**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»

(НГТУ)

ДЗЕРЖИНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет\_\_\_\_\_Инженерно-технологический (ИТФ)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление подготовки \_01.03.04 «Прикладная математика\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(код и наименование)

Направленность (профиль) образовательной программы \_\_\_\_”Математические и\_\_\_\_\_\_\_

\_\_компьютерные методы для современных технологий”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование)

Кафедра \_\_\_Автоматизация, энергетика, математика и информационные технологии\_\_\_\_\_

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе по дисциплине «Алгоритмы дискретной математики»

*(вид документа – проект дипломный, курсовой, исследовательская работа или часть и т.п.)*

на тему\_\_Разработка программы находяшей абсолютный центр графа\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование темы работы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студента \_Стацюры Владислава Николаевича\_\_\_\_\_\_\_\_\_\_\_\_\_\_ группы \_19ПМ\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.)

**РУКОВОДИТЕЛЬ**

Харитонова И.Ю.

*(подпись) (фамилия, и., о.)*

*(дата)*

**СТУДЕНТ**

Стацюра В.Н.

*(подпись) (фамилия, и., о.)*

**Работа защищена** *(дата)* **Протокол №**   **С оценкой**

*(подпись) (фамилия, и.,о.)*

2021 г

**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»

(НГТУ)

ДЗЕРЖИНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра \_\_\_ Автоматизация, энергетика, математика и информационные технологии \_\_\_\_

УТВЕРЖДАЮ

Зав. кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_Л.Ю.Вадова

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

**по направлению подготовки \_\_\_\_**01.03.04 «Прикладная математика **\_\_\_\_\_\_\_\_**

(код и наименование)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

студенту\_\_\_Стацюре В.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_группы\_19ПМ\_\_\_\_\_\_\_\_

(Ф.И.О.)

1. Тема КР \_Разработка программы находящей абсолютный центр графа \_\_\_\_\_\_\_

2. Срок сдачи студентом законченной работы\_\_\_27.12.2021\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе составить оконное приложение которое находит абсолютный центр графа

4. Содержание расчетно-пояснительной записки (перечень вопросов, подлежащих разработке)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Цели и задачи работы Справка из теоритической части Алгоритм и программная реализация Некоторые результаты работы программы  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

5. Перечень графического материала (с точным указанием количества слайдов презентации)

\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6.Дата выдачи задания\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Код и содержание**  **Компетенции** | **Задание** | **Проектируемый результат** | **Отметка о выполнении** |
| **ОПК-1** Способность применять знание фундаментальной матема-тики и естественно-научных дисциплин при решении задач в области естественных наук и инженерной практике | Создание программы на языке C++ предназначенной для нахождения абсолютного центра графа. | Выполнена программная реализация, получены правильные результаты работы на всех наборах тестовых данных | Поставленная задача выполнена полностью |

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.Ю.Харитонова

(подпись)

Задание принял к исполнению\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Н.Стацюра

(подпись)

Аннотация

Курсовая работа посвящена разработке программы, которая находит абсолютный центр графа.

Программа составлена в виде приложения QT, при использовании интегрированной среды разработки Microsoft Visual Studio Code.

Программа позволяет пользователю получить полную информацию о нахождении абсолютного центра графа на графе.

Приводятся описание алгоритма, программной реализации, некоторые

результаты работы программы и программный код.

Работа составляет 23 страницы, содержит 10 рисунков, список литературы состоит из 4 наименований, имеется 1 приложение.

Содержание

[Введение 5](#_Toc91380089)

[1 Цели и задачи работы 7](#_Toc91380090)

[2 Справка из теоритической части 8](#_Toc91380091)

[3 Алгоритм и программная реализация 10](#_Toc91380092)

[4 Некоторые результаты работы программы 15](#_Toc91380093)

[Заключение 16](#_Toc91380094)

[Список литературы 17](#_Toc91380095)

[ПРИЛОЖЕНИЕ А 18](#_Toc91380096)

# Введение

Изм.

Лист

№ докум.

