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 = 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;
    double h;
    int edgesCount;
    int hamilthonCount;
    int eulerCount;
    int totalCount;
};

#define LOWER_N 8
#define UPPER_N 10
#define ELEMENTS_FOR_N 20

#define CORES_AMOUNT 10
```

```cpp
std::atomic<int> takenCores;

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++) {
        double h = ((n * (n - 1)) / 2 - 1.0 * n) / (ELEMENTS_FOR_N);

        for (double e = n; e <= ((n * (n - 1)) / 2); e += h) {
            while (takenCores >= CORES_AMOUNT) {
                sleep(1);
                std::cout << "Working on it..." << "\n";
            }

            takenCores++;
            pool.push_back({std::chrono::system_clock::now(), std::async(std::launch::async, [n, h, 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, static_cast<int>(e));
                    totalGraphsGenerated++;

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

                    delete graph;
                }

                takenCores--;

                return (Report){n, h, static_cast<int>(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 << "====================\n";
        std::cout << "n = " << t.n << "\n";
        std::cout << "h = " << t.h << "\n";
        std::cout << "edges = " << t.edgesCount << "\n";
        std::cout << "hamilthon = " << t.hamilthonCount << "\n";
```

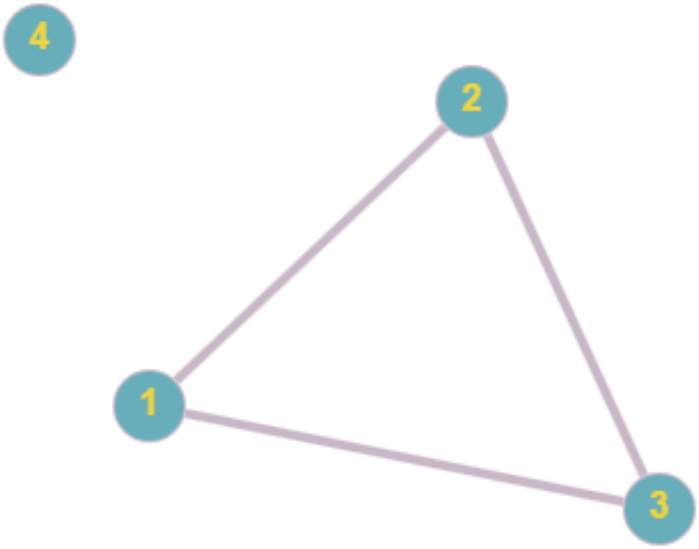
```
        std::cout << "euler = " << t.eulerCount << "\n";
        std::cout << "total = " << t.totalCount << "\n\n";
    }

    getchar();

