



SPARKS

< Student Peer Assistance for Review
and Knowledge Sharing />

DATA STRUCTURES AND
ALGORITHMS



TOPICS:

- ALGORITHMS
- LINKED LISTS
- STACKS
- QUEUES
- SORTING ALGORITHMS
(INTRO)
- SEARCHING ALGORITHMS
(INTRO)

INTRODUCTION TO DATA STRUCTURES AND ALGORITHMS



**Techno Paragons
LEAGUE**



WHAT IS A DATA STRUCTURE?



DATA STRUCTURE

- a way of organizing and storing data that allows for efficient execution of operations.
- enables effective data manipulation and retrieval

IMPORTANCE

- Enhances the performance of algorithms
- Optimizes memory usage
- Improves code reusability and readability
- Real-world problem application and problem-solving

PRIMITIVE

- Integer, Float,
Booleans,
Character

TYPES

NON-PRIMITIVE

- Linear - Arrays,
LinkedList, Stacks
Queues
- Non-Linear - Trees,
Graphs

IMPORTANCE

- Enhances the performance of algorithms
- Optimizes memory usage
- Improves code reusability and readability
- Real-world problem application and problem-solving



Techno Paragons
LEAGUE



ALGORITHM



ALGORITHM

- a finite set of step-by-step instructions to solve a problem or perform a computation.
- algorithms are clear and unambiguous, has a beginning and end and is designed to produce an output.

ALGORITHM APPLICATIONS:

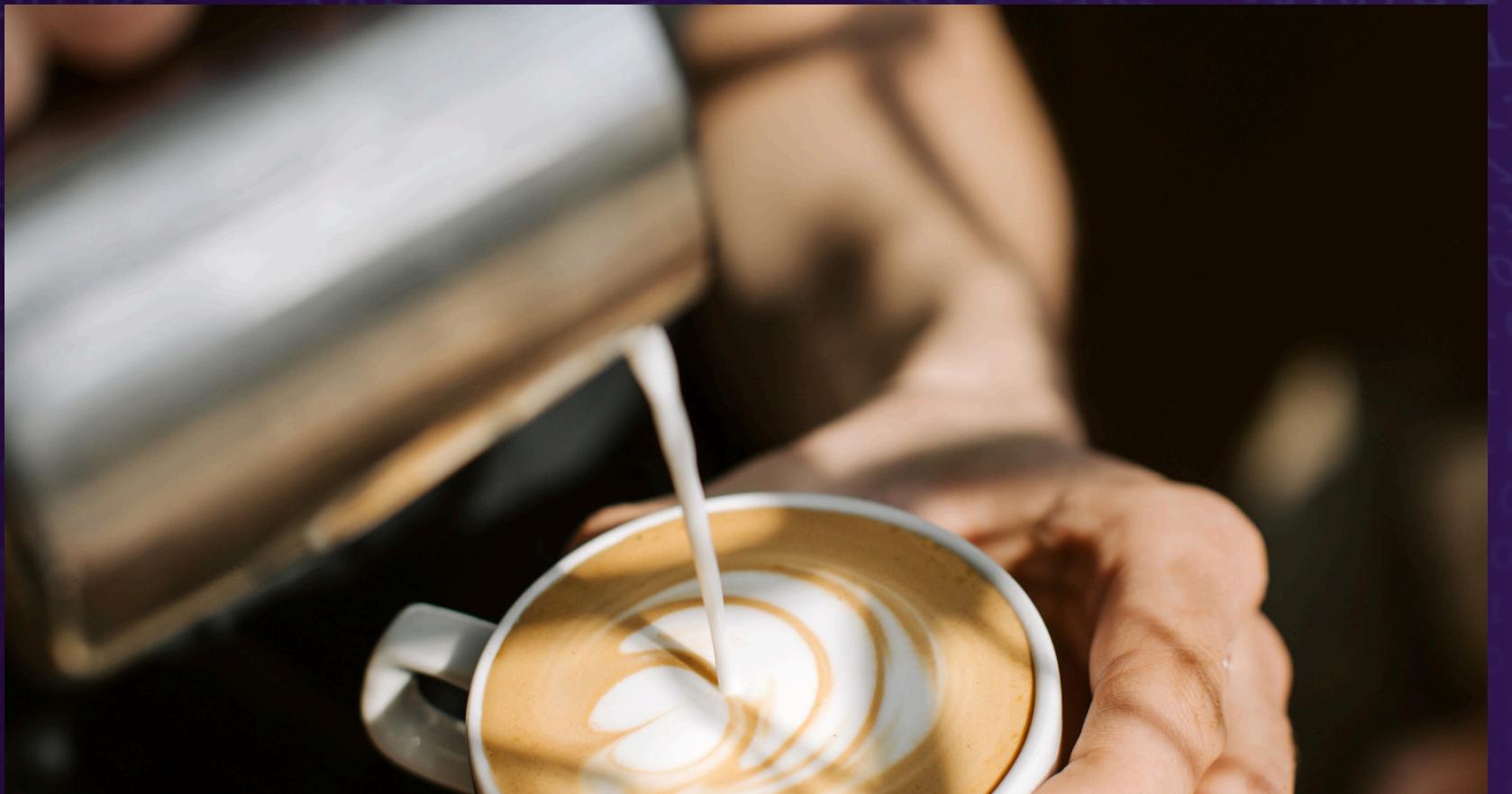
- Computer Science: Sorting, searching, image processing
- Daily Life: Cooking recipes, routines,
- Security: Detection and prevention of threats
- Finance: Data analysis, predictions



