|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 1 | | |
| по дисциплине «Дискретная математика» | | |
| **Алгоритмы поиска в ширину и глубину** | | |
|  | | |
|  | Группа ПМ-21 | пОРСИН Данил |
| Вариант 9 | дУДКИНА МАРИЯ |
|  | Егупов Иван |
|  | ЛИ АНДРЕЙ |
|  |  |
|  |  |
| Преподаватели | Рояк михаил эммануилович |
|  | ступаков илья михайлович |
| Новосибирск, 2023 | | |

1. **Задание**

Определить, существует ли путь между двумя заданными вершинами в ориентированном графе. Граф в памяти представлен в виде матрицы.

1. **Алгоритм решения задачи**

Входные данные: G = где – количество вершин, – количество ребер

Выходные данные: , где – путь не существует, – путь существует

Решение: «G – дерево» и «G – связный граф и m=n–1. Граф записан в виде вектора с векторами, где в каждом векторе «2го порядка» находятся все вершины, смежные соответствующие номеру этого.

***Поиск в глубину***: Двигаемся от начальной вершины к конечной, помечая уже посещенные вершины, до тех пор, пока не будет найден путь или не конец графа.

Для размещения всех посещенных вершин будем использовать стек – curpath.

***Поиск в ширину***: Двигаемся от начальной вершины к соседней и так далее, помечая уже посещенные, до тех пор, пока не будет найден путь или не конец графа.

Для реализации обхода используем очередь – next, состоящая ещё не из посещенных вершин графа.

1. **Текст программы**
   1. **Поиск в глубину**

#include <vector>

#include <fstream>

#include <set>

#include <iostream>

#include <queue>

#include <stack>

#include <set>

#include <string>

using namespace std;

static int path\_exist\_DFS(vector<vector<int>> graph, int a, int b) {

int start = a - 1;

int end = b - 1;

vector<bool> visited(graph.size());

stack<int> curpath;

curpath.push(start);

visited[start] = true;

while (!curpath.empty()) {

int curpoint = curpath.top();

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

if (graph[curpoint][i] == 1 && !visited[i]) {

if (i == end) {

return 1;

}

curpoint = i;

i = -1;

curpath.push(curpoint);

visited[curpoint] = true;

}

}

curpath.pop();

}

return 0;

}

* 1. **Поиск в ширину**

#include <vector>

#include <fstream>

#include <set>

#include <iostream>

#include <queue>

#include <stack>

#include <set>

#include <string>

using namespace std;

static int path\_exist\_BFS(vector<vector<int>> graph, int a, int b) {

int start = a - 1;

int end = b - 1;

vector<bool> visited(graph.size());

queue<int> next;

next.push(start);

visited[start] = true;

while (!next.empty()) {

int curpoint = next.front();

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

if (graph[curpoint][i] == 1 && !visited[i]) {

if (i == end) {

return 1;

}

next.push(i);

visited[i] = true;

}

}

next.pop();

}

return 0;

}

1. **Тесты для поиска ошибок**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Входные данные** | **Назначение** | **Ожидаемый результат** | **Результат работы программы** | |
| **Поиск в глубину** | **Поиск в ширину** |
| **1** | 1 2  1 0 | Простейший тест. Найти путь из 1 в 2. Граф состоит из одной вершины и не содержит ребер. | 0 | 0 | 0 |
| **2** | 4 1  4 0 | Найти путь из 4 в 1.  Граф состоит из изолированных точек без ребер. | 0 | 0 | 0 |
| **3** | 2 1  3 5  1 3  3 1  3 2  2 3  1 2 | Найти путь из 2 в 1.  Граф состоит из 3х вершин и 5 ребер. | 1 | 1 | 1 |
| **4** | 1 3  5 6  1 2  2 3  3 4  4 1  1 5  5 3 | Найти путь из 1 в 3.  Граф состоит из 5 вершин и 6 ребер | 1 | 1 | 1 |
| **5** | 2 5  5 7  1 2  1 3  1 4  1 5  2 3  3 4  4 5 | Найти путь из 2 в 5.  Граф состоит из 5 вершин и 7 ребер. | 1 | 1 | 1 |

1. **Тесты для оценки фактической сложности**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Входные данные** | | **Время работы программы в ms** | |
| **Поиск в глубину** | **Поиск в ширину** |
| **1** | 1 ребро у каждой вершины | Вершин: 5000 | 53 | 65 |
| **2** | 4 ребра у каждой вершины | Вершин: 5000 | 54 | 67 |
| **3** | 1 ребро у каждой вершины | Вершин: 10000 | 207 | 252 |
| **4** | 4 ребра у каждой вершины | Вершин: 10000 | 206 | 261 |
| **5** | 1 ребро у каждой вершины | Вершин: 20000 | 959 | 1092 |
| **6** | 4 ребра у каждой вершины | Вершин: 20000 | 984 | 1050 |
| **7** | 1 ребро у каждой вершины | Вершин: 40000 | 4612 | 5910 |
| **8** | 4 ребра у каждой вершины | Вершин: 40000 | 7612 | 6301 |
| **9** | 1 ребро у каждой вершины | Вершин: 5000 | 77 | 44 |
| **10** | 4 ребра у каждой вершины | Вершин: 5000 | 76 | 45 |
| **11** | 1 ребро у каждой вершины | Вершин: 10000 | 299 | 172 |
| **12** | 4 ребра у каждой вершины | Вершин: 10000 | 289 | 165 |
| **13** | 1 ребро у каждой вершины | Вершин: 20000 | 1219 | 685 |
| **14** | 4 ребра у каждой вершины | Вершин: 20000 | 1236 | 688 |
| **15** | 1 ребро у каждой вершины | Вершин: 40000 | 7366 | 3668 |
| **16** | 4 ребра у каждой вершины | Вершин: 40000 | 7870 | 3889 |

На данных графиках видно, что зависимость количества вершин и времени квадратичная- О(), как при

поиске в глубину, так и при поиске в ширину.