public class Parcial {

public RutaMinima resolver(Grafo<String> ciudades, String origen, int maxMonto) {

RutaMinima r = new RutaMinima();

r.setX(maxMonto);

if ((!ciudades.esVacio()) && (ciudades != null)) {

ListaGenerica<String> aux = new ListaGenericaEnlazada<String>();

boolean[] visitados = new boolean[ciudades.listaDeVertices().tamanio()];

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

if (origen.equals(ciudades.vertice(i).dato())) {

dfs(i, ciudades, visitados, i, aux, maxMonto, r);

break;

}

}

}

return r;

}

private void dfs(int i, Grafo<String> grafo, boolean[] visitados, int origen, ListaGenerica<String> aux, int monto,

RutaMinima r) {

Vertice<String> v = grafo.vertice(i);

visitados[i] = true;

aux.agregarFinal(v.dato());

if ((aux.tamanio() == r.getL().tamanio() && (r.getX() <= monto)) || (aux.tamanio() > r.getL().tamanio())) {

if (i != origen) {

r.setX(monto);

r.setL((ListaGenericaEnlazada<String>) aux.clonar());

}

}

ListaGenerica<Arista<String>> a = grafo.listaDeAdyacentes(v);

a.comenzar();

while (!a.fin()) {

Arista<String> arista = a.proximo();

int j = arista.verticeDestino().posicion();

if (!visitados[j]) {

if (monto - arista.peso() >= 0)

dfs(j, grafo, visitados, origen, aux, monto - arista.peso(), r);

}

}

visitados[i] = false;

aux.eliminarEn(aux.tamanio() - 1);

}

//// Itau

public ListaGenerica<String> QuedateEnCasa(Grafo<String> grafo, int separacion, String Empleado) {

ListaGenerica<String> aux = new ListaGenericaEnlazada<String>();

if ((!grafo.esVacio()) && (grafo != null)) {

boolean[] visitados = new boolean[grafo.listaDeVertices().tamanio()];

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

if (Empleado.equals(grafo.vertice(i).dato())) {

bfs(i, grafo, visitados, aux, separacion);

break;

}

}

}

return aux;

}

private void bfs(int i, Grafo<String> grafo, boolean[] visitado, ListaGenerica<String> aux, int separacion) {

Vertice<String> vertice = grafo.vertice(i);

ColaGenerica<Vertice<String>> cola = new ColaGenerica<Vertice<String>>();

int max = 0;

int cant = 0;

cola.encolar(vertice);

cola.encolar(null);

visitado[i] = true;

while ((!cola.esVacia()) && (max < separacion) && cant < 40) {

vertice = cola.desencolar();

if (vertice != null) {

ListaGenerica<Arista<String>> a = grafo.listaDeAdyacentes(vertice);

a.comenzar();

while (!a.fin()) {

Arista<String> arista = a.proximo();

int j = arista.verticeDestino().posicion();

if ((!visitado[j]) && (cant < 40)) {

visitado[j] = true;

cant++;

aux.agregarFinal(arista.verticeDestino().dato());

cola.encolar(arista.verticeDestino());

}

}

}

else if (!cola.esVacia()) {

cola.encolar(null);

max++;

}

}

}

}