ADA LAB PROGRAMS

Program: Quick sort(1)

```
import java.util.*;
public class QuickSort {
    static void quickSort(int[] a, int low, int high) {
        if (low >= high) return;
        int pivot = a[high], i = low - 1;
        for (int j = low; j < high; j++)
            if (a[j] <= pivot) swap(a, ++i, j);
        swap(a, i + 1, high);
        quickSort(a, low, i);
        quickSort(a, i + 2, high);
    }
    static void swap(int[] a, int i, int j) {
        int t = a[i]; a[i] = a[j]; a[j] = t;
    }
    public static void main(String[] args) {
        int[] a = {10, 7, 8, 9, 1, 5};
        long  start=System.nanoTime();
        quickSort(a, 0, a.length - 1);
         long  stop=System.nanoTime();
        for (int n : a) System.out.print(n + " ");
        long Time=stop-start;
        System.out.println();
```

```
 System.out.println("Time taken for execution="+Time);
    }
}
```

OUTPUT:- 1 5 7 8 9 10

Time taken for execution=3850

Program: Merge sort(2)

```
import java.util.Arrays;
import java.util.*;
public class MergeSort {
    static void mergeSort(int[] arr) {
        if (arr.length < 2) return;
        int mid = arr.length / 2;
        int[] left = Arrays.copyOfRange(arr, 0, mid);
        int[] right = Arrays.copyOfRange(arr, mid, arr.length);
        mergeSort(left);
        mergeSort(right);
        int i = 0, j = 0, k = 0;
        while (i < left.length && j < right.length)
            arr[k++] = (left[i] < right[j]) ? left[i++] : right[j++];
        while (i < left.length) arr[k++] = left[i++];
        while (j < right.length) arr[k++] = right[j++];
    }

    public static void main(String[] args) {
        int[] arr = {6, 3, 8, 5, 2};
        long  start=System.nanoTime();
        mergeSort(arr);
         long  stop=System.nanoTime();
        System.out.println(Arrays.toString(arr));
```

```
        long Time=stop-start;
        System.out.println();
        System.out.println("Time taken for execution="+Time);


    }
}
```

OUTPUT:- [2, 3, 5, 6, 8]

Time taken for execution=18270

Program: Sequential search(3)

```
import java.util.*;
public class BruteForceSearch {
    public static int search(int[] arr, int target) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == target) return i;
        }
        return -1;
    }
    public static void main(String[] args) {
        int[] arr = {4, 2, 7, 1, 3};
                long  start=System.nanoTime();
        System.out.println(search(arr, 7)); // Output: 2
                long  stop=System.nanoTime();
                long Time=stop-start;
        System.out.println();
        System.out.println("Time taken for execution="+Time);
    }
}
```

OUTPUT:- 2

Time taken for execution=325260

Program:- Factorial of a number(4)

```
import java.util.*;
public class FactorialRecursive {
    public static long factorial(int n) {
        if (n == 0) {
            return 1;
        }
        return n * factorial(n - 1);
    }
    public static void main(String[] args) {
        int num = 5;
        long  start=System.nanoTime();
        System.out.println("Factorial of " + num + " is: " + factorial(num));
        long  stop=System.nanoTime();
              long Time=stop-start;
        System.out.println();
        System.out.println("Time taken for execution="+Time);
    }
}
```

OUTPUT:- Factorial of 5 is: 120

Time taken for execution=11201244

Program:- Selection sort(5)

```
import java.util.*;
public class SelectionSort {
    public static void selectionSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if (arr[j] < arr[minIndex]) {
                    minIndex = j;
                }
            }
            swap(arr, i, minIndex);
        }
    }
    public static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
    public static void printArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
```

```
        System.out.println();
    }
    public static void main(String[] args) {
        int[] arr = {64, 25, 12, 22, 11};
        System.out.println("Unsorted Array:");
        printArray(arr);
        long  start=System.nanoTime();
        selectionSort(arr);
        System.out.println("Sorted Array:");
        printArray(arr);
        long  stop=System.nanoTime();
         long Time=stop-start;
        System.out.println();
        System.out.println("Time taken for execution="+Time);
    }
}
```

OUTPUT:- Unsorted Array:

64 25 12 22 11

Sorted Array:

11 12 22 25 64

Time taken for execution=11354299

Program:- Euclid's GCD(6)

```
import java.util.*;
public class EuclidGCD {
    public static int euclidGCD(int a, int b) {
        if (b == 0) {
            return a;
        }
        return euclidGCD(b, a % b);
    }
    public static void main(String[] args) {
        int a = 56;
        int b = 98;
        long  start=System.nanoTime();
        int gcd = euclidGCD(a, b);
        System.out.println("The GCD of " + a + " and " + b + " is: " + gcd);
        long  stop=System.nanoTime();
              long Time=stop-start;
        System.out.println();
        System.out.println("Time taken for execution="+Time);
    }
}
```

OUTPUT:- The GCD of 56 and 98 is: 14

Time taken for execution=6312519

Program:- Binary Search(7)

```
import java.util.*;
public class BinarySearch {
    public static int search(int[] arr, int target, int low, int high) {
        if (low > high) return -1;
        int mid = (low + high) / 2;
        if (arr[mid] == target) return mid;
        else if (arr[mid] < target) return search(arr, target, mid + 1, high);
        else return search(arr, target, low, mid - 1);
    }
    public static void main(String[] args) {
        int[] arr = {1, 3, 5, 7, 9};
        long  start=System.nanoTime();
        System.out.println(search(arr, 5, 0, arr.length - 1)); // Output: 2
        long  stop=System.nanoTime();
              long Time=stop-start;
        System.out.println();
        System.out.println("Time taken for execution="+Time);
    }
}
```