    return 0;
}
```

Результаты работы программы (n = 8, h = 1)

Количество вершин	Количество рёбер	Количество графов		
		эйлеровых	гамильтоновых	всех
8	8	103	103	200820
8	10	540	2897	183051
8	12	866	18442	164452
8	14	1172	54080	166376
8	16	1023	84642	147675
8	18	1132	112615	146409
8	20	1080	128453	144606
8	22	1007	121600	127881
8	24	981	128928	131627
8	26	991	129678	130709
8	28	1039	131660	132050
9	9	32	32	256243
9	11	258	1303	233387
9	14	584	22228	215795
9	17	565	66885	171215
9	19	581	100800	166941
9	22	618	133776	161496
9	25	572	135365	144932
9	27	556	146455	151423
9	30	554	133836	135290
9	33	496	128841	129238
9	36	446	124971	125083
10	10	8	8	222246
10	13	111	1475	211436
10	17	202	23527	146719
10	20	248	61836	141350
10	24	247	102376	134103
10	27	248	110193	123522
10	31	238	117785	121892
10	34	274	123179	124767
10	38	357	170638	171228
10	41	325	167164	167356
10	45	306	159020	159064

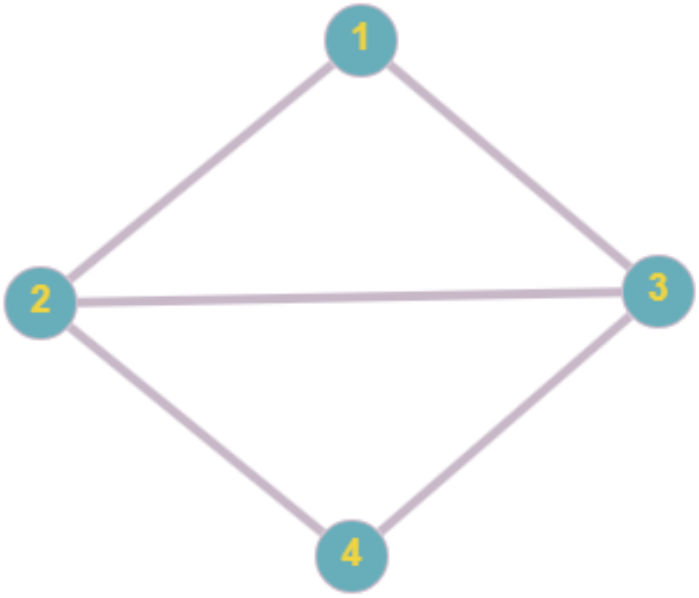
3. Выполнить программу при n = 8,9,10 и сделать выводы.
   Чем больше рёбер в графе, тем больше вероятность того, что он окажется гамильтоновым. Рост вероятности того, что граф окажется эйлеровым с ростом количества рёбер не такой значительный.

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



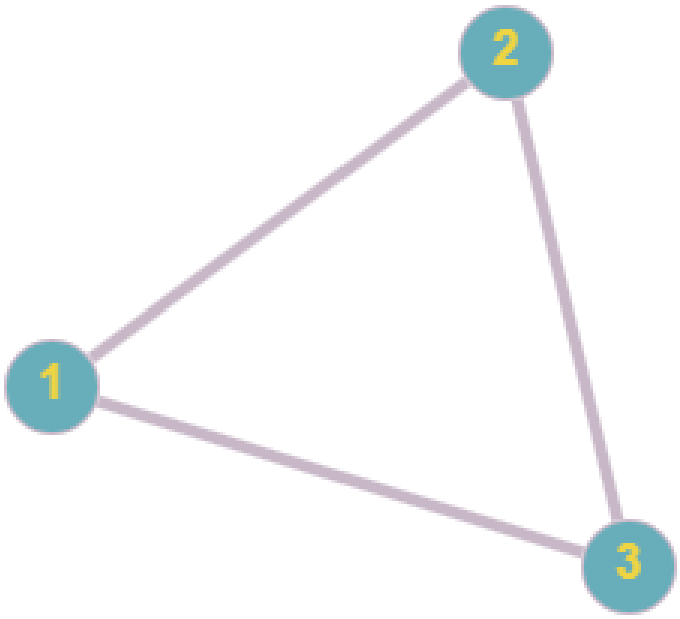
   Эйлеровы циклы:
   $\{1, 2, 3, 1\}$
   $\{2, 3, 1, 2\}$
   $\{3, 1, 2, 3\}$
   $\{1, 3, 2, 1\}$
   $\{3, 2, 1, 3\}$
   $\{2, 1, 3, 2\}$

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.



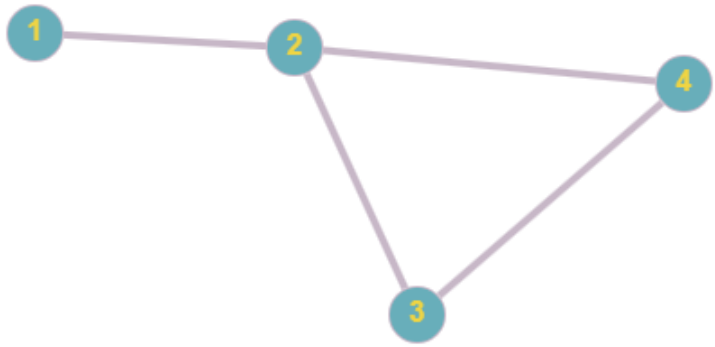
Эйлеровы циклы:

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

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.