Подпись

Дата

Лист

5

*КР-АДМ-НГТУ-019ПМ-017-21*

Разраб.

СтацюраВ.Н.

Пров.

Харитонова И.Ю.

Н. контр.

Утв.

Вадова Л.Ю.

Лит.

Листов

23

*Кафедра АЭМИС*

Изм.

Лист

№ докум.

Подпись

Дата

Лист

6

КР-АДМ-НГТУ-019ПМ-017-21

Данная работа представляет алгоритм нахождения абсолютного центра графа. Нахождение центра графа и абсолютного центра графа, очень частая задача при планировке местности или выбора места установки объекта.

Так как найти центр графа на бумаге очень длительный и кропотливый процесс, я решил его упростить, создав оконное приложение в котором пользователю нужно будет только построить граф и нажать на кнопку, а приложение найдет центр графа за него.

# 1 Цели и задачи работы

Изм.

Лист

№ докум.

Подпись

Дата

Лист

7

КР-АДМ-НГТУ-019ПМ-017-21

Целью курсовой работы является создание программы на языке C++ предназначенной для нахождения абсолютного центра графа.

Программа должна удовлетворять следующим требованиям:

* Удобный и понятный интерфейс
* Отсутствие ошибок в вычислительной части
* Как можно меньшее время работы программы

Основная задача работы заключается в более глубоком изучении языка программирования C++, а также фреймворка QT.

# 2 Справка из теоритической части

Изм.

Лист

№ докум.

Подпись

Дата

Лист

8

КР-АДМ-НГТУ-019ПМ-017-21

Задачи размещения связаны с решением проблем наилучшего расположения в определенных регионах таких систем обслуживания, как торговые центры, посты пожарной охраны, фабрики, аэропорты, склады и т.д. Математическая структура задачи размещения определяется конфигурацией области допустимых точек и способом оценки качества размещения. Вследствие этого существует много разнообразных задач размещения, и техническая литература изобилует методами их решения. Мы ограничимся рассмотрением только таких задач размещения, для которых областью допустимых точек размещения центров обслуживания является некоторый граф, т. е. эти центры могут располагаться в какой-либо вершине или на какой-либо дуге графа.

*Абсолютный центр* — это любая точка на дуге, расстояние от которой до наиболее отдаленной вершины графа минимально. Для поиска абсолютного центра мы должны найти такую точку f — (r, s), что максимальное расстояние точка-вершина для нее должно быть минимальным.

**МТВ(f — (r, s)) = min{МТВ(f — (t, u))}.**

Задача поиска абсолютного центра является более трудной, чем задачи поиска центра и главного центра графа, так как в процессе ее решения необходимо просматривать не только вершины, но и все точки дуг графа. Рассматривая свойства решения задачи, отметим, что ни одна из внутренних точек ориентированной дуги не может быть абсолютным центром. Действительно, из того, что обход ориентированной дуги может производиться только в одном направлении, следует, что конечная вершина, в которую входит эта дуга, ближе к каждой вершине графа, чем любая внутренняя точка этой дуги. Следовательно, при поиске абсолютного центра мы должны рассматривать только вершины графа и внутренние точки его неориентированных дуг. Рассмотрим произвольную неориентированную дугу (r, s). Расстояние d(f — (r, s), j) от точки f на дуге (r, s) до вершины j определяется из уравнения:

**d(f—(r,s ), j)=min{f×аr,s +dr,j, (1—f)×аr, s+ds,j}**,

где аr,s– длина дуги (r, s); dr,j– длина минимального пути между вершинами r, и j. Это расстояние легко представить как функцию от f. Ее график представляет собой кусочно-линейную кривую с не более чем двумя линейными отрезками. Построим графики d(f — (r, s), j) для 0≤f≤1 и для всех j. Участки кривых каждого из графиков, которые лежат выше соответствующих участков кривых всех других графиков, представляют собой значения max d(f — (r, s), j) для всех f. Точка f\* дуги (r, s), в которой достигается минимальное по f значение max d(f — (r, s), j), является наилучшим претендентом на размещение в ней абсолютного центра на дуге (r, s). С помощью этой процедуры может быть найдена точка размещения наилучшего претендента на каждой неориентированной дуге графа.  
Абсолютным центром является любая точка f\* — (r, s)\*, для которой расстояние до максимально удаленной вершины графа минимально, т. е.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

