МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

институт информационных технологий и технологического образования
кафедра информационных технологий и электронного обучения

Основная профессиональная образовательная программа
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль) «Технологии разработки программного обеспечения»
форма обучения – очная

Курсовая работа

по дисциплине «Технологии компьютерного моделирования»

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ДРЕВОВИДНЫХ СТРУКТУР

Обучающегося 2 курса
Стецук Максима Николаевича
_____

Руководитель:
д.п.н, профессор
_____ Власова Е. З.

«_____» _____ 2023 г.

Санкт-Петербург
2023

Оглавление

Введение

Актуальность изучения данной темы заключается в повсеместном использовании графов и различных древовидных структур в повседневной жизни, как отдельно взятого человека, так и крупных организаций. В 1736 году зародилась основа данной теории: благодаря Леонарду Эйлеру было получено решение такой известной задачи, как задача о семи кёнигсбергских мостах, которая впоследствии стала одной из классических задач теории графов.

В наше время актуальность и стремительное развитие науки в области теории графов, связано с развитием технологических процессов и появлением информационных технологий, в связи, с чем отражение и возможность применения данной теории было найдено в различных сферах жизни человека. Изначально данная теория применялась сугубо для решения теоретических задач различных разделов математики и химии, но постепенно, ей нашлось не только абстрактное, но и практическое применение в реальном мире, а также применение в различных технологических и компьютерных процессах.

Теория графов является инструментом, который, используется в науках о поведении, таких как: теория информации, кибернетика, теория игр, которая представляет собой задачи на нахождение оптимальной стратегии, теория систем и транспортные сети. Данная теория нашла своё отражение и в современных абстрактных дисциплинах: теория множеств, теория групп, теория матриц и многих других.

Исходя из всей важности теории графов для развития различных современных отраслей науки и техники, данное исследование, также является важным в современном мире. Такая структура, как остовное дерево графа (или остовный граф) нашла своё применение в задачах различного рода, таких как задачи оптимизации, в которых необходимо приводить к

оптимальному виду некоторую структуру или находить оптимальное решение вопроса для минимизации затрат, транспортных задачах, задачах на нахождение оптимальных стратегий и многих других. Также данная структура активно применяется при проектировании и создании различных систем, таких как базы данных, где большую роль играет оптимизация структуры и её читабельность для различных пользователей, компьютерные сети, а именно маршрутизация пакетов в компьютерной сети, искусственный интеллект, электрические цепи, системы компьютерного зрения и тому подобные.

Целью данного исследования является разработка алгоритма для работы с графами и компьютерное моделирование остовных деревьев.

Задачи исследования:

- изучение специальной литературы по теме данного исследования;
- рассмотрение содержания ключевых понятий и определений;
- разработка и программная реализация алгоритма для работы с графами;
- компьютерное моделирование остовных деревьев, путём преобразования и оптимизации структуры начальных графов на примере полных и неполных графов с различным количеством вершин.

При выполнении курсовой работы были использованы учебная и справочная литература, а также электронные ресурсы и ресурсы сети Интернет.

Глава 1. Теоретический блок

1.1 Основные определения и понятия

Теория графов - обширный раздел дискретной математики, в котором системно изучают свойства графов.

Граф - это геометрическая фигура, которая состоит из точек и линий, которые их соединяют. Точки называют вершинами графа, а линии – ребрами графа.

В некоторых случаях ребро графа определяют как неупорядоченную пару двух вершин. Эти вершины называются концевыми точками или концами ребра.

В математике под графом принято понимать как объект, который является совокупностью двух множеств - множества самих объектов, которое называют множеством вершин, и множества их парных связей, называемого множеством рёбер. Пример графа представлен на рисунке 1.1.

