Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

Исследование алгоритмов поиска на графах

БГУИР КП I–40 01 01 323 ПЗ

Выполнил

студент гр. 051003 Павловец С.В.

Проверил: Фадеева Е.П.

Минск 2021

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

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

(подпись)

Лапицкая Н.В. 2021г.

ЗАДАНИЕ

по курсовому проектированию

Студенту *Павловцу Сергею Валерьевичу*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы *Исследование алгоритмов поиска на графах\_\_\_\_\_\_\_\_\_\_\_\_\_*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

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

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

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

2. Срок сдачи законченной работы *11.06.2021г.*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе *Среда программирования Delphi. Реализация функций создания и редактирования графов, сохранения созданных графов в удобном формате, открытия файлов с графами, сохраненными в формате программы. Возможность отображения графа на экране, импорта графа из матрицы расстояний в таблице Excel. Реализация функций демонстрации работы алгоритмов поиска на созданных графах. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

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

*Введение*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*1 Анализ литературных источников\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*3 Разработка программного средства\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*4 Тестирование и проверка работоспособности программного средства\_\_\_\_*

*5 Руководство по установке и использованию программного средства\_\_\_\_\_*

*Заключение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Список использованных источников\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

\_*Приложения* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема алгоритма в формате А1*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе *Фадеева Е.П.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

7.Дата выдачи задания *16.02.2021г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего обьема работы):

*Раздел 1. Введение к 28.02.2021г. – 10 % готовности работы;\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 2 к 15.03.2021г. – 30% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 3 к 15.04.2021г. – 60% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 4 к 10.05.2021г. – 80% готовности работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Раздел 5.Заключение. Приложения к 20.05.2021г. – 90% готовности работы;*

*оформление пояснительной записки и графического материала к 01.06.2021г. – 100% готовности работы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защита курсового проекта с 02.06.2021г. по 11.06.2021г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

РУКОВОДИТЕЛЬ *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Фадеева Е.П.*

*(подпись)*

Задание принял к исполнению *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Павловец С.В. 16.02.2021г.*

*(дата и подпись студента)*

Содержание

[Введение 5](#_Toc73145703)

[1 Анализ литературных источников 6](#_Toc73145704)

[1.1 Понятие графа и его представление в памяти компьютера 6](#_Toc73145705)

[1.2 Структура данных «однонаправленный список» 7](#_Toc73145706)

[1.3 Обходы графов в глубину и ширину 8](#_Toc73145707)

[1.4 Алгоритм Дейкстры для взвешенных графов 10](#_Toc73145708)

[1.5 Работа с текстовыми файлами 12](#_Toc73145709)

[2 Постановка задачи 14](#_Toc73145710)

[3 Описание алгоритмов решения задачи 15](#_Toc73145711)

[4 Структура данных 22](#_Toc73145712)

[4.1 Структура типов программы 22](#_Toc73145713)

[4.2 Структура данных алгоритма InitializeGraph 24](#_Toc73145714)

[4.3 Структура данных алгоритма DestroyGraph 24](#_Toc73145715)

[4.4 Структура данных алгоритма DestroyAdjList 24](#_Toc73145716)

[4.5 Структура данных алгоритма AddVertice 25](#_Toc73145717)

[4.6 Структура данных алгоритма AddArc 25](#_Toc73145718)

[4.7 Структура данных алгоритма DeleteVertice 26](#_Toc73145719)

[4.8 Структура данных алгоритма DeleteArc 26](#_Toc73145720)

[4.9 Структура данных алгоритма IsNeighbour 27](#_Toc73145721)

[4.10 Структура данных алгоритма GetByNumber 27](#_Toc73145722)

[4.11 Структура данных алгоритма GetByPoint 27](#_Toc73145723)

[4.12 Структура данных алгоритма OpenGraph 28](#_Toc73145724)

[4.13 Структура данных алгоритма SaveGraph 28](#_Toc73145725)

[4.14 Структура данных алгоритма ImportGraph 29](#_Toc73145726)

[4.15 Структура данных алгоритма ToWeightMatrix 30](#_Toc73145727)

[4.16 Структура данных алгоритма DFS 30](#_Toc73145728)

[4.17 Структура данных алгоритма BFS 31](#_Toc73145729)

[4.18 Структура данных алгоритма Dijkstra 32](#_Toc73145730)

[4.19 Структура данных алгоритма RestorePath 33](#_Toc73145731)

[4.20 Структура данных алгоритма InitializeStack 33](#_Toc73145732)

[4.21 Структура данных алгоритма InitializeQueue 33](#_Toc73145733)

[4.22 Структура данных алгоритма DestroyList 34](#_Toc73145734)

[4.23 Структура данных алгоритма Push 34](#_Toc73145735)

[4.24 Структура данных алгоритма Enqueue 34](#_Toc73145736)

[4.25 Структура данных алгоритма Pop 35](#_Toc73145737)

[4.26 Структура данных алгоритма Dequeue 35](#_Toc73145738)

[4.27 Структура данных алгоритма MakeVisited 35](#_Toc73145739)

[4.28 Структура данных алгоритма MakePassive 36](#_Toc73145740)

[4.29 Структура данных алгоритма MakeRegPolygon 36](#_Toc73145741)

# Введение

Данный курсовой проект посвящен исследованию алгоритмов поиска на графах.

Теория графов – раздел дискретной математики, изучающий такие математические объекты, как графы. Её родоначальником считается Леонард Эйлер, который в 1736 году формулирует и предлагает решение задачи о семи кёнигсбергских мостах. Сам термин «граф» был предложен в 1878 году Сильвестром Джеймсом Джозефом. Теория графов развивалась в процессе решения загадок и головоломок, так как графы наглядно представляются графически с помощью фигур и соединяющих их линий. И хотя долгое время она не выделялась как отдельная математическая дисциплина, но в 20-21-ые столетия графы начались использоваться во многих сферах будь то гуманитарные науки или технические. Графы также активно применяются в программировании для описания различных алгоритмов, структур данных, синтаксиса языков и т.д.