**Techno Paragons
LEAGUE**



ALGORITHM REAL LIFE APPLICATION



BREWING A CUP OF COFFEE

1. Boil water
2. Add coffee to the cup
3. Pour hot water
4. Stir
5. Add sugar or milk if preferred
6. Serve and drink



ALGORITHM IN PROGRAMMING

Finding the smallest of three numbers

```
if ((a < b) && (a < c)) {  
    min = a;  
}  
else if (b < c) {  
    min = b;  
}  
else {  
    min = c;  
}
```

This algorithm uses selection logic



Techno Paragons
LEAGUE



EFFICIENCY IN ALGORITHMS



EFFICIENCY

- Time complexity - measures the time an algorithm takes to complete
- Space complexity - measures the amount of memory used

Both depends on the size of the input(n)



**Techno Paragons
LEAGUE**



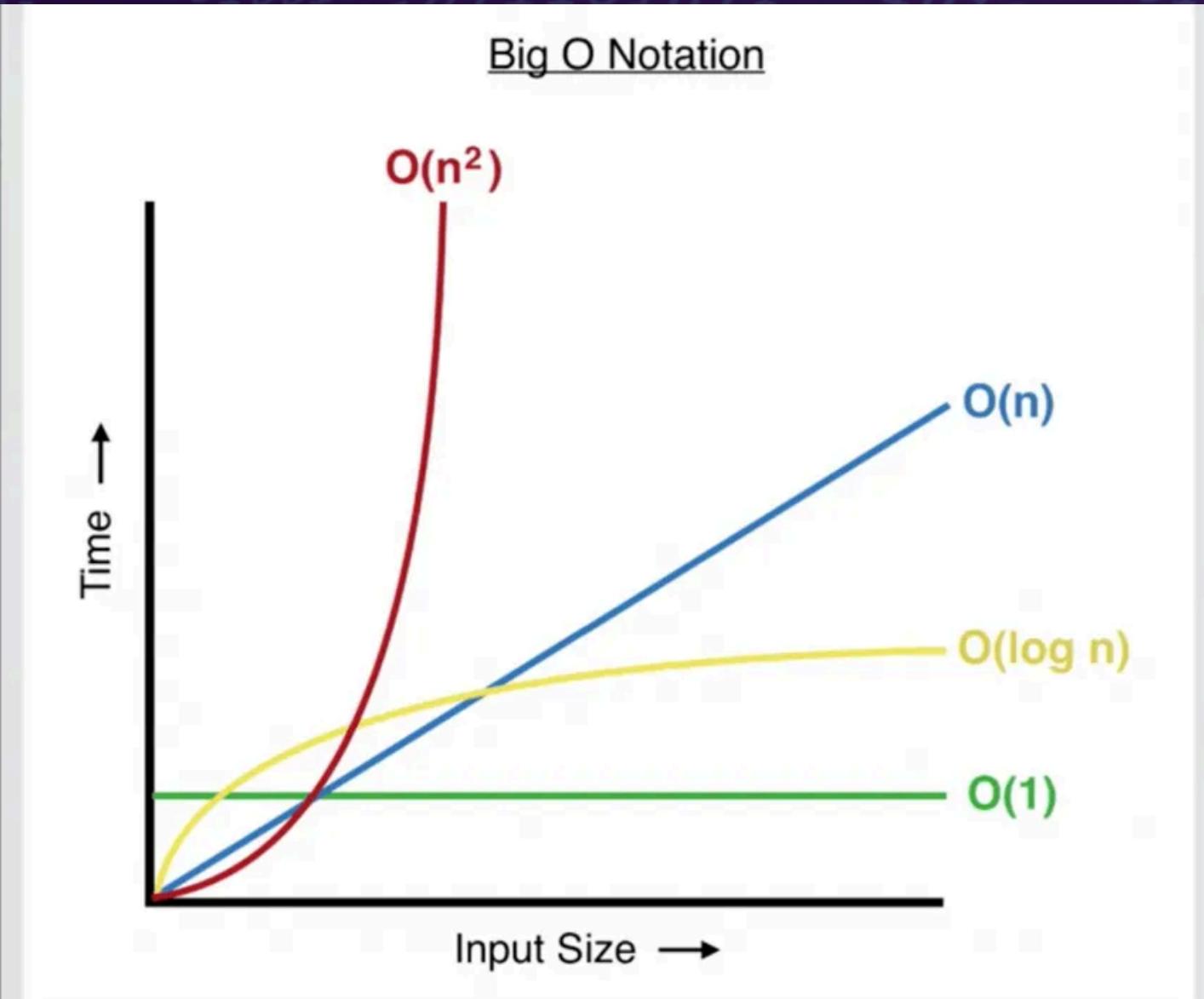
BIG-O NOTATION



BIG-O NOTATION

- the Big-O Notation helps you pick faster and better programs
- crucial when working with big data
- saves time and memory

BIG-O NOTATION



BIG-O	WHAT IT MEANS	EXAMPLES
$O(1)$	CONSTANT	ACCESS ELEMENT
$O(N)$	LINEAR	ONE FOR-LOOP
$O(N^2)$	QUADRATIC	NESTED LOOPS
$O(\log N)$	LOGARITHMIC	BINARY SEARCH



Techno Paragons
LEAGUE



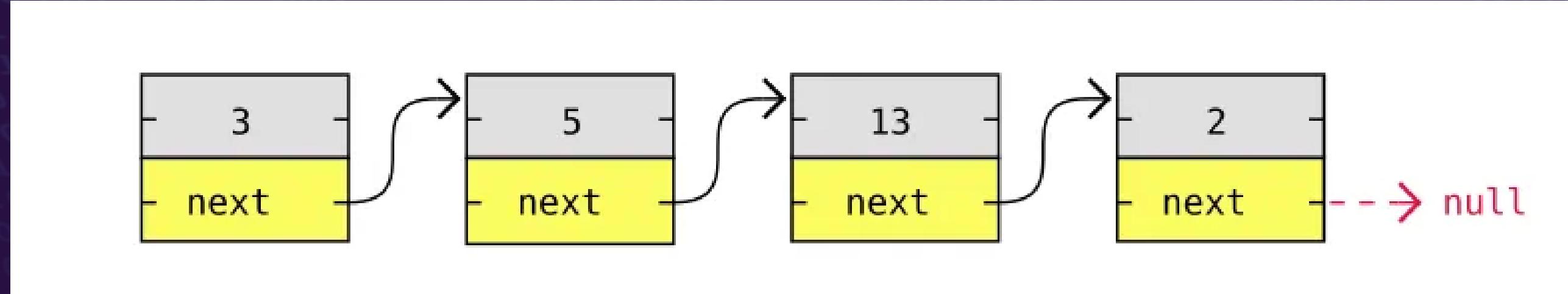
LINKED LIST



LINKED LIST

- a sequence of nodes where each node points to the next and elements are stored in nodes.

