Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе

Тема: «Задача Коммивояжера»

Выполнил

Студент группы РИС-22-2б

Юсупов Т.А.

Проверил доц. Кафедры ИТАС

Полякова О. А.

Пермь 2023

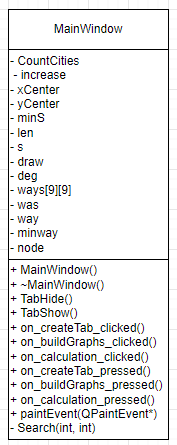
**Постановка задачи**

1) Построение и отображение диаграммы

2) Решение задачи TSP (Traveling Salesman Problem)

Программа работает по алгоритму TSP с применением метода ветвей и границ.

**Uml-Диаграмма**



**Код программы**

**point.h**

#ifndef POINT\_H

#define POINT\_H

#include <cmath>

*//* *Класс,* *представляющий* *точку* *на* *плоскости*

*class* **Point**

{

*public*:

**Point**(double x = 0, double y = 0) : x(x), y(y) {}

double **distanceTo**(*const* Point& other) *const*

{

double dx = x - other.x;

double dy = y - other.y;

*return* std::sqrt(dx \* dx + dy \* dy);

}

double x;

double y;

};

#endif *//* *POINT\_H*

**edge.h**

#ifndef EDGE\_H

#define EDGE\_H

*//* *Класс,* *представляющий* *ребро* *графа* *с* *весом*

*class* **Edge**

{

*public*:

**Edge**(int src, int dest, double weight) : src(src), dest(dest), weight(weight) {}

int src;

int dest;

double weight;

};

#endif *//* *EDGE\_H*

**graphwidget.h**

#ifndef GRAPHWIDGET\_H

#define GRAPHWIDGET\_H

#include <vector>

#include <algorithm>

#include <limits>

#include <QtWidgets/QGraphicsScene>

#include <QtWidgets/QGraphicsView>

#include <QtWidgets/QGraphicsItem>

#include <QtWidgets/QLabel>

#include <QtWidgets/QPushButton>

#include <QtCore/Qt>

#include <QtCore/QPoint>

#include <QtCore/QPointF>

#include "point.h"

#include "edge.h"

*//* *Класс* *для* *отображения* *графа* *и* *решения* *коммивояжера*

*class* **GraphWidget** : *public* QGraphicsView

{

*public*:

**GraphWidget**(std::vector<Point>& points, std::vector<Edge>& edges);

*private*:

std::vector<Point>& points;

std::vector<Edge>& edges;

QGraphicsScene\* scene;

std::vector<int> tour;

void **solveTSP**();

void **addPoint**();

void **removePoint**();

};

#endif *//* *GRAPHWIDGET\_H*

**graphwidget.cpp**

#include "graphwidget.h"

#include <iostream>

*//* *Конструктор* *класса* *GraphWidget*

GraphWidget::**GraphWidget**(std::vector<Point>& points, std::vector<Edge>& edges)

: points(points), edges(edges)

{

*//* *Создание* *сцены* *и* *установка* *фона*

scene = *new* QGraphicsScene(*this*);

scene->setBackgroundBrush(Qt::*white*);

setScene(*scene*);

*//* *Установка* *размеров* *области* *просмотра*

setRenderHint(QPainter::*Antialiasing*);

setMinimumSize(800, 600);

*//* *Создание* *точек* *на* *сцене*

*for* (int i = 0; i < points.size(); ++i)

{

QGraphicsEllipseItem\* item = scene->addEllipse(points[i].x - 5, points[i].y - 5, 10, 10, QPen(Qt::*black*), QBrush(Qt::*black*));

item->setFlag(QGraphicsItem::*ItemIsMovable*, *false*);

item->setFlag(QGraphicsItem::*ItemIsSelectable*, *false*);

QGraphicsSimpleTextItem\* textItem = scene->addSimpleText(QString::number(i));

textItem->setPos(points[i].x - 10, points[i].y - 20);

}

*//* *Создание* *ребер* *на* *сцене*

*for* (*const* Edge& edge : edges)

