Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Отчет №3

По лабораторной работе

Тема:” ***Нахождение кратчайшего пути в связном графе*** ”

Выполнил:

Студ. гр.ИИ-23

Романюк А.П.

Проверила:

Глущенко Т. А.

Брест 2023

Вариант №8

***Задание.***

1. Алгоритмом *Дейкстры* вычислить кратчайшие пути от вершины  ко всем остальным вершинам графа, указав их длины и сам путь для каждой пары вершин (последовательность вершин). Для хранения длин кратчайших путей рекомендуется использовать бинарную кучу *(min-heap).*
2. Алгоритмом *Флойда-Уоршелла* вычислить кратчайшие пути между всеми парами вершин взвешенного графа, указав их длины и пути.

Код программы:

#include <iostream>

#include <vector>

#include <queue>

#include <climits>

using namespace std;

const int INF = 999999;

vector<pair<int, int>> smej[1000];

int dist[1000]{};

void dekstra(int start) {

priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;

pq.push(make\_pair(0, start));

dist[start] = 0;

while (!pq.empty()) {

int u = pq.top().second;

pq.pop();

for (auto v : smej[u]) {

int w = v.first;

int to = v.second;

if (dist[u] + w < dist[to]) {

dist[to] = dist[u] + w;

pq.push(make\_pair(dist[to], to));

}

}

}

}

int main() {

cout << "введите кол-во вершин графа, кол-во ребер\n";

int v, r, start;

cin >> v >> r;

int graph[100][100];

for (int i = 1; i <= v; ++i) {

dist[i] = INF;

for (int j = 1; j <= v; j++) {

graph[i][j] = INF;

if (i == j) graph[i][j] = 0;

}

}

cout << "введите вершины и вес ребра\n";

for (int i = 1; i <= r;i++) {

int u, v, w;

cin >> u >> v >> w;

smej[u].push\_back(make\_pair(w, v));

graph[u][v] = w;

}

int choise;

cin >> choise;

cout << "1.deikstra\n2.orshala\n";

switch (choise) {

case 1:

cout << "введите стартовую точку";

cin >> start;

dekstra(start);

for (int i = 1; i <= v; ++i) {

if (dist[i] == INF) {

cout << "INF" << endl;

continue;

}

cout << "x" << i << " = " << dist[i] << endl;

}

break;

case 2:

for (int k = 1; k <= v; k++) {

for (int i = 1; i <= v; i++) {

for (int j = 1; j <= v; j++) {

if (graph[i][j] > graph[i][k] + graph[k][j]) {

graph[i][j] = graph[i][k] + graph[k][j];

}

}

}

}

for (int i = 1; i <= v; i++) {

for (int j = 1; j <= v; j++) {

if (i == j) continue;

cout << i << " -> " << j << " = " << graph[i][j]<<endl;

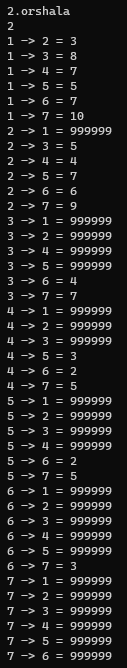
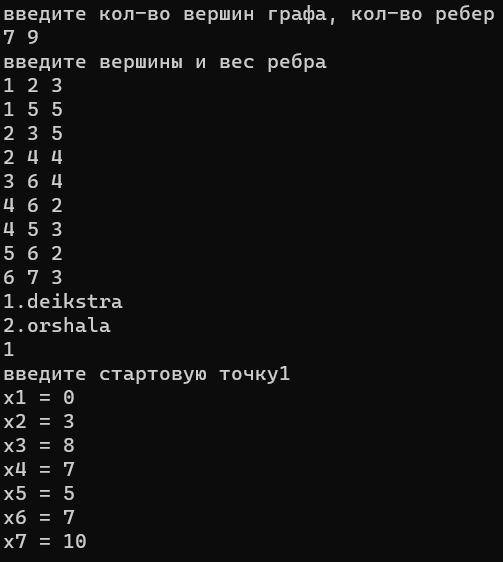
}

