

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АДЫГЕЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Инженерно-физический факультет
Кафедра автоматизированных систем обработки информации и
управления

ОТЧЕТ ПО ПРАКТИКЕ

*Написать программу Обход графа в глубину и
ширину. Вариант 8 .*

2 курс, группа 2ИВТ2

Выполнили:

_____ А. Ю. Коваленко

«___» _____ 2023 г.

Руководитель:

_____ С. В. Теплоухов

«___» _____ 2023 г.

Майкоп, 2023 г.

1. Введение

1.1 Обход графа в ширину

1.2 Код данной задачи

1.3 Скриншот программы обхода графа в ширину

1.4 Обход графа в глубину

1.5 Код данной задачи

1.6 Скриншот программы обхода графа в глубину

1.1. Обход графа в ширину

Теория

Поиск в ширину работает путём последовательного просмотра отдельных уровней графа, начиная с узла-источника.

Рассмотрим все рёбра, выходящие из узла. Если очередной узел является целевым узлом, то поиск завершается; в противном случае узел добавляется в очередь. После того, как будут проверены все рёбра, выходящие из узла, из очереди извлекается следующий узел, и процесс повторяется.

Работа алгоритма

1. Неформальное описание

2. Поместить узел, с которого начинается поиск, в изначально пустую очередь.

- Извлечь из начала очереди узел и пометить его как развёрнутый.
- Если узел является целевым узлом, то завершить поиск с результатом «успех».

В противном случае, в конец очереди добавляются все преемники узла, которые ещё не развёрнуты и не находятся в очереди.

3. Если очередь пуста, то все узлы связного графа были просмотрены, следовательно, целевой узел недостижим из начального; завершить поиск с результатом «неудача».

1.2. Код приложения

```
#include <iostream>
#include <queue>
#include <vector>

using namespace std;

vector<vector<int>> graph;
```

```

vector<bool> visited;

bool bfs(int startNode, int targetNode) {
    queue<int> q;

    q.push(startNode);
    visited[startNode] = true;

    while (!q.empty()) {
        int node = q.front();
        q.pop();

        if (node == targetNode) {
            return true; // Целевой узел достигнут, поиск успешен
        }

        for (int neighbor : graph[node]) {
            if (!visited[neighbor]) {
                q.push(neighbor);
                visited[neighbor] = true;
            }
        }
    }

    return false; // Очередь пуста, целевой узел недостижим, поиск неудачен
}

int main() {
    setlocale(LC_ALL, "Russian");
    // Пример графа в виде списка смежности
    int numNodes, numEdges;
    cout << "Введите количество вершин: ";
    cin >> numNodes;
    cout << "Введите количество ребер: ";
    cin >> numEdges;

    // Инициализация графа и массива посещенных вершин
    graph.resize(numNodes);
    visited.resize(numNodes, false);

    // Ввод ребер
    cout << "Введите ребра (вершина1 вершина2):" << endl;
    for (int i = 0; i < numEdges; i++) {
        int node1, node2;

```

```

    cin >> node1 >> node2;

    // Добавление ребер в список смежности
    graph[node1].push_back(node2);
    graph[node2].push_back(node1);
}

int startNode, targetNode;
cout << "Введите начальную вершину: ";
cin >> startNode;
cout << "Введите целевую вершину: ";
cin >> targetNode;

// Выполнение обхода в ширину, начиная с начальной вершины, и проверка достижимости
if (bfs(startNode, targetNode)) {
    cout << "Целевая вершина достижима." << endl;
}
else {
    cout << "Целевая вершина недостижима." << endl;
}

return 0;
}