9

КР-АДМ-НГТУ-019ПМ-017-21

**max{d(f\*— (r, s)\*, j) = min{max{d(f\* — (t, u), j)}}**

Таким образом, абсолютный центр находится в два этапа. Сначала находятся на каждой неориентированной дуге точки — претенденты на размещение в них абсолютного центра. Точками-предендентами являются точки на дуге, расстояние от которых до наиболее удаленных вершин минимально. Затем среди этих претендующих точек и всех вершин графа в качестве абсолютного центра выбирается такая точка или вершина, расстояние от которой до максимально удаленной вершины графа минимально, для выбора наилучшей претендующей точки на каждом ребре требуется построение функций, характеризующих расстояния точка — вершина для всех f-точек. Это относительно не сложная процедура, так как функции являются кусочно-линейными кривыми, содержащими не более двух отрезков. К сожалению, не существует неграфических методов определения наилучших точек-претендентов, где можно было бы избежать сравнения на графиках расстояний точка — вершина для различных вершин графа. Рассмотренный метод отыскания абсолютного центра был предложен Хакими в 1964 г.

# 3 Алгоритм и программная реализация

Изм.

Лист

№ докум.

Подпись

Дата

Лист

10

КР-АДМ-НГТУ-019ПМ-017-21

Алгоритм Хакими:

1. На вход получаем информацию о кол-ве вершин, наличие дуг между вершинами и вес этих вершин. Формируем полуенную информацию в матрицу.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 Получение входных данных

1. Формируем матрицу кратчайших расстояний из каждой вершины во все остальные при помощи алгоритма Дейкстры

Изм.

Лист

№ докум.

Подпись

Дата

Лист

11

КР-АДМ-НГТУ-019ПМ-017-21

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 Составление матрицы расстояния алгоритмом Дейкстры

1. Находим верхнюю границу графа

Изм.

Лист

№ докум.

Подпись

Дата

Лист

12

КР-АДМ-НГТУ-019ПМ-017-21

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 Нахождение верхней границы

1. Находим все существующие ребра графа и их вес.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 Поиск существующих ребер графа

1. Выбираем подходящие ребра (вес которых меньше величины верхней границы), и удаляем все не нужные.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

13

КР-АДМ-НГТУ-019ПМ-017-21

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 Выбор подходящих ребер

1. Проверяем, если подходящих ребер нет, то указываем то абсолютный центр графа лежит в верхней границу графа, если подходящие ребра есть, идем дальше.

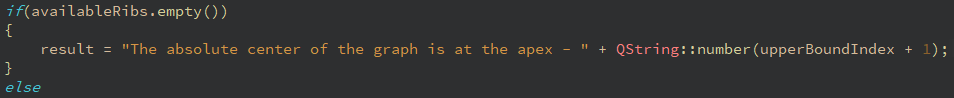


Рисунок 6 Проверка на наличее подходящих ребер

1. В цикле создаем новую матрицу по размеру равную начальной матрице плюс величина активного ребра -1 (таким образом мы распределяем на активном ребре вершины соединенные ребрами с весом 1), затем мы заполняем полученную матрицу, и в итоге при помощи алгоритмов Дейкстры и МиниМакса запоминаем активное ребро и точку на этом ребре с расстоянием до самой дальней вершины

Изображение выглядит как текст, внутренний, снимок экрана

Автоматически созданное описание

Рисунок 7 Нахождение точек на ребре

1. Из полученных данных находим вершину на ребре с наименьшим расстоянием до самой дальней вершины и выводим результат.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

