Prac qs

Introduction to ds

What is a ds

A **data structure** is a particular way of organizing data in a computer so that it can be used effectively.

Diff between calloc and malloc and dynamic memory allocation

Therefore, C **Dynamic Memory Allocation** can be defined as a procedure in which the size of a data structure (like Array) is changed during the runtime.

The name **malloc** and **calloc()** are library functions that allocate memory dynamically. It means that memory is allocated during runtime(execution of the program) from the heap segment.

- **Initialization:** malloc() allocates memory block of given size (in bytes) and returns a pointer to the beginning of the block. malloc() doesn't initialize the allocated memory. If we try to access the content of memory block(before initializing) then we'll get segmentation fault error(or maybe garbage values).

calloc() allocates the memory and also initializes the allocated memory block to zero. If we try to access the content of these blocks then we'll get 0.

free () method in C is used to dynamically **de-allocate** the memory. The memory allocated using functions malloc() and calloc() is not de-allocated on their own. Hence the free() method is used, whenever the dynamic memory allocation takes place. It helps to reduce wastage of memory by freeing it.

"realloc" or **"re-allocation"** method in C is used to dynamically change the memory allocation of a previously allocated memory. In other words, if the memory previously allocated with the help of malloc or calloc is insufficient, realloc can be used to **dynamically re-allocate memory**. re-allocation of memory maintains the already present value and new blocks will be initialized with default garbage value.

What is adt

Adt is description of data structure from user point of view. It tells only about what the data structure does and doesnt tell how it is done.

Linear and non linear ds

	Linear	Non Linear
1.	In a linear data structure, data elements are arranged in a linear order where each and every elements are attached to its previous and next adjacent.	In a non-linear data structure, data elements are attached in hierarchically manner.
2.	In linear data structure, single level is involved.	Whereas in non-linear data structure, multiple levels are involved.
3.	Its implementation is easy in comparison to non-linear data structure.	While its implementation is complex in comparison to linear data structure.
4.	In linear data structure, data elements can be traversed in a single run only.	While in non-linear data structure, data elements can’t be traversed in a single run only.
5.	In a linear data structure, memory is not utilized in an efficient way.	While in a non-linear data structure, memory is utilized in an efficient way.

6.	Its examples are: array, stack, queue, linked list, etc.	While its examples are: trees and graphs.
7.	Applications of linear data structures are mainly in application software development.	Applications of non-linear data structures are in Artificial Intelligence and image processing.

Linked list

Adt of linked list

Linked list consists of a series of structures. They are not required to be stored in adjacent memory locations.Each structure consists of data and address field.Address field consists of address of the successor.Structures can be used to define nodes in C.

Syntax:

```
Typedef struct node{
Int data;
Struct node *next;
}
```

It is a self referential structure in C. Memory is acquired during runtime.The memory is freed after the use of that variable is over. Memory can be acquired by using malloc and calloc.Linked list is known by address of the head node. Address is stored in a head pointer variable. Insertion requires changing values of two pointers and obtaining new node. Deletion can be performed in one pointer change.

Applications of LL

1. *Image viewer* – Previous and next images are linked, hence can be accessed by next and previous button.
2. *Previous and next page in web browser* – We can access previous and next url searched in web browser by pressing back and next button since, they are linked as linked list.
3. *Music Player* – Songs in music player are linked to previous and next song. you can play songs either from starting or ending of the list.

Types of Linked list

1. Singly linked list: two successive nodes are linked together with each other in a sequential linear manner.
2. Doubly ll: each node has 2 address fields and holds addresses of next as well as preceding nodes
3. Circular ll: first and last elements are adjacent. Address of first node is stored in next field of last node

Queue

Adt for queue

Queue is a ds which follows the principle of FIFO - first in first out. It is an ordered collection of data where insertion is done through the rear counter and deletion is done using the front counter.
A q can be implemented using array or linked list(dynamic memory allocation so no problems in over or under utilization of memory).
Operations :
Enqueue
Dequeue
Front

Why is circular q needed

In a Linear queue, once the queue is completely full, it's not possible to insert more elements.Even if we dequeue the queue to remove some of the elements, until the queue is reset, no new elements can be inserted.
When we **dequeue** any element to remove it from the queue, we are actually moving the **front** of the queue forward, thereby reducing the overall size of the queue. And we cannot insert new elements, because the **rear** pointer is still at the end of the queue.
Circular Queue is also a linear data structure, which follows the principle of **FIFO**(First In First Out), but instead of ending the queue at the last position, it again starts from the first position after the last, hence making the queue behave like a circular data structure.

Applications of cq

1. Computer controlled **Traffic Signal System** uses circular queue.
2. CPU scheduling and Memory management

Applications of Queue

Queue, as the name suggests is used whenever we need to manage any group of objects in an order in which the first one coming in, also gets out first while the others wait for their turn, like in the following scenarios:

Serving requests on a single shared resource, like a printer, CPU task scheduling etc.

In real life scenario, Call Center phone systems uses Queues to hold people calling them in an order, until a service representative is free.

Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive i.e First come first served.

Difference between lifo fifo

FIFO	LIFO
It stands for First-In-First-Out approach in programming.	It stands for Last-In-First-Out approach in programming.
In this, the new element is inserted below the existing element, So that the oldest element can be at the top and taken out first.	In this, the new element is inserted above the existing element, So that the newest element can be at the top and taken out first.
Therefore, the First element to be entered in this approach, gets out First.	Therefore, the First element to be entered in this approach, gets out Last.
In computing, FIFO approach is used as an operating system algorithm, which gives every process CPU time in the order they arrive.	In computing, LIFO approach is used as a queuing theory that refers to the way items are stored in types of data structures.
The data structure that implements FIFO is Queue.	The data structure that implements LIFO is Stack.

