**COURSE CODE:** DJS22ITL502 **DATE: 7-10-2024**

**COURSE NAME:** Advanced Data Structures Laboratory **CLASS:** TY B. TECH

**NAME:** Anish Sharma **DIV:** IT 1 **ROLL:** I011

# EXPERIME NT NO. 4

**CO/LO:** Choose appropriate data structure and use it to design algorithm for solving a specific problem

**AIM / OBJECTIVE:** To implement various operations on Fibonacci Heap.

Properties of Fibonacci Heap:

Amortized Efficiency: Insertions and decrease-key operations take O(1) amortized time.

Lazy Merging: Trees are merged lazily, with restructuring delayed until necessary.

Find Min: Finding the minimum element takes O(1) time by accessing the min pointer.

Consolidation on Extract Min: Extracting the minimum takes O(logn) by consolidating trees.

Decrease Key: Decreasing a key cuts the node and its subtree, and inserts it into the root list in O(1) amortized time.

**TECHNOLOGY STACK USED: C, C++, JAVA SOURCE CODE:**

import java.io.\*;

class Node {

    Node parent;

    Node child;

    Node left;

    Node right;

    int key;

}

class Main {

    static Node mini = null;

    static int no\_of\_nodes = 0;

    static void Insertion(int val)

    {

        Node new\_node = new Node();

        new\_node.key = val;

        new\_node.parent = null;

        new\_node.child = null;

        new\_node.left = new\_node;

        new\_node.right = new\_node;

        if (mini != null) {

            (mini.left).right = new\_node;

            new\_node.right = mini;

            new\_node.left = mini.left;

            mini.left = new\_node;

            if (new\_node.key < mini.key)

                mini = new\_node;

        }

        else {

            mini = new\_node;

        }

    }

    static void Display(Node mini)

    {

        Node ptr = mini;

        if (ptr == null)

            System.out.println("The Heap is Empty");

        else {

            System.out.println(

                "The root nodes of Heap are: ");

            do {

                System.out.print(ptr.key);

                ptr = ptr.right;

                if (ptr != mini) {

                    System.out.print("-->");

                }

            } while (ptr != mini && ptr.right != null);

            System.out.println();

            System.out.println("The heap has " + no\_of\_nodes

                            + " nodes");

        }

    }

    static void FindMin(Node mini)

    {

        System.out.println("min of heap is: " + mini.key);

    }

    public static void main(String[] args)

    {

        no\_of\_nodes = 7;

        Insertion(4);

        Insertion(3);

        Insertion(7);

        Insertion(5);

        Insertion(2);

        Insertion(1);

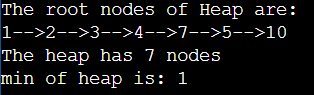
        Insertion(10);

        Display(mini);

        FindMin(mini);

    }}

**OUTPUT:**

****

# CONCLUSION: In this experiment we understood and implemented Fibonacci heaps

# REFERENCES:

1. Peter Brass, “Advanced Data Structures”, Cambridge University Press, 2008
2. Robert Sedgewick & Kevin Wayne, “Algorithms”, 4th Edition, Addison-Wesley Professional, 2011.