{

QGraphicsLineItem\* item = scene->addLine(points[edge.src].x, points[edge.src].y, points[edge.dest].x, points[edge.dest].y, QPen(Qt::*blue*));

item->setFlag(QGraphicsItem::*ItemIsMovable*, *false*);

item->setFlag(QGraphicsItem::*ItemIsSelectable*, *false*);

QGraphicsSimpleTextItem\* textItem = scene->addSimpleText(QString::number(edge.weight));

textItem->setPos((points[edge.src].x + points[edge.dest].x) / 2, (points[edge.src].y + points[edge.dest].y) / 2);

}

solveTSP();

*//* *Кнопка* *"Добавить* *пункт"*

QPushButton\* addButton = *new* QPushButton("Добавить пункт", *this*);

addButton->move(10, 10);

connect(addButton, &QPushButton::clicked, *this*, &GraphWidget::addPoint);

*//* *Кнопка* *"Удалить* *пункт"*

QPushButton\* removeButton = *new* QPushButton("Удалить пункт", *this*);

removeButton->move(10, 40);

connect(removeButton, &QPushButton::clicked, *this*, &GraphWidget::removePoint);

}

void GraphWidget::**solveTSP**()

{

double minDistance = std::numeric\_limits<double>::max(); *//* *Минимальное* *расстояние*

std::vector<int> vertexList(points.size());

*for* (int i = 0; i < points.size(); ++i)

vertexList[i] = i;

*do*

{

double distance = 0;

*for* (int i = 0; i < points.size() - 1; ++i)

distance += points[vertexList[i]].distanceTo(points[vertexList[i + 1]]);

distance += points[vertexList.back()].distanceTo(points[vertexList[0]]);

*if* (distance < minDistance)

{

minDistance = distance;

tour = vertexList;

}

} *while* (std::next\_permutation(vertexList.begin() + 1, vertexList.end()));

*//* *Отображение* *решения*

*for* (int i = 0; i < tour.size() - 1; ++i)

{

int src = tour[i];

int dest = tour[i + 1];

QGraphicsLineItem\* item = scene->addLine(points[src].x, points[src].y, points[dest].x, points[dest].y, QPen(Qt::*red*, 2));

item->setFlag(QGraphicsItem::*ItemIsMovable*, *false*);

item->setFlag(QGraphicsItem::*ItemIsSelectable*, *false*);

}

int src = tour.back();

int dest = tour[0];

QGraphicsLineItem\* item = scene->addLine(points[src].x, points[src].y, points[dest].x, points[dest].y, QPen(Qt::*red*, 2));

item->setFlag(QGraphicsItem::*ItemIsMovable*, *false*);

item->setFlag(QGraphicsItem::*ItemIsSelectable*, *false*);

std::cout << "Min Distance: " << minDistance << std::endl;

std::cout << "Tour: ";

*for* (int i : tour)

std::cout << i << " ";

std::cout << std::endl;

}

void GraphWidget::**addPoint**()

{

int newX = std::rand() % 600 + 100;

int newY = std::rand() % 400 + 100;

Point newPoint(newX, newY);

points.push\_back(newPoint);

scene->clear();

*for* (int i = 0; i < points.size(); ++i)

{

QGraphicsEllipseItem\* item = scene->addEllipse(points[i].x - 5, points[i].y - 5, 10, 10, QPen(Qt::*black*), QBrush(Qt::*black*));

item->setFlag(QGraphicsItem::*ItemIsMovable*, *false*);

item->setFlag(QGraphicsItem::*ItemIsSelectable*, *false*);

QGraphicsSimpleTextItem\* textItem = scene->addSimpleText(QString::number(i));

textItem->setPos(points[i].x - 10, points[i].y - 20);

}

*for* (*const* Edge& edge : edges)

{

QGraphicsLineItem\* item = scene->addLine(points[edge.src].x, points[edge.src].y, points[edge.dest].x, points[edge.dest].y, QPen(Qt::*blue*));

item->setFlag(QGraphicsItem::*ItemIsMovable*, *false*);

item->setFlag(QGraphicsItem::*ItemIsSelectable*, *false*);

QGraphicsSimpleTextItem\* textItem = scene->addSimpleText(QString::number(edge.weight));

textItem->setPos((points[edge.src].x + points[edge.dest].x) / 2, (points[edge.src].y + points[edge.dest].y) / 2);

