

Basic Operations on Stack

push() to insert an element into the stack

pop() to remove an element from the stack

top() Returns the top element of the stack.

isEmpty() returns true is stack is empty else false

size() returns the size of stack

Implement Stack using Queues

Stack can be implemented using two queues.

to implement the push(s, x) operation:

1.Enqueue x to q2.

2.One by one dequeue everything from q1 and enqueue to q2.

3.Swap the queues of q1 and q2.

In Brief:

Add x to q2

q1->q2

swap

to implement the pop(s) operation:

Dequeue an item from q1 and return it.

Queue:

https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/

Queue is a linear data structure for storing data that follows a particular order First In First Out (FIFO).

Basic Operations on Queue:

enqueue(): Inserts an element at the end of the queue i.e. at the rear end.

dequeue(): This operation removes and returns an element that is at the front end of the queue.

front(): This operation returns the element at the front end without removing it.

rear(): This operation returns the element at the rear end without removing it.

Empty(): This operation indicates whether the queue is empty or not.

size(): This operation returns the size of the queue i.e. the total number of elements it contains.

Implementation Queue using Stacks

enQueue(q, x):

While stack1 is not empty, push everything from stack1 to stack2.

Push x to stack1 (assuming size of stacks is unlimited).

Push everything back to stack1.

In Brief:

s1->s2

x->s1

s2->s1

deQueue(q):

If stack1 is empty then error

Pop an item from stack1 and return it

Heap:

Heap is also a data structure or memory used to store the global variables.

By default, all the global variables are stored in the heap memory. It allows dynamic memory allocation.

The heap memory is not managed by CPU. Heap data structure can be implemented either using arrays or trees.

Hash:

A hash table is a data structure which is used to store key-value pairs.

Hash function is used by hash table to compute an index into an array in which an element will be inserted or searched.

Basic Operations:

HashTable: This operation is used in order to create a new hash table.

Delete: This operation is used in order to delete a particular key-value pair from the hash table. O(1) i.e. constant time.

Get: This operation is used in order to return the value that is associated with that key.O(1) i.e. constant time.

Put: This operation is used in order to insert a new key-value pair inside the hash table. O(1) i.e. constant time.

DeleteHashTable: This operation is used in order to delete the hash table

https://www.geeksforgeeks.org/hashing-set-1-introduction/

Tree--

A tree is a popular data structure that is non-linear and hierarchical structure in nature.

Other data structures like array, stack, queue, and linked list which are linear in nature.

The ordering information of a tree is not important.

A tree contains nodes and 2 pointers. These two pointers are the left child and the right child of the parent node.

Let us understand the terms of tree in detail.

Root: The root of a tree is the topmost node of the tree that has no parent node. There is only one root node in every tree.

Edge: Edge acts as a link between the parent node and the child node.

Leaf: A node that has no child is known as the leaf node. It is the last node of the tree. There can be multiple leaf nodes in a tree.

Depth: The depth of the node is the distance from the root node to that particular node.

Height: The height of the node is the distance from that node to the deepest node of the tree.

Height of tree: The Height of the tree is the maximum height of any node.

Why we use?

Trees provide moderate access/search (quicker than Linked List and slower than arrays).

Trees provide moderate insertion/deletion (quicker than Arrays and slower than Unordered Linked Lists).

Application of trees

a Manipulate hierarchical data

Make information easy to search (see tree traversal)

Manipulate sorted lists of data

Binary Tree Traversals:

PreOrder Traversal: Here, the traversal is: root – left child – right child. It means that the root node is traversed first then its left child and finally the right child.

InOrder Traversal: Here, the traversal is: left child – root – right child. It means that the left child is traversed first then its root node and finally the right child.

PostOrder Traversal: Here, the traversal is: left child – right child – root. It means that the left child is traversed first then the right child and finally its root node.

Time complexity: O(n)

Graph

Basically, graph is a type of non-linear data structure which is the combination of nodes(vertices) and edges.Nodes are connected with the help of edges.

Types:::>>

1) Directed Graph: A graph in which edge has direction.(Twitter)

2) Undirected Graph: A graph in which edges do not have any direction.(Facebook)

3) Connected Graph: The graph in which from one node we can visit any other node in the graph is known as a connected graph.

4) Disconnected Graph: The graph in which at least one node is not reachable from a node is known as a disconnected graph.

5) Regular Graph: The graph in which the degree of every vertex is equal to the other vertices of the graph.

6) Complete Graph: The graph in which from each node there is an edge to each other node.

7) Cycle Graph: The graph in which the graph is a cycle in itself, the degree of each vertex is 2.

8) Cyclic Graph: The graph in which contain at least one cycle is known as a Cyclic graph.

9)Directed Acyclic Graph: A Directed Graph that does not contain any cycle.

10) Bipartite Graph: A graph in which vertex can be divided into two sets such that vertex in each set does not contain any edge between them.

11) Weighted Graph: A graph in which the edges are already specified with suitable weight is known as a weighted graph.

12) Null Graph: A graph is known as a null graph if there are no edges in the graph.

13) Trivial Graph: Graph having only a single vertex, it is also the smallest graph possible.

Trees are the restricted types of graphs, just with some more rules.

Every tree will always be a graph but not all graphs will be trees.

Linked List, Trees, and Heaps all are special cases of graphs.

What is stack unwinding?

When we call some functions, it stores the address into call stack.

The stack unwinding is a process where the function call stack entries are removed at runtime.

To remove stack elements, we can use exceptions. If an exception is thrown from the inner function,

then all of the entries of the stack is removed, and return to the main invoker function.

Tell me about Exceptional Handling.

One of the advantages of C++ over C is Exception Handling.

Exceptions can handle at runtime bcoz it occurs due to some unexpected conditions.

to handle runtime errors.

exception handling is performed using try/catch statement.

try block is used to place the code that may occur exception.

The catch block is used to handle the exception.

What are VTABLE and VPTR?

The vtable is a table of function pointers. It is maintained per class. vptr is a pointer to vtable.

It is maintained per object (See this for an example). The compiler adds additional code at two places to maintain and use vtable and vptr.

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

Queue is used for BFS

(Breadth-first search is an algorithm for searching a node in tree data structure.

The Queue data structure is used for the Breadth First Search traversal. When we use the BFS algorithm for the traversal in a graph,

we can consider any node as a root node.

DFS::>

Depth-first search is an algorithm for traversing or searching tree or graph data structures.

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

It is a recursive algorithm to search all the vertices of a tree data structure or a graph.

The depth-first search (DFS) algorithm starts with the initial node of graph G and goes deeper until we find the goal node or the node with no children.

What is this pointer?

'this' pointer is a constant pointer that holds the memory address of the current object.
