mМИНОБРНАУКИ РОССИИ

ФГБОУ ВО Череповецкий государственный университет

Институт информационных технологий

Кафедра: МПО ЭВМ

Дисциплина: Структуры и алгоритмы обработки данных

ЛАБОРАТОРНАЯ РАБОТА № 5

Динамическое программирование

Выполнила: Пикалова А.С.,

студент гр. 1ПИб-01-21оп.

Проверил: Пышницкий К.М.

Череповец, 2023 г.

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.

4**. 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 прохождения:

1. Проверяется условие что матрица размером не больше 20.
2. Берется начальная вершина и присваивается ей путь из файла
3. Находят все соседние клетки и вычисляется стоимость маршрута поочередно в каждую из них заходя.
4. Т.к. вероятно, что есть несколько путей в ту или иную вершину, пути придется пересматривать, тогда наименьшее значение ставиться в соответствие клетки.
5. После обработки начальная клетка отмечается как посещенная и активной становиться та, путь в которую из нач. клетки минимален.
6. Далее клетка снова помечается пройдённой, и вся процедура повторяется для следующей клеткой.
7. Алгоритм продолжается, пока все доступные клетки из начальной не будут исследованы.
8. Теперь найдем все кратчайшие пути между истоковой и всеми остальными клетками идя по меткам и устанавливаем флаги true.
9. Когда все клетки, будут помечены как посещенные, тогда работа алгоритма завершится, и все найденные пути будут кратчайшими и окрасятся в красный цвет.
10. Выведется сумма минимального пути.

**Алгоритм:**

Функция main:

1. Открытие файла input.txt;
2. Инициализация i1, j1, i2, j2;
3. Инициализация N1, N2;
4. Чтение из файла N1, N2;
5. Инициализация sum, и присваивание значения 1;
6. Создание двумерного массива data
7. Пока int i = 0; i < N1; i++:
   1. data[i] = new int[N2];
   2. Пока int j = 0; j < N2; j++;
      1. Читаем и записываем элементы массива из файла input >> data[i][j];
      2. sum += data[i][j];
8. Создание двумерного массива sums
9. Пока int i = 0; i < N1; i++:
   1. sums[i] = new int[N2];
   2. Пока int j = 0; j < N2; j++;
      1. sums[i][j] = sum;
10. sums[i1][j1] = data[i1][j1];
11. Создание двумерного символьного массива way
12. Пока int i = 0; i < N1; i++:
    1. way[i] = new char[N2];
    2. Пока int j = 0; j < N2; j++;
       1. way[i][j] = '0';
13. Инициализация флага IsChanged значением true;
14. Пока IsChanged = true:
    1. Метке IsChanged присваиваем значение false;
    2. Пока int i = 0; i < N1; i++
       1. Пока int j = 0; j < N2; j++
          1. Если не выполняется i == i1 && j == j1) && data[i][j] != 0, то:
             1. Если j - 1 >= 0, то:

Если data[i][j] + sums[i][j - 1] < sums[i][j], то:

sums[i][j] = data[i][j] + sums[i][j - 1]; складываем значения;

way[i][j] = '<';// присваиваем элементу массива значение <;

IsChanged = true;

* + - * 1. Если i - 1 >= 0, то:

Если data[i][j] + sums[i - 1][j] < sums[i][j], то:

sums[i][j] = data[i][j] + sums[i - 1][j]; складываем значения;

way[i][j] = '^';// присваиваем элементу массива значение ^;

IsChanged = true;

* + - * 1. Если j + 1 < N2, то:

Если data[i][j] + sums[i][j + 1] < sums[i][j], то:

sums[i][j] = data[i][j] + sums[i][j + 1]; складываем значения;

way[i][j] = '>'; // присваиваем элементу массива значение >

IsChanged = true;

* + - * 1. Если i + 1 < N1

Если data[i][j] + sums[i + 1][j] < sums[i][j], то:

sums[i][j] = data[i][j] + sums[i + 1][j]; складываем значения;

way[i][j] = 'v';// присваиваем элементу массива значение v;

IsChanged = true;