Важной частью теории графов являются задачи поиска. К примеру, согласно формулировке задачи, может быть необходимо найти некую вершину, начиная просмотр из заданной, или же некий путь, удовлетворяющий определённым условиям, или какой-либо подграф заданного графа, обладающий указанными свойствами.

Программное средство предназначено для создания и редактирования графов, а также для демонстрации некоторых алгоритмов поиска.

Целью работы является создание проекта, сопровождающегося документацией в виде пояснительной записки.

В реализации проекта попутно решаются следующие задачи:

* создание динамических структур;
* работа с файлами (текстовыми, типизированными);
* чтение/запись данных из файла;
* разработка пользовательского интерфейса для реализации перечисленных функций;
* представление содержимого файла для работы в виде динамической структуры: однонаправленного списка.

# Анализ литературных источников

## Понятие графа и его представление в памяти компьютера

Простым (n, m)-графом называется пара конечных множеств вершин и рёбер таких, что множество вершин состоит из n>0 элементов, а множество рёбер - из m элементов, каждый из которых является неупорядоченной парой из вершин. Если же помимо перечисленных условий каждому ребру ставится в соответствие некое число (вес), то такие множества называют взвешенным графом. Существует несколько способов задания графов в памяти компьютера.

Первый из них – матрица смежности. У неё номерам столбцов и строк соответствуют номера вершин графа. В каждой ячейке записывается число, определяющее смежность вершин: если вершины смежны, то элемент равен 1, иначе – 0. Для взвешенных графов вместо матрицы смежности используется матрица весов, где вместо нулей и единиц записывают вес ребра, если же ребра не существует, то в элемент записывают 0 или ∞ в зависимости от задачи. Подобный способ представления занимает O(n2) места в памяти.

Следующий способ – списками смежности. В этом случае граф представляет собой множество окрестностей всех вершин. Окрестность вершины, в свою очередь, представляется множеством смежных с ней вершин. Для взвешенного графа вместе с номером смежной вершины хранится вес соответствующего ребра. Таким образом, подобная структура занимает места в памяти.

Видно, что для графов с относительно небольшим количеством рёбер (их называют разреженными) меньше места в памяти занимает второй способ представления. Если же рёбер много (такие графы называют плотными), в частности для полного графа, у которого , списки смежности могут занимать больше памяти, чем матрица смежности. Стоит также учесть, что временная сложность проверки смежности вершин в матрице смежности составляет O(1), а в списках смежности – O(n).

Существуют и такие способы задания графов, как матрица инцидентности, строки которой соответствуют вершинам, а столбцы – рёбрам, и список рёбер, который является перечислением каждого ребра графа. Однако, они реже используются для решения задач на графах: матрица инцидентности имеет размер O(n\*m), а временная сложность проверки смежности вершин в списке рёбер – O(m), что хуже, чем у списка смежности для плотных графов. Более того, список рёбер не полностью описывает графы с изолированными вершинами, поэтому чтобы необходимо хранить дополнительный список вершин.

В следующих таблицах приведены примеры различного представления графа, изображённого на рисунке 1.

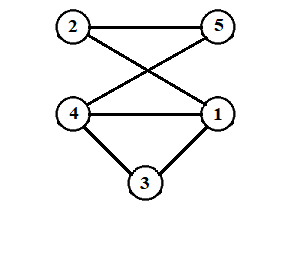


Рисунок 1.1 - Пример графа

Таблица 1 - Матрица смежности

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** |
| **1** | 0 | 1 | 1 | 1 | 0 |
| **2** | 1 | 0 | 0 | 0 | 1 |
| **3** | 1 | 0 | 0 | 1 | 0 |
| **4** | 1 | 0 | 1 | 0 | 1 |
| **5** | 0 | 1 | 0 | 1 | 0 |

Таблица 2 - Списки смежности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **→** | 2 | 3 | 4 |
| **2** | **→** | 1 | 5 |  |
| **3** | **→** | 1 | 4 |  |
| **4** | **→** | 1 | 3 | 5 |
| **5** | **→** | 2 | 4 |  |

Таблица 3 - Матрица инцидентности

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **{1, 2}** | **{1, 3}** | **{1, 3}** | **{2, 5}** | **{3, 4}** | **{4, 5}** |
| **1** | 1 | 1 | 0 | 0 | 0 | 0 |
| **2** | 1 | 0 | 1 | 1 | 0 | 0 |
| **3** | 0 | 1 | 1 | 0 | 1 | 0 |
| **4** | 0 | 0 | 0 | 0 | 1 | 1 |
| **5** | 0 | 0 | 0 | 1 | 0 | 1 |

Таблица 4 - Список рёбер

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **{1, 2}** | **{1, 3}** | **{1, 3}** | **{2, 5}** | **{3, 4}** | **{4, 5}** |

## Структура данных «однонаправленный список»

Списки смежности можно хранить, используя два вложенных однонаправленных списка. Однонаправленный список – структура данных, состоящая из элементов, последовательно связанных между собой с помощью указателей. Каждый элемент списка имеет указатель на следующий элемент, а последний элемент списка указывает на nil. В подобном списке можно передвигаться только в сторону его конца: узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно.

## Обходы графов в глубину и ширину

Обходом графа называется процесс посещения каждой вершины графа. Результатом обхода может служить новая нумерация вершин в графе. Существуют два варианта обхода графа: в глубину и в ширину, отличающиеся порядком посещения вершин. В любом случае процесс начинается из указанной заранее вершины.

Обход в глубину использует метод поиска с возвращением, при котором после посещения очередной вершины поиск может либо продолжиться c переходом к смежной вершине, либо произойти откат к предыдущей рассмотренной вершине, если у рассматриваемой не осталось смежных с ней и не посещённых ранее вершин.

Противоположный метод обхода – обход в ширину. В этом случае у очередной вершины сначала посещаются вершины из её окрестности, после чего процесс переходит к окрестностям этих вершин.

Обходы могут описываться в терминах рекурсии, но для итеративной реализации потребуются такие структуры данных как стек и очередь. При посещении новой вершины обходом в глубину её добавляют в стек, а при необходимости отката извлекают последнюю добавленную вершину из стека. Обратный процесс происходит для поиска в ширину: для посещения очередной вершины её сначала извлекают из начала очереди, а после посещения и перед переходом к следующей вершине все вершины из её окрестности помещают в конец очередь. Обходы завершаются в том случае, когда посещены все вершины, иначе говоря, когда стек или очередь станут пусты.

