Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по творческой работе**

Дисциплина: «Информатика»

Тема Задача Коммивояжера

Выполнили:

Студенты группы РИС-23-2б

Хабарова А.А.

Крючков К.П.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь 2024**

**Цель работы**

1. Освоение алгоритмов решения задачи коммивояжера: Изучить и внедрить алгоритмы поиска оптимального маршрута, в частности метод ветвей и границ.
2. Разработка и реализация программного решения: Создать программу для нахождения оптимального маршрута с использованием метода ветвей и границ.

**Постановка задачи**

Задача коммивояжера (Travelling Salesman Problem, TSP) является классической задачей комбинаторной оптимизации, где требуется найти оптимальный маршрут, проходящий через все заданные города ровно один раз и возвращающийся в начальный город, с минимальными затратами. В рамках данной лабораторной работы необходимо:

1. Разработать программное обеспечение для решения задачи коммивояжера, используя метод ветвей и границ.
2. Реализовать пользовательский интерфейс для ввода данных и визуализации маршрута.
3. Обеспечить возможность ввода данных о расстояниях между городами в матричной форме через интерфейс программы.
4. Визуализировать процесс поиска оптимального маршрута, отображая промежуточные шаги алгоритма и конечный результат.
5. Обеспечить корректное выполнение алгоритма на различных наборах данных, в том числе на больших и сложных графах.
6. Провести тестирование программы с целью выявления и исправления ошибок, а также оптимизации производительности

**Анализ задачи**

1. Определить цель лабораторной работы и основные задачи, которые необходимо решить.
2. Ознакомиться с теоретическим материалом по задаче коммивояжера и методу ветвей и границ.
3. Проектирование пользовательского интерфейса
4. Разработать макет окна приложения, включающего элементы для ввода данных о городах и расстояниях между ними.
5. Предусмотреть кнопки для выполнения основных действий: создания матрицы расстояний, запуска алгоритма и отображения результатов.
6. Реализовать функции инициализации, которые будут считывать количество городов и генерировать необходимые элементы интерфейса для ввода расстояний.
7. Сбор данных о расстояниях
8. Обеспечить ввод данных пользователем через интерфейс, используя соответствующие текстовые поля.
9. Считать введенные значения и сохранить их в матрице расстояний.
10. Реализация метода ветвей и границ
11. Написать функции, выполняющие основные шаги алгоритма ветвей и границ: поиск минимальных значений в строках и столбцах, расчет рангов нулей и выбор оптимальных ветвей для дальнейшего разбиения.
12. Реализовать поиск оптимального маршрута с использованием полученных данных.
13. Написать код для графической визуализации городов и маршрутов на плоскости.
14. Отображать промежуточные результаты выполнения алгоритма, показывая, как изменяются маршруты и какие ветви исключаются.
15. Запустить алгоритм на различных наборах данных, проверить корректность его работы.
16. Оптимизировать код для повышения производительности и исправить выявленные ошибки.
17. Реализовать функцию отображения конечного результата в виде сообщения, показывающего оптимальный маршрут и его длину.
18. Обеспечить наглядное представление маршрута на графике.
19. Подготовка отчетной документации
20. Составить отчет по проделанной работе, включающий описание задачи, методику решения, описание алгоритма и интерфейса, примеры запуска и тестирования программы.

**Реализация задачи на языке С++**

**Заголовочный файл comi.h**

#ifndef COMI\_H

#define COMI\_H

#include <QMainWindow>

#include <QPainter>

QT\_BEGIN\_NAMESPACE

namespace Ui { class Comi; }

QT\_END\_NAMESPACE

class Comi : public QMainWindow

{

Q\_OBJECT

public:

Comi(QWidget \*parent = nullptr);

~Comi();

void Close();

void Print();

void Search\_min();

void Search\_null(int num);

private slots:

void on\_spinBox\_valueChanged(int arg1);

void on\_create\_clicked();

void on\_build\_clicked();

void on\_create\_pressed();

void on\_create\_released();

void on\_build\_pressed();

void on\_build\_released();

void on\_run\_pressed();

void on\_run\_released();

void on\_run\_clicked();

private:

Ui::Comi \*ui;

protected:

void paintEvent(QPaintEvent\*) override;

};

#endif // COMI\_H

**Файл comi.cpp**

#include "comi.h"

#include "ui\_comi.h"