14

КР-АДМ-НГТУ-019ПМ-017-21

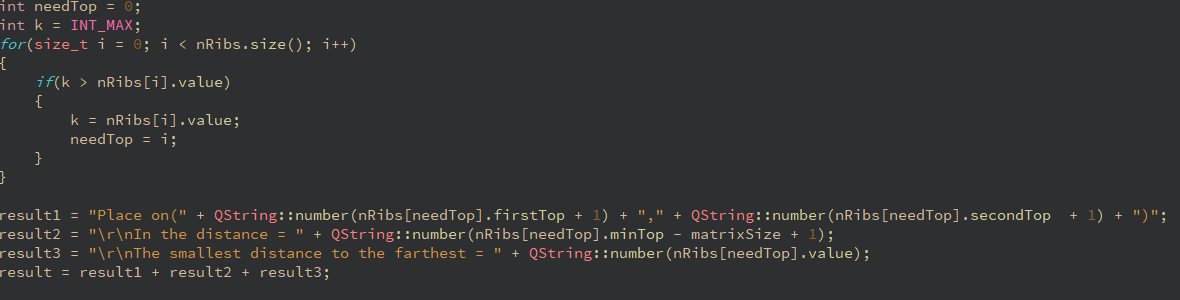


Рисунок 8 Получение результата

# 4 Некоторые результаты работы программы

Изм.

Лист

№ докум.

Подпись

Дата

Лист

15

КР-АДМ-НГТУ-019ПМ-017-21

Входные данные программы в виде треугольной матрицы:

8

41 0 0 37 63 0 0

29 50 55 32 0 0

0 0 64 46 62

0 0 0 25

0 39 0

0 0

106

Входные данные программы в виде графа в приложении:

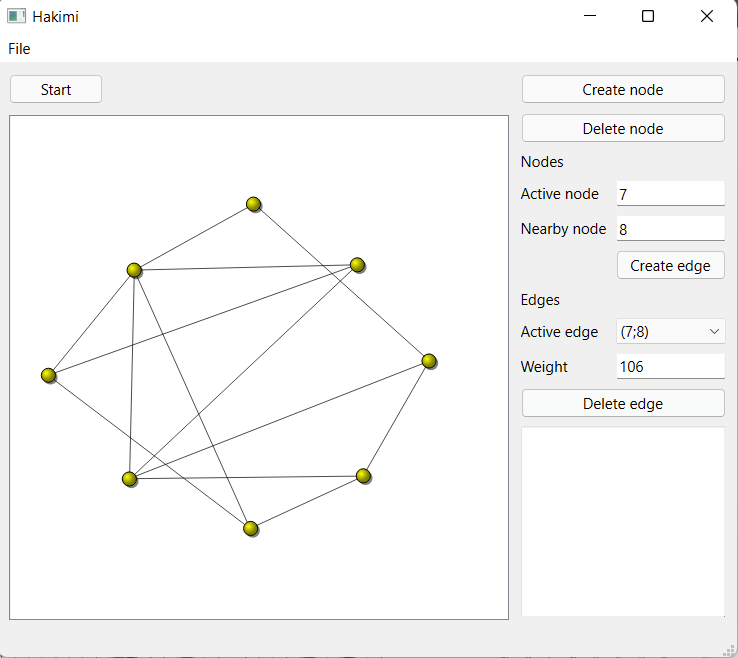


Рисунок 9 Окно приложения с входными данными

Результаты работы программы:

