



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практической работе №7

по дисциплине «Структуры и алгоритмы обработки данных»
по теме «основные алгоритмы работы с графами»

Выполнил:

Студент группы ИКБО-13-22

Руденко Алексей Дмитриевич

Проверил:

ассистент Муравьёва Е.А.

МОСКВА 2023 г.

Практическая работа № 7

Цель работы

Получить навыки применения методов, позволяющих сократить число переборов в задачах, которые могут быть решены только методом перебора всех возможных вариантов решения.

Задание 1

Задание:

1. Разработать алгоритм решения задачи с применением метода, указанного в варианте и реализовать программу.

Вариант 26:

26	Разработать программу поиска и вывода всех гамильтоновых циклов в произвольном графе.	метод ветвей и границ
----	---	-----------------------

2. Оценить количество переборов при решении задачи стратегией «в лоб» - грубой силы. Сравнить с числом переборов при применении метода.

Код решения задачи методом ветвей и границ

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

class Graph {
    int vertices;
public:
    vector<vector<int>> graph;

    Graph(int v) : vertices(v) {
        graph.resize(vertices, vector<int>(vertices, 0));
    }

    bool isSafe(int v, int pos, const vector<int>& path) {
        if (graph[path[pos - 1]][v] == 0)
            return false;

        for (int vertex : path)
            if (vertex == v)
                return false;

        return true;
    }

    void hamiltonianCycle(vector<int>& path, vector<bool>& visited) {
        if (path.size() == vertices) {
            cout << "Гамильтонов цикл: ";
            for (int vertex : path)
                cout << vertex << " ";
            cout << path[0] << endl;
        }
    }
};
```

```

        return;
    }

    for (int v = 0; v < vertices; ++v) {
        if (isSafe(v, path.size(), path) && !visited[v]) {
            path.push_back(v);
            visited[v] = true;

            hamiltonianCycle(path, visited);

            visited[v] = false;
            path.pop_back();
        }
    }
}

void findHamiltonianCycles() {
    vector<int> path;
    vector<bool> visited(vertices, false);

    path.push_back(0);
    visited[0] = true;

    hamiltonianCycle(path, visited);
}

};

int main() {
    setlocale(LC_ALL, "Russian");
    Graph g(5);

    vector<vector<int>> adjacencyMatrix = {
        {0, 1, 0, 1, 0},
        {1, 0, 1, 1, 1},
        {0, 1, 0, 0, 1},
        {1, 1, 0, 0, 1},
        {0, 1, 1, 1, 0}
    };

    g.graph = adjacencyMatrix;

    g.findHamiltonianCycles();

    return 0;
}

```

Рисунок 1 – листинг решения задачи методом динамического программирования

Этот код решает задачу поиска гамильтоновых циклов методом границ и ветвей.

Алгоритм работает, избегая повторных вычислений циклов благодаря методу ветвей и границ, чего не скажешь об обходе грубой силой.

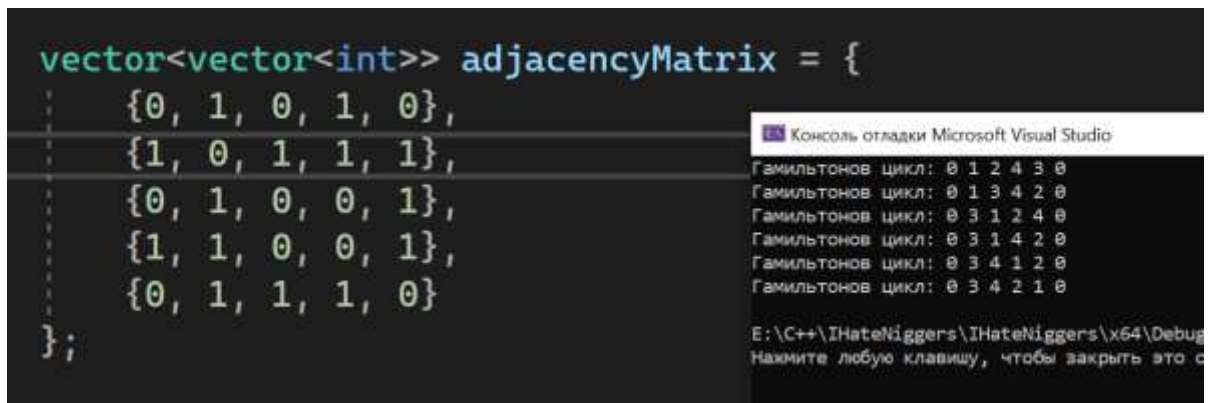


Рисунок 2 – тест №1 решения методом динамического программирования

Код решения задачи методом “грубой силы”