Частным случаем обхода графа в глубину и ширину является обход дерева. Деревом называют граф, не содержащим циклов. Это свойство позволяет быть уверенным, что при обходе никакая из вершин не будет посещена дважды, однако в графах с циклами неизвестно заранее, была ли посещена вершина ранее или нет из-за чего обход может замедлиться или зациклиться. Чтобы избежать этого, вершины после посещения «закрашивают» в некий цвет и перед посещением следующей проверяют цвет вершины. В программе это означает, что необходимо дополнительно хранить массив флагов размером, равным количеству вершин, и изменять значения флагов при посещении очередной.

На деревьях удобно изображать процесс обхода графа. К тому же, понятно их название: при обходе в глубину процесс «спускается всё ниже», пока не дойдёт до самой отдалённой от корня, «глубокой», вершины, а при обходе в ширину процесс идёт «поярусно», посещая все вершины, одинаково удалённые от корня.

В таблице 5 номерами вершин показан порядки обхода дерева.

Таблица 5 - Обходы графа

|  |  |
| --- | --- |
| Обход в глубину | Обход в ширину |
|  |  |

Что касается временной сложности, оба алгоритма имеют одну временную сложность, равную O(n+m).

Для работы алгоритмов необходимы стек или очередь, хранящие от 0 до n вершин графов и n флагов для хранения посещённости вершин, тогда пространственная сложность алгоритмов составляет O(n).

Оба алгоритма одинаково сложны, но для разных видов графов результаты могут отличаться: в графах, где много «тупиков» произойдёт много откатов, а в графе простой цикл откаты будут происходить один за другим после того, как все вершины уже будут помечены, и в задачах поиска могут не происходить, если вершина была найдена.

Обходы в глубину и ширину используют для решения многих задач:

* проверка связности вершин;
* проверка графа на ацикличность;
* проверка графа на двудольность;
* топологическая сортировка;
* выделение связных компонент;
* поиск точек сочленения, мостов;
* и т.д.

Главное отличие алгоритмов состоит в следующем свойстве обхода в ширину: вершина в каждом «ярусе» удалена от начальной на столько же рёбер, каков номер этого «яруса». Таким образом, алгоритм поиска в ширину можно использовать для эффективного нахождения расстояний между вершинами в простых графах. При движении «в глубину» для этого приходилось бы перебирать все простые пути между двумя вершинами.

К тому же алгоритмы используются и для задач, которые можно сформулировать в терминах графов. К примеру, область применения обхода в ширину – поиск в социальных сетях, нахождение соседних узлов в одноранговых сетях вроде BitTorrent, поисковые системы для построения индексов веб-страниц. А наиболее известное применение обхода в глубину – решения головоломок с одним решением вроде лабиринта.

В графах с ориентированными рёбрами (орграфы), петлями (псевдографы) и кратными рёбрами (мультиграфы) алгоритмы работают аналогично.

## Алгоритм Дейкстры для взвешенных графов

Алгоритм был изобретён нидерландским учёным Эдсгером Дейкстрой в 1959 году и используется для поиска кратчайшего расстояния между двумя вершинами во взвешенном графе с неотрицательными весами. При этом с помощью некоторых модификаций возможно сохранение кратчайшего пути, а также нахождение всех расстояний и кратчайших путей из заданной вершины до остальных. Алгоритм можно использовать для простых графов, орграфов, псевдографов и мультиграфов.

Важными понятиями для алгоритма являются понятия временных и постоянных меток, а также процесса релаксации.

Временная метка вершины - верхняя оценка расстояния между текущей вершиной и начальной. Она вычисляется с помощью релаксаций рёбер, инцидентных с вершиной, у которой рассматривается постоянная метка.

Постоянная метка вершины - расстояние между текущей вершиной и начальной. На каждом шаге алгоритма для одной вершины выбирается постоянная метка как минимальная из временных меток.

Изначально, метка начальной вершины считается постоянной и равна 0, а для остальных вершин временным меткам присваивается максимально допустимое значение.

Релаксация по ребру (i, j) заключается в следующем: из найденной временной метки вершины j и суммы текущей постоянной метки вершины i с весом ребра (i, j) выбирается минимальное значение и сохраняется во временную метку j. Таким образом, релаксация - процесс уменьшения временных меток.

Тогда суть алгоритма заключается в следующем: на очередном шаге находится постоянная метка и для соответствующей ей вершины происходят релаксации по инцидентным данной вершине рёбрам. Если же постоянной метке соответствовала искомая вершина, то процесс завершается.

Если необходимо найти сам кратчайший путь, то для сохранения пути необходимо на каждом шаге записывать вершину, соответствующую постоянной метке, в конец уже найденного пути и по завершении цикла записать в конец пути искомую вершину.

Метод схож с поиском в ширину в том смысле, что исследуется окрестность текущей вершины, после чего вершина помечается как посещённая (постоянная метка и закрашенные вершины). При этом вместо добавления в очередь не закрашенных вершин, происходят релаксации по рёбрам, инцидентным этой вершине, а вместо извлечения вершины из начала очереди выбирается вершина с минимальной временной меткой.

Если искомая вершина самая удалённая, то происходит n выборов постоянных меток, а также происходит m релаксаций. В зависимости от используемых структур данных для хранения меток сложность релаксации и выбора постоянной метки могут разниться. В случае массива: релаксация происходит за O(1), а поиск минимума за O(n), тогда в худшем случае временная сложность алгоритма - O(n2+m). Для алгоритма могут быть применены такие древовидные структуры как двоичная и фибоначиева кучи.

Таблица – Сложность алгоритма Дейкстры

|  |  |  |  |
| --- | --- | --- | --- |
| Структура данных | Поиск минимума | Запись элемента | Сложность алгоритма |
| Массив |  |  |  |
| Двоичная куча |  |  |  |
| Фибоначиева куча |  |  |  |

Алгоритм является жадным, то есть на каждом шаге принимает наиболее оптимальное решение для текущего шага, которое в дальнейшем не пересматривается.