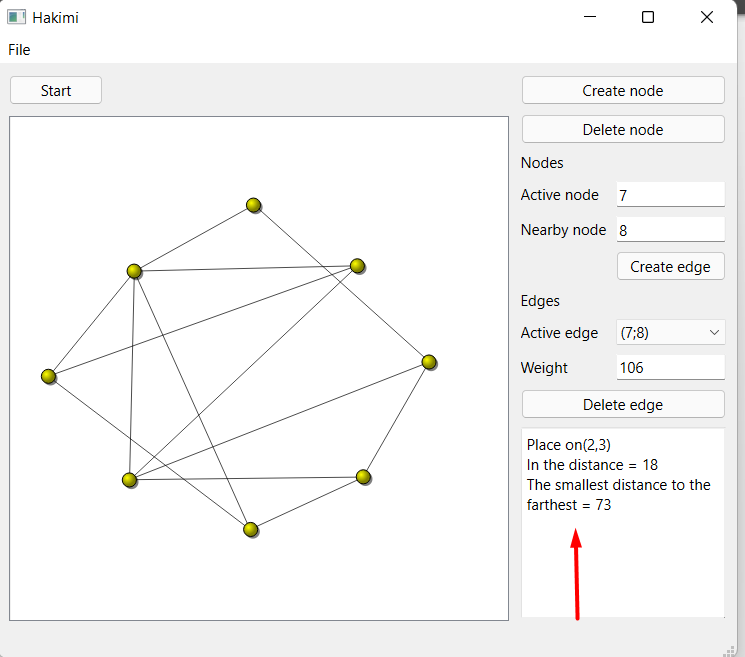


Рисунок 10 Результаты работы программы

# Заключение

Подведем итог нашей работы.

Создана программа которая находит абсолютный центр графа с помощью алгоритма Хакими. Программа реализована в виде приложения на языке C++ при использовании среды разработки Microsoft Visual Studio Code и QT Creator

Программа позволяет:

* Оптимизировать процесс решения задач путем мгновенного получения результатов.
* Получить ответ для любой задачи данного типа.

Выполненные тестовые примеры демонстрируют работоспособность созданной программы.

Распечатка программного кода приводится в приложении пояснительной записки.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

16

КР-АДМ-НГТУ-019ПМ-017-21

# Список литературы

Изм.

Лист

№ докум.

Подпись

Дата

Лист

17

КР-АДМ-НГТУ-019ПМ-017-21

1. Павловская, Т.А. C#. Программирование на языке высокого уровня. Учебник для вузов / Т.А. Павловская. – СПб.: Питер, 2014. – 432 с.
2. Т. Пратт, М. Зелковиц. Языки программирования: разработка и реализация = Terrence W. Pratt, Marvin V. Zelkowitz. Programming Languages: Design and Implementation. — 4-е издание. — Питер, 2002. — 688 с.
3. А. В. Ахо, Р. Сети, Д. Д. Ульман. Компиляторы: принципы, технологии и инструменты. М.: «Вильямс», 2003. С. 51.
4. Шилдт, Г. C# 4.0 Полное руководство / Г. Шилдт. – Спб.: Вильямс, 2011 – 1056 с.

# ПРИЛОЖЕНИЕ А

Изм.

Лист

№ докум.

Подпись

Дата

Лист

18

КР-АДМ-НГТУ-019ПМ-017-21

Код программы описанной в работе.

#include "alghoritmhakimi.h"

#include "graphwidget.h"

#include "edge.h"

#include "node.h"

void **deikstra**(std::vector<std::vector<int>> matrix, std::vector<std::vector<int>>& matrixD)

{

*for*(size\_t startTop = 0; startTop < matrix.size(); startTop++)

{

std::vector<bool> used;

used.assign(matrix.size(), *false*);

std::vector<int> tops;

tops.assign(matrix.size(), INT\_MAX);

tops[startTop] = 0;

int z = 0;

int s = startTop;

*while*(z != used.size())

{

size\_t v = 0;

*for*(size\_t j = 0; j < matrix.size(); j++)

{

*if*(matrix[s][j] == 0)

{

*continue*;

}

int summ = tops[s] + matrix[s][j];

*if*(summ < tops[j])

{

tops[j] = summ;

}

int k = INT\_MAX;

*for* (size\_t i = 0; i < tops.size(); i++)

{

*if* (k > tops[i] && !used[i] && i != s)

{

v = i;

k = tops[i];

}

}

}

used[s] = *true*;

s = v;

z = 0;

*for* (size\_t i = 0; i < used.size(); i++)

{

*if* (used[i] == *true*)

{

z++;

}

}

}

Изм.

Лист

№ докум.

