Лабораторная работа №5

Динамическое программирование

Задание:

Составить программы, решающие следующие задачи. Входные данные находятся в текстовом файле input.txt. В первой строке файла указаны через пробел размеры массива, каждый из которых не превосходит 20. В последующих строках перечислены элементы массива - целые числа. Результатом работы программы должны быть оптимальное значение целевой функции и путь, при котором оно достигается.

Для указания пути (кроме задачи 6) вывести массив, отметив элементы пути либо другим цветом, либо некоторым другим образом.

Дан двумерный числовой массив размером N1xN2. Найти такой путь из клетки [i1, j1] в клетку [i2, j2], чтобы сумма чисел по данному пути была минимальной. Из каждой клетки массива допустимо двигаться вправо, влево, вверх или вниз. В клетки с нулями заходить нельзя. Числа i1, i2, j1, j2 вводятся с клавиатуры.

Алгоритм:

1. Ввод данных и проверка на корректность введённых данных:

Открытие файла input.txt для чтения. Если файл не удается открыть, выводится сообщение об ошибке, и программа завершает работу. Чтение размеров матрицы N1 и N2 и занесение их в соответствующие переменные. Проверка корректность размеров матрицы (0 < N1, N2 <= 20).

Чтение координат: запрашивает у пользователя начальные (i1, j1) и конечные (i2, j2) координаты для начала и конца пути, который нужно найти. Проверка корректности координат (0 <= i1, j1 < N1; 0 <= i2, j2 < N2).

Инициализация двумерного массива data[N1][N2], который заполняется значениями из файла.

Проверяет, что начальная и конечная точки не находятся в клетках со значением 0. Нет необходимости проверять путь между координатами, когда начальная или конечная клетка являются непроходимыми.

1. Алгоритм поиска кратчайшего пути (алгоритм Дейкстры):

Инициализация массивов для пути и сумм:

Создается двумерный массив minSum[N1][N2], который хранит минимальную сумму весов пути от начальной точки до каждой клетки. Изначально заполняется большими значениями (1e9). Минимальная сумма весов пути начальной точки устанавливается равным значению data[i1][j1].

Создается двумерный массив way[N1][N2], который хранит направление движения (прихода) в каждую клетку по кратчайшему пути ('<', '>', '^', 'v').

Запускается алгоритм, использующий флаг IsChanged, который указывает, была ли выполнена модификация массива сумм.

Внутри цикла программа проходит по каждой ячейке массива и проверяет соседей (верхнюю, нижнюю, левую, правую). Если соседняя клетка находится внутри границ матрицы, проходима (значение не равно 0) и сумма весов пути через текущую клетку меньше текущего минимального значения для соседней клетки, то: minSum для соседней клетки обновляется, way для соседней клетки устанавливается в направление из текущей клетки, а IsChanged устанавливается в true, сигнализируя об изменении.

1. Поиск пути:

Создается двумерный массив color[N1][N2], который помечает клетки, которые входят в минимальный путь.

Начиная с конечной точки (i2, j2) и двигаясь обратно, следуя указаниям из массива way, реконструируется путь, пока не достигнет начальной координаты (i1, j1).

Если минимальная сумму весов пути minSum[i2][j2] остается равным 1e9, значит путь не найден.

Происходит вывод всех элементов массива. Ячейки, которые входят в минимальный путь, отображаются другим цветом, используя функции Windows API для изменения цвета текста.

На экран выводится минимальная сумма весов на кратчайшем пути totalSum.

Текст программы:

#include <iostream>

#include <fstream>

#include <Windows.h>

#include <iomanip>

using namespace std;

int main() {

setlocale(LC\_ALL, "rus");

ifstream input("input.txt");

if (!input.is\_open()) {

cout << "Не удается открыть файл input.txt" << endl;

return 1;

}

int N1, N2;

input >> N1 >> N2;

if (N1 > 20 || N2 > 20) {

cout << "Размеры массива не должны превышать 20!" << endl;

return 2;

}

else if (N1 < 0 || N2 < 0) {

cout << "Размеры массива не могут быть меньше 0!" << endl;

return 3;