#include <cmath>

#include <QMessageBox>

bool \*prev, notall=0;

int arr[9][9], coun, increm, temp=0, str\_min[9], clm\_min[9],

optimate[9][2], mid\_arg, mid\_func, way[9];

double grad=360;

class Node

{

public:

int x,y;

};

Node \*node;

Comi::Comi(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::Comi)

{

ui->setupUi(this);

ui->build->hide();

ui->run->hide();

Close();

}

Comi::~Comi()

{

delete ui;

}

void Comi::on\_create\_clicked()

{

ui->run-> hide();

ui->build->show();

Close();

Print();

node = new Node[coun];

mid\_arg=800;

mid\_func=350;

temp=1;

this->repaint();

}

void Comi::on\_build\_clicked()

{

ui->run-> hide();

QString searcher,buffer;

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

{

for (int j=0; j<coun; j++)

{

if(i != j)

{

searcher="u"+QString::number(i+1) + QString::number(j+1);

buffer=this->findChild<QLineEdit\*>(searcher)->text();

if (!buffer.length())

buffer=QString::number(INT32\_MAX);

arr[i][j]=buffer.toInt();

}

if(i == j)

arr[i][j] = -1;

}

}

ui->run->show();

temp=2;

this->repaint();

}

void Comi::on\_run\_clicked()

{

QString dialog, mistake;

QString searcher,buffer;

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

{

Search\_min();

Search\_null(i);

}

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

{

for (int j=0; j<coun; j++)

{

if(i != j)

{

searcher="u"+QString::number(i+1) + QString::number(j+1);

buffer=this->findChild<QLineEdit\*>(searcher)->text();

if (!buffer.length())

buffer=QString::number(INT32\_MAX);

arr[i][j]=buffer.toInt();

}

else

arr[i][j] = -1;

}

}

dialog += "\nОптимальный путь: " + QString::number(optimate[0][0]+1) + " -> " + QString::number(optimate[0][1]+1);

int point = optimate[0][1], summ = arr[optimate[0][0]][optimate[0][1]];

way[0] = optimate[0][0];

way[1] = optimate[0][1];

int z = 1;

while(z < coun)

{

for(int i = 1; i < coun; i++)

{

if(optimate[i][0]==point && optimate[i][0]!=optimate[i][1])

{

way[z+1] = point = optimate[i][1];

dialog += " -> " + QString::number(optimate[i][1]+1);

summ += arr[optimate[i][0]][optimate[i][1]];

optimate[i][0] = -1;

optimate[i][1] = -1;

break;

}

}

z++;

}

temp = 3;

this->repaint();

dialog += "\nПродолжительность: " +QString::number(summ);

QMessageBox::information(this,"Result",dialog);

}

void Comi::Search\_null(int num)

{

int min\_in\_str = INT32\_MAX;

int min\_in\_clm = INT32\_MAX;

int max\_rank\_null = 0;

int max\_i=100, max\_j=100;

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

{

for(int j=0;j<coun;j++)

{

if(arr[i][j] == 0 && i!=j)

{

for (int k=0;k<coun;k++)

{

if(arr[k][j]>-1 && arr[k][j] < min\_in\_clm && k != j && k!=i)

min\_in\_clm = arr[k][j];

if(arr[i][k]>-1 && arr[i][k] < min\_in\_str && i != k && k!=j)

min\_in\_str = arr[i][k];

}

if(min\_in\_str > 1000)

min\_in\_str = 0;

if(min\_in\_clm > 1000)

min\_in\_clm = 0;

if(min\_in\_str>-1 && min\_in\_clm>-1 && i!=j && max\_rank\_null <= (min\_in\_str + min\_in\_clm))

{

max\_rank\_null = min\_in\_str + min\_in\_clm;

max\_i = i;

max\_j = j;

}

}

}

}

optimate[num][0] = max\_i;

optimate[num][1] = max\_j;

for(int z=0;z<coun;z++)

{

arr[max\_i][z] = -1;

arr[z][max\_j] = -1;

}

arr[max\_j][max\_i] = -1;

};

void Comi::Search\_min()

{

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

{

str\_min[i] = INT32\_MAX;

clm\_min[i] = INT32\_MAX;

}

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

{

for (int j=0; j<coun; j++)

if(i != j && arr[i][j]>-1 && arr[i][j] < str\_min[i])

str\_min[i] = arr[i][j];

for (int j=0; j<coun; j++)

if(i != j && str\_min[i] > 0 && str\_min[i]<1000)

arr[i][j] -= str\_min[i];