- Each node contains:
Data (value)



Pointer (link to the next and/or previous node)

Unlike arrays, it is not stored in a contiguous memory



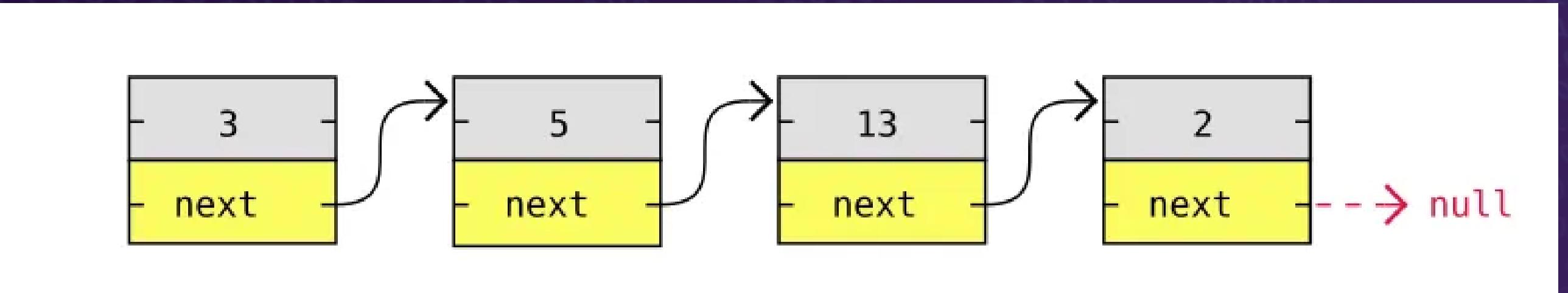
Techno Paragons
LEAGUE



SINGLY LINKED LIST

SINGLY LINKED LIST

- this model only contains the data node and pointer node to the next node.



SINGLY LINKED LIST

Operations:

- insertAtBeginning(data)
- insertAtEnd(data)
- insertAtIndex(index, data)
- deleteAtBeginning()
- deleteAtEnd()
- deleteByValue(value)
- search(data)
- display()
- reverse()



