package lab4;

import java.io.File;

import java.util.ArrayList;

import java.util.Scanner;

/\*\*

\*

\* @author gawitt

\*/

public class Lab4 {

public static final int CITI = 19; //cities

int[][] distance = new int[CITI][CITI]; //create an array with the distances between the cities

int totalCost = 0; // accumulator

int bestCost = 100000000; // compare costs to a large number for pruning

public void populateMatrix(){

try{ //checked error prevention through a read file

File data = new File("tsp19.txt"); //12 cities.txt from canvas with values

Scanner input = new Scanner(data); // declaring io file reader scanner

int value; // index values

int i; //rows

int j; // columns

for (i = 0; i < CITI && input.hasNext(); i++) {

for ( j = i; j < CITI && input.hasNext(); j++) {

if(j == i){ //the diagonal in the file

distance[i][j] = 0; // two cities in the rows and columns are the same

}//if

else{

value = input.nextInt(); // keep reading the file

distance[i][j] = value; //first position

distance[j][i] = value; //flip position for diagonal

System.out.print(value + " : "); //echo

}//else

}//for j

System.out.println("");//skip a line

}//for i

}//try

catch(Exception ex){

System.out.println(ex); //display the exception

}//catch

}//populateMatrix

public int computeCost(ArrayList<Integer> tour){

int cost = 0;

for (int i = 0; i < tour.size()-1; i++) { //last element in the distance array

cost += (distance[tour.get(i)][tour.get(i+1)]); // adding the edges of the matrix together into the cost variable

int[] is = distance[i];

}//for

if (tour.size() == CITI) // if the number of cities is met

cost += (distance[tour.get(tour.size()-1)][0]); // back to start: add the cost of the final city to the total cost at index zero

return cost;

}//ComputeCost

public void recDFS(ArrayList<Integer> partialtour, ArrayList<Integer> remainingtour){

if(remainingtour.isEmpty()){

int partialcost = computeCost(partialtour);

if(partialcost < bestCost){

bestCost = partialcost;

System.out.println("Best so far: " + partialtour + " " + bestCost);

}//if

}//if

else{

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

ArrayList<Integer> newpartialtour = new ArrayList<Integer>(partialtour);

newpartialtour.add(remainingtour.get(i));

int pcost = computeCost(newpartialtour);

if(pcost < bestCost){

ArrayList<Integer> newremainingtour = new ArrayList<Integer>(remainingtour);

newremainingtour.remove(i);

recDFS(newpartialtour, newremainingtour);//recursive call

}//if

}//for

}//else

}//recDFS

public void show(){

for (int i = 0; i < distance.length; i++) {

for (int j = 0; j < distance.length; j++) {

System.out.print(distance[i][j]+ "\t");

}//for j

System.out.println("");//line break

}//for i

System.out.println(""); //line break

}//show

public static void main(String[] args) {

Lab4 ans = new Lab4();

ArrayList<Integer> partialtour = new ArrayList<>();

partialtour.add(0);

ArrayList<Integer> remainingcities = new ArrayList<>();

for (int i = 1; i < CITI; i++) {

remainingcities.add(i);

}//for

ans.populateMatrix();

ans.show();

ans.recDFS(partialtour, remainingcities);

}//main

}//class