FINAL EXAM

- DATA STRUCTURE AND ALGORITHM (SECJ2013)
- 9.00 am – 12.00 pm , 15.2.2025 (Saturday)
- DEWAN SULTAN ISKANDAR (DSI) – UTM

QUESTIONS (Covered Chapter 6 to Chapter 10)

Type of question similar with Test 1 ,

- Part A : Multiple Choice Question, 20 Questions
- Part B : Structured, 5 Questions

Marks : 30%

CHAPTER 6 : SEARCHING

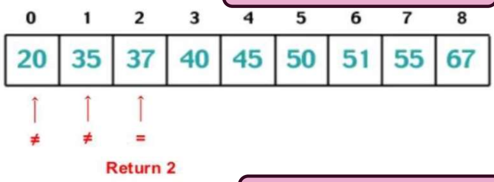
Linear Search vs Binary Search

LINEAR SEARCH	BINARY SEARCH
An algorithm to find an element in a list by sequentially checking the elements of the list until finding the matching element	An algorithm that finds the position of a target value within a sorted array
Also called sequential search	Also called half-interval search and logarithmic search
Time complexity is O(N)	Time complexity is O(log2N)
Best case is to find the element in the first position	Best case is to find the element in the middle position
It is not required to sort the array before searching the element	It is necessary to sort the array before searching the element
Less efficient	More efficient
Less complex	More complex

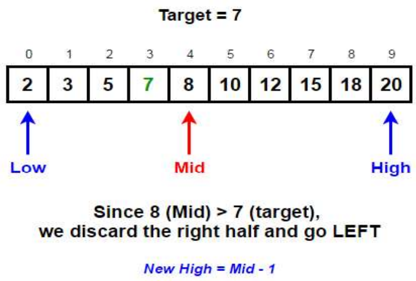
Visit www.PEDIAA.com

Sequential search

- Find 37?



Binary search

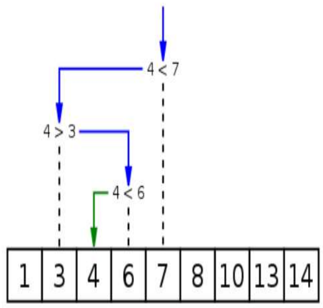


There are three cases used in the binary search:

Case 1: data<a[mid] then left = mid+1.

Case 2: data>a[mid] then right=mid-1

Case 3: data = a[mid] // element is found



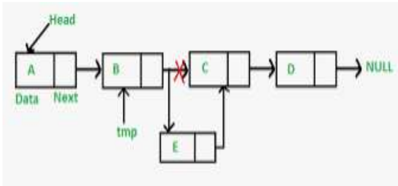
CHAPTER 7 : LINKED LIST

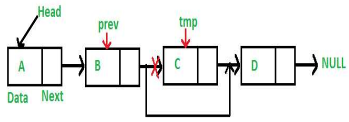
- Pointer Concepts
- Introduction to Linked lists
- Linked lists operations
- Types of Linked List

- Linked List Implementations
  - o Declaring Nodes and Linked Lists class
  - o Insert Node, Delete Node, Find Node, Print Node

CHAPTER 7 : LINKED LIST

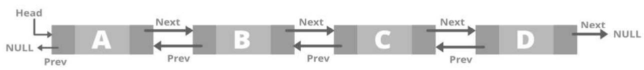
- A linked list is a linear data structure.
- Nodes make up linked lists.
- Nodes are structures made up of data and a pointer to another node.
- Usually the pointer is called next.



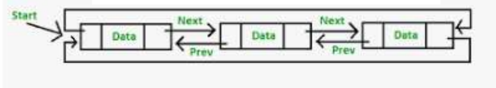


Circular Linked List

Doubly Linked List



Circular Doubly linked list



CHAPTER 8 : STACK

LIFO

- Introduction to Stack
- Stack Operations
  - create(), push(), pop(),stackTop(), isEmpty(), isFull()
- Stack implementations
  - Array based
  - Pointer based
- Stack Applications
  - Infix, Prefix, Postfix

CHAPTER 8 : STACK APPLICATION

Examples of infix to prefix and post fix

Infix	PostFix	Prefix
A+B	AB+	+AB
(A+B) * (C + D)	AB+CD+*	*+AB+CD
A-B/(C*D^E)	ABCDE^*/-	-A/B*C^DE

The infix expression is

(P/(Q−R)*S+T)

Symbol	Stack	Expression
(	(	—
P	(	P
/	(/	P
(	(/(	P
Q	(/(	PQ
−	(/(−	PQ
R	(/(−	PQR
)	(/	PQR−
*	(*	PQR−/
S	(*	PQR−/S
+	(+	PQR−/S*
T	(+	PQR−/S*T
)		PQR−/ST*+

So, the postfix expression is PQR−/ST+*.

