DATA STRUCTURES AND ALGORITHMS

TOPIC: Melodic Stacks in Music Players

By: BHASWAT LENKA[RA2211026010522]

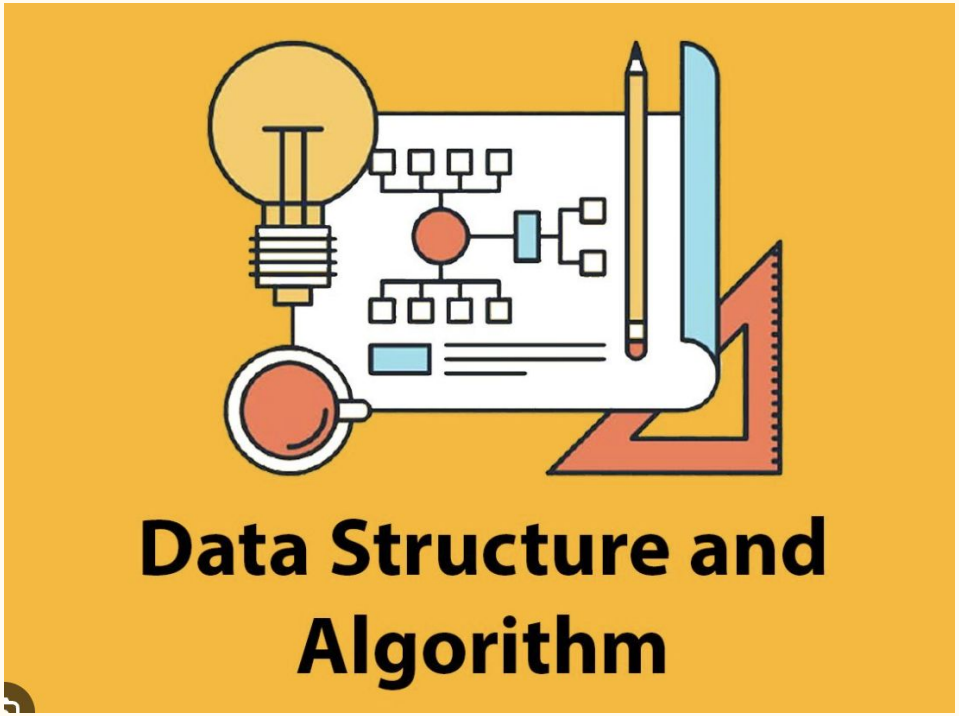
SHASHANKA NIRAULA[RA2211026010528]

WHAT IS DATA STRUCTURES?

Simply put, they are essential in today's computer science world. It refers to the format comprising an assortment of data values, relationships, and functions applied to data. Data structures can arrange data such that it gets accessed & worked on with specified algorithms.

Data Structures and Their Classification:

In the universe of computer science, data structures are used for arranging data in memory. They are essential in organizing, accessing, processing, and storing data. Besides, different types of data structures have their characteristics, data structures, applications, features, pros, and cons.
They have various uses in our daily life. Some of them are used for solving mathematical problems. Implementing their use can assist in organizing and processing a large amount of data in a short span. Here's a rundown of each type and application of data structures in the following pointers:



LINEAR DATA STRUCTURE

A linear data structure is a type of data organization where elements are stored sequentially, with each element having a unique predecessor and successor, except for the first and last elements. This linear arrangement facilitates easy traversal, insertion, and deletion of elements. Common examples of linear data structures include arrays, linked lists, stacks, and queues.

1)ARRAYS: store elements of the same data type in contiguous memory locations, making them efficient for random access.

2)LINKED LIST: consist of nodes connected by pointers, allowing for dynamic memory allocation and easy insertion/deletion but slowing down random access.

3)STACKS: follow the Last-In-First-Out (LIFO) principle, and are used for tasks like function calls and managing data temporarily.

4)QUEUES: adhere to the First-In-First-Out (FIFO) principle and are suitable for tasks like scheduling and managing tasks in a linear order.

