БУ ВО Ханты-Мансийского автономного округа – Югры

«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

ОТЧЁТ

по лабораторной работе №9

по дисциплине: «Алгоритмы и структуры данных»

Выполнил: студент(ка) группы №609-31,

Гаврилов Е.Е.

Принял: старший преподаватель кафедры АиКС

Назаров Е.В.

Сургут

2025г.

**Цель работы:** изучить алгоритмы поиска эйлеровых и гамильтоновых путей на графах, исследовать асимптотические характеристики этих алгоритмов, закрепить навыки структурного и объектно-ориентированного программирования.

**Задание:**

1. Используя исходный код предыдущей лабораторной работы, модифицировать его и разработать методы генерации эйлерова и гамильтонова графов.
2. Реализовать методы поиска эйлерова и гамильтонова циклов, используя любые известные алгоритмы, направленные на решение этих задач.
3. Исследовать асимптотическую сложность указанных алгоритмов поиска циклов. Построить графики зависимости времени работы алгоритмов от количества ребер/количества вершин. Подготовить отчет, в который включить указанные графики и выводы по работе.

**Ход работы:**

Листинг 1. Функция генерации эйлеровых графов.

SGraph GenerateEulersGraph(int size)

{

SGraph graph(size);

if (size == 3 || size == 4)

{

for (int i = 0; i < size; i++)

graph.AddEdge(i, (i + 1) % size);

return graph;

}

int next\_v = 0;

next\_v = random(4, size);

if (next\_v == size - 1)

next\_v = size;

for (int i = 0; i < next\_v; i++)

graph.AddEdge(i, (i + 1) % next\_v);

while (next\_v < size)

{

int a = random(0, next\_v - 1);

int b = random(0, next\_v - 1);

while (a == b)

b = random(0, next\_v - 1);

int next\_vs = random(next\_v + 2, size);

if (size - next\_vs <= 1)

next\_vs = size;

int local\_v = random(next\_v + 1, next\_vs - 1);

graph.AddEdge(a, next\_v++);

while (next\_v < local\_v)

{

graph.AddEdge(next\_v - 1, next\_v);

next\_v++;

}

next\_v--;

graph.AddEdge(b, next\_v++);

graph.AddEdge(a, next\_v++);

while (next\_v < next\_vs)

{

graph.AddEdge(next\_v - 1, next\_v);

next\_v++;

}

next\_v--;

graph.AddEdge(b, next\_v++);

}

return graph;

}

Листинг 2. Функция генерации гамильтоновых графов.

SGraph GenerateHamiltonGraph(int size)

{

SGraph graph(size);

std::vector<int> vec(size);

for (int i = 0; i < size; i++)

vec[i] = i;

std::random\_device rd;

std::mt19937 g(rd());

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

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

graph.AddEdge(vec[i],vec[(i+1)%size]);

}

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

graph.AddEdge(random(0,size-1),random(0,size-1));

}

return graph;

}

Листинг 3. Функция нахождения эйлорова пути

std::vector<int> GetEulersCycle(SGraph graph)

{

std::stack<int> s;

std::vector<int> c;

s.push(0);

while (!s.empty())

{

int v = s.top();

int i;

for (i = 0; i < graph.Size(); i++)

{

if (v != i && graph(v, i) < MAXDOUBLE)

{

graph.DeleteEdge(v, i);

graph.DeleteEdge(i, v);

s.push(i);

break;

}

}

if (i == graph.Size())

{

c.push\_back(s.top());

s.pop();

}

}

return c;

}

Листинг 4. Функция нахождения гамильтонова пути

std::vector<int> GetHamiltonCycle(SGraph graph)

{

std::vector<int> path;

std::vector<bool> visited(graph.Size(), false);

std::stack<std::pair<int, int>> stack;

stack.push({0, 0});

visited[0] = true;

path.push\_back(0);

while (!stack.empty())

{

auto &[current, next] = stack.top();

if (path.size() == graph.Size() && graph(path.back(), 0) < MAXDOUBLE && path.back() != 0)

{

return path;

}

bool found = false;

for (int neighbor = next; neighbor < graph.Size(); ++neighbor)

{

if (graph(current, neighbor) < MAXDOUBLE && !visited[neighbor])

{

stack.top().second = neighbor + 1;

stack.push({neighbor, 0});

visited[neighbor] = true;

path.push\_back(neighbor);

found = true;

break;

}

}

if (!found)

{

stack.pop();

visited[current] = false;

path.pop\_back();

}

}

return {};

}

Рисунок . Время поиска эйлерова графа от количества вершин

Рисунок . Время поиска эйлерова графа от количества ребер

Рисунок . Время поиска гамильтонова графа от количества вершин

Рисунок . Время поиска гамильтонова графа от количества ребер

**Анализ**

Для поиска эйлерова пути использован алгоритм Хиэхолзера. Его асимптотическая сложность O(p + q), где p – количество вершин, q –количество ребер.

Для поиска гамильтонова пути использован алгоритм поиска с возвратом. Его асимптотическая сложность O(p!), где p – количество вершин. Кроме того, время увеличивается по мере увеличения насыщенности графа.

**Вывод:** изучены алгоритмы поиска эйлерова и гамильтонова путей на графах, исследована асимптотическая характеристика этих алгоритмов.