﻿using \_16189\_16191\_Projeto3ED;

using Arvores;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace \_16189\_16191\_Projeto4ED

{

public partial class frmAcharCaminho : Form

{

private ListaSimples<string> cidades = new ListaSimples<string>();

private Caminho[,] caminhos;

private Grafo g;

public frmAcharCaminho()

{

InitializeComponent();

}

public void btnManterCidades\_Click(object sender, EventArgs e)

{

new frmManterCidades().ShowDialog();

}

private void btnAcharCaminhos\_Click(object sender, EventArgs e)

{

bool cidadesValidas = true;

if (!CidadeValida(txtOrigem.Text))

{

cidadesValidas = false;

MessageBox.Show("A cidade de origem não é válida!", "", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

if (!CidadeValida(txtDestino.Text))

{

cidadesValidas = false;

MessageBox.Show("A cidade de destino não é válida!", "", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

lsbCaminhos.Items.Clear();

// GerarGrafo();

if (cidadesValidas)

{

if (rbRecursao.Checked)

AcharCaminhoComRecursao();

else if (rbBacktracking.Checked)

AcharCaminhoComBacktracking();

else if (rbDijkstra.Checked)

AcharCaminhoComDijkstra();

}

}

private bool CidadeValida(string cidade)

{

if (string.IsNullOrEmpty(cidade))

return false;

foreach (char letra in cidade)

if (char.IsDigit(letra))

return false;

return true;

}

private void montarGrafo()

{

//supondo que a lista de cidades e a matriz de caminhos tenham mesmos "índices"

g = new Grafo(null);

cidades.iniciarPercursoSequencial();

while (cidades.podePercorrer())

{

g.NovoVertice(cidades.atual.Info);

}

for (int i = 0; i < Math.Pow(caminhos.Length, 0.5); i++) {

for (int j = 0; j < Math.Pow(caminhos.Length, 0.5); j++)

{

if (caminhos[i, j] != null)

{

g.NovaAresta(i, j, caminhos[i,j].Custo);

}

} }

}

private void AcharCaminhoComDijkstra()

{

montarGrafo();

lsbCaminhos.Text = g.Caminho(g.indice(txtOrigem.Text), g.indice(txtDestino.Text));

}

private void AcharCaminhoComRecursao()

{ cidades.insereAposFim(new NoLista<string>("Araraquara", null));

cidades.insereAposFim(new NoLista<string>("Americana", null));

cidades.insereAposFim(new NoLista<string>("Bertioga", null));

cidades.insereAposFim(new NoLista<string>("Campinas", null));

cidades.insereAposFim(new NoLista<string>("Guarujá", null));

cidades.insereAposFim(new NoLista<string>("Ilhabela", null));

cidades.insereAposFim(new NoLista<string>("Itatiba", null));

cidades.insereAposFim(new NoLista<string>("Jundiaí", null));

cidades.insereAposFim(new NoLista<string>("Piracicaba", null));

cidades.insereAposFim(new NoLista<string>("Santos", null));

cidades.insereAposFim(new NoLista<string>("São Paulo", null));

cidades.insereAposFim(new NoLista<string>("Sorocaba", null));

cidades.insereAposFim(new NoLista<string>("Sumaré", null));

cidades.insereAposFim(new NoLista<string>("Ubatuba", null));

cidades.insereAposFim(new NoLista<string>("Valinhos", null));

cidades.insereAposFim(new NoLista<string>("Vinhedo", null));

caminhos = new Caminho[cidades.QtosNos, cidades.QtosNos];

caminhos[PosicaoDe("Araraquara"), PosicaoDe("Campinas")] = new Caminho("Araraquara", "Campinas",185);

caminhos[PosicaoDe("Campinas"), PosicaoDe("Sorocaba")] = new Caminho("Campinas", "Sorocaba",86);

caminhos[PosicaoDe("Sorocaba"), PosicaoDe("Santos")] = new Caminho("Sorocaba", "Santos",176);

caminhos[PosicaoDe("Santos"), PosicaoDe("Guarujá")] = new Caminho("Santos", "Guarujá",10);

caminhos[PosicaoDe("Guarujá"), PosicaoDe("Bertioga")] = new Caminho("Guarujá", "Bertioga",44);