```

1.3. Скриншот программы обхода графа в ширину

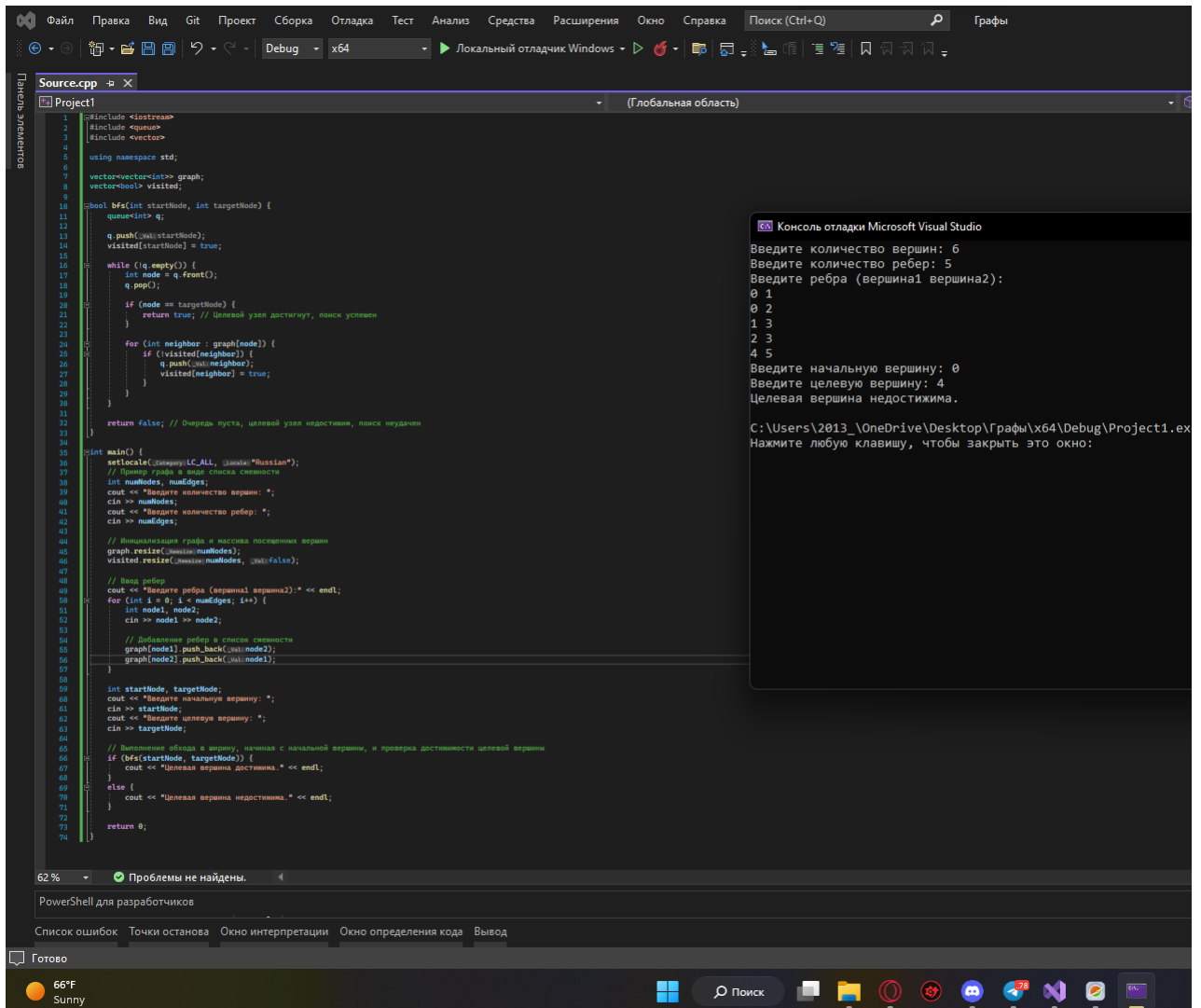


Рис. 1. В ширину

1.4. Обход графа в глубину

Теория

Стратегия поиска в глубину, как и следует из названия, состоит в том, чтобы идти «вглубь» графа, насколько это возможно. Алгоритм поиска описывается рекурсивно: перебираем все исходящие из рассматриваемой вершины рёбра. Если ребро ведёт в вершину, которая не была рассмотрена ранее, то запускаем алгоритм от этой нерассмотренной вершины, а после возвращаемся и продолжаем перебирать рёбра. Возврат происходит в том случае, если в рассматриваемой вершине не осталось рёбер, которые ведут в нерассмотренную вершину. Если после завершения алгоритма не все вершины были рассмотрены, то необходимо запустить алгоритм от одной из нерассмотренных вершин.

Работа алгоритма

Пусть задан граф $G = (V, E)$, где V — множество вершин графа, E — множество ребер графа. Предположим, что в начальный момент времени все вершины графа окрашены в белый цвет. Выполним следующие действия:

Пройдём по всем вершинам $u \in V$

Если вершина белая, выполним для неё $\text{DFS}(v)$.

Процедура DFS (параметр — вершина)

1. Перекрашиваем вершину u в серый цвет.
2. Для всякой вершины w , смежной с вершиной u и окрашенной в белый цвет, рекурсивно выполняем процедуру $\text{DFS}(w)$.
3. Перекрашиваем вершину u в чёрный цвет.