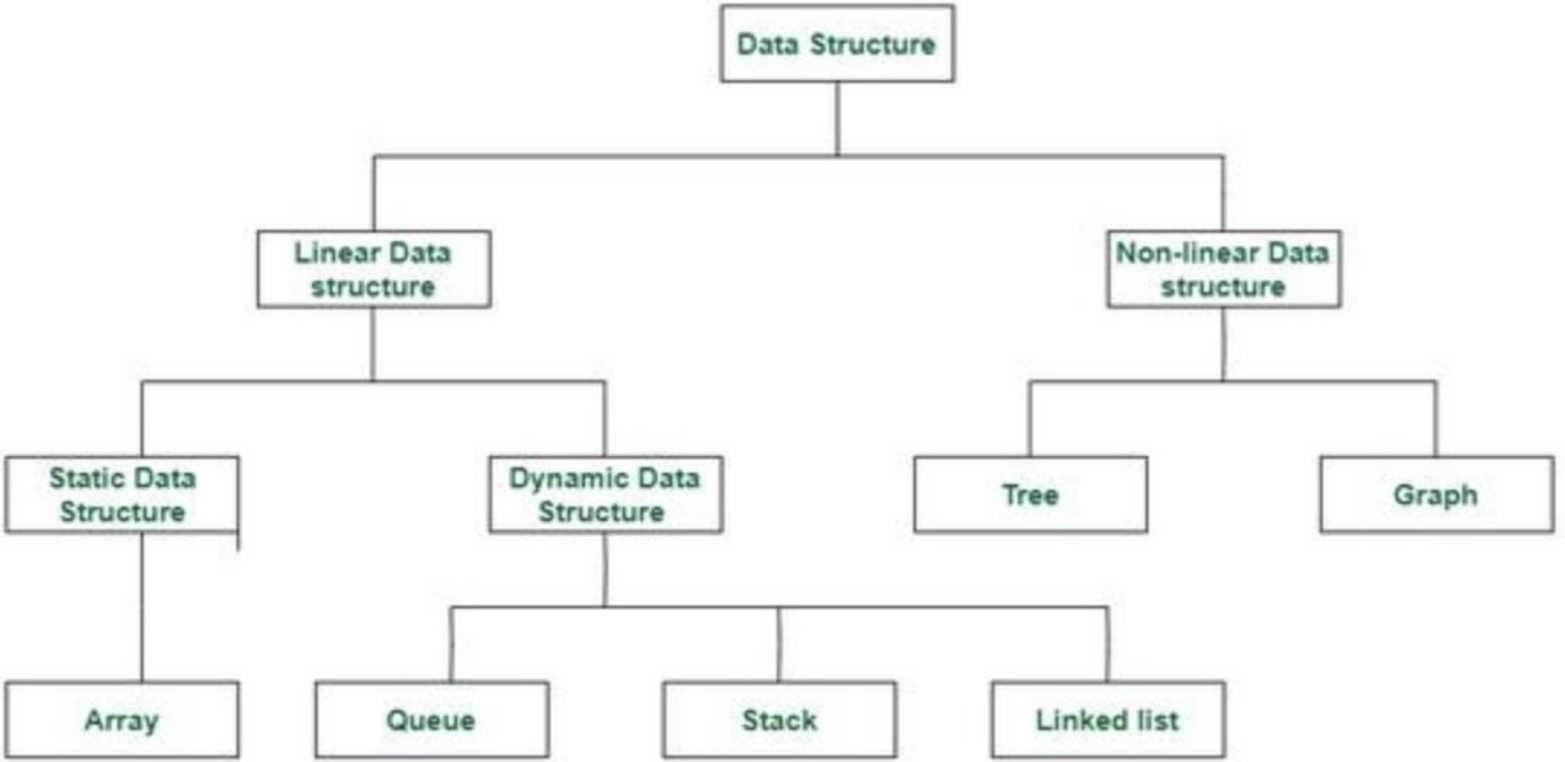
NON-LINEAR DATA STRUCTURE

In contrast, non-linear data structures do not organize elements in a sequential manner. Instead, they establish relationships between elements that can be hierarchical or network-based, enabling more complex data representations. Common non-linear data structures include trees and graphs.

1) TREES: are hierarchical structures with a root node and child nodes, and are used for tasks like organizing data in file systems or implementing search algorithms (e.g., binary search trees).
2) GRAPHS: are versatile structures that consist of nodes and edges, allowing for complex connections and relationships between data points. Graphs are essential for modeling social networks, transportation systems, and more

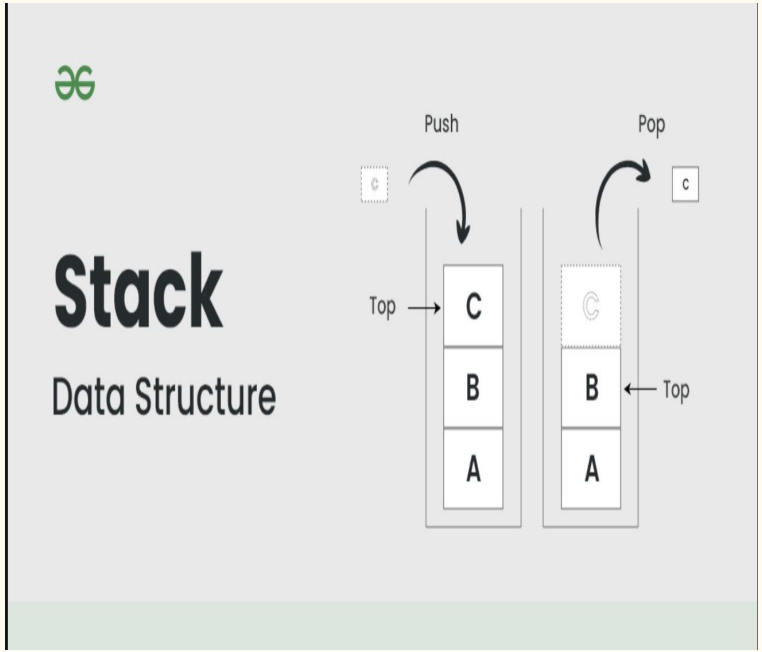
Non-linear data structures enable the representation of complex relationships and are vital in solving many real-world problems efficiently.