Таким образом, после выбора постоянной пометки вершины предполагается, что любой другой путь приведёт к большей длине, чем рассмотренный на текущем шаге. Однако в графах с отрицательными весами это не всегда верно: даже одно ребро не рассмотренного пути может уменьшить расстояние так, что оно станет меньше, чем постоянная метка.

Это можно увидеть на примере орграфа с рисунка 1.2: при поиске из вершины 1 до вершины 3 постоянные метки (указаны около вершин) равны 0, 1, 2, а действительные расстояния до каждой из вершин: 0, -8, 2.

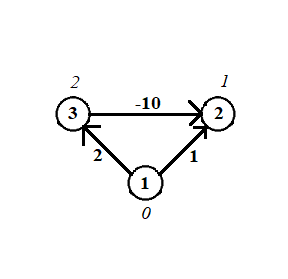


Рисунок .2 – Орграф с отрицательными весами

К тому же, в графах с отрицательными весами кратчайших путей может не существовать вовсе, так как дополнительные проходы по циклу с отрицательным суммарным весом уменьшают кратчайший путь, что верно для графа на рисунке 1.3.

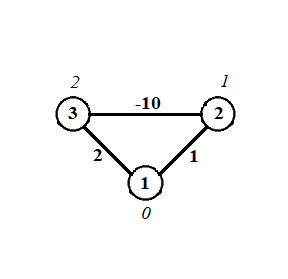


Рисунок 1.3 - Простой цикл с отрицательными весами

Таким образом, для поиска кратчайших путей в графах с отрицательными весами используют другие алгоритмы, к примеру, алгоритмы Беллмана-Форда и Флойда-Уоршелла.

Поиск кратчайшего пути может использоваться для построения маршрута в приложениях вроде Google Maps, при создании компьютерных сетей для обеспечения минимальной задержки, в абстрактных автоматах для определения оптимальной стратегии и т.д.

## Работа с текстовыми файлами

Файловый тип – произвольная последовательность элементов, длина которой заранее не определена, а конкретизируется в процессе выполнения программы. Это определение логического файла, т.е. того, который используется в программе. Физический файл – это поименованная область памяти на внешнем носителе, в которой хранится некоторая информация.

Всего существует три типа файлов:

* типизированные файлы;
* текстовые файлы;
* нетипизированные файлы.

Типизированные файлы связываются с файловыми переменными, объявленными как file of <Тип>. Файл считается состоящим из элементов, каждый из которых имеет тип <Тип>.

Нетипизированные файлы могут быть связаны только с теми файловыми переменными, которые были объявлены как file. Файл считается состоящим из элементов, размер которых определяется при открытии файла.

Текстовый файл представляет собой последовательность символов, однако он не эквивалентен файлу типа file of Char. Текстовые файлы связываются с файловыми переменными, принадлежащими стандартному типу TextFile. Особенность текстовых файлов состоит в том, что содержащиеся в них символы разбиваются на строки. Строки имеют различную длину, а в конце каждой помещается специальный управляющий символ: возврат каретки и символ перехода на новую строку.

Для доступа к отдельным элементам в Delphi существуют специальные стандартные процедуры и функции. Их называют процедурами и функциями ввода-вывода.

Процедура AssignFile (F, Name) связывает файловую переменную с внешним файлом на диске. Здесь F – имя файловой переменной любого типа; Name – выражение строкового типа. Назначение процедуры: организует связь между конкретным физическим файлом на внешнем устройстве (конкретным набором данных) и файловой переменной (логическим файлом).

Процедура Reset (F) открывает существующий файл F. При этом открывается внешний файл с именем, присвоенным переменной F процедурой AssignFile. Если файла не существует, возникает сообщение об ошибке. Указатель файла устанавливается на первую позицию файла. Если файл был предварительно открыт, то он закрывается и повторно открывается. При выполнении процедуры содержимое файла не изменяется. Для текстовых файлов файл открывается в режиме только для чтения (read-only).

Процедура Rewrite (F) создает новый файл и открывает его. Если файл уже открыт, то закрывает и снова открывает его. Указатель файла устанавливается на начало файла (файл создается пустым). Для текстовых файлов – в режиме только для записи (write-only).

Процедура Append (F) предопределена только для текстовых файлов. Она открывает существующий файл для добавления. Если файл уже открыт, то закрывает его, а затем вновь открывает. В данном случае указатель файла устанавливается на конец файла. Файл открывается в режиме только для записи.

Для закрытия файла используется процедура CloseFile (F), где F – файловая переменная, открытая с помощью Reset, Rewrite или Append.

Функции Eof (F) и Eol(F) проверяют на конец файла или символ конца строки соответственно.

Для файлов с типом предопределены процедуры считывания компонентов файла в переменные и записи в файл компонентов из переменных Read (F, V1, V2, …, Vn) и Write (F, V1, V2, …, Vn).

Для файлов без типа есть аналоги процедур Read и Write. Это процедуры BlockRead (F, Buf, Count [, Done]) и BlockWrite (F, Buf, Count [, Done]).

Также есть функции и процедуры, перемещающие указатель файла в указанную позицию, возвращающие текущее положение указателя файла в байтах, возвращающие размер файла в компонентах и др.

# Постановка задачи

Реализовать программное средство для создания ориентированных графов со взвешенными рёбрами положительного веса без петель и кратных рёбер и демонстрации работы обхода в ширину, обхода в глубину и алгоритма Дейкстры.

Для чтения и записи графа использовать 2 вида типизированных файлов: для вершин и рёбер.

Для хранения введённого графа использовать представление графа списками смежности.

Доступный функционал программы:

* добавление и удаление вершины;
* добавление и удаление ребра;
* ввод веса дуги;
* возможность перемещения вершины;
* загрузка и сохранение графа из типизированного файла;
* импорт графа из файлов формата xlsx, xls, xlsb, xlsm;
* поиск пути между вершинами с помощью обхода в глубину;
* поиск пути между вершинами с помощью обхода в ширину;
* поиск пути между вершинами с помощью алгоритма Дейкстры;
* возможность удаления созданного графа;
* возможность чтения файла с информацией о программе.