Evaluate Postfix

Infix Expression: A * B + C $\xrightarrow{\text{Step-1}}$ Postfix Expression: A B * C +

Values: A=3, B=5, C=5 Postfix Expression: A B * C + ⟶ 3 5 * 5 +

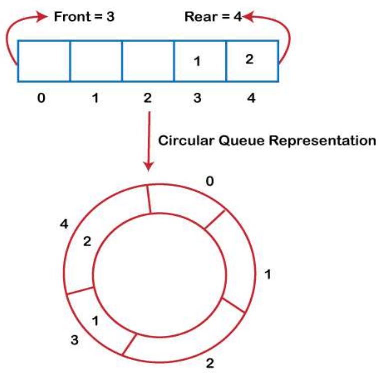
3 5 * 5 +

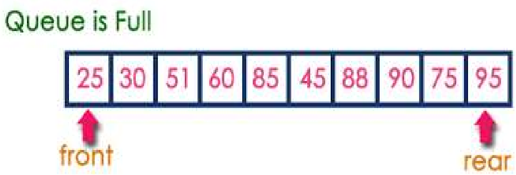
Step	Symbol	Stack	Operation
1	3	3	Push(3)
2	5	3, 5	Push(5)
3	*	15	y = Pop() // 5, x = Pop() // 3, result = x operator y // 3*5, Push(result)
4	5	15, 5	Push(5)
5	+	20	y = Pop() // 5, x = Pop() // 15, result = x operator y // 15+5, Push(result)

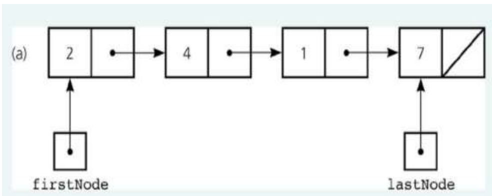
CHAPTER 9 : QUEUE

FIFO

- Introduction to Queue
- Queue Implementations
  - Array based - Linear and Circular Queue
  - Pointer based - Linear and Circular Queue

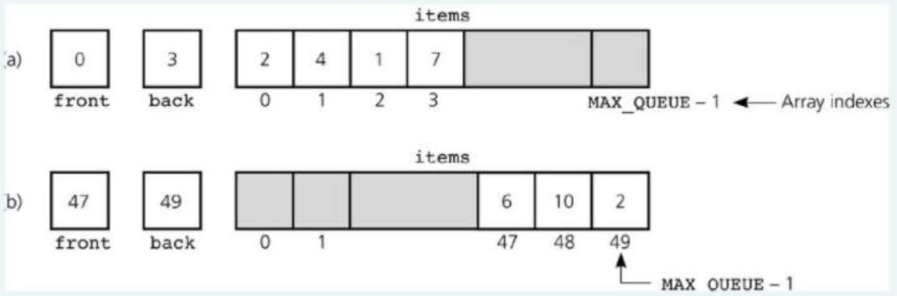






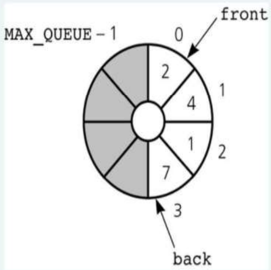
CHAPTER 9 : QUEUE

FIFO



a) A naive array-based implementation of a queue; b) rightward drift can cause the queue to appear full

A circular array eliminates the problem of rightward drift



A circular implementation of a queue

https://slideplayer.com/slide/17434573/

CHAPTER 9 : QUEUE

FIFO

- To detect queue-full and queue-empty conditions
  - Keep a count of the queue items
- To initialize the queue, set
  - `front` to 0
  - `back` to `MAX_QUEUE - 1`
  - `count` to 0

