МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №4.2

по дисциплине: Дискретная математика
тема: «Циклы»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
ст. пр. Рязанов Юрий Дмитриевич
ст. пр. Бондаренко Татьяна Владимировна

Белгород 2023 г.

Лабораторная работа №4.2

Циклы

Цель работы: изучить разновидности циклов в графах, научиться генерировать случайные графы, определять их принадлежность к множеству эйлеровых и гамильтоновых графов, находить все эйлеровы и гамильтоновы циклы в графах.

1. Разработать и реализовать алгоритм генерации случайного графа, содержащего n вершин и m ребер.

```
#include "../../libs/alg/alg.h"

#include <random>

typedef Node<int> IntNode;

Graph<Edge<IntNode>>* generateRandomGraph(int nodesCount, int edgeCount) {
    auto graph = new AdjacencyMatrixGraph<Edge<IntNode>>();

    for (int i = 1; i <= nodesCount; i++) {
        Node<int> n(i);

        graph->addNode(n);
    }

    std::vector<std::pair<int, int>> edges;
    for (int i = 1; i <= nodesCount; i++) {
        for (int j = i + 1; j <= nodesCount; j++) {
            if (i == j) continue;

            edges.push_back({i, j});
        }
    }

    std::random_device rd;
    std::mt19937 g(rd());

    std::shuffle(edges.begin(), edges.end(), g);

    for (int i = 0; i < edgeCount && i < edges.size(); i++)
        graph->addEdge({{(*graph)[edges[i].first],
        (*graph)[edges[i].second]}, false});

    return graph;
}
```

2. Написать программу, которая:

    а) в течение десяти секунд генерирует случайные графы, содержащие n вершин и m ребер;

б) для каждого полученного графа определяет, является ли он эйлеровым или гамильтоновым;

в) подсчитывает общее количество сгенерированных графов и количество графов каждого типа.

Результат работы программы представить в виде таблицы. Величину h подобрать такой, чтобы в таблице количество строк было в диапазоне от 20 до 30. *task1.tpp*

```
#include "../lab11/graph.tpp"

template <typename E, typename N>
bool AdjacencyMatrixGraph<E, N>::isHamiltonian() {
    if (this->nodes.size() < 3) return false;

    std::vector<bool> cache(this->nodes.size(), false);

    return hasHamiltonianCycle(0, 0, cache);
}

template <typename E, typename N>
bool AdjacencyMatrixGraph<E, N>::hasHamiltonianCycle(int originIndex, int startIndex, std::vector<bool>&
↪  takenNodes) {
    takenNodes[startIndex] = true;

    bool anyElementFound = false;
    for (int i = 0; i < this->nodes.size(); i++) {
        if (takenNodes[i] || this->edges[startIndex][i] == nullptr) continue;

        anyElementFound = true;

        if (hasHamiltonianCycle(originIndex, i, takenNodes)) return true;
    }

    if (!anyElementFound) {
        if (this->edges[startIndex][originIndex] == nullptr) {
            takenNodes[startIndex] = false;

            return false;
        }

        for (auto takenFlag : takenNodes)
            if (!takenFlag) {
                takenNodes[startIndex] = false;

                return false;
            }

        return true;
    }

    takenNodes[startIndex] = false;
    return false;
}
```

```cpp
template <typename E, typename N>
bool AdjacencyMatrixGraph<E, N>::isEuler() {
    if (this->nodes.size() < 3) return false;

    // Строим матрицу M
    BoolMatrixRelation linkMatrix(this->nodes.size(), [this](int x, int y) {
        return this->edges[x - 1][y - 1] != nullptr;
    });

    // Получаем матрицу C = I + M+ и формируем из C факторможество
    auto factorSet = BoolMatrixRelation::getIdentity(this->nodes.size())
                                        .unite(linkMatrix.transitiveClosureWarshall(nullptr))
                                        .getPackedFactorSet();

    // Если в полученном факторможестве несколько классов эквивалентности, то множество несвязное.
    for (int i = 1; i < factorSet.size(); i++)
        if (factorSet[0] != factorSet[i]) return false;

    for (int i = 0; i < this->nodes.size(); i++) {
        int nodePow = 0;
        for (int j = 0; j < this->nodes.size(); j++)
            nodePow += (this->edges[i][j] != nullptr);

        // Степень каждой вершины должна быть чётна.
        if (nodePow % 2 != 0) return false;
    }

    return true;
}
```