# Описание алгоритмов решения задачи

Таблица – Описание алгоритмов решения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование алгоритма | Назначение алгоритма | Формальные параметры | Предлагаемый тип реализации |
| 1 | Инициализация главной формы | Загрузка настроек программы. Инициализация графа, установка режима работы с холстом,  инициализация активной вершины |  |  |
| 2 | InitializeGraph  (Graph) | Инициализация графа Graph | Graph – получает от фактического параметра адрес | Процедура |
| 3 | DestroyGraph  (Graph) | Освобождение динамической памяти, выделенной для графа Graph | Graph – получает от фактического параметра адрес | Процедура |
| 4 | DestroyAdjList  (Head) | Освобождение динамической памяти, выделенной для списка смежности с головой Head и её обнуление | Head – получает от фактического параметра адрес | Процедура |
| 5 | AddVertice  (Graph, C) | Создание и добавление в конец списка вершин графа Graph новой вершины с центром в точке C | Graph – получает от фактического параметра адрес;  C – получает от фактического параметра значение | Процедура |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | AddArc  (Graph, v, u, w) | Создание и добавление в начало списка смежности графа Graph вершины v соседа u с инцидентной дугой веса w | Graph – получает от фактического параметра адрес;  v, u, w – получают от фактического параметра значение | Процедура |
| 7 | DeleteVertice  (Graph, v) | Удаление и освобождение памяти, выделенной для вершины v из графа Graph, вместе со всеми инцидентными дугами, и уменьшение номеров вершин, которые больше v | Graph – получает от фактического параметра адрес;  v – получает от фактического параметра значение | Процедура |
| 8 | DeleteArc  (Graph, v, u) | Удаление и освобождение памяти, выделенной для соседа u вершины v графа Graph | Graph – получает от фактического параметра адрес;  v, u – получают от фактического параметра значение | Процедура |
| 9 | IsNeighbour  (Graph,  Vertice,  u,  Res) | Проверка принадлежности вершины u соседям вершины со звеном Vertice в графе Graph | Graph – получает от фактического параметра адрес с защитой;  Vertice – получает от фактического параметра значение;  u – получает от фактического параметра значение;  Res – получает от фактического параметра адрес | Функция  (Res – возвращаемый функцией параметр) |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | GetByNumber  (Graph, v, Res) | Получение звена графа Graph с номером вершины v | Graph – получает от фактического параметра адрес с защитой;  v – получает от фактического параметра значение;  Res – получает от фактического параметра адрес | Функция  (Res – возвращаемый функцией параметр) |
| 11 | GetByPoint  (Graph, P, Res) | Получение звена графа Graph с помощью проверки принадлежности точки P геометрическому месту точек вершины | Graph – получает от фактического параметра адрес с защитой;  P – получает от фактического параметра значение;  Res – получает от фактического параметра адрес | Функция  (Res – возвращаемый функцией параметр) |
| 12 | OpenGraph  (Graph,  VerFileName,  ArcFileName) | Чтение графа Graph из типизированных файлов вершин и дуг с именами VerFileName и ArcFileName соответственно | Graph – получает от фактического параметра адрес;  VerFileName – получает от фактического параметра значение;  ArcFileName – получает от фактического параметра значение | Процедура |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 13 | SaveGraph  (Graph,  VerFileName,  ArcFileName) | Запись графа Graph в типизированные файлы вершин и дуг с именами VerFileName и ArcFileName соответственно | G – получает от фактического параметра адрес с защитой;  VerFileName – получает от фактического параметра значение;  ArcFileName – получает от фактического параметра значение | Процедура |
| 14 | ImportGraph  (Graph,  ExcelFileName) | Импорт графа Graph из электронной таблицы с помощью Excel с именем FileName | Graph – получает от фактического параметра адрес;  ExcelFileName – получает от фактического параметра значение | Процедура |
| 15 | ToWeightMatrix  (Graph, Res) | Преобразование графа Graph из списков смежности в матрицу весов Res | G – получает от фактического параметра адрес с защитой;  Res – получает от фактического параметра адрес | Функция  (Res – возвращаемый функцией параметр) |
| 16 | DFS  (Graph,  Src,  Dest,  Res) | Обход графа Graph в глубину от вершины Src до вершины Dest с информацией об обходе в Res | G – получает от фактического параметра адрес с защитой;  Src, Dest – получают от фактического параметра значение;  Res – получает от фактического параметра адрес | Функция  (Res – возвращаемый функцией параметр) |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 17 | BFS  (Graph,  Src,  Dest,  Res) | Обход графа Graph в ширину от вершины Src до вершины Dest с информацией об обходе в Res | Graph – получает от фактического параметра адрес с защитой;  Src, Dest – получают от фактического параметра значение;  Res – получает от фактического параметра адрес | Функция  (Res – возвращаемый функцией параметр) |
| 18 | Dijkstra  (Graph,  Src,  Dest,  Res) | Поиск с помощью алгоритма Дейкстры в графе Graph пути от вершины Src до вершины Dest с информацией о поиске в Res | Graph – получает от фактического параметра адрес;  Src, Dest – получают от фактического параметра значение;  Res – получает от фактического параметра адрес | Функция  (Res – возвращаемый функцией параметр) |
| 19 | RestorePath  (Graph,  Parents,  Src,  Dest,  Res) | Восстановление пути из вершины Src в Dest графа Graph по массиву предков Parents с сохранением его свойств в Res | Graph – получает от фактического параметра адрес с защитой;  Parents – получает от фактического параметра адрес с защитой;  Src, Dest – получают от фактического параметра значение;  Res – получает от фактического параметра адрес |  |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 20 | InitializeStack  (Stack) | Инициализация стека Stack | Stack – получает от фактического параметра адрес | Процедура |
| 21 | InitializeQueue  (Queue) | Инициализация очереди Queue | Queue – получает от фактического параметра адрес | Процедура |
| 22 | DestroyList  (Head) | Освобождение памяти, выделенной для списка с головой Head и её обнуление | Head – получает от фактического параметра адрес | Процедура |
| 23 | Push  (Stack, n) | Вставка в стек Stack числа n | Stack – получает от фактического параметра адрес;  n – получает от фактического параметра значение | Процедура |
| 24 | Enqueue  (Queue, n) | Добавление в очередь Queue числа n | Queue – получает от фактического параметра адрес;  n – получает от фактического параметра значение | Процедура |
| 25 | Pop  (Stack, Res) | Извлечение числа из стека Stack в Res | Stack – получает от фактического параметра адрес;  Res – получает от фактического параметра адрес | Функция  (Res – возвращаемый функцией параметр) |
| 26 | Dequeue  (Queue, Res) | Извлечение числа из очереди Queue в Res | Queue – получает от фактического параметра адрес;  Res – получает от фактического параметра адрес | Функция  (Res – возвращаемый функцией параметр) |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 27 | MakeVisited  (G, Path) | Изменение стиля раскраски вершин и дуг из пути Path графа Graph на посещённые | Graph – получает от фактического параметра адрес;  Path – получает от фактического параметра адрес | Процедура |
| 28 | MakePassive  (Graph) | Изменение стиля раскраски всех вершин и дуг графа Graph на изначальный | Graph – получает от фактического параметра адрес | Процедура |
| 30 | MakeRegPolygon  (Graph,  Width,  Height) | Изменение местоположение всех вершин графа Graph в вершины правильного многоугольника на холсте | Graph – получает от фактического параметра адрес;  Width, Height – получают от фактического параметра значение | Процедура |