OUTPUT:- 2

Time taken for execution=257370

Program:- Dijkstra's(8)

```
import java.util.*;
public class DijkstraAlgorithm {
        public void dijkstraAlgorithm(int[][] graph, int source) {
        int nodes = graph.length;
        boolean[] visited_vertex = new boolean[nodes];
        int[] dist = new int[nodes];
        for (int i = 0; i < nodes; i++) {
          visited_vertex[i] = false;
          dist[i] = Integer.MAX_VALUE;
        }
        dist[source] = 0;
        for (int i = 0; i < nodes; i++) {
          int u = find_min_distance(dist, visited_vertex);
          visited_vertex[u] = true;
          for (int v = 0; v < nodes; v++) {
            if (!visited_vertex[v] && graph[u][v] != 0 && (dist[u] + graph[u][v] <
dist[v])) {
              dist[v] = dist[u] + graph[u][v];
            }
          }
        }
        for (int i = 0; i < dist.length; i++) {
          System.out.println(String.format("Distance from Vertex %s to Vertex
%s is %s", source, i, dist[i]));
```

```
  }
}
private static int find_min_distance(int[] dist, boolean[] visited_vertex) {
  int minimum_distance = Integer.MAX_VALUE;
  int minimum_distance_vertex = -1;
  for (int i = 0; i < dist.length; i++) {
    if (!visited_vertex[i] && dist[i] < minimum_distance) {
      minimum_distance = dist[i];
      minimum_distance_vertex = i;
    }
  }
  return minimum_distance_vertex;
}

  public static void main(String[] args) {
  int graph[][] = new int[][] {
    { 0, 1, 1, 2, 0, 0, 0 },
    { 0, 0, 2, 0, 0, 3, 0 },
    { 1, 2, 0, 1, 3, 0, 0 },
    { 2, 0, 1, 0, 2, 0, 1 },
    { 0, 0, 3, 0, 0, 2, 0 },
    { 0, 3, 0, 0, 2, 0, 1 },
    { 0, 2, 0, 1, 0, 1, 0 }
  };
```

```
        long  start=System.nanoTime();
        DijkstraAlgorithm Test = new DijkstraAlgorithm();
        Test.dijkstraAlgorithm(graph, 0);
        long  stop=System.nanoTime();
        long Time=stop-start;
      System.out.println();
      System.out.println("Time taken for execution="+Time);
      }
}
```

OUTPUT:- Distance from Vertex 0 to Vertex 0 is 0

Distance from Vertex 0 to Vertex 1 is 1

Distance from Vertex 0 to Vertex 2 is 1

Distance from Vertex 0 to Vertex 3 is 2

Distance from Vertex 0 to Vertex 4 is 4

Distance from Vertex 0 to Vertex 5 is 4

Distance from Vertex 0 to Vertex 6 is 3

Time taken for execution=56188507

Program:-Warshal's and Floyd's(9)

```
import java.lang.*;
import java.util.*;
public class AllPairShortestPath {
    final static int INF = 99999, V = 4;
    void floydWarshall(int dist[][])
    {
        int i, j, k;
        for (k = 0; k < V; k++) {
            for (i = 0; i < V; i++) {
                for (j = 0; j < V; j++) {
                    if (dist[i][k] + dist[k][j]
                        < dist[i][j])
                        dist[i][j]
                            = dist[i][k] + dist[k][j];
                }
            }
        }
        printSolution(dist);
    }
    void printSolution(int dist[][])
    {
        System.out.println(
            "The following matrix shows the shortest "
```

```
        + "distances between every pair of vertices");
    for (int i = 0; i < V; ++i) {
        for (int j = 0; j < V; ++j) {
            if (dist[i][j] == INF)
                System.out.print("INF ");
            else
                System.out.print(dist[i][j] + "   ");
        }
        System.out.println();
    }
}
public static void main(String[] args)
{
    int graph[][] = { { 0, 5, INF, 10 },
                      { INF, 0, 3, INF },
                      { INF, INF, 0, 1 },
                      { INF, INF, INF, 0 } };
                      long  start=System.nanoTime();
    AllPairShortestPath a = new AllPairShortestPath();
    a.floydWarshall(graph);
    long  stop=System.nanoTime();
          long Time=stop-start;
    System.out.println();
    System.out.println("Time taken for execution="+Time);
```

```
    }
}
```

OUTPUT:- The following matrix shows the shortest distances between every pair of vertices

```
0   5   8   9
INF 0   3   4
INF INF 0   1
INF INF INF 0
```

Time taken for execution=1182870

Program:-LCM(10)

```
import java.util.*;
public class LCMCalculator {
    private static int gcd(int a, int b) {
        if (b == 0)
            return a;
        return gcd(b, a % b);
    }
    private static int lcm(int a, int b) {
        return (a * b) / gcd(a, b);
    }
    public static int lcmArray(int[] arr) {
        int result = arr[0];
        for (int i = 1; i < arr.length; i++) {
            result = lcm(result, arr[i]);
        }
        return result;
    }

    public static void main(String[] args) {
        int[] numbers = {12, 15, 20, 25};
        long  start=System.nanoTime();
        int result = lcmArray(numbers);
        System.out.println("LCM of the array is: " + result);
```

```
        long  stop=System.nanoTime();
              long Time=stop-start;
        System.out.println();
        System.out.println("Time taken for execution="+Time);
    }
}
```

OUTPUT:- LCM of the array is: 300

Time taken for execution=424930