```
#include <iostream>
#include <vector>
using namespace std;

const int MAX = 5;

void printSolution(vector<int>& path, int V) {
    cout << "Гамильтонов цикл: ";
    for (int i = 0; i < V; ++i)
        cout << path[i] << " ";
    cout << path[0] << endl;
}

bool isValid(int v, int pos, vector<int>& path, bool graph[MAX][MAX]) {
    if (!graph[path[pos - 1]][v])
        return false;

    for (int i = 0; i < pos; ++i)
        if (path[i] == v)
            return false;

    return true;
}

void hamiltonianCycle(bool graph[MAX][MAX], int V, vector<int>& path, int pos) {
    if (pos == V) {
        if (graph[path[pos - 1]][path[0]] == 1) {
            printSolution(path, V);
            return;
        }
    }

    for (int v = 1; v < V; ++v) {
        if (isValid(v, pos, path, graph)) {
            path[pos] = v;
            hamiltonianCycle(graph, V, path, pos + 1);
            path[pos] = -1;
        }
    }
}

void findAllHamiltonianCycles(bool graph[MAX][MAX], int V) {
    vector<int> path(V, -1);
    path[0] = 0;

    hamiltonianCycle(graph, V, path, 1);
}
```

```

int main() {
    setlocale(LC_ALL, "");
    int V;
    cout << "Введите количество вершин в графе: ";
    cin >> V;

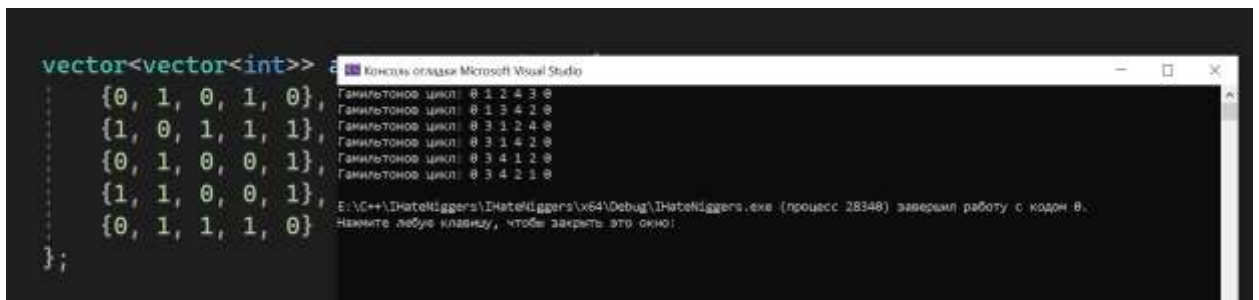
    bool graph[MAX][MAX] = {
        {0, 1, 1, 1, 1},
        {1, 0, 1, 1, 1},
        {1, 1, 0, 0, 1},
        {1, 1, 0, 0, 1},
        {1, 1, 1, 1, 0}
    };

    cout << "Гамильтоновы циклы в графе:\n";
    findAllHamiltonianCycles(graph, V);

    return 0;
}

```

Рисунок 3 – листинг решения задачи “грубой силы”



```

vector<vector<int>>>
{
    {0, 1, 0, 1, 0},
    {1, 0, 1, 1, 1},
    {0, 1, 0, 0, 1},
    {1, 1, 0, 0, 1},
    {0, 1, 1, 1, 0}
};

```

Гамильтонов цикл: 0 1 2 4 3 0
Гамильтонов цикл: 0 1 3 4 2 0
Гамильтонов цикл: 0 3 1 2 4 0
Гамильтонов цикл: 0 3 1 4 2 0
Гамильтонов цикл: 0 3 4 1 2 0

E:\C++\IhateNiggers\IhateNiggers\x64\Debug\IhateNiggers.exe (процесс 28348) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно.

Рисунок 4 – тест №1 решения методом “грубой силы”

Вывод

Метод ветвей и границ не всегда сможет выдать правильный ответ, но сможет определённо сильно снизить круг поиска истинного гамильтонова цикла, то есть на некоторых маленьких объёмах данных есть смысл использовать перебор, а далее переходить на этот метод.