# Структура данных

## Структура типов программы

Таблица – Структура типов программы

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| TVerticeStyle | (stPassive = 0, stActive, stVisited) | Стиль раскраски вершины графа |
| TPNeighbour | ^TNeighbour | Указатель на звено списка смежности |
| TNeighbour | record  Number: Integer;  Weight: Integer;  isVisited: Boolean;  Next: TPNeighbour;  end; | Звено списка смежности;  Number – номер соседа вершины:  Weight – вес дуги, ведущей к соседу;  isVisited – флаг о посещённости дуги, ведущей к соседу, при поиске пути;  Next – ссылка на следующее звено списка |
| TPVertice | ^TVertice | Указатель на звено списка вершин |
| TVertice | record  Number: Integer;  Center: TPoint;  OutDeg: Integer;  Style: TVerticeStyle;  Head: TPNeighbour;  Next: TPVertice;  end; | Звено списка вершин:  Number – номер вершины;  Center – геометрический центр вершины;  OutDeg – полустепень исхода вершины;  Style – стиль раскраски вершины графа;  Head – ссылка на головной элемент списка смежности вершины;  Next – ссылка на следующее звено списка вершин |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| TGraph | record  Head: TPVertice;  Tail: TPVertice;  Order: Integer;  isPainted: Boolean;  R: Integer;  end; | Взвешенный орграф без петель, кратных рёбер и отрицательных весов:  Head – ссылка на голову списка вершин;  Tail – ссылка на хвост списка вершин;  Order – порядок графа;  isPainted – флаг об изменении цветов у элементов графа;  R – радиус окружности, соответствующий вершине на холсте |
| TWeightMatrix | Array of Array of Integer | Матрица весов графа |
| TSearchInfo | record  Path: TStack;  PathSring: String;  ArcsCount: Integer;  Distance: Integer;  VisitsCount: Integer;  end; | Путь с информацией:  Path – список вершин в пути;  PathString – строка с вершинами пути через разделители;  ArcsCount – число дуг в пути;  Distance – длина пути в единицах веса;  Visits – число посещений при поиске |
| TPItem | ^TItem | Ссылка на звено списка натуральных чисел |
| TItem | record  Number: Integer;  Next: TPItem;  end; | Звено списка натуральных чисел:  Number – число в списке;  Next – ссылка на следующее звено списка |
| TStack | TPItem | Стек натуральных чисел |
| TQueue | record  Head: TPItem;  Tail: TPItem;  end; | Очередь натуральных чисел:  Head – голова очереди;  Tail – хвост очереди |

## Структура данных алгоритма InitializeGraph

Таблица – Структура данных алгоритма InitializeGraph (Graph)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Инициализируемый граф | Формальный |

## Структура данных алгоритма DestroyGraph

Таблица – Структура данных алгоритма DestroyGraph (Graph)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Граф, полученный для освобождения памяти, выделенной под него | Формальный |
| Vertice | TPVertice | Ссылка на удаляемую вершину из списка вершин | Локальный |

## Структура данных алгоритма DestroyAdjList

Таблица – Структура данных алгоритма DestroyAdjList (Head)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Head | TPNeighbour | Ссылка на голову списка смежности | Формальный |
| Neighbour | TPNeighbour | Ссылка на удаляемого соседа из списка смежности | Локальный |

## Структура данных алгоритма AddVertice

Таблица – Структура данных алгоритма AddVertice (Graph, C)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Граф, в хвост которого добавляется вершина | Формальный |
| C | TPoint | Геометрическое место центра вершины на холсте | Формальный |
| Vertice | TPVertice | Ссылка на добавляемую вершину графа | Локальный |

## Структура данных алгоритма AddArc

Таблица – Структура данных алгоритма AddArc (Graph, v, u, w)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Граф, в который добавляется дуга  (v, u) | Формальный |
| v | Integer | Начало добавляемой дуги | Формальный |
| u | Integer | Конец добавляемой дуги | Формальный |
| w | Integer | Вес добавляемой дуги | Формальный |
| Vertice | TPVertice | Ссылка на вершину с номером v | Локальный |
| AdjVertice | TPVertice | Ссылка на вершину с номером u | Локальный |
| Neighbour | TPNeighbour | Ссылка на добавляемого соседа | Локальный |
| isIncorrect | Boolean | Флаг об отсутствии кратных дуг и петель в случае добавления дуги:  true – таких дуг нет;  false - есть | Локальный |