main.cpp

```cpp
#include "task1.h"

#include <thread>
#include <future>
#include <unistd.h>

struct Report {
    int n;
    int edgesCount;
    int hamilthonCount;
    int eulerCount;
    int totalCount;
};

#define LOWER_N 8
#define UPPER_N 10
#define ELEMENTS_FOR_N 10

#define CORES_AMOUNT 10
std::atomic<int> takenCores;
```

```cpp
int main() {
    std::vector<std::pair<std::chrono::_V2::system_clock::time_point, std::future<Report>>> pool;
    for (int n = LOWER_N; n <= UPPER_N; n++) {
        for (int i = 0; i < ELEMENTS_FOR_N; i++) {
            while (takenCores >= CORES_AMOUNT) {
                sleep(1);
                std::cout << "Working on it..." << "\n";
            }

            int e = n;
            if (i != 0)
                e += (i * ((((n * (n - 1)) / 2 - n))))  / (ELEMENTS_FOR_N - 1);

            takenCores++;
            pool.push_back({std::chrono::system_clock::now(), std::async(std::launch::async, [n, e] {
                int totalGraphsGenerated = 0;
                int hamilthonGraphs = 0;
                int eulerGraphs = 0;
                auto start = std::chrono::system_clock::now();
                while (std::chrono::system_clock::now() - start < std::chrono::seconds(10)) {
                    auto graph = generateRandomGraph(n, e);
                    totalGraphsGenerated++;

                    if (graph->isHamiltonian()) hamilthonGraphs++;
                    if (graph->isEuler()) eulerGraphs++;

                    delete graph;
                }

                takenCores--;

                return (Report){n, e, hamilthonGraphs, eulerGraphs, totalGraphsGenerated};
            })});
        }
    }



    std::cout << "Waiting for results...\n\n";
    for (auto& future : pool) {
        if (future.second.wait_until(future.first + std::chrono::seconds(60)) != std::future_status::ready) {
            std::cout << "====================\n";
            std::cout << "Thread is not responing, consider no result is present.\n";

            continue;
        }

        auto t = future.second.get();

        std::cout << "\\hline\n";
        std::cout << t.n << "&"<<t.edgesCount << "&" << t.eulerCount << "&" << t.hamilthonCount << "&" <<
        ↪  t.totalCount << "\\\\\n";
    }
```

