```java
import java.io.*;
import java.lang.reflect.Array;
import java.util.*;

public class Main {
	static ArrayList<nodo> la;
	static int n,m;
	static char caracter = '1';
	static ArrayDeque<Integer> topo = new ArrayDeque<Integer>(); // topo is
LIFO

	public static void main(String[] args) throws Exception {
		lectura txt
		StringBuilder sb = new StringBuilder();
		String ent;
		char linea[];

		while ((ent = br.readLine()) != null) {
			StringTokenizer st = new StringTokenizer(ent);
			n = Integer.parseInt(st.nextToken());
			m = Integer.parseInt(st.nextToken());
			la  = new ArrayList<nodo>();
			//llenado de la lista de adyacencia
			for(int i = 0; i< n; i++){
				linea = br.readLine().toCharArray();
				la.add(new nodo(i));
				for(int k=0; k<m; k++){
					if(linea[k] == caracter)
						la.get(i).setChilds(k);
				}
			}
			//recorrer la lista.
			exploredfalse();
			for(int i = 0; i< n; i++){

				//DFSr(la.get(i));
				//DFSit(la.get(i));
				//System.out.println();
				//BFS(la.get(i));
				if(!la.get(i).isExplored())
					topologicalSort(la.get(i));
			}

			Iterator<Integer> it = topo.iterator();
			//while(it.hasNext())
			//	System.out.print(la.get(it.next()).getIdNodo() +" ->
");
			//System.out.print(conectedComponents());
			sb.append("").append("\n");
		}
		System.out.print(sb);
	}
	public static void exploredfalse(){
		for(int i = 0; i<n; i++) la.get(i).setExplored(false);
	}
```

```
public static void DFSr(nodo nd){
	if(!nd.isExplored()){
		nd.setExplored(true);
		System.out.print("->> "+nd.getIdNodo()+"  ");
		Iterator<Integer> it = nd.getChilds().iterator();
		while(it.hasNext())
			DFSr(la.get(it.next()));
	}
}

public static void DFSit(nodo nd){
	Stack<nodo> lista = new Stack<nodo>();
	lista.push(nd);
	while(!lista.isEmpty()){
		nodo u = lista.pop();
		if(!u.isExplored()){
			System.out.print("->> "+u.getIdNodo()+"  ");
			u.setExplored(true);
			Iterator<Integer> it = u.getChilds().iterator();
			while(it.hasNext())
				lista.push(la.get(it.next()));
		}
	}
}
public static void BFS(nodo nd){
	nd.setExplored(true);
	Queue<nodo> q = new ArrayDeque<nodo>();
	q.add(nd);

	while(!q.isEmpty()){
		nodo u = q.poll();
		System.out.println("->> "+u.getIdNodo());
		Iterator<Integer> it = u.getChilds().iterator();
		while(it.hasNext()){
			nodo key = la.get(it.next());
			if(!key.isExplored()){
				key.setExplored(true);
				q.add(key);
			}
		}
	}
}

public static void topologicalSort(nodo nd){
	nodo u;
	nd.setExplored(true);
	Iterator<Integer> it = nd.getChilds().iterator();
	while(it.hasNext()){
		u = la.get(it.next());
		if(!u.isExplored())
			topologicalSort(u);
	}
	System.out.print(" ->> "+ nd.getIdNodo()+"  ");
	topo.add(nd.getIdNodo());
```

```
	}

	public static int conectedComponents(){
		exploredfalse();
		int c=0;
		for(int i = 0; i< n; i++){
			nodo nd = la.get(i);
			if(!nd.isExplored()){
				BFS(nd);
				c++;
			}
		}
		return c;
	}
	//public static
}
class nodo{
	int idNodo;
	long value;
	boolean explored;
	ArrayList<Integer> childs;
	public nodo(){
		idNodo = -1;
		value = -1;
		explored = false;
		childs = new ArrayList<Integer>();
	}
	public nodo(int id){
		idNodo = id;
		value = -1;
		explored = false;
		childs = new ArrayList<Integer>();
	}
	public int getIdNodo() {
		return idNodo;
	}
	public void setIdNodo(int idNodo) {
		this.idNodo = idNodo;
	}
	public long getValue() {
		return value;
	}
	public void setValue(long value) {
		this.value = value;
	}
	public boolean isExplored() {
		return explored;
	}
	public void setExplored(boolean explored) {
		this.explored = explored;
	}
	public ArrayList<Integer> getChilds() {
		return childs;
	}
	public void setChilds(int child) {
```

```
                this.childs.add(child);
        }
    }
```