Часто используют двухцветные метки — без серого, на 1-м шаге красят сразу в чёрный цвет.

1.5. Код приложения

```
#include <iostream>
#include <vector>

using namespace std;

vector<vector<int>> graph;
vector<bool> used;

void dfs(int node_index) {
    used[node_index] = true; // Помечаем текущую вершину как посещенную (черную)

    // Проходимся по всем смежным вершинам
    for (int i = 0; i < graph[node_index].size(); ++i) {
        int neighbor = graph[node_index][i];
        if (!used[neighbor]) { // Если смежная вершина не была посещена (неокрашена)
            dfs(neighbor); // Рекурсивно вызываем dfs для нее
        }
    }
}

int main() {
    setlocale(LC_ALL, "Russian");
    int numNodes, numEdges;
    cout << "Введите количество вершин: ";
    cin >> numNodes;
```

```

cout << "Введите количество ребер: ";
cin >> numEdges;

// Инициализация графа и массива посещенных вершин
graph.resize(numNodes);
used.resize(numNodes, false);

// Ввод ребер
cout << "Введите ребра (вершина1 вершина2):" << endl;
for (int i = 0; i < numEdges; i++) {
    int node1, node2;
    cin >> node1 >> node2;

    // Добавление ребер в список смежности
    graph[node1].push_back(node2);
    graph[node2].push_back(node1);
}

int startNode;
cout << "Введите начальную вершину: ";
cin >> startNode;

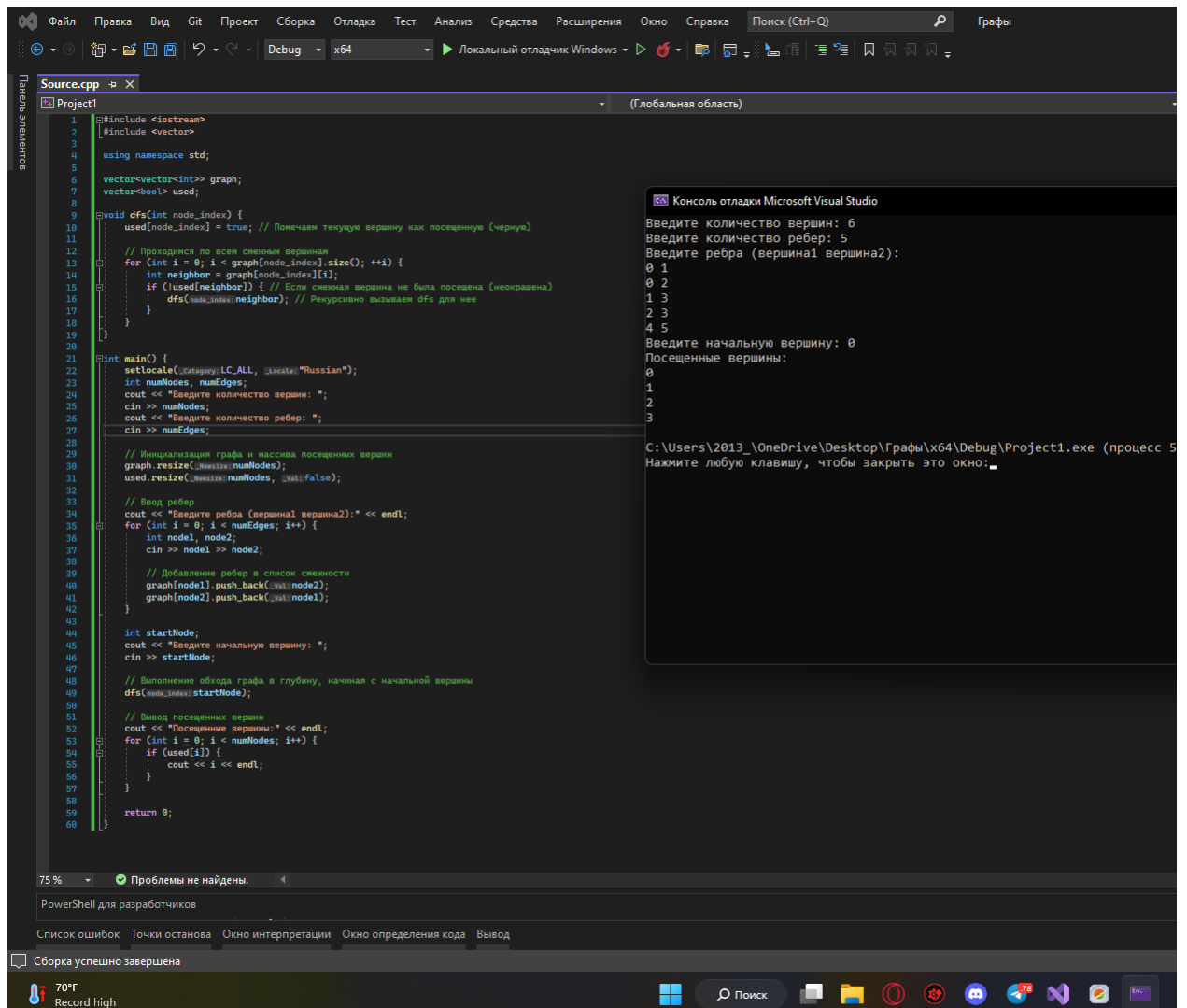
// Выполнение обхода графа в глубину, начиная с начальной вершины
dfs(startNode);

// Вывод посещенных вершин
cout << "Посещенные вершины:" << endl;
for (int i = 0; i < numNodes; i++) {
    if (used[i]) {
        cout << i << endl;
    }
}

return 0;
}