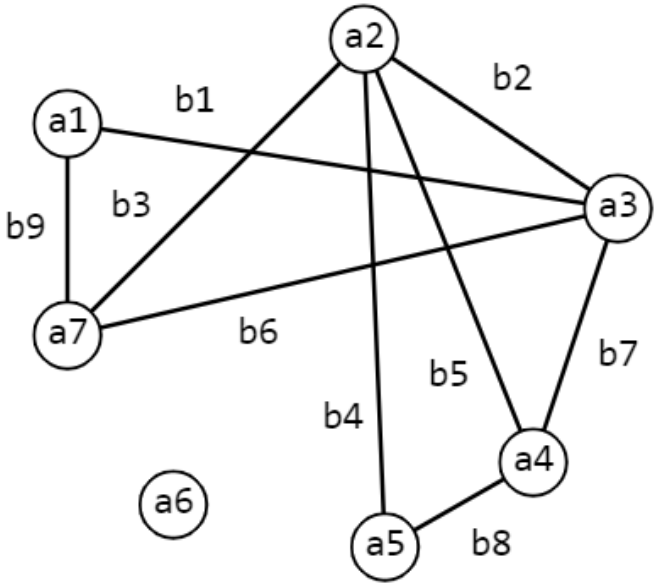

Рисунок 1.1. Граф 1

Пояснение к определению графа: Пусть множество A = {a1, a2, a3, …, a7} - множество городов некоторой страны, а множество B = {b1, b2, b3, …, b9} - множество железнодорожных путей между этими городами. Тогда отношение связи городов из множества A, железнодорожными путями из множества B, представляется графически в виде графа, где элементы множества A являются вершинами, а элементы множества B - рёбрами.

В данном случае видно, что вершина a6 не связана ни с одной из других вершин. В теории графов такие вершины принято называть изолированными вершинами графа.

Степенью вершины называют количество ребер графа, для которых она является концевой (то есть, количество рёбер, которые заканчиваются в данной вершине).

Изолированная вершина графа - это вершина, степень которой равна нулю. То есть это вершина, не являющаяся концевой ни для какого ребра.

Висячая вершина графа (лист) - это вершина, степень которой равна единице.

Компонента связности графа - некоторое множество вершин графа такое, что для любых двух вершин из этого множества существует путь из одной в другую, и не существует пути из вершины этого множества в вершину не из этого множества.

Связный граф - граф, содержащий ровно одну компоненту связности. Это означает, что между любой парой вершин этого графа существует как минимум один путь.

Если рассмотреть рисунок 1.1, то видно, что данный граф имеет 2 компоненты связности, G_1 = {a1, a2, a3, a4, a5, a7} и G_2 = {a6}.

Если отдельно рассматривать связь между множествами A’ = {a1, a2, a3, a4, a5, a7} и B, то полученный граф будет иметь одну компоненту связности, и будет являться связным, данный граф представлен на рисунке 1.2.

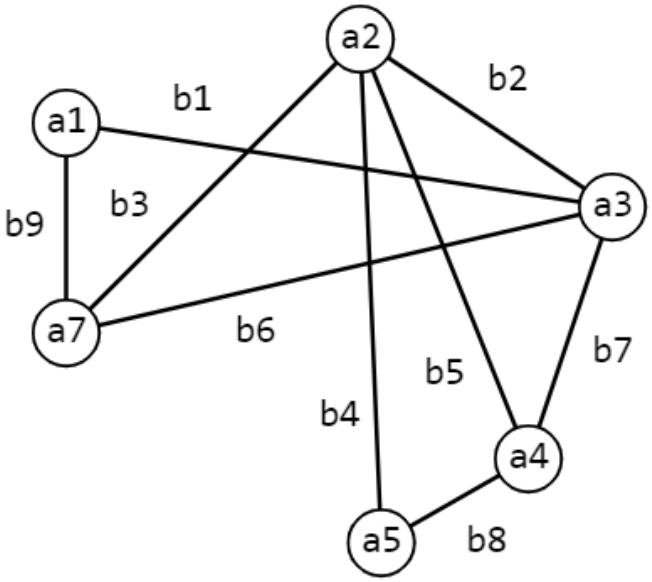

Рисунок 1.2. Граф 2

Полный граф - это граф, в котором каждые две вершины соединены одним ребром. Пример полного графа представлен на рисунке 1.3.

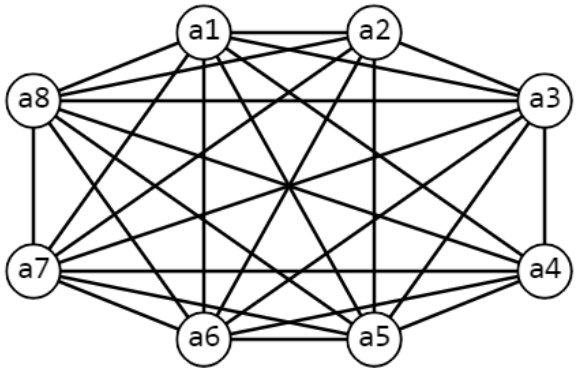

Рисунок 1.3. Полный граф

Важно отметить, что теория графов не учитывает происхождение рассматриваемых множеств A и B. То есть граф - это топологичекая модель, которая состоит из множества вершин и множества соединяющих их рёбер. При этом значение имеет только сам факт, какая вершина, с какой соединена.