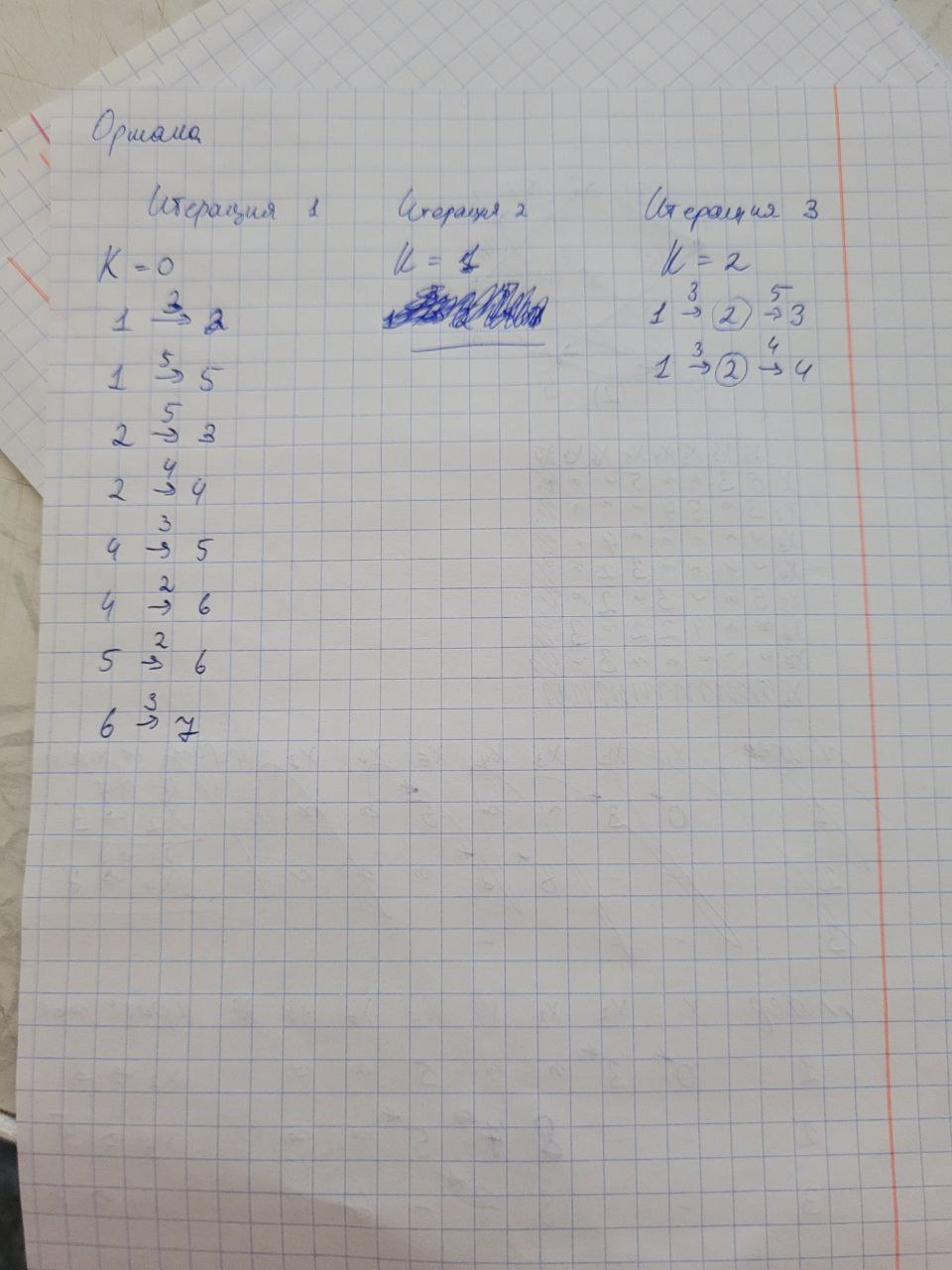
}

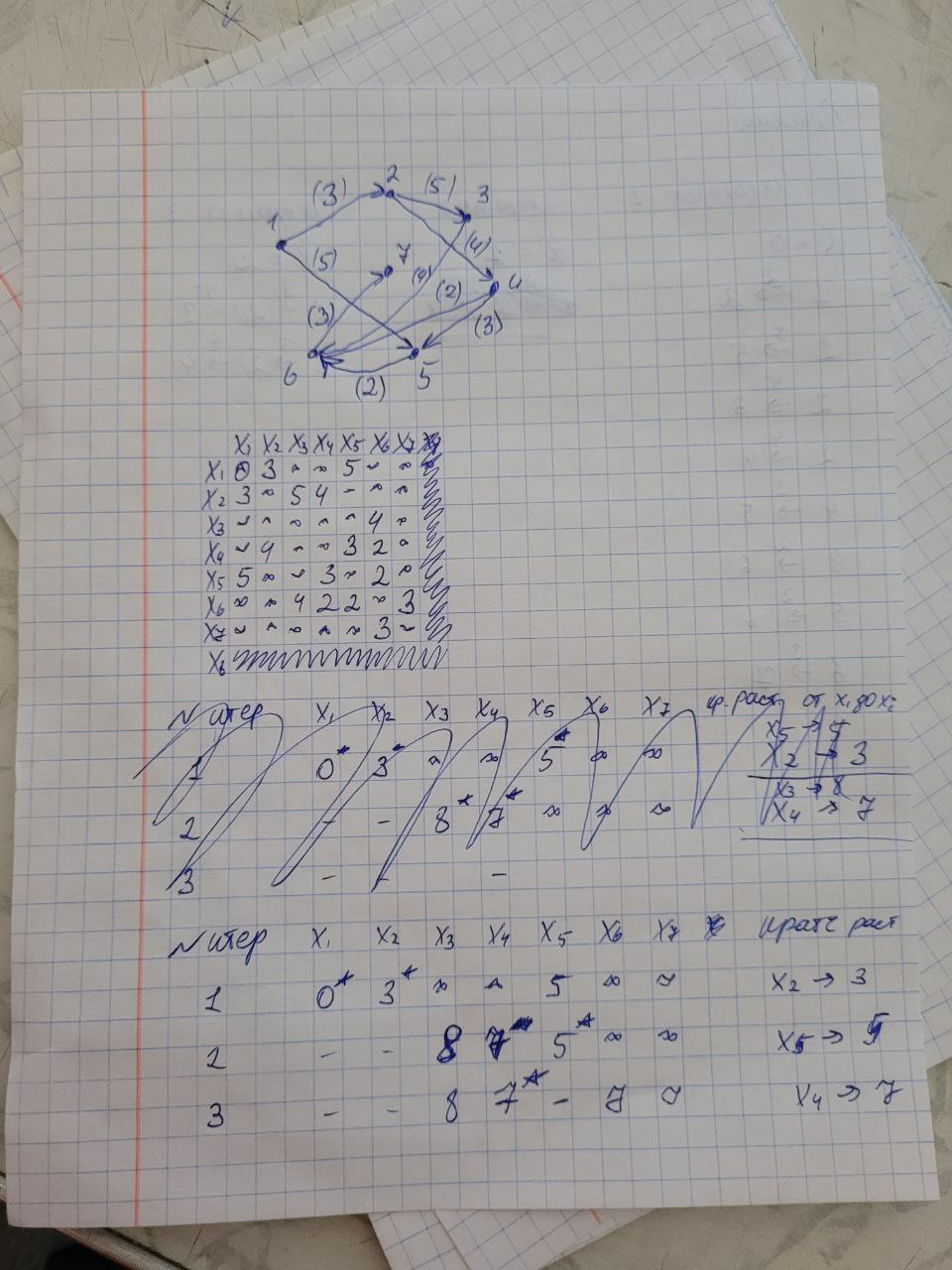