Подпись

Дата

Лист

19

КР-АДМ-НГТУ-019ПМ-017-21

*for*(size\_t j = 0; j < matrixD.size(); j++)

{

matrixD[startTop][j] = tops[j];

}

}

}

void **deikstra**(std::vector<std::vector<int>> matrix, int matrixSize, std::pair<int,int>& topWithMax)

{

std::vector<std::pair<int, int>> maxCounts;

*for*(size\_t startTop = matrixSize; startTop < matrix.size(); startTop++)

{

std::vector<bool> used;

used.assign(matrix.size(), *false*);

std::vector<int> tops;

tops.assign(matrix.size(), INT\_MAX);

tops[startTop] = 0;

int z = 0;

int s = startTop;

*while*(z != used.size())

{

size\_t v = 0;

*for*(size\_t j = 0; j < matrix.size(); j++)

{

*if*(matrix[s][j] == 0)

{

*continue*;

}

int summ = tops[s] + matrix[s][j];

*if*(summ < tops[j])

{

tops[j] = summ;

}

int k = INT\_MAX;

*for* (size\_t i = 0; i < tops.size(); i++)

{

*if* (k > tops[i] && !used[i] && i != s)

{

v = i;

k = tops[i];

}

}

}

used[s] = *true*;

s = v;

z = 0;

*for* (size\_t i = 0; i < used.size(); i++)

{

*if* (used[i] == *true*)

{

z++;

}

Изм.

Лист

№ докум.

Подпись

Дата

Лист

20

КР-АДМ-НГТУ-019ПМ-017-21

}

}

int k = 0;

*for*(size\_t j = 0; j < tops.size(); j++)

{

*if*(k < tops[j])

{

k = tops[j];

}

}

maxCounts.push\_back({*startTop*, *k*});

}

int k = INT\_MAX;

*for*(size\_t i = 0; i < maxCounts.size(); i++)

{

*if*(k > maxCounts[i].second)

{

k = maxCounts[i].second;

topWithMax.first = maxCounts[i].first;

topWithMax.second = maxCounts[i].second;

}

}

}

int **findMaxMin**(std::vector<std::vector<int>> matrixD, int& index)

{

std::vector<int> maxValues;

int maxValue = 0;

*for*(size\_t i = 0; i < matrixD.size(); i++)

{

*for*(size\_t j = 0; j < matrixD.size(); j++)

{

*if*(maxValue < matrixD[i][j])

{

maxValue = matrixD[i][j];

}

}

maxValues.push\_back(maxValue);

maxValue = 0;

}

int minValue = INT\_MAX;

*for*(size\_t i = 0; i < maxValues.size(); i++)

{

*if*(minValue > maxValues[i])

{

minValue = maxValues[i];

index = i;

}

}

*return* minValue;

}

void **findExistingRibs**(std::vector<std::vector<int>> matrix, std::vector<ribs>& availableRibs, std::vector<std::vector<int>> matrixD)

{

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

Изм.

Лист

№ докум.

Подпись

Дата

Лист

21

КР-АДМ-НГТУ-019ПМ-017-21

{

*for*(int j = i + 1; j < matrix.size(); j++)

{

*if*(matrix[i][j] != 0)

{

availableRibs.push\_back({i, j, 0});

}

}

}

int z = 0;

*for*(size\_t i = 0; i < availableRibs.size(); i++)

{

*for*(size\_t j = 0; j < matrixD.size(); j++)

{

*if*(matrixD[availableRibs[i].firstTop][j] > matrixD[availableRibs[i].secondTop][j])

{

z = matrixD[availableRibs[i].secondTop][j];

}

*else*

{

z = matrixD[availableRibs[i].firstTop][j];

}

*if*(z > availableRibs[i].edgeEvaluation)

{

availableRibs[i].edgeEvaluation = z;

}

}

}

}

void **findAvailableRibs**(std::vector<ribs>& ribs, int upperBound)

{

*for*(size\_t i = 0; i < ribs.size(); )

{

*if*(ribs[i].edgeEvaluation >= upperBound)

{

ribs.erase(ribs.begin() + i);

i = 0;

}

*else*

{

i++;

}

}

}

void **fillNewMatrix**(std::vector<std::vector<int>> matrix, std::vector<std::vector<int>>& newMatrix, ribs activeRib)

{

*for*(size\_t i = 0; i < matrix.size(); i++)

{

*for*(size\_t j = 0; j < matrix.size(); j++)

{

newMatrix[i][j] = matrix[i][j];

}

}

newMatrix[matrix.size()][activeRib.firstTop] = 1;

Изм.

