Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Информатика»

Тема: "Коммивояжер"

Семестр 2

Выполнил работу

Студент группы РИС-24-3Б

Гузий С.А.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2025

# Постановка задачи

# 1. В качестве варианта для демонстрации работы программы взять свой вариант задания из лабораторной работы «ГРАФЫ» (если в варианте граф имеет менее 6 вершин,то надо изменить граф так, чтобы в нём было не менее 6 вершин, и граф должен быть двунаправленным).

# Не на каждом графе решается задача коммивояжера, поэтому, при необходимости, необходимо

# Модифицировать граф таким образом, чтобы для этого графа можно было решить задачу Коммивояжера. Можно придумать собственную альтернативную задачу, которую можно решить методом ветвей и границ.

# Это может быть игра, построенная по типу пошаговых настольных игр, к примеру.

# Разработать программу, которая будет универсальной на любом наборе исходных данных.

# 2. Проработать визуализирующую часть в программе средствами OpenGL или иных открытых кроссплатформенных графических библиотек

# в части построения графа (SFML).

# Интересные дизайнерские и конструкторские решения в интерфейсе применить: добавление новых узлов, перемещение узлов, установка связей между узлами, разрыв связей и прочие варианты демонстрации своего таланта.

# 3. Исходные данные должны приниматься с консоли, либо через графический интерфейс с помощью Qt, Windows Forms или других фреймворков и библиотек в экосистеме языка C++.

# 4. Задокументировать программу диаграммой классов UML.

# Код программы

#include <GL/glut.h>

#include <iostream>

#include <vector>

#include <cmath>

#include <climits>

#include <locale>

#include <string>

const int N = 6;

int graph[N][N];

float coords[N][2] = {

{100, 400}, {200, 500}, {300, 400},

{400, 300}, {300, 200}, {200, 300}

};

std::vector<int> bestPath;

int minCost = INT\_MAX;

void drawText(float x, float y, const std::string& text, float r = 0, float g = 0, float b = 0) {

glColor3f(r, g, b);

glRasterPos2f(x, y);

for (char c : text)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, c);

}

void tspBranchBound(std::vector<int>& path, std::vector<bool>& visited, int cost, int level) {

if (level == N) {

if (graph[path.back()][path[0]] != INT\_MAX) {

int totalCost = cost + graph[path.back()][path[0]];

if (totalCost < minCost) {

minCost = totalCost;

bestPath = path;

bestPath.push\_back(path[0]);

}

}

return;

}

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

if (!visited[i] && graph[path.back()][i] != INT\_MAX) {

visited[i] = true;

path.push\_back(i);

tspBranchBound(path, visited, cost + graph[path[path.size() - 2]][i], level + 1);

visited[i] = false;

path.pop\_back();

}

}

}

void drawCircle(float cx, float cy, float r) {

glBegin(GL\_TRIANGLE\_FAN);

for (int i = 0; i < 100; ++i) {

float theta = 2.0f \* 3.1415926f \* i / 100;

glVertex2f(cx + r \* cos(theta), cy + r \* sin(theta));

}

glEnd();

}

void drawLine(float x1, float y1, float x2, float y2, bool highlight, int weight = -1) {

if (highlight) glColor3f(1.0, 0.0, 0.0);

else glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_LINES);

glVertex2f(x1, y1);

glVertex2f(x2, y2);

glEnd();

if (weight >= 0) {

float midX = (x1 + x2) / 2;

float midY = (y1 + y2) / 2;

// Вектор нормали (перпендикуляр к линии) для смещения надписи

float dx = x2 - x1;

float dy = y2 - y1;

float length = sqrt(dx \* dx + dy \* dy);

float offsetX = -dy / length \* 10;

float offsetY = dx / length \* 10;

drawText(midX + offsetX, midY + offsetY, std::to\_string(weight));

}

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

// Рисуем рёбра

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

for (int j = i + 1; j < N; ++j)

if (graph[i][j] != INT\_MAX)

drawLine(coords[i][0], coords[i][1], coords[j][0], coords[j][1], false, graph[i][j]);

// Оптимальный путь

if (!bestPath.empty()) {

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

int from = bestPath[i];

int to = bestPath[i + 1];

drawLine(coords[from][0], coords[from][1], coords[to][0], coords[to][1], true);

}

}

// Вершины

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

glColor3f(0.2, 0.6, 0.9);

drawCircle(coords[i][0], coords[i][1], 15);

drawText(coords[i][0] - 5, coords[i][1] - 5, std::to\_string(i));

}

// Вывод стоимости пути

if (minCost != INT\_MAX) {

drawText(20, 20, "MIN: " + std::to\_string(minCost), 0.0f, 0.0f, 1.0f);

}

glutSwapBuffers();

}

void init() {

glClearColor(1, 1, 1, 1);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0, 600, 0, 600);

std::vector<int> path = { 0 };

std::vector<bool> visited(N, false);

visited[0] = true;

tspBranchBound(path, visited, 0, 1);

std::cout << "Лучший маршрут: ";

for (int v : bestPath) std::cout << v << " ";

std::cout << "\nМинимальная стоимость: " << minCost << "\n";

}

void inputGraph() {

std::cout << "Введите матрицу смежности (" << N << "x" << N << ", -1 = нет ребра):\n";

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

for (int j = 0; j < N; ++j) {

int val;

std::cin >> val;

graph[i][j] = (val == -1) ? INT\_MAX : val;

}

}

int main(int argc, char\*\* argv) {

setlocale(LC\_ALL, "Russian");

std::cout << "ЗАДАЧА КОММИВОЯЖЁРА\nМетод ветвей и границ\n";

inputGraph();

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(600, 600);

glutCreateWindow("TSP - Ветви и границы (OpenGL)");

init();

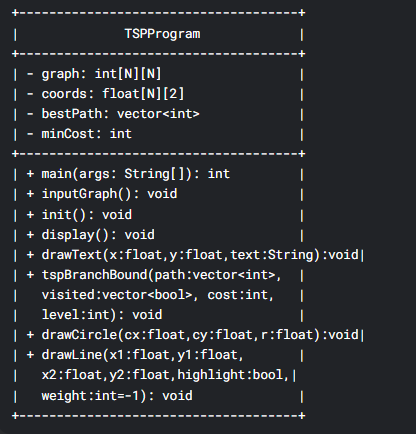
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

UML – диаграмма



# Визуализация графа

Вводим:  
 0 10 15 20 25 30

10 0 35 25 17 28

15 35 0 30 20 22

20 25 30 0 18 26

25 17 20 18 0 24

30 28 22 26 24 0