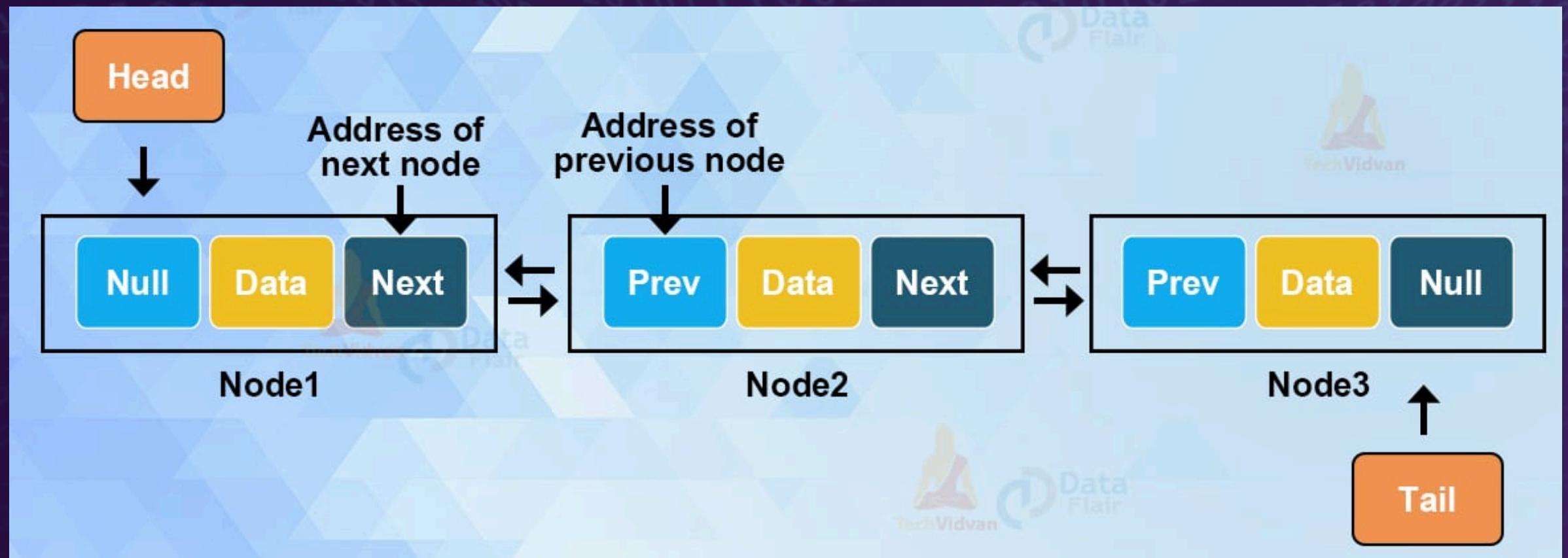
Techno Paragons
LEAGUE



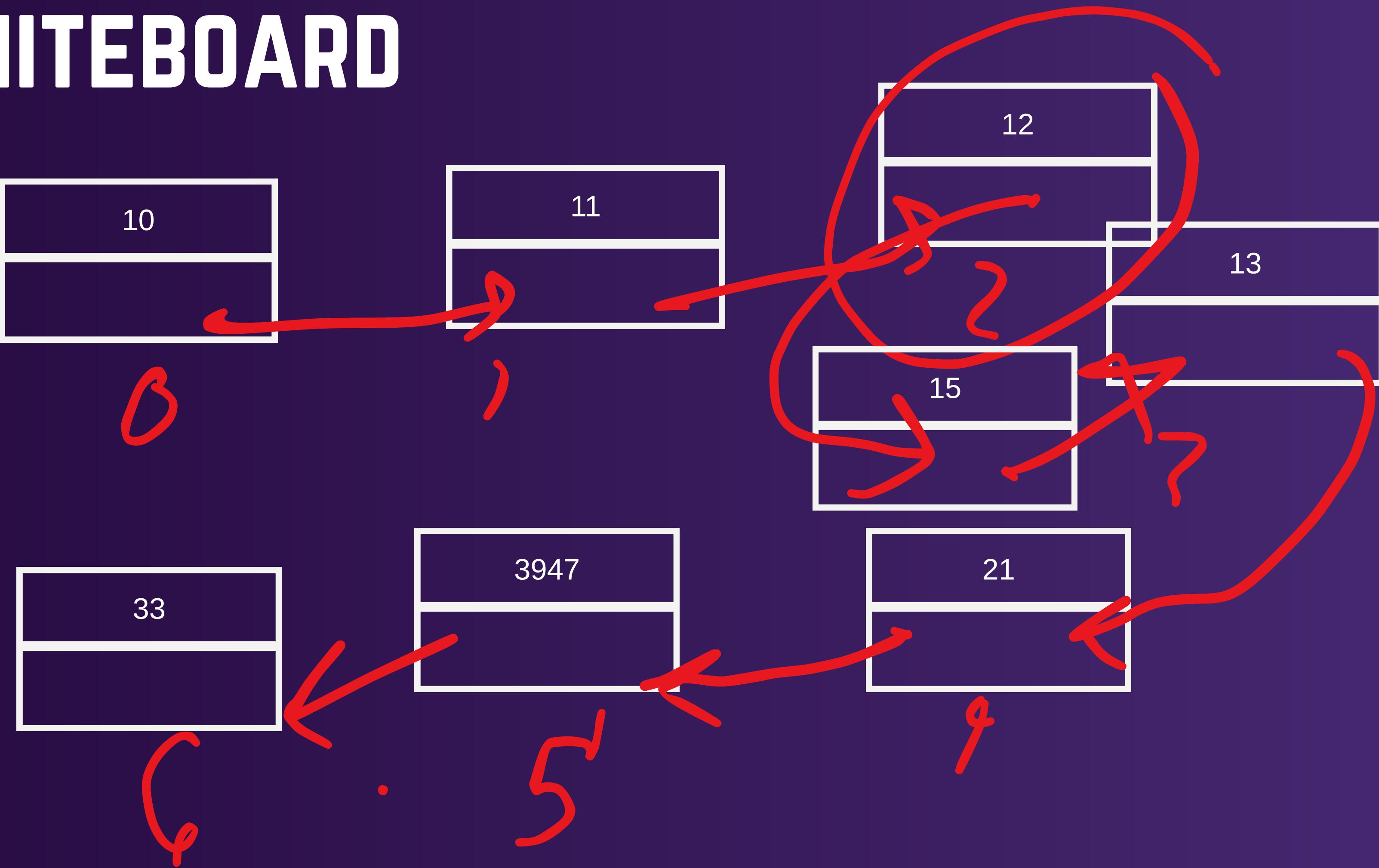
DOUBLY LINKED LIST

DOUBLY LINKED LIST

- The same as a singly Linked List, but comes with an additional previous node for backtracking.