caminhos[PosicaoDe("Bertioga"), PosicaoDe("Ilhabela")] = new Caminho("Bertioga", "Ilhabela",100);

caminhos[PosicaoDe("Ilhabela"), PosicaoDe("Ubatuba")] = new Caminho("Ilhabela", "Ubatuba",79);

caminhos[PosicaoDe("Ubatuba"), PosicaoDe("Itatiba")] = new Caminho("Ubatuba", "Itatiba",259);

caminhos[PosicaoDe("Itatiba"), PosicaoDe("Sumaré")] = new Caminho("Itatiba", "Sumaré",62);

caminhos[PosicaoDe("Sumaré"), PosicaoDe("Americana")] = new Caminho("Sumaré", "Americana",12);

caminhos[PosicaoDe("Americana"), PosicaoDe("Piracicaba")] = new Caminho("Americana", "Piracicaba",38);

caminhos[PosicaoDe("Piracicaba"), PosicaoDe("Araraquara")] = new Caminho("Piracicaba", "Araraquara",141);

caminhos[PosicaoDe("Piracicaba"), PosicaoDe("Valinhos")] = new Caminho("Piracicaba", "Valinhos",86);

caminhos[PosicaoDe("Valinhos"), PosicaoDe("Campinas")] = new Caminho("Valinhos", "Campinas",10);

caminhos[PosicaoDe("Valinhos"), PosicaoDe("Jundiaí")] = new Caminho("Valinhos", "Jundiaí",36);

caminhos[PosicaoDe("Jundiaí"), PosicaoDe("Sorocaba")] = new Caminho("Jundiaí", "Sorocaba",90);

caminhos[PosicaoDe("Jundiaí"), PosicaoDe("Guarujá")] = new Caminho("Jundiaí", "Guarujá",144);

caminhos[PosicaoDe("Jundiaí"), PosicaoDe("São Paulo")] = new Caminho("Jundiaí", "São Paulo",97);

caminhos[PosicaoDe("São Paulo"), PosicaoDe("Bertioga")] = new Caminho("São Paulo", "Bertioga",109);

caminhos[PosicaoDe("São Paulo"), PosicaoDe("Ubatuba")] = new Caminho("São Paulo", "Ubatuba",224);

caminhos[PosicaoDe("São Paulo"), PosicaoDe("Vinhedo")] = new Caminho("São Paulo", "Vinhedo",77);

caminhos[PosicaoDe("Vinhedo"), PosicaoDe("Itatiba")] = new Caminho("Vinhedo", "Itatiba",18);

caminhos[PosicaoDe("Vinhedo"), PosicaoDe("Americana")] = new Caminho("Vinhedo", "Americana",55);

caminhos[PosicaoDe("Vinhedo"), PosicaoDe("Valinhos")] = new Caminho("Vinhedo", "Valinhos",9);

for(int i=0;i<Math.Pow(caminhos.Length, 0.5); i++)

{

for(int j = 0; j < Math.Pow(caminhos.Length, 0.5); j++)

{

if(caminhos[i,j]==null && caminhos[j, i] != null)

{

caminhos[i, j] = new Caminho(caminhos[j, i].CidadeDestino, caminhos[j, i].CidadeOrigem, caminhos[j, i].Custo );

}

}

}

string orig = Origem;

string dest = Destino;

ListaSimples<Trajeto> listaCaminhosAchados = new ListaSimples<Trajeto>();

bool[] jaPassou = new bool[cidades.QtosNos];

for (int i = 0; i < jaPassou.Length; i++)

jaPassou[i] = false;

acharCaminho(ref listaCaminhosAchados, jaPassou, PosicaoDe(orig), PosicaoDe(orig), PosicaoDe(dest));

Trajeto achado = Trajeto.menor(listaCaminhosAchados);

string[] linhas = achado.ToString().Split('\n');

foreach (string linha in linhas)

{

lsbCaminhos.Items.Add(linha);

}

}

private void acharCaminho(ref ListaSimples<Trajeto> caminhos, bool[] jaPassou, int origAbs, int origem, int destino)

