import java.util.List;

import java.util.LinkedList;

import java.util.Stack;

/\*class Pair{

int origin,destination;

Pair(int p,int q){

origin=p;

destination=q;

}

}\*/

public class FFMinimalCut {

public int[][] adj; //directed weighted graph where weights signify mag of similarity

public int[][] flow;

public boolean[] visited;

public boolean[][] sat;

public boolean[] source;

public int size;

public List<Integer> AugPath;

public List<String> minCut;

Stack<Integer> s;

public FFMinimalCut(int n) {

adj=new int[n][n];

sat=new boolean[n][n];

flow =new int[n][n];

size=n;

s=new Stack<Integer>();

AugPath=new LinkedList<Integer>();

minCut=new LinkedList<String>();

source=new boolean[n];

}

public void MaxFlow() {

while(findAugPath()!=null)

addFlow();

visited=new boolean[size];

boolean b;

s.clear();

do {

b=idSource();

}while(b!=false);

for(int i=0;i<size;i++)

for(int j=0;j<size;j++)

if(adj[i][j]!=0 && source[i]==true && source[j]==false)

minCut.add(""+i+" "+j);

}

public List<Integer> findAugPath() {

AugPath.clear();

visited=new boolean[size];

visited[0]=true;

s.push(0);

int k=-1;

while(!s.isEmpty()) {

k=getNextUnvNei(s.peek());

if(k==size-1) {

s.push(k);

break;

}

if(k==-1)

s.pop();

else {

visited[k]=true;

s.push(k);

}

}

if(k!=size-1)

return null;

else{

Stack<Integer> t=new Stack<Integer>();

while(!s.isEmpty())

t.push(s.pop());

while(!t.isEmpty()) {

Integer z=t.pop();

AugPath.add(z);

}

return AugPath;

}

}

public int getNextUnvNei(int pos) {

for(int i=0;i<size;i++) {

if(adj[pos][i]!=0 && !visited[i] && !sat[pos][i])

return i;

}

return -1;

}

public void addFlow() {

int min=adj[AugPath.get(0)][AugPath.get(1)];

for(int i=1;i<AugPath.size()-1;i++) {

if(adj[AugPath.get(i)][AugPath.get(i+1)]<min)

min=adj[AugPath.get(i)][AugPath.get(i+1)];

}

for(int i=0;i<AugPath.size()-1;i++) {

flow[AugPath.get(i)][AugPath.get(i+1)]+=min;

if(flow[AugPath.get(i)][AugPath.get(i+1)]==adj[AugPath.get(i)][AugPath.get(i+1)])

sat[AugPath.get(i)][AugPath.get(i+1)]=true;

}

}

public boolean idSource() {

visited[0]=true;

boolean flag=false;

s.push(0);

source[0]=true;

while(!s.isEmpty()) {

int k=getNextUnvNei(s.peek());

if(k==-1)

s.pop();

else {

visited[k]=true;

source[k]=true;

s.push(k);

flag=true;

}

}

return flag;

}

public static void main(String args[]) {

FFMinimalCut g=new FFMinimalCut(8);

int[][] w={{0,40,60,0,0,0,0,0},{0,0,0,20,0,0,0,0},{0,0,0,10,15,0,0,0},{0,0,0,0,0,30,0,0},{0,0,0,0,0,0,10,0},{0,0,0,0,0,0,10,60},{0,0,0,0,0,0,0,50},{0,0,0,0,0,0,0,0}};

g.adj=w;

g.MaxFlow();

System.out.println("For g");

for(int i=0;i<g.minCut.size();i++)

System.out.println(g.minCut.get(i));

int[][] v= {{0,4,3,0},{0,0,1,3},{0,0,0,4},{0,0,0,0}};

FFMinimalCut h=new FFMinimalCut(4);

h.adj=v;

h.MaxFlow();

System.out.println("For h");

for(int i=0;i<h.minCut.size();i++)

System.out.println(h.minCut.get(i));

int[][] y= {{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}};

FFMinimalCut x=new FFMinimalCut(6);

x.adj=y;

x.MaxFlow();

System.out.println("For x");

for(int i=0;i<x.minCut.size();i++)

System.out.println(x.minCut.get(i));

}

}