```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class PushRelabel {
    public static void main(String[] args) throws FileNotFoundException {
        PushRelabel test=new PushRelabel("src/source.txt");
        test.printCapacity();
        test.printFlow();
        test.printResidual();
        test.printL();
        test.printVertices();

        test.RelabelToFront();

        test.printFlow();
        test.printResidual();
        test.printL();
        test.printVertices();
        test.printMaxFlow();
    }

    private class Vertex{
        private int name;
        private int height;
        private int excess;
        private List<Vertex> neighbours;
        private Vertex next;
        private int nextNeigh;
        public Vertex(int name){
            this.name=name;
            this.height=0;
            this.excess=0;
            this.neighbours=new ArrayList<>();
            this.next=null;
            nextNeigh=0;
        }

        public Vertex nextNeighbour(){
            if(nextNeigh<neighbours.size()) return neighbours.get(nextNeigh++);
            else {
                nextNeigh=0;
                return null;
            }
        }
    }
    private final Vertex dummyHead;
    private final int[][] capacity;
    private final int[][] flow;
    private final int[][] residual;
    private final Vertex[] vertices;

    public PushRelabel(String path) throws FileNotFoundException {
        dummyHead=new Vertex(-1);
        Scanner in =new Scanner(new File(path));
        int row= in.nextInt();
        capacity=new int[row][row];
        flow=new int[row][row];
        residual=new int[row][row];

        int i=0,j=0;
        while(in.hasNextInt()){
            capacity[i][j++]=in.nextInt();
            if(j==row){i++;j=0;}
        }

        Vertex iter=dummyHead;
        vertices=new Vertex[row];
        for(int k=0;k<row;k++) vertices[k]=new Vertex(k);
```

```
    for(int k=1;k<row-1;k++){
        for(int s=0;s<row;s++){
            if(capacity[k][s]!=0||capacity[s][k]!=0) vertices[k].neighbours.add(vertices[s]);
        }
        iter.next=vertices[k];
        iter=iter.next;
    }

    for(int k=0;k<row;k++){
        for(int s=0;s<row;s++){
            if(capacity[k][s]>0) residual[k][s]=capacity[k][s]-flow[k][s];
            else residual[k][s]=flow[s][k];
        }
    }

    vertices[0].height=row;
    for(int k=1;k<row;k++){
        flow[0][k]=capacity[0][k];
        vertices[k].excess=capacity[0][k];
        vertices[0].excess-=capacity[0][k];
        residual[0][k]-=capacity[0][k];
        residual[k][0]+=capacity[0][k];
    }
}

private void Push(Vertex u, Vertex v){
    int delta=Math.min(u.excess,residual[u.name][v.name]);
    if(capacity[u.name][v.name]>0)
        flow[u.name][v.name]+=delta;
    else
        flow[v.name][u.name]-=delta;

    residual[u.name][v.name]-=delta;
    residual[v.name][u.name]+=delta;
    u.excess-=delta;
    v.excess+=delta;
}
private void Relabel(Vertex u){
    int minHeight=Integer.MAX_VALUE;
    for(Vertex v:u.neighbours){
        if(residual[u.name][v.name]>0) minHeight=Math.min(minHeight,v.height);
    }
    u.height=minHeight+1;
}
private void Discharge(Vertex u){
    Vertex tmp;
    while(u.excess>0){
        tmp=u.nextNeighbour();
        if(tmp==null)
            Relabel(u);
        else if(residual[u.name][tmp.name]>0&&u.height==tmp.height+1)
            Push(u,tmp);
    }
}

public void RelabelToFront(){
    Vertex iter=dummyHead.next;
    Vertex prev=dummyHead;
    while(iter!=null){
        int oldHeight=iter.height;
        Discharge(iter);
        if(iter.height>oldHeight){
            prev.next=iter.next;
            iter.next=dummyHead.next;
            dummyHead.next=iter;
        }
        prev=iter;
        iter=iter.next;
    }
}

public void printResidual(){
```

```
        System.out.println("This is residual matrix:\n");
        for(int i=0;i<residual.length;i++){
            for(int j=0;j<residual.length;j++){
                System.out.print(residual[i][j]+"   ");
            }
            System.out.print("\n");
        }
    }

    public void printCapacity(){
        System.out.println("This is capacity matrix:\n");
        for(int i=0;i<capacity.length;i++){
            for(int j=0;j<capacity.length;j++){
                System.out.print(capacity[i][j]+"   ");
            }
            System.out.print("\n");
        }
    }

    public void printFlow(){
        System.out.println("This is flow matrix:\n");
        for(int i=0;i<flow.length;i++){
            for(int j=0;j<flow.length;j++){
                System.out.print(flow[i][j]+"   ");
            }
            System.out.print("\n");
        }
    }

    public void printL(){
        System.out.println("This is L and N table:");
        Vertex iter=dummyHead.next;
        while(iter!=null){
            System.out.println(iter.name+" has neighbours: ");
            for(Vertex v:iter.neighbours) System.out.print(v.name+" ");
            System.out.print("\n\n");
            iter=iter.next;
        }
    }

    public void printVertices(){
        System.out.println("This is summary of vertices:");
        for(Vertex v: vertices) System.out.println("Vertex "+v.name+" has excess "+v.excess+" and height "+v.height);
    }

    public void printMaxFlow(){
        System.out.println("Max flow is "+vertices[vertices.length-1].excess);
    }
}
```