Лист

№ докум.

Подпись

Дата

Лист

22

КР-АДМ-НГТУ-019ПМ-017-21

newMatrix[activeRib.firstTop][matrix.size()] = 1;

newMatrix[matrix.size()][matrix.size() + 1] = 1;

newMatrix[matrix.size() + 1][matrix.size()] = 1;

*for*(size\_t i = matrix.size() + 1; i < newMatrix.size() - 1; i++)

{

newMatrix[i][i+1] = 1;

newMatrix[i+1][i] = 1;

}

newMatrix[newMatrix.size() - 1][activeRib.secondTop] = 1;

newMatrix[activeRib.secondTop][newMatrix.size() - 1] = 1;

}

QString **mainAlghoritm**(GraphWidget\* graph)

{

std::vector<std::vector<int>> matrix;

std::vector<std::vector<int>> matrixD;

int matrixSize = graph->GetNodes().size();

QString result;

QString result1;

QString result2;

QString result3;

matrix.assign(matrixSize, std::vector<int>(matrixSize));

matrixD.assign(matrixSize, std::vector<int>(matrixSize));

*for*(size\_t i = 0; i < matrix.size(); i++)

{

*for*(size\_t j = i + 1; j < matrix.size(); j++)

{

bool make = *false*;

int p = 0;

*for*(; p < graph->GetEdges().size(); p++)

{

int node1 = graph->GetNeedEdge(p)->edgeNode1();

int node2 = graph->GetNeedEdge(p)->edgeNode2();

*if*(node1 == i && node2 == j)

{

make = *true*;

*break*;

}

}

*if*(make)

{

matrix[i][j] = graph->GetNeedEdge(p)->GetWeight();

}

*else*

{

matrix[i][j] = 0;

}

matrix[j][i] = matrix[i][j];

}

}

deikstra(matrix, *matrixD*);

Изм.

Лист

№ докум.

Подпись

Дата

Лист

23

КР-АДМ-НГТУ-019ПМ-017-21

int upperBoundIndex;

int upperBound = findMaxMin(matrixD, *upperBoundIndex*);

std::vector<ribs> availableRibs;

findExistingRibs(matrix, *availableRibs*, matrixD);

findAvailableRibs(*availableRibs*, upperBound);

std::vector<neededRibs> nRibs;

*if*(availableRibs.empty())

{

result = "The absolute center of the graph is at the apex - " + QString::number(upperBoundIndex + 1);

}

*else*

{

*for*(size\_t i = 0; i < availableRibs.size(); i++)

{

std::vector<std::vector<int>> newMatrix;

newMatrix.assign(matrixSize + matrix[availableRibs[i].firstTop][availableRibs[i].secondTop] - 1, std::vector<int>(matrixSize + matrix[availableRibs[i].firstTop][availableRibs[i].secondTop] - 1));

fillNewMatrix(matrix, *newMatrix*, availableRibs[i]);

std::pair<int,int> topWithMax;

deikstra(newMatrix, matrixSize, *topWithMax*);

nRibs.push\_back({availableRibs[i].firstTop, availableRibs[i].secondTop, topWithMax.first, topWithMax.second});

}

int needTop = 0;

int k = INT\_MAX;

*for*(size\_t i = 0; i < nRibs.size(); i++)

{

*if*(k > nRibs[i].value)

{

k = nRibs[i].value;

needTop = i;

}

}

result1 = "Place on(" + QString::number(nRibs[needTop].firstTop + 1) + "," + QString::number(nRibs[needTop].secondTop + 1) + ")";

result2 = "\r\nIn the distance = " + QString::number(nRibs[needTop].minTop - matrixSize + 1);

result3 = "\r\nThe smallest distance to the farthest = " + QString::number(nRibs[needTop].value);

result = result1 + result2 + result3;

}

*return* result;

}