Classification of Data Structure



Classification of Data Structure

STACK



Stack is a linear data structure that follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out). LIFO implies that the element that is inserted last, comes out first and FILO implies that the element that is inserted first, comes out last.

There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)/FILO(First In Last Out) order.

PROBLEM STATEMENT:

Problem Statement: Develop a music player application with "Melodic Stacks" to efficiently manage multiple playlists. Using a stack data structure, this feature enables users to create, switch, and manipulate playlists of up to 1000 songs each. Users can add, remove, and shuffle songs within the active playlist, switch between playlists instantly, and view the current song title without removing it. The system must ensure that songs are not repeated consecutively during shuffle. The challenge is to extend the existing code to provide a user-friendly and feature-rich music player for seamless playlist management while meeting the specified constraints.

CODE:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_SONGS 100

#define MAX_SONG_TITLE_LENGTH 50

// Define a structure to represent a song

typedef struct {

    char title[MAX_SONG_TITLE_LENGTH];

} Song;

// Define the stack data structure

typedef struct {

    Song songs[MAX_SONGS];

    int top;

} Stack;

// Initialize the stack

void initialize(Stack *stack) {

    stack->top = -1;

}

// Push a song onto the stack

void push(Stack *stack, const char *title) {

    if (stack->top < MAX_SONGS - 1) {

        stack->top++;
strcpy(stack->songs[stack->top].title, title);

        printf("Added '%s' to the playlist.\n", title);

    } else {

        printf("Stack is full. Cannot add more songs.\n");

    }

}
```

CODE:

```
// Pop a song from the stack and play it
void play(Stack *stack) {
    if (stack->top >= 0) {
        printf("Now playing: '%s'\n",
stack->songs[stack->top].title);
        stack->top--;
} else {
        printf("No more songs in the playlist.\n");
    }
}
int main() {
Stack playlist;
initialize(&playlist);
    printf("Welcome to the Music Player!\n");
    while (1) {
        printf("\nMenu:\n");
        printf("1. Add a song to the playlist\n");
        printf("2. Play the next song\n");
        printf("3. Exit\n");
        int choice;
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

CODE:

```
switch (choice) {
            case 1:
                printf("Enter the title of the song: ");
                char title[MAX_SONG_TITLE_LENGTH];
                scanf(" %[^\n]", title); // Read the song
title with spaces
                push(&playlist, title);
                break;
            case 2:
                play(&playlist);
  break;
case 3:
                printf("Goodbye!\n");
                exit(0);
            default:
                printf("Invalid choice. Please try
again.\n");
        }
    }
    return 0;
}
```

```
/tmp/VwEGOp5VTc.o
Welcome to the Music Player!

Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 1
Enter the title of the song: YOUNG AND BEAUTIFUL
Added 'YOUNG AND BEAUTIFUL' to the playlist.

Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 1
Enter the title of the song: CREEP
Added 'CREEP' to the playlist.

Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 1
Enter the title of the song: KABIRA
Added 'KABIRA' to the playlist.
```

OUTPUT

```
Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 1
Enter the title of the song: ILLAHI
Added 'ILLAHI' to the playlist.

Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 2
Now playing: 'ILLAHI'

Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 2
Now playing: 'KABIRA'

Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
```

OUTPUT

```
Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 1
Enter the title of the song: ILLAHI
Added 'ILLAHI' to the playlist.

Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 2
Now playing: 'ILLAHI'

Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 2
Now playing: 'KABIRA'

Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 2
Now playing: 'CREEP'
```

```
Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 2
Now playing: 'YOUNG AND BEAUTIFUL'Menu:
1. Add a song to the playlist
2. Play the next song
3. Exit
Enter your choice: 3
Goodbye!
```

Conclusion:

In the grand symphony of life, we often overlook the little notes, like how a simple stack of tunes powers our musical adventures. These melodic stacks in music players keep the groove going, ensuring that we dance, sing, and sometimes cringe to our favorite songs. So, the next time you're tapping your feet or hitting the skip button with enthusiasm, remember that behind every catchy beat is a mischievous stack making sure your playlist rocks! Keep the music playing, and let the stacks stack – because in the end, they're the unsung heroes of our toe-tapping, earworm-filled lives.