Priority q

In computer science, a priority queue is an abstract data type similar to a regular queue or stack data structure in which each element additionally has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority.

Priority Queue is an extension of queue with following properties.

1. Every item has a priority associated with it.
2. An element with high priority is dequeued before an element with low priority.
3. If two elements have the same priority, they are served according to their order in the queue.

Applications of Priority Queue:

1) CPU Scheduling

2) Graph algorithms like Dijkstra's shortest path algorithm, Prim's Minimum Spanning Tree, etc

Applications of Priority Queue

- Prim's algorithm implementation can be done using priority queues.
- Dijkstra's shortest path algorithm implementation can be done using priority queues.
- A* Search algorithm implementation can be done using priority queues.
- Priority queues are used to sort heaps.
- Priority queues are used in operating system for load balancing and interrupt handling.
- Priority queues are used in huffman codes for data compression.
- In traffic light, depending upon the traffic, the colors will be given priority.

Stack

Adt of stack

Stack is a lifo data structure.it is an ordered list of same type of elements. It is a linear list where all insertions and deletions are possible from only one end of stack namely front end of list known as top. When elements are added to stack it grows at one end and when elements are deleted it shrinks from the same end.

Structure used for stack:-

```
Typedef struct stack{

Int data[size];

Int top;

}
```

Operations on stack are:-

1. initialize()
2. empty()
3. full()
4. push()
5. pop()
6. peek()

Applications of stack

Infix to Postfix or Infix to Prefix Conversion − and evaluation

Parenthesis checking

Function call and return process

Backtracking problems - n queen problem , dfs,

Infix postfix and prefix

Infix: expressions are evaluated from left to right but operator precedence must be taken into consideration.they are not used for representation inside a computer

Prefix: An expression is called the prefix expression if the operator appears in the expression before the operands.

Postfix: An expression is called the postfix expression if the operator appears in the expression after the operands.

Graphs

Applications graph

In **Computer science** graphs are used to represent the flow of computation.

Facebook for linking friends -- undirected

Google maps

World wide web - -directed graph

Why is stack used in dfs

Depth First Search (DFS) algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.
Helps in backtracking to the previous vertex and explore further.

Why is q in bfs
Queue data structures are considered inherently "fair". The FIFO concept that underlies a Queue will ensure that those things that were *discovered first* will be *explored first*, before exploring those that were discovered subsequently.

Difference between bfs and dfs

	bfs	dfs
1.	BFS stands for Breadth First Search.	DFS stands for Depth First Search.
2.	BFS(Breadth First Search) uses Queue data structure for finding the shortest path.	DFS(Depth First Search) uses Stack data structure.

3.	BFS can be used to find single source shortest path in an unweighted graph, because in BFS, we reach a vertex with minimum number of edges from a source vertex.	In DFS, we might traverse through more edges to reach a destination vertex from a source.
3.	BFS is more suitable for searching vertices which are closer to the given source.	DFS is more suitable when there are solutions away from source.
4.	BFS considers all neighbors first and therefore not suitable for decision making trees used in games or puzzles.	DFS is more suitable for game or puzzle problems. We make a decision, then explore all paths through this decision. And if this decision leads to win situation, we stop.
5.	The Time complexity of BFS is O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges.	The Time complexity of DFS is also O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges.

Trees

What is bst

Binary Search Tree is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys lesser than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- The left and right subtree each must also be a binary search tree.

Applications of bst

1. used to efficiently store data in sorted form in order to access and search stored elements quickly
2. They can be used to represent arithmetic expressions (Refer here 408 for more info)
3. BST used in Unix kernels for managing a set of virtual memory areas (VMAs).

Types of traversals

1. Preorder: root left right
2. Inorder: left root right
3. Postorder: left right root

What is tree

A tree is a nonlinear hierarchical data structure that consists of nodes connected by edges.

Root: special node in a tree and the entire tree is referenced through it.This node does not have a parent.

Parent: Immediate predecessor of node

Child: All immediate successors of a node

Siblings: Nodes with the same parent and on the same level are siblings

Path: Number of source nodes from source node to destination node

Leaf Node: A node is a leaf node if both left and right child nodes of it are NULL.

Graph and tree

NO.	GRAPH	TREE
1	Graph is a non-linear data structure.	Tree is a non-linear data structure.
2	It is a collection of vertices/nodes and edges.	It is a collection of nodes and edges.
3	Each node can have any number of edges.	General trees consist of the nodes having any number of child nodes. But in case of binary trees every node can have at the most two child nodes.
4	There is no unique node called root in graph.	There is a unique node called root in trees.
5	A cycle can be formed.	There will not be any cycle.

6	Applications: For finding shortest path in networking graph is used.	Applications: For game trees, decision trees, the tree is used.

Avl trees

AVL tree is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees cannot be more than one for all nodes.

Hashing

Hashing is an important Data Structure which is designed to use a special function called the Hash function which is used to map a given value with a particular key for faster access of elements. The efficiency of mapping depends of the efficiency of the hash function used.

Collisions

A situation when the resultant hashes for two or more data elements in the data set *U*, maps to the same location in the hash table, is called a hash collision. In such a situation two or more data elements would qualify to be stored/mapped to the same location in the hash table.

Probing

2 types:

1. Linear Probing: A **hash** collision occurs when the **hash** function maps a key into a cell that is already occupied by a different key. Linear **probing** is a strategy for resolving collisions, by placing the new key into the closest following empty cell.
2. Quadratic Probing: Quadratic probing is an open-addressing scheme where we look for i^2'th slot in i'th iteration if the given hash value x collides in the hash table.