{

ListaSimples<Trajeto> conexoes = acharConexoes(origem, jaPassou);

int numero = conexoes.QtosNos;

conexoes.iniciarPercursoSequencial();

while (conexoes.podePercorrer())

{

jaPassou[PosicaoDe(conexoes.atual.Info.origemAbs)] = true;

origem = PosicaoDe(conexoes.atual.Info.destinoAbs);

caminhos.insereAposFim(new NoLista<Trajeto>((conexoes.atual.Info), null));

if (origem != destino)

{

acharCaminho(ref caminhos, jaPassou, origAbs, origem, destino);

}

else

{

ListaSimples<Trajeto> aux = new ListaSimples<Trajeto>();

caminhos.iniciarPercursoSequencial();

int tamanho = caminhos.QtosNos;

int i;

int j = 0;

bool achou = false;

while (caminhos.podePercorrer())

{

if (achou || caminhos.atual.Prox == null || caminhos.atual.Prox.Info.origemAbs != ValorDe(origAbs))

{

achou = true;

aux.insereAposFim(new NoLista<Trajeto>(caminhos.atual.Info, null));

}

}

caminhos.iniciarPercursoSequencial();

while (caminhos.podePercorrer())

{

if (aux.existe(caminhos.atual.Info))

{

caminhos.removerNo(caminhos.atual, caminhos.anterior);

if (caminhos.anterior == null)

{

caminhos.iniciarPercursoSequencial();

}

else

{

caminhos.atual = caminhos.anterior;

}

}

}

caminhos.insereAposFim(new NoLista<Trajeto>(new Trajeto(aux), null));

aux.iniciarPercursoSequencial();

while (aux.podePercorrer())

{

caminhos.insereAposFim(new NoLista<Trajeto>(aux.atual.Info, null));

}

if(caminhos.ultimo!=null)

if (!(caminhos.ultimo.Info.origemAbs==(ValorDe(origAbs)) && caminhos.ultimo.Info.destinoAbs==(ValorDe(destino)) ))

caminhos.removerUltimoNo();

}

jaPassou[PosicaoDe(conexoes.atual.Info.origemAbs)] = false;

}

if(caminhos.ultimo!=null)

if (!(caminhos.ultimo.Info.origemAbs==(ValorDe(origAbs)) && caminhos.ultimo.Info.destinoAbs==(ValorDe(destino)) ))

caminhos.removerUltimoNo();

}

private ListaSimples<Trajeto> acharConexoes(int cidade, bool[] jaPassou)

{

ListaSimples<Trajeto> retorno = new ListaSimples<Trajeto>();

for (int i = 0; i < Math.Pow(caminhos.Length, 0.5); i++)

{

if (caminhos[cidade, i] != null)

{

if (!jaPassou[i])

{

ListaSimples<Caminho> caminho = new ListaSimples<Caminho>();

caminho.insereAposFim(new NoLista<Caminho>(caminhos[cidade, i], null));

retorno.insereAposFim(new NoLista<Trajeto>(new Trajeto(caminho), null));

}

}

}

return retorno;

}

private void AcharCaminhoComBacktracking()

{

lsbCaminhos.Items.Clear();

int distancia, tempo, valor;

distancia = tempo = valor = 0;

if (CaminhoEntre(Origem, Destino) == null)

{

bool achou = false;

bool caminhoCompleto = false;

PilhaDinamica<Caminho> caminhosJaPassou = new PilhaDinamica<Caminho>();

bool[] jaPassou = new bool[cidades.QtosNos];

for (int i = 0; i < cidades.QtosNos; i++)

jaPassou[i] = false;

Caminho novoCaminho, naoDeuCerto;

naoDeuCerto = null;

string novaOrigem = Origem;

string origemAnt = "";

while (!caminhoCompleto)

{

achou = false;

for (int i = 0; i < cidades.QtosNos; i++)

{

if (achou == false)

{

if (caminhos[PosicaoDe(novaOrigem), i] != null && jaPassou[i] == false)

{

novoCaminho = caminhos[PosicaoDe(novaOrigem), i];

if (novoCaminho != null)

{

caminhosJaPassou.empilhar(novoCaminho);

achou = true;

jaPassou[PosicaoDe(novaOrigem)] = true;

origemAnt = novaOrigem;

novaOrigem = ValorDe(i);