}

for (int j=0; j<coun;j++)

{

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

if(i != j && arr[i][j]>-1 && arr[i][j] < clm\_min[j])

clm\_min[j] = arr[i][j];

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

if(i != j && clm\_min[j] > 0 && clm\_min[j]<1000)

arr[i][j] -= clm\_min[j];

}

}

void Comi::Close()

{

QString str;

for (int i=1;i<10;i++)

{

str="clm" + QString::number(i);

this->findChild<QWidget\*>(str)->hide();

}

for (int i=1;i<10;i++)

{

str="stg" + QString::number(i);

this->findChild<QWidget\*>(str)->hide();

str="z" + QString::number(i);

this->findChild<QWidget\*>(str)->hide();

for (int j=i+1;j<10;j++)

{

str="u" + QString::number(i) + QString::number(j);

this->findChild<QWidget\*>(str)->hide();

}

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

{

str="u" + QString::number(i) + QString::number(j);

this->findChild<QWidget\*>(str)->hide();

}

}

}

void Comi::Print()

{

QString str;

for (int i=1;i<coun+1;i++)

{

str="clm" + QString::number(i);

this->findChild<QWidget\*>(str)->show();

}

for (int i=1;i<coun+1;i++)

{

str="stg" + QString::number(i);

this->findChild<QWidget\*>(str)->show();

str="z" + QString::number(i);

this->findChild<QWidget\*>(str)->show();

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

{

str="u" + QString::number(i) + QString::number(j);

this->findChild<QWidget\*>(str)->show();

}

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

{

str="u" + QString::number(i) + QString::number(j);

this->findChild<QWidget\*>(str)->show();

}

}

}

void Comi::paintEvent(QPaintEvent \*)

{

if (temp){

int otstup;

grad /= coun;

increm=grad;

grad=0;

QPainter paint(this);

QPen pen;

QBrush brush;

QString str;

QFont font;

paint.begin(this);

paint.setPen(pen);

paint.setFont(font);

paint.setBrush(brush);

pen.setColor(Qt::gray);

pen.setWidth(7);

pen.setStyle(Qt::SolidLine);

font.setPointSize(20);

brush.setStyle(Qt::Dense3Pattern);

brush.setColor(Qt::white);

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

{

otstup=-10;

if (grad>180) otstup=130;

str=QString::number(i+1);

node[i].x=mid\_arg+(200\*cos(grad/57));

node[i].y=mid\_func+(200\*-sin(grad/57));

paint.setPen(QPen(Qt::white, 2, Qt::SolidLine, Qt::FlatCap));

paint.setBrush(QBrush(Qt::darkBlue, Qt::SolidPattern));

paint.drawEllipse(node[i].x,node[i].y,100,100);

paint.drawText(node[i].x+48,node[i].y+otstup,str);

grad+=increm;

}

if (temp>1){

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

{

for (int j=0; j<coun; j++)

{

if (arr[i][j]<1000&&arr[i][j]>0)

{

pen.setStyle(Qt::DashLine);

pen.setWidth(3);

paint.setPen(pen);

paint.drawLine(node[i].x+50,node[i].y+50,node[j].x+50,node[j].y+50);

}

}

}

}

if (temp==3)

{

pen.setStyle(Qt::SolidLine);

pen.setWidth(5);

pen.setColor(Qt::white);

paint.setPen(pen);

for (int i=1; i<coun+1; i++)

paint.drawLine(node[way[i-1]].x+50,node[way[i-1]].y+50,node[way[i]].x+50,node[way[i]].y+50);

}

paint.end();

}

}

void Comi::on\_spinBox\_valueChanged(int arg1)

{

coun = arg1;

}

void Comi::on\_create\_pressed()

{

ui->create->setStyleSheet("color: rgb(205, 205, 255);font-size: 24px;border: 2px solid rgb(205, 205, 255);");

}

void Comi::on\_create\_released()

{

ui->create->setStyleSheet("background-color: rgb(205, 205, 255);color: rgb(30, 30, 100);");

}

void Comi::on\_build\_pressed()

{

ui->build->setStyleSheet("color: rgb(205, 205, 255);font-size: 24px;border: 2px solid rgb(205, 205, 255);");

}

void Comi::on\_build\_released()

{

ui->build->setStyleSheet("background-color: rgb(205, 205, 255);color: rgb(30, 30, 100);");