WHITEBOARD



PRACTICE:

Implement a basic Linked List that includes these methods:

- insertAtEnd(data)
- InsertAtStart(data)
- insertAtIndex(index, data)
- deleteAtEnd()
- display()
- reverse()



Techno Paragons

LEAGUE



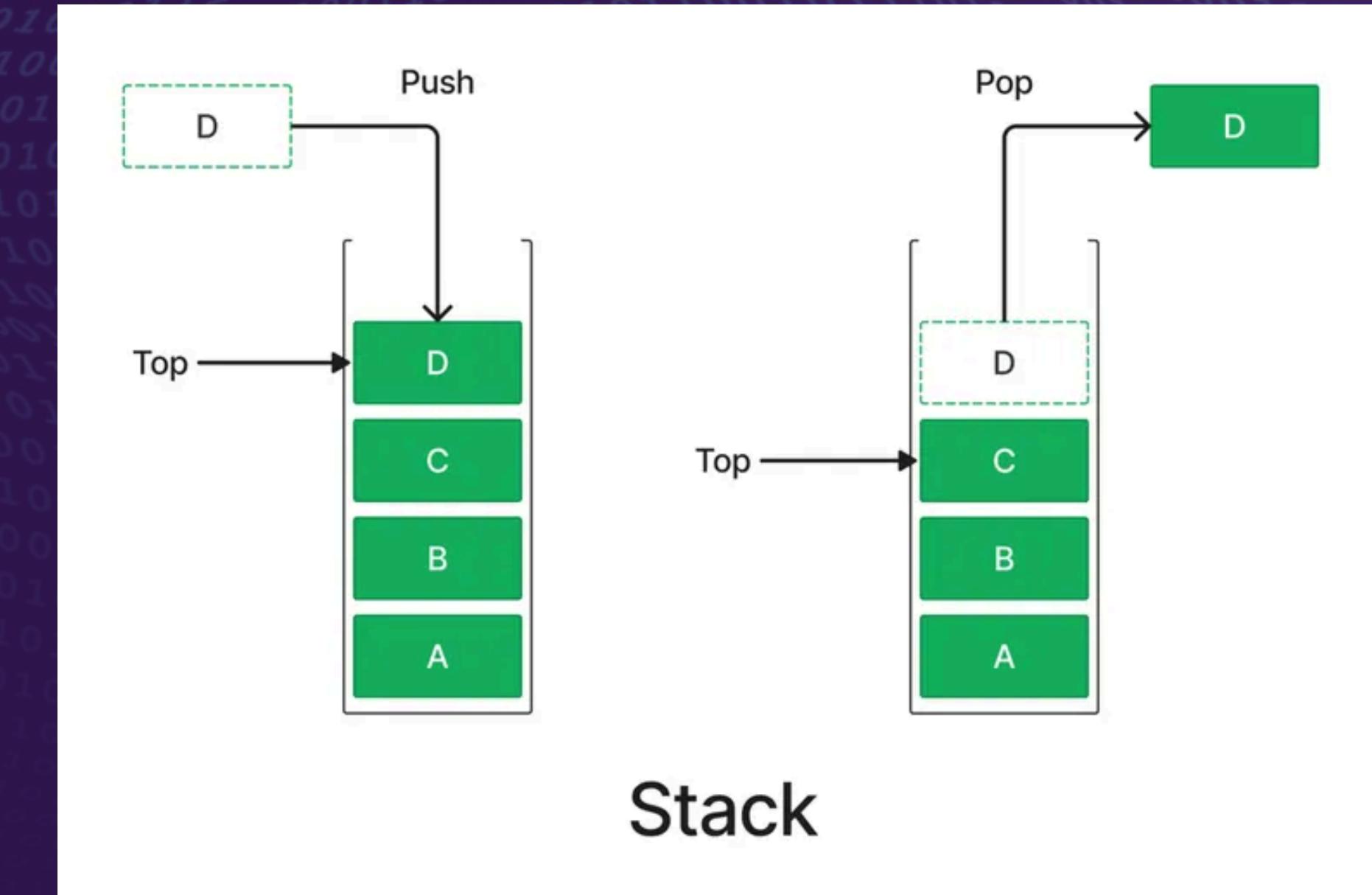
STACKS



STACKS

- LIFO (Last In, First Out)
OPERATIONS:

- push()
- pop()
- peek()
- size()



PRACTICE:

Check for a balanced parentheses using stacks

Example:

