//

// main.cpp

// FinalProjectCS560

//

// Created by YusufShaikh ,ErnestViola, WilliamMerz, VinhTa, StephenLuu, GregWoods on 4/18/17.

// Copyright (c) 2017 San Diego State University. All rights reserved.

//

#include <iostream>

#include <fstream>

#include <istream>

#include <ctime>

#include <chrono>

#include <climits>

using namespace std;

using namespace std::chrono;

int cost[234];

struct Nodes

{

int Weight;

int SelfIndex;

Nodes \*N, \*NE, \*SE, \*S, \*SW, \*NW;

int parent;

bool InPath = false;

}Hexagon[234];

void dikstra();

void setCostInArray(int index);

int main(int argc, const char \* argv[])

{

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

int flagin = 0, flagout = 0;

ifstream file("INPUT.txt");

ofstream outputfile("OUTPUT.txt");

// ifstream file( argv[1] );

// ofstream outputfile( argv[2] );

// try

// {

// file;

// }

// catch(int e)

// {

// ifstream file("INPUT.txt");

// }

// try

// {

// outputfile;

// }

// catch(int e)

// {

// ofstream outputfile("OUTPUT11.txt");

// }

string output;

int count = 1;

int i = 1;

int index;

int arrayValues[234];

int temp = 8;

int outputarray[100000];

while(file >> output)

{

if (count%2 == 0)

{

arrayValues[i] = (int)strtol(output.c\_str(),NULL,10);

i++;

}

count++;

}

cost[0] = INT\_MAX;

//set Hex nodes

for (i = 1;i <=233;i++ )

{

cost[i] = INT\_MAX;

Hexagon[i].Weight = arrayValues[i];

Hexagon[i].SelfIndex = i;

Hexagon[i].N = &Hexagon[i-15];

Hexagon[i].NE = &Hexagon[i-7];

Hexagon[i].NW = &Hexagon[i-8];

Hexagon[i].S = &Hexagon[i+15];

Hexagon[i].SE = &Hexagon[i+8];

Hexagon[i].SW = &Hexagon[i+7];

if (i <=8)

{

Hexagon[i].N = NULL;

Hexagon[i].NE = NULL;

Hexagon[i].NW = NULL;

}

else if(9 <= i && i <= 15)

{

Hexagon[i].N = NULL;

}

if ((i-8)%15 == 0)

{

Hexagon[i].NE = NULL;

Hexagon[i].SE = NULL;

}

if ((i-1)%15 == 0)

{

Hexagon[i].NW = NULL;

Hexagon[i].SW = NULL;

}

if (i >= 226)

{

Hexagon[i].S = NULL;

Hexagon[i].SE = NULL;

Hexagon[i].SW = NULL;

}

else if(i >= 219 && i <= 225)

{

Hexagon[i].S = NULL;

}

}

// int num = 233;

// if(Hexagon[num].N != NULL)

// cout << "N = " << Hexagon[num].N->Weight <<endl;

// if(Hexagon[num].NE != NULL)

// cout << "NE = " << Hexagon[num].NE->Weight <<endl;

// if(Hexagon[num].SE != NULL)

// cout << "SE = " << Hexagon[num].SE->Weight <<endl;

// if(Hexagon[num].S != NULL)

// cout << "S = " << Hexagon[num].S->Weight <<endl;

// if(Hexagon[num].SW != NULL)

// cout << "SW = " << Hexagon[num].SW->Weight <<endl;

// if(Hexagon[num].NW != NULL)

// cout << "NW = " << Hexagon[num].NW->Weight <<endl;

dikstra();

int outcounter = 1;

int sum = 0;

outputarray[0] = temp;

while(temp != 226)

{

outputarray[outcounter] = Hexagon[temp].parent;

sum = sum + Hexagon[temp].Weight;

temp = Hexagon[temp].parent;

outcounter++;

}

sum = sum + Hexagon[226].Weight;

for (i = (outcounter-1); i >=0; i--)

{

outputfile << outputarray[i] << endl;;

}

outputfile << "MINIMAL-COST PATH COSTS: "<< sum << endl;

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

auto duration = duration\_cast<microseconds>( t2 - t1 ).count();

cout << "0.000" << duration << " seconds" <<endl;

outputfile.close();

}

void dikstra ()

{

int index = 226;

int temp;

cost[226] = Hexagon[226].Weight;

for (int i = 0; i <= 233; i++) //50

{

Hexagon[index].InPath = true;

setCostInArray(index);

temp = 0;

for (int j = 1; j <= 234; j++) //COST ARRAY

{

if (Hexagon[j].InPath == false)

{

if (cost[j] < cost[temp])

temp = j;

}

}

index = temp;

}

}

void setCostInArray(int index)

{

if (Hexagon[index].N != NULL && Hexagon[index].N->InPath != true)

if (cost[Hexagon[index].N->SelfIndex] > cost[index] + Hexagon[index].N->Weight)

{

cost[Hexagon[index].N->SelfIndex] = cost[index] + Hexagon[index].N->Weight;

Hexagon[index].N->parent = index;

}

if (Hexagon[index].NE != NULL && Hexagon[index].NE->InPath != true)

if (cost[Hexagon[index].NE->SelfIndex] > cost[index] + Hexagon[index].NE->Weight)

{

cost[Hexagon[index].NE->SelfIndex] = cost[index] + Hexagon[index].NE->Weight;

Hexagon[index].NE->parent = index;

}

if (Hexagon[index].SE != NULL && Hexagon[index].SE->InPath != true)

if (cost[Hexagon[index].SE->SelfIndex] > cost[index] + Hexagon[index].SE->Weight)

{

cost[Hexagon[index].SE->SelfIndex] = cost[index] + Hexagon[index].SE->Weight;

Hexagon[index].SE->parent = index;

}

if (Hexagon[index].S != NULL && Hexagon[index].S->InPath != true)

if (cost[Hexagon[index].S->SelfIndex] > cost[index] + Hexagon[index].S->Weight)

{

cost[Hexagon[index].S->SelfIndex] = cost[index] + Hexagon[index].S->Weight;

Hexagon[index].S->parent = index;

}

if (Hexagon[index].SW != NULL && Hexagon[index].SW->InPath != true)

if (cost[Hexagon[index].SW->SelfIndex] > cost[index] + Hexagon[index].SW->Weight)

{

cost[Hexagon[index].SW->SelfIndex] = cost[index] + Hexagon[index].SW->Weight;

Hexagon[index].SW->parent = index;

}

if (Hexagon[index].NW != NULL && Hexagon[index].NW->InPath != true)

if (cost[Hexagon[index].NW->SelfIndex] > cost[index] + Hexagon[index].NW->Weight)

{

cost[Hexagon[index].NW->SelfIndex] = cost[index] + Hexagon[index].NW->Weight;

Hexagon[index].NW->parent = index;

}

}