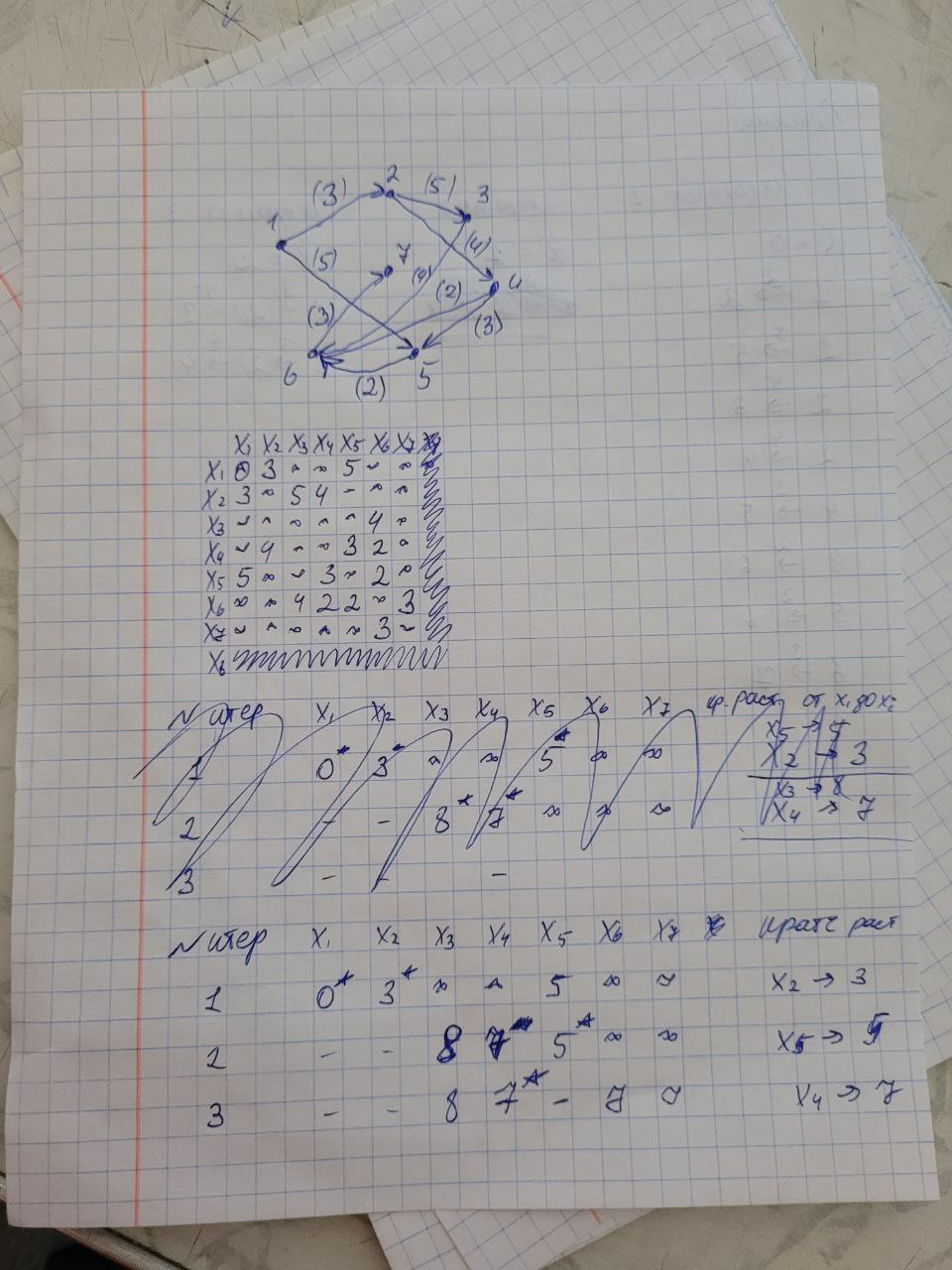
break;