Input: `([{}])` → Output: true

Input: `([]})` → Output: false

WHITEBOARD



create document

hello world

hello joseph



**Techno Paragons
LEAGUE**



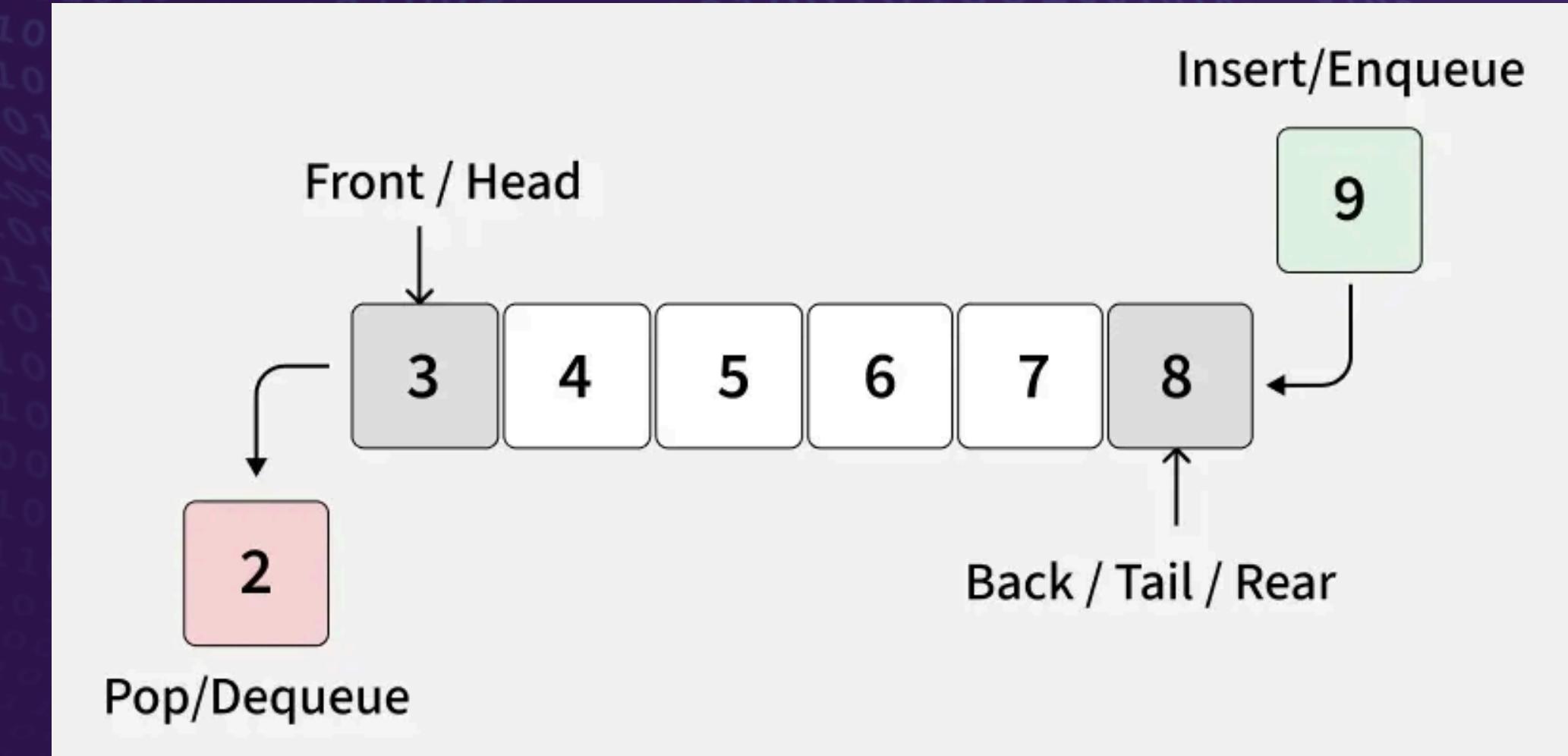
QUEUES



QUEUE

- FIFO (First In, First Out) OPERATIONS:

- enqueue()
- dequeue()
- peak()
- isEmpty()
- size()