Теория графов применяется при решении задач различных областей, и в некоторых случаях можно временно опускать содержание конкретных элементов множеств, однако данная специфика не отражается на ходе решения конкретных задач.

1.2 Остовное дерево графа

Подграф - совокупность вершин, которые связанны рёбрами, и выбраны в начальном графе.

Например: Подграфом для графа из рисунка 1.1, может являться граф, который графически представляет связь между двумя подмножествами множеств A и B, A’ = {a1, a3, a4, a7} и B’ = {b1, b6, b7, b9}. Данный подграф выделен на рисунке 1.4.

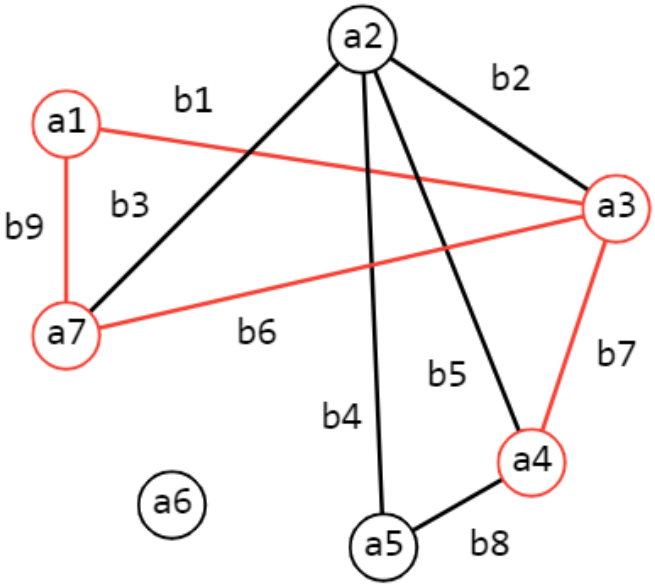

Рисунок 1.4. Подграф графа 1

Дерево - связный граф, в котором не содержится циклов (то есть, для любой вершины существует один способ добраться до любой другой вершины).

Из определения вытекает важное свойство о количестве вершин и рёбер в дереве: количество рёбер на ё меньше, чем количество вершин.

Корень дерева - вершина, которая при “подвешивании” дерева окажется на нулевом уровне, при нумерации от 0 до n сверху вниз. Пример дерева изображён на рисунке 1.5 (вершина a0 в дереве 1, является его корнем).

Из данного определения вытекает понятие древовидной структуры, а именно - это такая структура, в которой каждая вершина связана хотя бы с одной другой вершиной, причём существует один и только один способ попасть из одной вершины в другую.

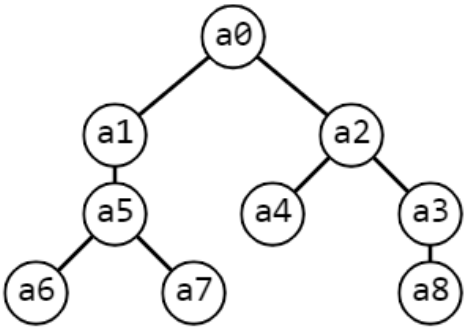

Рисунок 1.5. Дерево 1

Остовное дерево графа - это дерево, которое является подграфом начального графа, причём оно содержит все его вершины.

Можно сказать, что для получения остовного дерева, необходимо из начального графа удалить лишние рёбра, которые входят в циклы, однако при этом не должна быть нарушена связность графа. Пример выделения остовного дерева для графа из рисунка 1.6, с корнем в вершине a1, представлен на рисунке 1.7 (слева остовное дерево выделено прямо на графе 3, а справа оно представлено в виде древовидной структуры).