1. Создание двумерного символьного массива color
2. Пока int i = 0; i < N1; i++
   1. color[i] = new bool[N2];
   2. Пока int j = 0; j < N2; j++
      1. color[i][j] = false;
3. инициализируем элементы и присваиваем им значения:
   1. int i\_way = i2;
   2. int j\_way = j2;
4. Пока не выполняется i\_way == i1 && j\_way == j1
   1. color[i\_way][j\_way] = true; Присваиваем элементу значение true$
   2. Если way[i\_way][j\_way] == '<'
      1. j\_way--;
   3. Иначе если way[i\_way][j\_way] == '^'
      1. i\_way--;
   4. Иначе если way[i\_way][j\_way] == '^'
      1. j\_way++;
   5. Иначе если way[i\_way][j\_way] == 'v'
      1. i\_way++;
5. color[i\_way][j\_way] = true;
6. Инициализируем ko;
7. Пока int i = 0; i < N1; i++
   1. Пока int j = 0; j < N2; j++
      1. Если color[i][j] == true, то:
         1. SetConsoleTextAttribute(hConsole, 12);
         2. Выводим закрашенный элемент data[i][j];
         3. Складываем все значения ko;
         4. SetConsoleTextAttribute(hConsole, 7);
      2. Иначе
         1. data[i][j]; вывод элемента массива.
8. Выводим ko.

**Текст программы:**

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

void main() {

ifstream input("input.txt");

setlocale(LC\_ALL, "Russian");

int i1, j1, i2, j2;

cout << "i1="; //ввод 1 кледки

cin >> i1;

cout << "j1=";

cin >> j1;

cout << "i2=";//ввод 2 клетки

cin >> i2;

cout << "j2=";

cin >> j2;

int N1, N2; //инициализация размера массива

input >> N1 >> N2;//чтение из файла

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

int sum = 1;

int\*\* data = new int\* [N1]; // создание массива

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

data[i] = new int[N2];

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

input >> data[i][j]; //

sum += data[i][j];//

}

}

int\*\* sums = new int\* [N1];// путь

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

sums[i] = new int[N2];

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

sums[i][j] = sum;

}

}

sums[i1][j1] = data[i1][j1];//

char\*\* way = new char\* [N1];// будем присваивать метки в этот массив

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

way[i] = new char[N2];

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

way[i][j] = '0';// присваиваем 0

}

}

bool IsChanged = true; // флаг. прошли ли вершину

while (IsChanged == true) {

IsChanged = false;

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

for (int j = 0; j < N2; j++) { //идем по массиву

if (!(i == i1 && j == j1) && data[i][j] != 0) {

if (j - 1 >= 0) {//

if (data[i][j] + sums[i][j - 1] < sums[i][j]) {//если соседняя клетка меньше sums

sums[i][j] = data[i][j] + sums[i][j - 1];// присваиваем значение соседней клетки

way[i][j] = '<';// присваиваем элементу массива значение <

IsChanged = true;// массив изменен

}

}

if (i - 1 >= 0) {//

if (data[i][j] + sums[i - 1][j] < sums[i][j]) {//если соседняя клетка меньше sums

sums[i][j] = data[i][j] + sums[i - 1][j];// присваиваем значение соседней клетки

way[i][j] = '^';// присваиваем элементу массива значение ^

IsChanged = true;// массив изменен

}

}

if (j + 1 < N2)

if (data[i][j] + sums[i][j + 1] < sums[i][j]) {//если соседняя клетка меньше sums

sums[i][j] = data[i][j] + sums[i][j + 1];// присваиваем значение соседней клетки

way[i][j] = '>';// присваиваем элементу массива значение >

IsChanged = true;// массив изменен

}

}

if (i + 1 < N1) {//

if (data[i][j] + sums[i + 1][j] < sums[i][j]) {//если соседняя клетка меньше sums

sums[i][j] = data[i][j] + sums[i + 1][j];// присваиваем значение соседней клетки

way[i][j] = 'v';// присваиваем элементу массива значение v

IsChanged = true;// массив изменен

}

}

}

}

}

}

bool\*\* color = new bool\* [N1]; //массив цветов

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

color[i] = new bool[N2];

