LAB ASSIGNMENT-9

21BCE9905

K. Madhu Sudhana Rao

1. Write a program to implement the simplex Method.

```
import java.util.Arrays;

Public class SimplexMethod {

 Public Static double[] simplex(double[][] tableau) {

    int m = tableau.length - 1;
    int n = tableau[0].length - 1;

   while(true) {
     int q = 0;
     for(int j = 1; j < n; j++) {
       if ( tableau[m][j] < tableau[m][q]) {
           q = j;
       }
     }
   if ( tableau[m][q] >= 0) {
      break;
   }
   int p = -1;
   double minRatio = Double.MAX_VALUE;
  for(int i = 0; i < m; i++) {
    if ( tableau[i][q] > 0) {
     double ratio = tableau[i][n] / tableau[i][q];
     if (ratio < minRatio) {
        minRatio = ratio;
```

```
                p=i;
            }
        }
    }
    if (p == -1){
        throw new ArithmeticException("Unbound Solution");
    }
    for(int i=0; i<=m; i++){
        for(int j=0; j<=n; j++){
            if(i != ptr[j] && j != q){
                tableau[i][j] -= tableau[p][j] * tableau[i][q]/tableau[p][q];
            }
        }
    }
    for(int i=0; i<=m; i++){
        if(i != p){
            tableau[i][q] = 0;
        }
    }
    for(int j=0; j<=n; j++){
        if(j != q){
            tableau[p][j] /= tableau[p][q];
        }
    }
    tableau[p][q] = 1;
}
public static double[] solution(){
    double[] solution = new double[n];
    Arrays.fill(solution, 0);
    for(int i=0; i<m; i++){
        if(tableau[i][n] == 0){
            continue;
        }
        boolean found = false;
        for(int j=0; j<n; j++){
            if(tableau[i][j] == 1){
                if(!found){
                    found = true;
                }else{
```

```
                throw new ArithmeticException ("Degenerate solution");
            }
            solution[j] = tableau[i][n];
        }
    }
    return solution;
}

public static void main(String[] args) {
    double[][] tableau = {
        {2, 3, 1, 0, 12},
        {4, 1, 0, 1, 16},
        {-3, -4, 0, 0, 0}
    };
    double[] solution = simplex(tableau);
    printSolution(solution);
}
```

// output: Optimal solution $x_1 = 0.0$, $x_2 = 0.0$, $x_3 = 0.0$, $x_4 = 0.0$

2. Write a program to implement - The Maximum-flow problem

```
import java.util.*;

public class MaximumFlow {

    public static int fordFulkerson(int[][] graph, int source, int sink) {
        int n = graph.length;
        int[][] residualGraph = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                residualGraph[i][j] = graph[i][j];
            }
        }

        int[] parent = new int[n];
        int maxFlow = 0;
```

```
while (bfs (residualGraph, source, sink, parent)) {
    int pathFlow = Integer.MAX_VALUE;
    for(int v = sink; v != source; v = parent[v]){
        int u = parent[v];
        pathFlow = Math.min(pathFlow, residualGraph[u][v]);
    }
    for (int v = sink; v != source; v = parent[v]){
        int u = parent[v];
        residualGraph[u][v] -= pathFlow;
        residualGraph[v][u] += pathFlow;
        maxFlow = maxFlow + pathFlow;
    }
    return maxFlow;
}
Private static boolean bfs (int[][] residualGraph, int source,
    int sink, int[] parent) {
    int n = residualGraph.length;
    boolean[] visited = new boolean[n];
    Arrays.fill(visited, false);
    Queue<Integer> queue = new LinkedList<>();
    queue.add(source);
    visited[source] = true;
    parent[source] = -1;
    while (!queue.isEmpty()) {
        int u = queue.poll();
        for(int v=0; v<n; v++){
            if(!visited[v] && residualGraph[u][v] > 0) {
                queue.add(v);
                parent[v] = u;
                visited[v] = true;
            }
        }
    }
    return visited[sink];
}
```

```
Public static void main (String[] args) {
    int[][] graph = {
        {0, 16, 13, 0, 0, 0},
        {0, 0, 10, 12, 0, 0},
        {0, 4, 0, 0, 14, 0},
        {0, 0, 9, 0, 0, 20},
        {0, 0, 0, 7, 0, 4},
        {0, 0, 0, 0, 0, 0}
    };
    int source = 0;
    int sink = 5;
int maxFlow = FordFulkerson(graph, source, sink);
    System.out.println("Maximum Flow" + maxFlow);
}
}
```

Output :-

Maximum flow : 23