}

solveTSP();

}

void GraphWidget::**removePoint**()

{

*if* (!points.empty())

points.pop\_back();

scene->clear();

*for* (int i = 0; i < points.size(); ++i)

{

QGraphicsEllipseItem\* item = scene->addEllipse(points[i].x - 5, points[i].y - 5, 10, 10, QPen(Qt::*black*), QBrush(Qt::*black*));

item->setFlag(QGraphicsItem::*ItemIsMovable*, *false*);

item->setFlag(QGraphicsItem::*ItemIsSelectable*, *false*);

QGraphicsSimpleTextItem\* textItem = scene->addSimpleText(QString::number(i));

textItem->setPos(points[i].x - 10, points[i].y - 20);

}

*for* (*const* Edge& edge : edges)

{

QGraphicsLineItem\* item = scene->addLine(points[edge.src].x, points[edge.src].y, points[edge.dest].x, points[edge.dest].y, QPen(Qt::*blue*));

item->setFlag(QGraphicsItem::*ItemIsMovable*, *false*);

item->setFlag(QGraphicsItem::*ItemIsSelectable*, *false*);

QGraphicsSimpleTextItem\* textItem = scene->addSimpleText(QString::number(edge.weight));

textItem->setPos((points[edge.src].x + points[edge.dest].x) / 2, (points[edge.src].y + points[edge.dest].y) / 2);

}

solveTSP();

}

**main.cpp**

#include <QApplication>

#include <iostream>

#include "graphwidget.h"

#include <QMainWindow>

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

{

QApplication app(*argc*, *argv*);

*//* *Задаем* *координаты* *точек*

std::vector<Point> points = {

{50, 50},

{200, 50},

{150, 200},

{100, 150},

{250, 150}

};

*//* *Задаем* *ребра* *с* *весом*

std::vector<Edge> edges = {

{0, 1, points[0].distanceTo(points[1])},

{1, 2, points[1].distanceTo(points[2])},

{2, 3, points[2].distanceTo(points[3])},

{3, 4, points[3].distanceTo(points[4])},

{4, 0, points[4].distanceTo(points[0])}

};

*//* *Создаем* *окно* *и* *отображаем* *граф*

QMainWindow window;

GraphWidget\* graphWidget = *new* GraphWidget(*points*, *edges*);

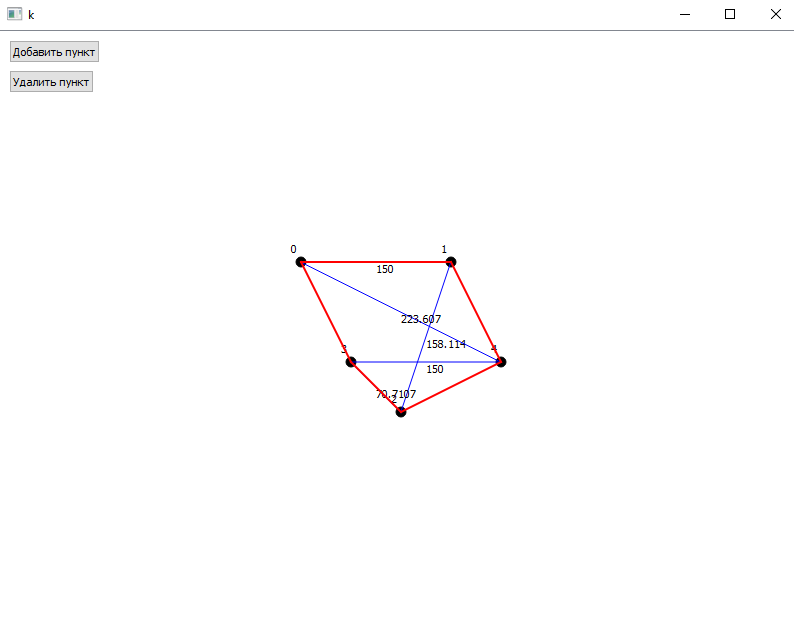
window.setCentralWidget(*graphWidget*);

window.show();

*return* app.exec();

}

**Вывод**

****