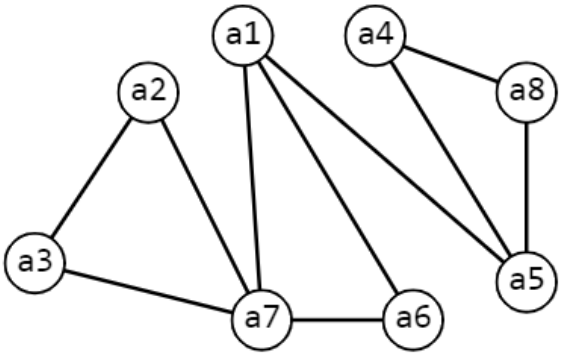

Рисунок 1.6. Граф 3

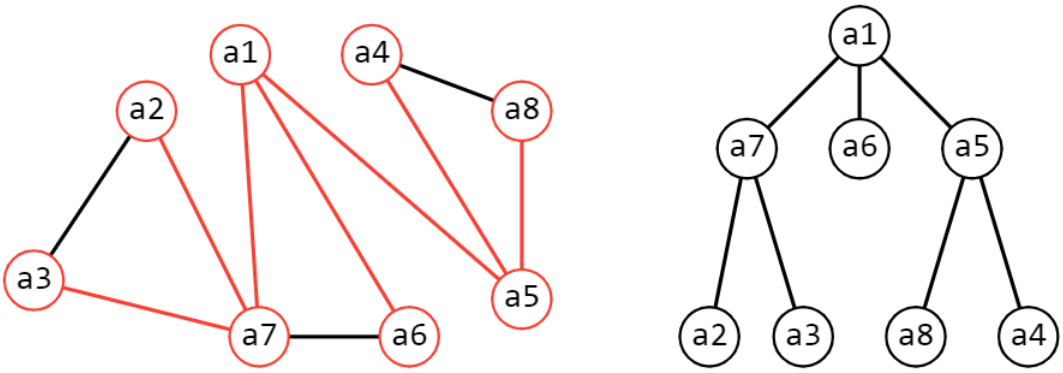

Рисунок 1.7. Остовное дерево графа 3

Глава 2. Практический блок

2.1 Алгоритм преобразования графа и моделирования остовного дерева

В ходе выполнения курсовой работы, мной был разработан алгоритм для преобразования графа к остовному дереву и моделирования полученного дерева, путём удаления лишних рёбер из начального графа. Полная программная реализация разработанного алгоритма на языке программирования Python представлена в приложении 1, отдельные части программы сопровождаются комментариями, которые описывают задачу отдельных фрагментов кода.

Алгоритм состоит из четырёх блоков, каждый из которых отвечает за определённый процесс в данном алгоритме.

АЛГОРИТМ

Блок 1. Ввод начального графа

В данном блоке происходит получение информации о начальном графе, получаются сведения о количестве вершин начального графа и о связях между этими вершинами (рёбрах данного графа).

В программном варианте ввод реализован путём выбора количества вершин в начальном графе, а затем последовательным указанием связей для каждой вершины. Полученные данные программа преобразует в два массива: массив вершин и массив связей, причём массив связей реализуется путём создания массива, элементами которого являются массивы, содержащие два элемента, а именно имена вершин, между которыми есть связь в начальном графе.

Блок 2. Размещение вершин по уровням

В данном блоке происходит непосредственное размещение вершин начального графа по уровням, на которых они будут располагаться при выделении остовного дерева из начального графа, путём обхода вершин от первой до последней, пока все вершины не будут размещены по уровням.

На данном этапе программа создаёт два массива, сначала создаётся вспомогательный массив элементов по уровням, а затем массив по уровням без вспомогательного элемента. Массив элементов по уровням с вспомогательных элементов будет использоваться программой в следующем блоке.

В ходе выполнения данного блока программы, пользователь выбирает вершину, которая станет корнем дерева, а программа записывает её в качестве первого элемента для обоих массивов.

Для создания массива элементов по уровням со вспомогательным элементом программа использует массив всех элементов, из которого в ходе добавления новых вершин в массив элементов по уровням, удаляются вершины, которые уже добавлены, во избежание повторного добавления уже использованной вершины. Каждый элемент в массиве элементов по уровням со вспомогательным элементов, является массивом, первый элемент каждого из которых указывает на связь следующих элементов с ним, кроме первого, который является корнем для будущего дерева.

Затем по аналогии, используя массив всех элементов для удаления и проверки на наличие в нём проверяемой вершины, программа создаёт массив вершин по уровням, каждый из уровней является массивом, который содержит все вершины конкретного уровня, без каких либо вспомогательных элементов.

Блок 3. Массив финальных связей

В данном блоке происходит выборка связей из начального графа, которые необходимо оставить, для построения остовного дерева, которое будет являться связным графом, причём никаких новых, ранее не существовавших связей не добавляется.