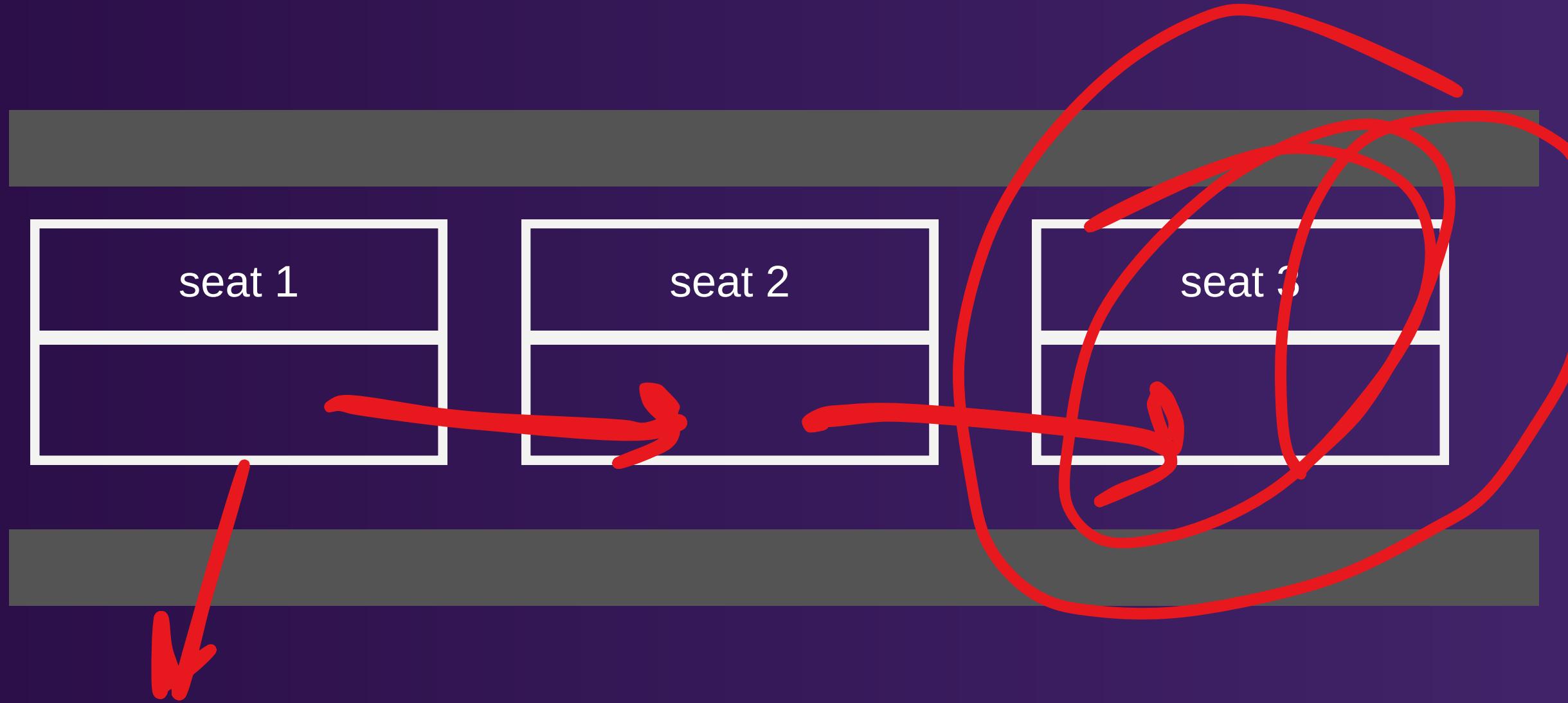


QUEUE

TYPES:

- Simple
- Circular
- Priority
- Double Ended

WHITEBOARD



Head

PRACTICE:

Reverse a queue (made with array) using stacks

Example:

Input size: 3

input data: 10

input data: 5

input data: 12

Output:

==== Reversed Data ====

12

5

10





**Techno Paragons
LEAGUE**



SORTING

SORTING

Searching means finding the location of a particular element within a data structure.

Example:

- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort



**Techno Paragons
LEAGUE**



SEARCHING



Techno Paragons

LEAGUE



SEARCHING

Searching means finding the location of a particular element within a data structure.

Example:

- Linear Search
- Binary Search

PRACTICE:

Given an array of integers ([link_here](#)), implement:

- `linearSearch(arr, target)` → Returns the index if found, else return -1
- `binarySearch(arr, target)` → Returns the index if found, else return -1

At the end, we shall compare the two in approach and speed.

SPARKS DSA MATERIALS:

<https://github.com/jeiya1/school/tree/main/CC105/SPARKS>

