```java
class edges {
    private String vertex1;
    private String vertex2;
    private int distance;
    private boolean add = false;;

    edges(String edges){
        String[] str = edges.split(" ");
        vertex1 = str[0];
        vertex2 = str[1];
        distance = Integer.parseInt(str[2]);
    }
    public String getVertex1() {
        return vertex1;
    }
    public String getVertex2() {
        return vertex2;
    }
    public int getDistance() {
        return distance;
    }
    public boolean getAdd() {
        return add;
    }
    public void setAdd(boolean add) {
        this.add = add;
    }
    String addtoString(){
        if(add)return "added";
        return "Not added";
    }
    @Override
    public String toString() {
        return vertex1+" "+vertex2+" "+distance+" "+addtoString();
    }
}
```

```java
public class Main{
    public static void main(String[] args) {
        String[] input = {
            "Minneapolis Chicago 355",
            "Louisville Cincinnati 83",
            "Chicago Milwaukee 74",
            "St.Louis Louisville 242",
            "Louisville Milwaukee 348",
            "Louisville Nashville 151",
            "Chicago Louisville 269",
            "Minneapolis Nashville 695",
            "Louisville Detroit 306",
            "St.Louis Chicago 262",
            "Cincinnati Detroit 230"
        };
        Kruskal q1 = new Kruskal(input);
        q1.printMatrix();
        q1.printAns();
    }
}
```

```
0 74 0 0 0 0 0 0
74 0 0 0 0 0 262 355
0 0 0 83 0 230 0 0
0 0 83 0 151 0 242 0
0 0 0 151 0 0 0 0
0 0 230 0 0 0 0 0
0 262 0 242 0 0 0 0
0 355 0 0 0 0 0 0
Chicago Milwaukee 74 added
Louisville Cincinnati 83 added
Louisville Nashville 151 added
Cincinnati Detroit 230 added
St.Louis Louisville 242 added
St.Louis Chicago 262 added
Chicago Louisville 269 Not added
Louisville Detroit 306 Not added
Louisville Milwaukee 348 Not added
Minneapolis Chicago 355 added
Minneapolis Nashville 695 Not added
```

```java
import java.util.ArrayList;
import java.util.HashMap;

class Kruskal{
    int Matrix[][];
    HashMap<String,Integer> vertex;
    edges[] input;

    Kruskal(String[] ip){
        input= new edges[ip.length];
        for(int i=0; i<input.length; i++){input[i] = new edges(ip[i]); }
        input = Sort(input);
        setVertex(input);
        setMatrix(vertex.size());;
    }

    edges[] Sort(edges[] input){
        for(int i=0; i<input.length; i++){
            int min=i;
            for(int j=i+1; j<input.length; j++){
                if(input[min].getDistance()>input[j].getDistance()){
                    min=j;
                }
            }
            if(min != i){
                edges y = input[i];
                input[i] = input[min];
                input[min] = y;
            }
        }
        return input;
    }

    void setVertex(edges[] input){
        vertex = new HashMap<>();
        ArrayList<String> Strvertex = new ArrayList<>();
        for(edges ip : input){
            if(!Strvertex.contains(ip.getVertex2()))Strvertex.add(ip.getVertex2());
            if(!Strvertex.contains(ip.getVertex1()))Strvertex.add(ip.getVertex1());
        }
        for(int i=0; i<Strvertex.size(); i++){
            vertex.put(Strvertex.get(i), i);
        }
    }

    void setMatrix(int quantity){
        Matrix = new int[quantity][quantity];
        for(int i=0; i<input.length; i++){
            if(checkcontinue2(vertex.get(input[i].getVertex1()),vertex.get(input[i].getVertex2()), -999)){
                Matrix[vertex.get(input[i].getVertex2())][vertex.get(input[i].getVertex1())] = input[i].getDistance();
                Matrix[vertex.get(input[i].getVertex1())][vertex.get(input[i].getVertex2())] = input[i].getDistance();
                input[i].setAdd(true);
            }
        }
    }

    boolean checkcontinue2(int start, int end, int k){
        if(start == end) return false;
            for(int j=0; j<Matrix[0].length; j++){
                if(Matrix[start][j] != 0 && k != j){
                    if(!checkcontinue2(j, end ,start)){
                        return false;
                    }
                }
            }
        return true;
    }

    void printMatrix(){
        for(int i=0; i<Matrix.length; i++){
            for(int j=0; j<Matrix[i].length; j++){
                System.out.print(Matrix[i][j]+" ");
            }
            System.out.println();
        }
    }

    void printAns(){
        for(edges ip : input)System.out.println(ip);
    }
}
```