}

else {

cout << "N1 = " << N1 << ", N2 = " << N2 << endl << endl;

}

int i1, j1, i2, j2;

cout << "Введите координаты первой точки:\nx = ";

cin >> i1;

cout << "y = ";

cin >> j1;

cout << "Введите координаты второй точки:\nx = ";

cin >> i2;

cout << "y = ";

cin >> j2;

cout << endl;

if (i1 >= N1 || j1 >= N1 || j1 < 0 || i1 < 0) {

cout << "Координаты первой клетки не должны выходить за пределы число-вого массива!" << endl;

return 4;

}

if (i2 < 0 || j2 < 0 || i2 >= N2 || j2 >= N2) {

cout << "Координаты второй клетки не должны выходить за пределы число-вого массива!" << endl;

return 5;

}

bool minusValues = false;

int\*\* data = new int\* [N1];

for (int i = 0; i < N1; i++) {

data[i] = new int[N2];

for (int j = 0; j < N2; j++) {

input >> data[i][j];

if (data[i][j] < 0) {

minusValues = true;

}

}

}

if (minusValues) {

cout << "В матрице присутствуют отрицательные значения. Алгоритм не гарантирует корректный результат!\n" << endl;

}

if (!data[i1][j1]) {

cout << "Клетка начальных координат с нулём. В клетки с нулями захо-дить нельзя." << endl;

}

else if (!data[i2][j2]) {

cout << "Клетка конечных координат с нулём. В клетки с нулями заходить нельзя." << endl;

}

int\*\* minSum = new int\* [N1];

for (int i = 0; i < N1; i++) {

minSum[i] = new int[N2];

for (int j = 0; j < N2; j++) {

minSum[i][j] = 1e9; // Инициализация большим значением

}

}

minSum[i1][j1] = data[i1][j1];

char\*\* way = new char\* [N1];

for (int i = 0; i < N1; i++) {

way[i] = new char[N2];

for (int j = 0; j < N2; j++) {

way[i][j] = '0';

}

}

bool IsChanged = true;

while (IsChanged) {

IsChanged = false;

for (int i = 0; i < N1; i++) {

for (int j = 0; j < N2; j++) {

int dx[] = { -1, 0, 1, 0 };

int dy[] = { 0, 1, 0, -1 };

char dir[] = { '^', '>', 'v', '<' };

for (int k = 0; k < 4; ++k) {

int ni = i + dx[k];

int nj = j + dy[k];

if (ni >= 0 && ni < N1 && nj >= 0 && nj < N2) { // Не гра-ница ли

if (data[ni][nj] != 0) { // Не 0 ли

if (data[i][j] + minSum[ni][nj] < minSum[i][j]) { // Минимальный ли

minSum[i][j] = data[i][j] + minSum[ni][nj];

way[i][j] = dir[k];

IsChanged = true;

}

}

}

}

}

}

}

bool\*\* color = new bool\* [N1];

for (int i = 0; i < N1; i++) {

color[i] = new bool[N2];

for (int j = 0; j < N2; j++) {

color[i][j] = false;

}

}

int x = i2;

int y = j2;

if (minSum[x][y] == 1e9) {

cout << "Пути между координатами нет!" << endl;

return 6;

}

while (x != i1 || y != j1) {

color[x][y] = true;

if (way[x][y] == '<') y--;

else if (way[x][y] == '^') x--;

else if (way[x][y] == '>') y++;

else if (way[x][y] == 'v') x++;

}

color[x][y] = true;

int totalSum = 0;

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

for (int i = 0; i < N1; i++) {

for (int j = 0; j < N2; j++) {

if (color[i][j]) {

SetConsoleTextAttribute(hConsole, 12);

cout << setw(2) << data[i][j];

cout << " ";

totalSum += data[i][j];

SetConsoleTextAttribute(hConsole, 7);

}

else {

cout << setw(2) << data[i][j] << " ";

}

}

cout << endl;

}

cout << endl << "Минимальная сумма чисел пути: " << totalSum << endl << endl;

system("pause");

return 0;

}

Результат работы программы:

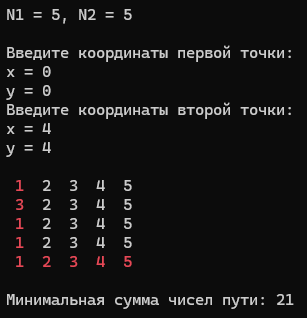


Рис. 1.