## Структура данных алгоритма DeleteVertice

Таблица – Структура данных алгоритма DeleteVertice (Graph, v)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Граф, из которого удаляется вершина v | Формальный |
| v | Integer | Номер удаляемой вершины | Формальный |
| Vertice | TPVertice | Ссылка на текущую вершину | Локальный |
| PrVertice | TPVertice | Ссылка на вершину в списке вершин перед Vertice | Локальный |
| Neighbour | TPNeighbour | Ссылка на текущего соседа вершины | Локальный |
| PrNeighbour | TPNeighbour | Ссылка на соседа в списке смежности перед Neighbour | Локальный |
| Exists | Boolean | Флаг о наличии вершины | Локальный |

## Структура данных алгоритма DeleteArc

Таблица – Структура данных алгоритма DeleteArc (Graph, v, u)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Граф, из которого удаляется дуга (v, u) | Формальный |
| v | Integer | Номер вершины-начала дуги | Формальный |
| u | Integer | Номер вершины-конца дуги | Формальный |
| Vertice | TPVertice | Ссылка на вершину-начало дуги | Локальный |
| Neighbour | TPNeighbour | Ссылка на искомого удаляемого соседа вершины | Локальный |
| PrNeighbour | TPNeighbour | Ссылка на соседа в списке смежности перед Neighbour | Локальный |
| Exists | Boolean | Флаг о наличии дуги | Локальный |

## Структура данных алгоритма IsNeighbour

Таблица – Структура данных алгоритма IsNeighbour (Graph, Vertice, u, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Полученный граф | Формальный |
| Vertice | TPVertice | Ссылка на вершину-начало дуги | Формальный |
| u | Integer | Номер вершины-конца дуги | Формальный |
| Res | Boolean | Результат проверки | Формальный |
| Neighbour | TPNeighbour | Ссылка на сосед с номером u | Локальный |

## Структура данных алгоритма GetByNumber

Таблица – Структура данных алгоритма GetByNumber (Graph, v, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Полученный граф | Формальный |
| v | Integer | Номер вершины для поиска | Формальный |
| Res | TPVertice | Ссылка на искомое звено списка вершин | Формальный |

## Структура данных алгоритма GetByPoint

Таблица – Структура данных алгоритма GetByPoint (Graph, P, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Полученный граф | Формальный |
| P | TPoint | Переданная координата холста | Формальный |
| Res | TPVertice | Ссылка на искомое звено списка вершин | Формальный |
| Vertice | TPVertice | Ссылка на текущую вершину графа | Локальный |

## Структура данных алгоритма OpenGraph

Таблица – Структура данных алгоритма OpenGraph (Graph, VerFileName, ArcFileName)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Неинициализированный граф для чтения из файлов | Формальный |
| VerFileName | String | Имя файла с информацией о вершинах графа | Формальный |
| ArcFileName | String | Имя файла с информацией о дугах графа | Формальный |
| VerFile | File of TVertice | Типизированный файл вершин | Локальный |
| ArcFile | File of TNeighbour | Типизированный файл соседей | Локальный |
| Vertice | TPVertice | Ссылка на текущую вершину графа | Локальный |
| Neighbour | TPNeighbour | Ссылка на текущего соседа вершины | Локальный |
| v | Integer | Параметр цикла по соседям вершины | Локальный |

## Структура данных алгоритма SaveGraph

Таблица – Структура данных алгоритма SaveGraph (Graph, VerFileName, ArcFileName)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Граф для чтения из файлов | Формальный |
| VerFileName | String | Имя файла с информацией о вершинах графа | Формальный |
| ArcFileName | String | Имя файла с информацией о дугах графа | Формальный |
| VerFile | File of TVertice | Типизированный файл вершин | Локальный |

Продолжение таблицы 15

|  |  |  |  |
| --- | --- | --- | --- |
| ArcFile | File of TNeighbour | Типизированный файл соседей | Локальный |
| Vertice | TPVertice | Ссылка на текущую вершину графа | Локальный |
| Neighbour | TPNeighbour | Ссылка на текущего соседа вершины | Локальный |

## Структура данных алгоритма ImportGraph

Таблица – Структура данных алгоритма ImportGraph (Graph, ExcelFileName)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Граф для импорта из книги Excel | Формальный |
| ExcelFileName | String | Имя книги Excel с матрицей весов | Формальный |
| MyExcel | Variant | Объект для открытия Excel | Локальный |
| Sheet | OLEVariant | Объект активного листа книги | Локальный |
| Weights | OLEVariant | Объект диапазона ячеек с матрицей весов | Локальный |
| CLSID | TCLSID | Идентификатор, определяющий тип COM объекта | Локальный |
| i | Integer | Параметр цикла по строкам Weights | Локальный |
| j | Integer | Параметр цикла по столбцам Weights | Локальный |
| Rows | Integer | Количество строк Weights | Локальный |
| Cols | Integer | Количество столбцов Weights | Локальный |
| w | Integer | Значение ячейки Weights для записи веса дуги | Локальный |

Продолжение таблицы 16

|  |  |  |  |
| --- | --- | --- | --- |
| ww | Integer | Значение ячейки Weights, симметричное w относительно главной диагонали | Локальный |
| IsIncorrect | Boolean | Флаг о существовании в Weights недопустимых дуг | Локальный |

## Структура данных алгоритма ToWeightMatrix

Таблица – Структура данных алгоритма ToWeightMatrix (Graph, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Полученный для преобразования граф | Формальный |
| Res | TWeightMatrix | Матрица весов полученного графа | Формальный |
| Vertice | TPVertice | Ссылка на текущую вершину графа | Локальный |
| Neighbour | TPNeighbour | Ссылка на текущего соседа вершины | Локальный |
| v | Integer | Параметр цикла по вершинам | Локальный |
| u | Integer | Параметр цикла по соседям | Локальный |

## Структура данных алгоритма DFS

Таблица – Структура данных алгоритма DFS (Graph, Src, Dest, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TWeightMatrix | Полученный для поиска граф | Формальный |
| Src | Integer | Номер вершины-начала пути | Формальный |
| Dest | Integer | Номер вершины-конца пути | Формальный |

Продолжение таблицы 18

|  |  |  |  |
| --- | --- | --- | --- |
| Res | TSearchInfo | Результат поиска в глубину | Формальный |
| v | Integer | Номер посещаемой вершины | Локальный |
| u | Integer | Номер соседа посещаемой вершины | Локальный |
| Order | Integer | Порядок графа | Локальный |
| Stack | TStack | Стек вершин для итеративного поиска в глубину | Локальный |
| isVisited | Array of Boolean | Массив флагов о посещённости вершин | Локальный |
| Parents | Array of Integer | Массив предков вершин | Локальный |