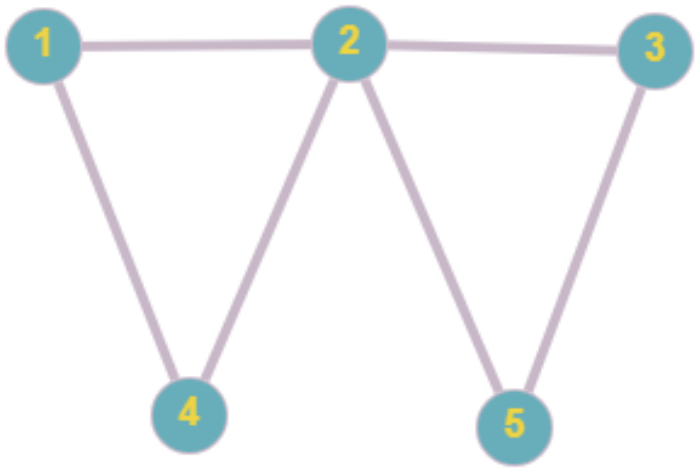
```
    getchar();

    return 0;
}
```

Результаты работы программы

Количество вершин	Количество рёбер	Количество графов		
		эйлеровых	гамильтоновых	всех
8	8	102	102	138589
8	10	647	4777	139096
8	12	834	30119	121321
8	14	897	75835	121929
8	16	1766	195543	223236
8	19	918	113577	114899
8	21	845	103548	103659
8	23	898	102016	102016
8	25	0	94443	94443
8	28	0	92151	92151
9	9	82	82	325377
9	12	366	6651	164324
9	15	532	58577	157293
9	18	491	105194	133994
9	21	526	126056	131892
9	24	476	124748	125456
9	27	415	109102	109134
9	30	433	110253	110253
9	33	1174	101690	101690
9	36	98078	98078	98078
10	10	6	6	126670
10	13	175	2991	209716
10	17	184	37283	115934
10	21	169	75209	95293
10	25	193	94958	98776
10	29	249	145783	146400
10	33	182	96687	96711
10	37	160	86866	86866
10	41	0	85518	85518
10	45	0	79227	79227

3. Выполнить программу при n = 8,9,10 и сделать выводы.
   Чем больше рёбер в графе, тем больше вероятность того, что он окажется гамильтоновым. Рост вероятности того, что граф окажется эйлеровым с ростом количества рёбер сначала растёт, потом - падает.

4. Привести пример диаграммы графа, который является эйлеровым, но не гамильтоновым. Найти в нем все эйлеровы циклы.



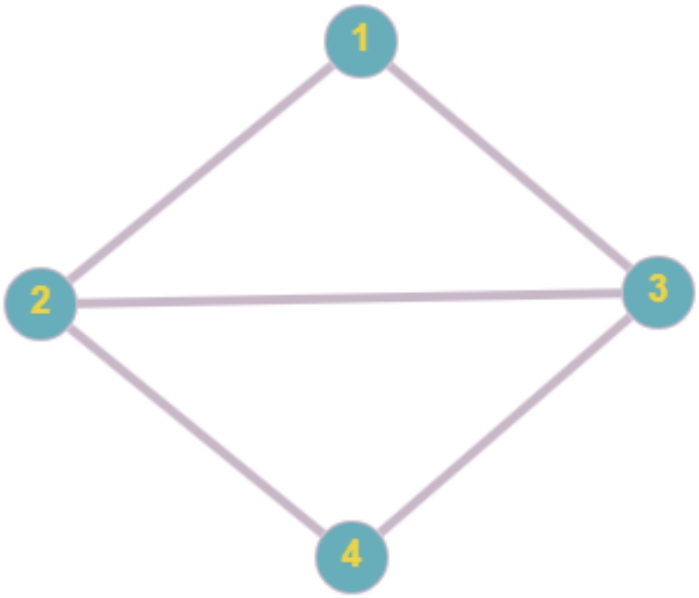
Эйлеровы циклы:
$\{1, 2, 3, 5, 2, 4, 1\}$
$\{2, 3, 5, 2, 4, 1, 2\}$
$\{3, 5, 2, 4, 1, 2, 3\}$
$\{5, 2, 4, 1, 2, 3, 5\}$
$\{2, 4, 1, 2, 3, 5, 2\}$
$\{4, 1, 2, 3, 5, 2, 4\}$

$\{1, 4, 2, 5, 3, 2, 1\}$
$\{4, 2, 5, 3, 2, 1, 4\}$
$\{2, 5, 3, 2, 1, 4, 2\}$
$\{5, 3, 2, 1, 4, 2, 5\}$
$\{3, 2, 1, 4, 2, 5, 3\}$
$\{2, 1, 4, 2, 5, 3, 2\}$

$\{1, 2, 5, 3, 2, 4, 1\}$
$\{2, 5, 3, 2, 4, 1, 2\}$
$\{5, 3, 2, 4, 1, 2, 5\}$
$\{3, 2, 4, 1, 2, 5, 3\}$
$\{2, 4, 1, 2, 5, 3, 2\}$
$\{4, 1, 2, 5, 3, 2, 4\}$