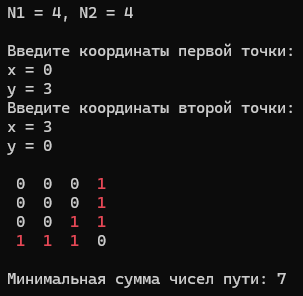


Рис. 2.

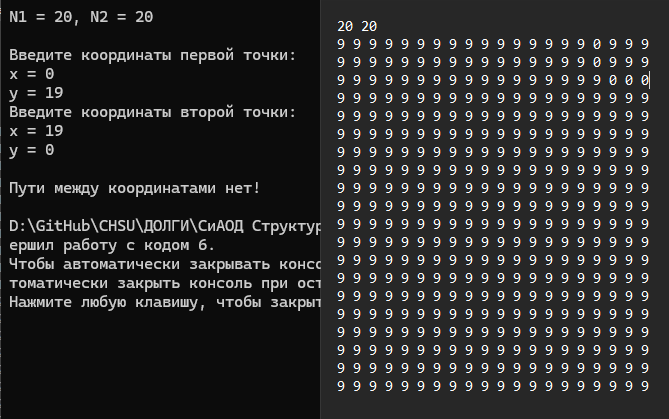


Рис. 3.

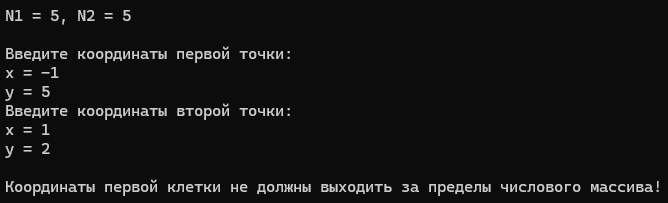


Рис. 4.

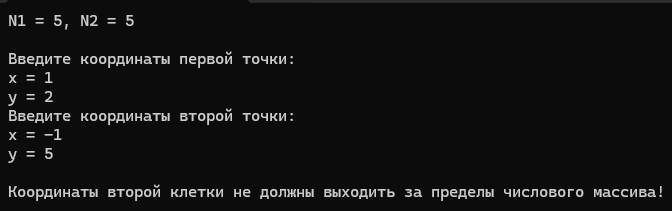


Рис. 5.

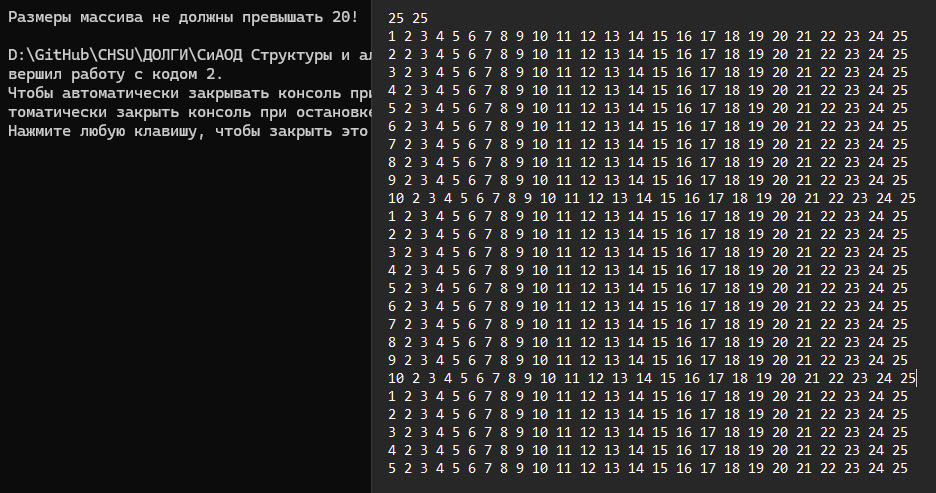


Рис. 6.