深度优先算法：

代码：

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <ctime>

using namespace std;

const int num = 100;

int map[8][8];

int x\_coord[4] = {-1, 0, 1, 0};

int y\_coord[4] = {0, 1, 0, -1};

int x\_queue[num];

int y\_queue[num];

int n;

void out(ofstream& file) {

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

file << '(' << x\_queue[i] << ',' << y\_queue[i] << ')' << endl;

}

}

bool dfs(int x, int y) {

x\_queue[n] = x;

y\_queue[n] = y;

if (x == 7 && y == 7) return true;

else {

bool flag = false;

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

int x\_now = x + x\_coord[i];

int y\_now = y + y\_coord[i];

if ((x\_now >= 0 && x\_now < 8) && (y\_now >= 0 && y\_now < 8) && (map[x\_now][y\_now] == 0)) {

flag = true;

map[x\_now][y\_now] = 2;

n++;

if (dfs(x\_now, y\_now))

break;

else

flag = false;

map[x\_now][y\_now] = 0;

n--;

}

}

return flag;

}

}

int main() {

srand(time(0));

n = 0;

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

map[i][j] = rand() % 2;

}

}

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

printf("%d ",map[i][j]);

}

printf("\n");

}

ofstream outputFile("output.txt");

if (!outputFile) {

cout << "Failed to open the file." << endl;

return 1;

}

if (dfs(0, 0)) {

outputFile << "存在破解路径" << endl;

out(outputFile);

} else {

outputFile << "不存在破解路径" << endl;

outputFile << n << endl;

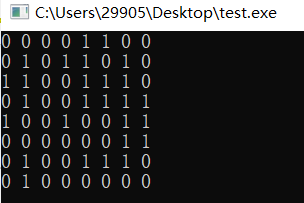
}

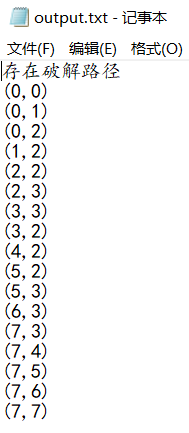
outputFile.close();

return 0;

}

运行截图：





广度优先算法：

代码：

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <ctime>

using namespace std;

const int num = 100;

int map[8][8];

int x[4] = {-1,0,1,0}; //向当前地点相邻的四个点遍历(坐标顺序分别为:上，右，下，左)

int y[4] = {0,1,0,-1};

int x\_queue[num]; //存储队列中的点的坐标

int y\_queue[num];

int route[num]; //存储每个点的前驱点

int front, end; //队头，队尾

void out(int n, ofstream& file) { //输出函数

if (n == 0) {

file << '(' << x\_queue[0] << ',' << y\_queue[0] << ')' << endl;

return;

}

out(route[n], file);

file << '(' << x\_queue[n] << ',' << y\_queue[n] << ')' << endl;

}

bool bfs() { //遍历函数

while (front != end) {

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

int x\_now = x\_queue[front] + x[i];

int y\_now = y\_queue[front] + y[i];

if ((x\_now >= 0 && x\_now < 8) && (y\_now >= 0 && y\_now < 8) && (map[x\_now][y\_now] == 0)) {

//当周围的点符合条件，入队，条件为点在矩阵内，点不是墙，点没有走过

map[x\_now][y\_now] = 2; //走过的点赋值为2，不重复走

x\_queue[end] = x\_now;

y\_queue[end] = y\_now;

route[end] = front; //记录该入队的点的前驱，即当前正在遍历的点

if (x\_now == 7 && y\_now == 7)

return true;

end++;

}

}

front++;

}

return false;

}

int main() {

srand(time(0)); // Seed the random number generator

front = 0;

end = 1;

x\_queue[0] = 0;

y\_queue[0] = 0;

route[0] = 0;

map[0][0] = 2;

// Generate random map

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

map[i][j] = rand() % 2; // Generate 0 or 1 randomly

}

}

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

printf("%d ",map[i][j]);

}

printf("\n");

}

ofstream outputFile("output.txt"); // Open the output file

if (!outputFile) {

cout << "Failed to open the file." << endl;

return 1;

}

if (bfs()) {

outputFile << "存在破解路径" << endl;

out(end - 1, outputFile);

} else {

outputFile << "不存在破解路径" << endl;

}

outputFile.close(); // Close the output file

return 0;

}

运行截图：