}

}







1. задача о количестве островов: **200. Number of Islands** ресурса https://leetcode.com /

class Solution {

public:

int numIslands(vector<vector<char>>& grid) {

int islands=0;

for(int i=0;i<grid.size();i++)

{

for(int j=0;j<grid[0].size();j++)

{

if(grid[i][j]=='0' || grid[i][j]=='2')

{

}

else

{

islands++;

maketwo(grid,i,j+1);

maketwo(grid,i,j-1);

maketwo(grid,i-1,j);

maketwo(grid,i+1,j);

}

}

}

return islands;

}

void maketwo(vector<vector<char>>& grid,int a,int b)

{

int r=grid.size();

int c=grid[0].size();

if(a<0 || b<0 || a>=r || b>=c)

return;

if(grid[a][b]=='0')

return;

if(grid[a][b]=='2')

return;

if(grid[a][b]=='1')

{

grid[a][b]='2';

maketwo(grid,a,b+1);

maketwo(grid,a,b-1);

maketwo(grid,a-1,b);

maketwo(grid,a+1,b);

}

}

};



1. Описать алгоритм решения задачи о *8 ферзях*.

Шаги алгоритма

Создать пустую доску размером 8x8.

Разместить первого ферзя в любой клетке первой строки.

Рекурсивно рассмотреть все возможные варианты расстановки оставшихся 7 ферзей на оставшихся 7 строках.

Для каждого из этих вариантов проверить, не находятся ли ферзи под ударом друг друга.

Если ферзи не находятся под ударом, то сохранить текущую расстановку ферзей в качестве решения задачи.

Вернуться к предыдущей расстановке и продолжить перебор.