}

void Comi::on\_run\_pressed()

{

ui->run->setStyleSheet("color: rgb(205, 205, 255);font-size: 24px;border: 2px solid rgb(205, 205, 255);");

}

void Comi::on\_run\_released()

{

ui->run->setStyleSheet("background-color: rgb(205, 205, 255);color: rgb(30, 30, 100);");

}

**Файл с главной программой main.cpp**

#include "comi.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

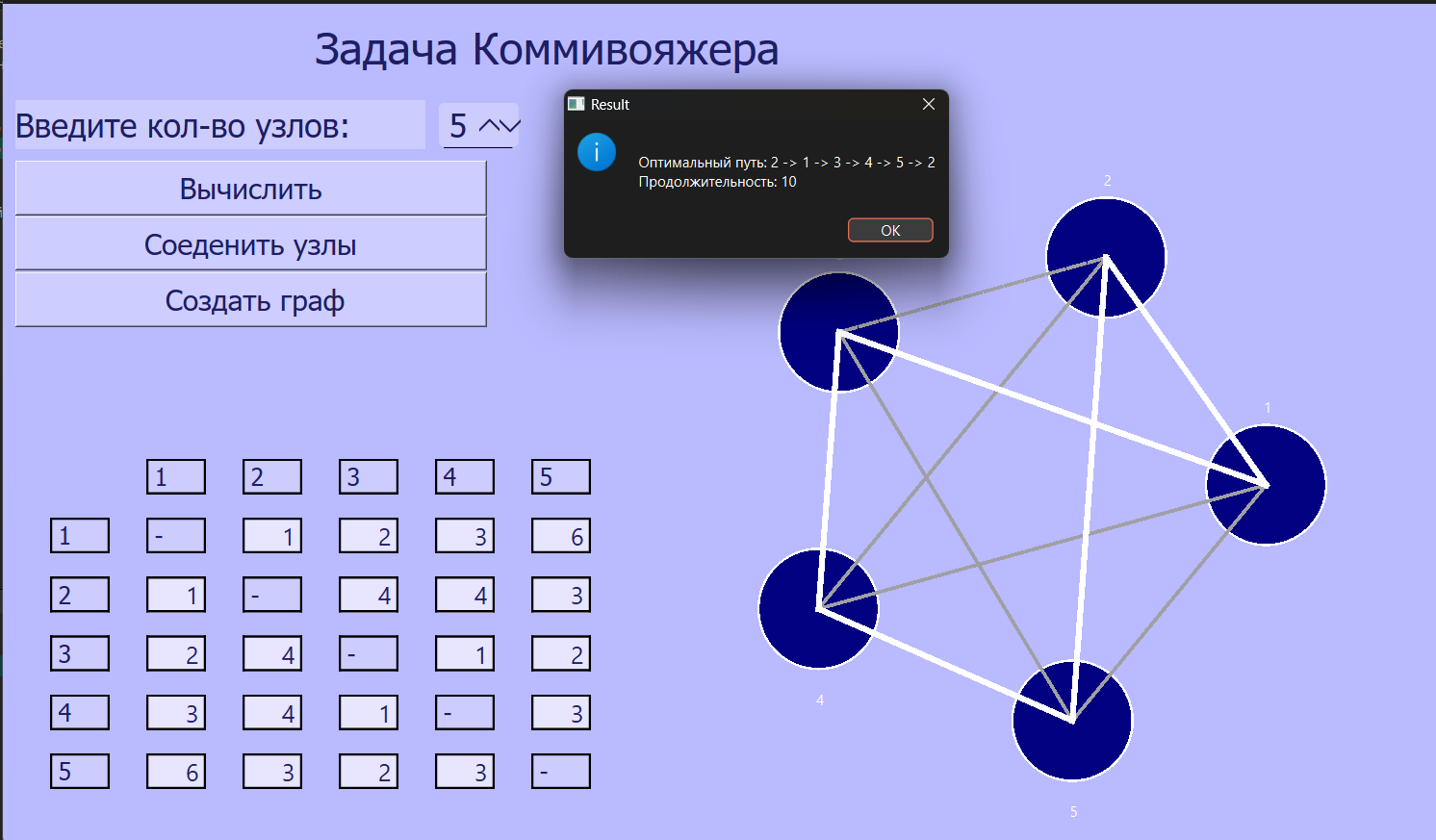
Comi w;

w.show();

return a.exec();

}

**Скриншот работы программы**

****