}

}

if (novaOrigem != null && novaOrigem != "")

if (caminhos[PosicaoDe(origemAnt), PosicaoDe(novaOrigem)] != null)

if (PosicaoDe(novaOrigem) == PosicaoDe(Destino))

caminhoCompleto = true;

}

}

if (achou == false)

{

try

{

naoDeuCerto = caminhosJaPassou.Desempilhar();

}

catch { }

if (naoDeuCerto != null)

{

novaOrigem = naoDeuCerto.CidadeOrigem;

jaPassou[PosicaoDe(novaOrigem)] = false;

jaPassou[PosicaoDe(naoDeuCerto.CidadeDestino)] = true;

}

}

}

PilhaDinamica<Caminho> pilhaInvertida = new PilhaDinamica<Caminho>();

while (!caminhosJaPassou.EstaVazia())

{

pilhaInvertida.empilhar(caminhosJaPassou.Desempilhar());

valor += pilhaInvertida.Topo.Info.Custo;

}

while (!pilhaInvertida.EstaVazia())

{

var caminhoDeExibicao = pilhaInvertida.Desempilhar().ToString();

lsbCaminhos.Items.Add(caminhoDeExibicao);

}

}

else

lsbCaminhos.Items.Add(CaminhoEntre(Origem, Destino));

}

private string ValorDe(int indiceDaCidade)

{

var atual = cidades.primeiro;

int indice = 0;

while (indice < indiceDaCidade)

{

atual = atual.Prox;

indice++;

}

return atual.Info;

}

private Caminho CaminhoEntre(string cidadeOrigem, string cidadeDestino)

{

int indiceOrigem = -1;

int indiceDestino = -1;

try

{

indiceOrigem = PosicaoDe(cidadeOrigem);

indiceDestino = PosicaoDe(cidadeDestino);

}

catch (Exception e)

{

MessageBox.Show(e.Message + "\n" + e.StackTrace);

}

return caminhos[indiceOrigem, indiceDestino];

}

public int PosicaoDe(string cidade)

{

int resultado = 0;

NoLista<string> atual = cidades.primeiro;

while (atual.Info != cidade)

{

resultado++;

atual = atual.Prox;

}

return resultado;

}

private void GerarGrafo(ListaSimples<Caminho> listaCaminhos)

{

caminhos = new Caminho[cidades.QtosNos, cidades.QtosNos];

NoLista<Caminho> atual = listaCaminhos.primeiro;

do

{

caminhos[PosicaoDe(atual.Info.CidadeOrigem), PosicaoDe(atual.Info.CidadeDestino)] = atual.Info;

atual = atual.Prox;

} while (atual != null);

}

private void LerArquivo()

{

try

{

var listaCaminhos = new ListaSimples<Caminho>();

// StreamReader arquivo = new StreamReader(NomeArquivo);

string linha = "";

/\* while ((linha = arquivo.ReadLine()) != null)

{

//Entrada de dados e criação dos objetos a serem armazenados

string origem = linha.Substring(0, 15);

string destino = linha.Substring(15, 15);

string distancia = linha.Substring(30, 4

string velocidade = linha.Substring(34, 4);

string custo = linha.Substring(38);

Caminho aIncluir = new Caminho(origem, destino, int.Parse(distancia), int.Parse(velocidade), int.Parse(custo));

cidades.inserirEmOrdem(origem); //Se essa cidade já foi armazenada, ela não será armazenada novamente

cidades.inserirEmOrdem(destino); //

listaCaminhos.inserirEmOrdem(aIncluir); //Todos os caminhos devem estar registrados antes de se criar o grafo,

//pois é necessário saber o número total de cidades para, depois, gerar

//a matriz, que terá tamanho fixo

}

GerarGrafo(listaCaminhos);

arquivo.Close();

arquivo.Dispose();\*/

}

catch (Exception e)

{

MessageBox.Show(e.Message);

MessageBox.Show(e.StackTrace);

}

}

private string Origem

{

get

{

return txtOrigem.Text;

}

}

private string Destino

{

get

{

return txtDestino.Text;

}

}

}

}