## Структура данных алгоритма BFS

Таблица – Структура данных алгоритма BFS (Graph, Src, Dest, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TWeightMatrix | Полученный для поиска граф | Формальный |
| Src | Integer | Номер вершины-начала пути | Формальный |
| Dest | Integer | Номер вершины-конца пути | Формальный |
| Res | TSearchInfo | Результат поиска в ширину | Формальный |
| v | Integer | Номер посещаемой вершины | Локальный |
| u | Integer | Номер соседа посещаемой вершины | Локальный |
| Order | Integer | Порядок графа | Локальный |
| Queue | TQueue | Очередь вершин для итеративного поиска в ширину | Локальный |

Продолжение таблицы 19

|  |  |  |  |
| --- | --- | --- | --- |
| isVisited | Array of Boolean | Массив флагов о том, что вершина уже была добавлена в стек | Локальный |
| Parents | Array of Integer | Массив предков вершин | Локальный |

## Структура данных алгоритма Dijkstra

Таблица – Структура данных алгоритма Dijkstra (Graph, Src, Dest, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TWeightMatrix | Полученный для поиска граф | Формальный |
| Src | Integer | Номер вершины-начала пути | Формальный |
| Dest | Integer | Номер вершины-конца пути | Формальный |
| Res | TSearchInfo | Результат поиска в ширину | Формальный |
| v | Integer | Номер посещаемой вершины | Локальный |
| u | Integer | Номер соседа посещаемой вершины | Локальный |
| Order | Integer | Порядок графа | Локальный |
| Marks | Array of Integer | Массив меток (расстояний от начальной) вершин | Локальный |
| isVisited | Array of Boolean | Массив флагов о том, что вершина уже была посещена | Локальный |
| Parents | Array of Integer | Массив предков вершин | Локальный |
| d | Integer | Расстояние до текущей вершины | Локальный |

## Структура данных алгоритма RestorePath

Таблица 21 – Структура данных алгоритма RestorePath (Graph, Parents, Src, Dest, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TWeightMatrix | Полученный для поиска граф | Формальный |
| Parents | Array of Integer | Массив предков вершин | Формальный |
| Src | Integer | Номер вершины-начала пути | Формальный |
| Dest | Integer | Номер вершины-конца пути | Формальный |
| Res | TSearchInfo | Результат поиска в ширину | Формальный |
| v | Integer | Номер вершины-начала дуги | Локальный |
| u | Integer | Номер вершины-конца дуги | Локальный |
| Splitter | String | Разделитель между вершинами в строке пути | Локальный |

## Структура данных алгоритма InitializeStack

Таблица – Структура данных алгоритма InitializeStack (Stack)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Stack | TStack | Стек для инициализации нулевой ссылкой | Формальный |

## Структура данных алгоритма InitializeQueue

Таблица – Структура данных алгоритма InitializeQueue (Queue)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Queue | TQueue | Очередь для инициализации её хвоста и головы нулевыми ссылками | Формальный |

## Структура данных алгоритма DestroyList

Таблица – Структура данных алгоритма DestroyList (Head)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Head | TPItem | Ссылка на голову освобождаемого списка | Формальный |
| Item | TPItem | Ссылка освобождаемый на данном шаге узел | Локальный |

## Структура данных алгоритма Push

Таблица – Структура данных алгоритма Push (Stack, n)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Stack | TStack | Стек для вставки | Формальный |
| n | Integer | Значение вставляемого звена | Формальный |
| Item | TPItem | Ссылка на вставляемое звено | Локальный |

## Структура данных алгоритма Enqueue

Таблица – Структура данных алгоритма Enqueue (Queue, n)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Queue | TQueue | Очередь для добавления | Формальный |
| n | Integer | Значение вставляемого звена | Формальный |
| Item | TPItem | Ссылка на вставляемое звено | Локальный |

## Структура данных алгоритма Pop

Таблица – Структура данных алгоритма Pop (Stack, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Stack | TStack | Стек для извлечения | Формальный |
| Res | Integer | Значение извлекаемого звена | Формальный |
| Item | TStack | Ссылка на извлекаемое звено | Локальный |

## Структура данных алгоритма Dequeue

Таблица – Структура данных алгоритма Dequeue (Queue, Res)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Queue | TQueue | Очередь для извлечения | Формальный |
| Res | Integer | Значение извлекаемого звена | Формальный |
| Item | TPItem | Ссылка на извлекаемое звено | Локальный |

## Структура данных алгоритма MakeVisited

Таблица – Структура данных алгоритма MakeVisited (Graph, Path)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Полученный граф | Формальный |
| Path | TStack | Полученный путь | Формальный |
| Vertice | TPVertice | Ссылка на текущую вершину | Локальный |
| Neighbour | TPNeighbour | Ссылка на текущего соседа | Локальный |
| v | Integer | Номер вершины, извлечённой из пути | Локальный |

## Структура данных алгоритма MakePassive

Таблица – Структура данных алгоритма MakePassive (Graph)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Полученный граф | Формальный |
| Vertice | TPVertice | Ссылка на текущую вершину | Локальный |
| Neighbour | TPNeighbour | Ссылка на текущего соседа | Локальный |

## Структура данных алгоритма MakeRegPolygon

Таблица – Структура данных алгоритма MakeRegPolygon (Graph, Width, Height)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Graph | TGraph | Полученный граф | Формальный |
| Width | Integer | Ширина квадрата на холсте, который ограничивает граф | Формальный |
| Height | Integer | Высота квадрата на холсте, который ограничивает граф | Формальный |
| Vertice | TPVertice | Ссылка на текущую вершину | Локальный |
| ImageCenter | TPoint | Координата центра квадрата, который ограничивает граф | Локальный |
| PolygonRadius | Integer | Радиус правильного многоугольника | Локальный |
| Angle | Real | Текущий угол наклона радиус-вектора вершины относительно начального положения | Локальный |