- Inserting into a queue

```
back = (back+1) % MAX_QUEUE;
items[back] = newItem;
++count;
```

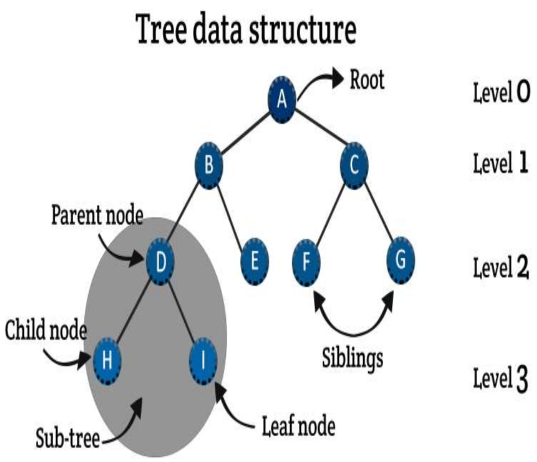
- Deleting from a queue

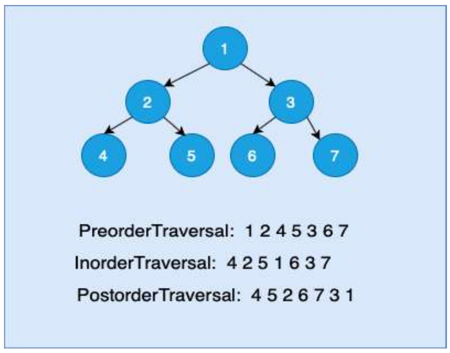
```
front = (front+1) % MAX_QUEUE;
--count;
```

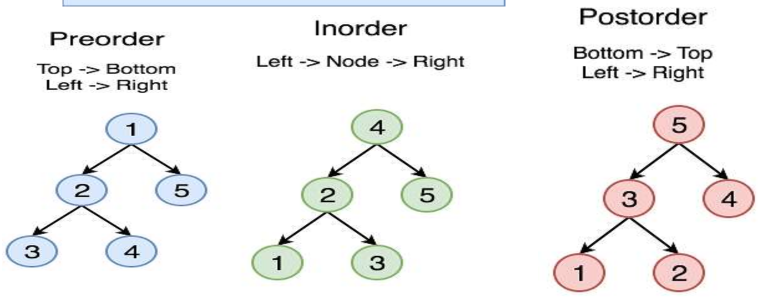
CHAPTER 10 : TREE

- **Introduction to Tree**
  - Terms related to Tree concepts
  - Binary Search Tree
- **Binary Search Tree Implementations**
  - Declaring Tree node, Tree class
  - Create Node, Insert Node, Delete Node, Search Node
  - Tree Traversals

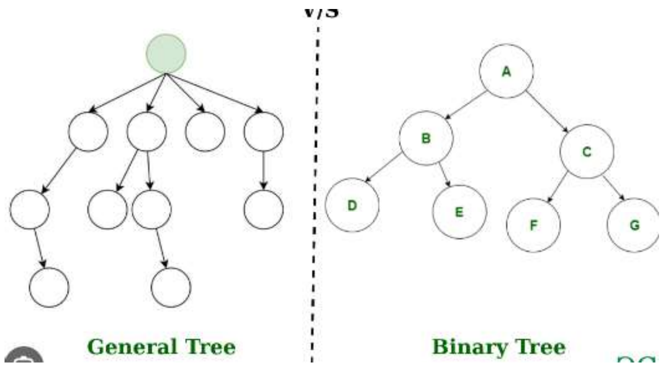
CHAPTER 10 : TREE





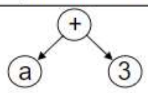
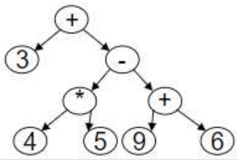


CHAPTER 10 : TREE VS BINARY TREE



Binary Tree is defined as a tree data structure where each node has at most 2 children (internal node) . Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

ALGEBRAIC EXPRESSION – INORDER, PREDORDER,POSTORDER

Expression	Expression Tree	Inorder Traversal Result
(a+3)		a + 3
3+(4*5-(9+6))		3+4*5-9+6

BST