Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Решения задач блока 4

Алгоритмам и структурам данных

Работу выполнил:

Гаврилин О.С.

Группа:

P3230

Санкт-Петербург,

2025

1. **M**

#include <algorithm>

#include <iostream>

#include <queue>

#include <string>

#include <vector>

using namespace std;

void solve() {

int n, m;

cin >> n >> m;

int start\_x, start\_y, end\_x, end\_y;

cin >> start\_x >> start\_y >> end\_x >> end\_y;

--start\_x, --start\_y, --end\_x, --end\_y;

vector<string> grid(n);

for (auto& row : grid)

cin >> row;

const int INF = 1e9;

vector<vector<int>> cost(n, vector<int>(m, INF));

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

for (int j = 0; j < m; ++j)

cost[i][j] = (grid[i][j] == '.') ? 1 : (grid[i][j] == 'W' ? 2 : INF);

vector<vector<int>> dist(n, vector<int>(m, INF));

vector<vector<pair<int, int>>> prev(n, vector<pair<int, int>>(m, {-1, -1}));

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

dist[start\_x][start\_y] = 0;

pq.push({

0, {start\_x, start\_y}

});

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

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

char move\_dir[4] = {'N', 'E', 'S', 'W'};

while (!pq.empty()) {

auto [d, pos] = pq.top();

auto [x, y] = pos;

pq.pop();

if (d > dist[x][y])

continue;

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

int nx = x + dx[i], ny = y + dy[i];

if (nx >= 0 && nx < n && ny >= 0 && ny < m && cost[nx][ny] != INF) {

int nd = d + cost[nx][ny];

if (nd < dist[nx][ny]) {

dist[nx][ny] = nd;

prev[nx][ny] = {x, y};

pq.push({

nd, {nx, ny}

});

}

}

}

}

if (dist[end\_x][end\_y] == INF) {

cout << -1 << '\n';

return;

}

cout << dist[end\_x][end\_y] << '\n';

string path;

int x = end\_x, y = end\_y;

while (x != start\_x || y != start\_y) {

auto [px, py] = prev[x][y];

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

if (px + dx[i] == x && py + dy[i] == y)

path += move\_dir[i];

x = px, y = py;

}

reverse(path.begin(), path.end());

cout << path << '\n';

}

int main() {

ios::sync\_with\_stdio(false);

cin.tie(nullptr);

solve();

return 0;

}

**Описание:** Имеем взвешенный граф, точку начала и конца пути – при помощи алгоритма Дейкстры можем оценить наименьшую цену попадания в каждую вершину графа из исходной точки. Для увеличения эффективности алгоритма используем приоритетную очередь.

**Сложность по времени:**

**Сложность по памяти:**

**2. N**

**Описание:**

**Сложность по времени:**

**Сложность по памяти:**

**3. O**

**Описание:**

**Сложность по времени:**

**Сложность по памяти:**

**4. P**

**Описание:**

**Сложность по времени:**

**Сложность по памяти:**