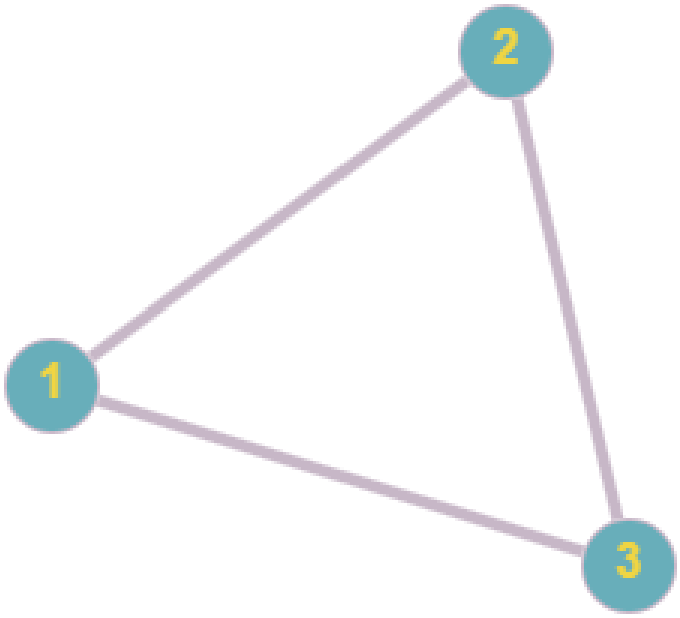
$\{1, 4, 2, 3, 5, 2, 1\}$
$\{4, 2, 3, 5, 2, 1, 4\}$
$\{2, 3, 5, 2, 1, 4, 2\}$
$\{3, 5, 2, 1, 4, 2, 3\}$
$\{5, 2, 1, 4, 2, 3, 5\}$
$\{2, 1, 4, 2, 3, 5, 2\}$

5. Привести пример диаграммы графа, который является гамильтоновым, но не эйлеровым. Найти в нем все гамильтоновы циклы.



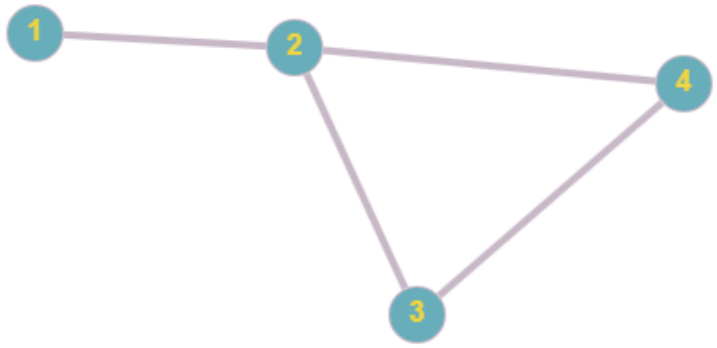
Гамильтоновы циклы:
$\{1, 2, 4, 3, 1\}$
$\{2, 4, 3, 1, 2\}$
$\{4, 3, 1, 2, 4\}$
$\{3, 1, 2, 4, 3\}$
$\{1, 3, 4, 2, 1\}$
$\{3, 4, 2, 1, 3\}$
$\{4, 2, 1, 3, 4\}$
$\{2, 1, 3, 4, 2\}$

6. Привести пример диаграммы графа, который является эйлеровым и гамильтоновым. Найти в нем все эйлеровы и гамильтоновы циклы.



Эйлеровы циклы:
$\{1, 2, 3, 1\}$
$\{2, 3, 1, 2\}$
$\{3, 1, 2, 3\}$
$\{1, 3, 2, 1\}$
$\{3, 2, 1, 3\}$
$\{2, 1, 3, 2\}$
Гамильтоновы циклы:
$\{1, 2, 3, 1\}$
$\{2, 3, 1, 2\}$
$\{3, 1, 2, 3\}$
$\{1, 3, 2, 1\}$
$\{3, 2, 1, 3\}$
$\{2, 1, 3, 2\}$

7. Привести пример диаграммы графа, который не является ни эйлеровым, ни гамильтоновым.



Вывод: в ходе лабораторной работы изучили разновидности циклов в графах, научились

генерировать случайные графы, определять их принадлежность к множеству эйлеровых и гамильтоновых графов, находить все эйлеровы и гамильтоновы циклы в графах.