```

1.6. Скриншот программы обхода графа в глубину



```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6 vector<vector<int>> graph;
7 vector<bool> used;
8
9 void dfs(int node_index) {
10     used[node_index] = true; // Помечаем текущую вершину как посещенную (черную)
11
12     // Проходимся по всем смежным вершинам
13     for (int i = 0; i < graph[node_index].size(); ++i) {
14         int neighbor = graph[node_index][i];
15         if (!used[neighbor]) { // Если смежная вершина не была посещена (неокрашена)
16             dfs(neighbor); // Рекурсивно вызываем dfs для нее
17         }
18     }
19 }
20
21 int main() {
22     setlocale(_LC_ALL, "Russian");
23     int numNodes, numEdges;
24     cout << "Введите количество вершин: ";
25     cin >> numNodes;
26     cout << "Введите количество ребер: ";
27     cin >> numEdges;
28
29     // Инициализация графа и массива посещенных вершин
30     graph.resize(numNodes);
31     used.resize(numNodes, false);
32
33     // Ввод ребер
34     cout << "Введите ребра (вершина1 вершина2):" << endl;
35     for (int i = 0; i < numEdges; i++) {
36         int node1, node2;
37         cin >> node1 >> node2;
38
39         // Добавление ребер в список смежности
40         graph[node1].push_back(node2);
41         graph[node2].push_back(node1);
42     }
43
44     int startNode;
45     cout << "Введите начальную вершину: ";
46     cin >> startNode;
47
48     // Выполнение обхода графа в глубину, начиная с начальной вершины
49     dfs(startNode);
50
51     // Вывод посещенных вершин
52     cout << "Посещенные вершины:" << endl;
53     for (int i = 0; i < numNodes; i++) {
54         if (used[i]) {
55             cout << i << endl;
56         }
57     }
58
59     return 0;
60 }
```

Консоль отладки Microsoft Visual Studio

Введите количество вершин: 6
Введите количество ребер: 5
Введите ребра (вершина1 вершина2):
0 1
0 2
1 3
2 3
4 5
Введите начальную вершину: 0
Посещенные вершины:
0
1
2
3

C:\Users\2013_OneDrive\Desktop\Графы\x64\Debug\Project1.exe (процесс 5
Нажмите любую клавишу, чтобы закрыть это окно: _

75% Проблемы не найдены.

PowerShell для разработчиков

Список ошибок Точки останова Окно интерпретации Окно определения кода Вывод

Сборка успешно завершена

70°F
Record high

Рис. 2. В глубину

2. библиографические ссылки

Для изучения «внутренностей» $\text{T}_\text{E}\text{X}$ необходимо изучить [1], а для использования $\text{L}_\text{A}\text{T}_\text{E}\text{X}$ лучше почитать [2, 3].

Список литературы

- [1] Кнут Д.Э. Всё про $\text{T}_\text{E}\text{X}$. — Москва: Изд. Вильямс, 2003 г. 550 с.
- [2] Львовский С.М. Набор и верстка в системе $\text{L}_\text{A}\text{T}_\text{E}\text{X}$. — 3-е издание, исправленное и дополненное, 2003 г.
- [3] Воронцов К.В. $\text{L}_\text{A}\text{T}_\text{E}\text{X}$ в примерах. 2005 г.