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

color[i][j] = false;// пройден

}

}

int i\_way = i2;//элементы матрицы стрелочек(меток)

int j\_way = j2;

while (!(i\_way == i1 && j\_way == j1)) {// идем обратно - ищем путь

color[i\_way][j\_way] = true;// присваиваем метке цвета значение true

if (way[i\_way][j\_way] == '<') {// если знак < то идем на клетку меньше j\_way--

j\_way--;

}

else if (way[i\_way][j\_way] == '^') {// если знак ^ то идем на клетку меньше i\_way--

i\_way--;

}

else if (way[i\_way][j\_way] == '>') {// если знак > то идем на клетку больше j\_way++

j\_way++;

}

else if (way[i\_way][j\_way] == 'v') {// если знак v то идем на клетку больше i\_way++

i\_way++;

}

}

color[i\_way][j\_way] = true;// присваиваем метке цвета значение true

int ko = 0;

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

cout << endl;

for (int i = 0; i < N1; i++) { //двигаемся по массиву

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

if (color[i][j] == true) {//если элемент тот, что нам нужен, то закрашиваем

SetConsoleTextAttribute(hConsole, 12);// изменение цвета

cout << data[i][j] << " ";//выводим закрашенный элемент

ko += data[i][j];

SetConsoleTextAttribute(hConsole, 7);// изменение цвета в белый

}

else {//иначе просто выводим элемент клетки

cout << data[i][j] << " ";

}

}

cout << endl;

}

cout << "Сумма подсчитанных элементов:" << endl;

cout << ko << endl;

}

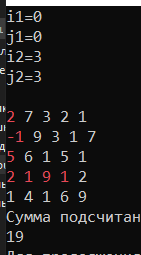
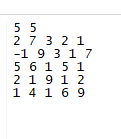
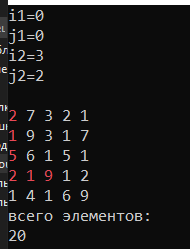
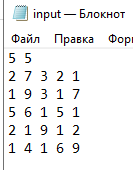
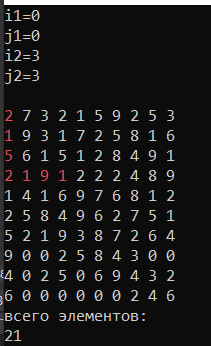
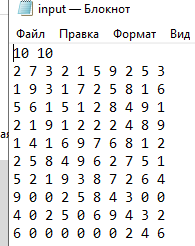
else

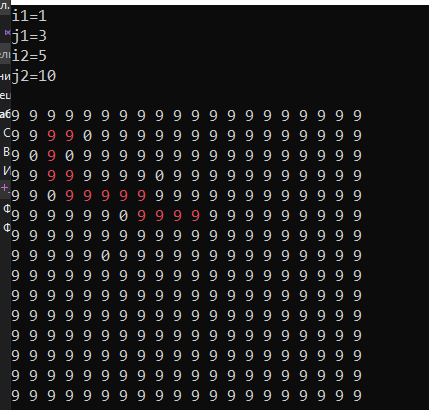
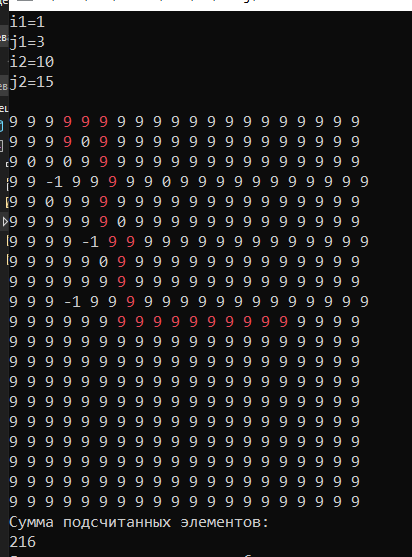
cout << "Матрица больше 20!" << endl;

system("pause");

}

Тестирование:

Вывод: на основе своего варианта была написана программа, которая ищет путь минимальных значений от заданных пользователем параметров и выводящая сумму этих элементов.