Программа, используя массив вершин по уровням со вспомогательным элементом, создаёт новый массив связей, который, как и начальный массив связей состоит из массивов, содержащих пары вершин, которые связаны ребром. Она проходит по вспомогательному массиву вершин, пропуская первый элемент, который является корнем дерева, и создаёт пары вершин, которые состоят из первого элемента каждого массива из массива вершин по уровням и каждой из оставшихся вершин в данном массиве. Например, если массив вершин по уровням имел вид [a1, [a1, a2, a3], [a2, a4]], то программа создаст массив связей [[a1, a2], [a1, a3], [a2, a4]].

Блок 4. Графическое представление дерева

В данном блоке используются связи, которые не были исключены из списка начальных связей, а также используются списки вершин для каждого уровня дерева, для графического представления остовного дерева начального графа.

Программа задаёт каждой вершине графа координаты, при этом создаётся словарь, ключами которого являются вершины, а значениями ключей являются пары чисел, которые являются координатами x и y, располагая их на различных уровнях, для чего используется массив вершин по уровням. После этого происходит моделирование данного графа и его графическое представление, для чего программа использует массив связей, который был создан в блоке 3.

Для реализации графического вывода программа использует библиотеку tKinter языка Python, которая предназначена для работы с графикой в различных iDE, поддерживающих данный язык программирования.

Итерируясь по массиву связей из блока 3, программа строит рёбра остовного дерева, концами которых являются соответствующие координаты из словаря координат вершин. А затем, с помощью итераций по массиву вершин по уровням без вспомогательных элементов, программа изображает вершины в виде окружностей с координатой центра, берущейся из словаря координат вершин. А также указывает для каждой вершины её имя.

2.2 Полный граф

В параграфах 2.2 и 2.3 будет демонстрироваться работа алгоритма на примере решения задачи теории графов, общее условие которой представлено далее.

Условие задачи: В некоторой стране X, необходимо построить железнодорожные пути между городами, группа инженеров спроектировала план, представленный на изображение Y, однако данный план необходимо оптимизировать, чтобы количество путей было наименьшим, причём путём необходимо считать прямую связь между двумя городами.

Анализируя условие, приходим к выводу, что для решения данной задачи необходимо в представленном в условии плане Y выделить остовное дерево, т.к. из определения именно оно является связным графом с наименьшим возможным количеством рёбер среди всех подграфов исходного графа.

Пример 1

План из условия задачи представлен на рисунке 2.1.

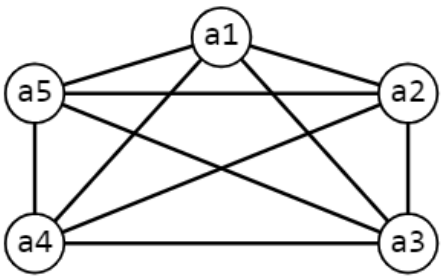

Рисунок 2.1. Пример 1

Программа получила на вход количество вершин, равное 5 и массив связей, который имеет вид: [['a1', 'a2'], ['a1', 'a3'], ['a1', 'a4'], ['a1', 'a5'], ['a2', 'a3'], ['a2', 'a4'], ['a2', 'a5'], ['a3', 'a4'], ['a3', 'a5'], ['a4', 'a5']].

Затем, для получения решения необходимо выбрать вершину графа из рисунка 2.1, которая будет являться корнем остовного дерева. Выберем вершину a1.

После выполнения алгоритма, программа получила массив вершин, распределённых по уровням A и массив итоговых связей B:

A = [['a1'], ['a2', 'a3', 'a4', 'a5']];

B = [['a1', 'a2'], ['a1', 'a3'], ['a1', 'a4'], ['a1', 'a5']].

Графическое представление связи между множествами A и B, которое является остовным деревом графа из рисунка 2.1 и решением для заданной условием в примере 1 задачи, также было смоделировано программой и представлено на рисунке 2.2.

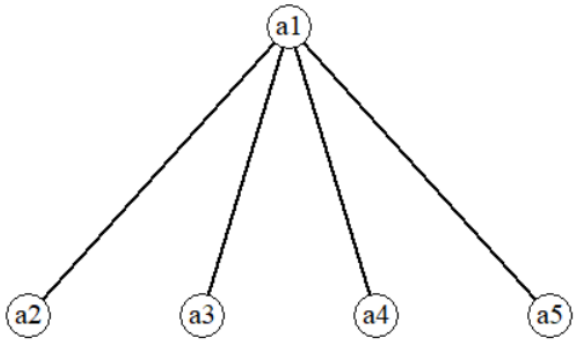

Рисунок 2.2. Остовное дерево для примера 1

Пример 2

План из условия задачи представлен на рисунке 2.3.

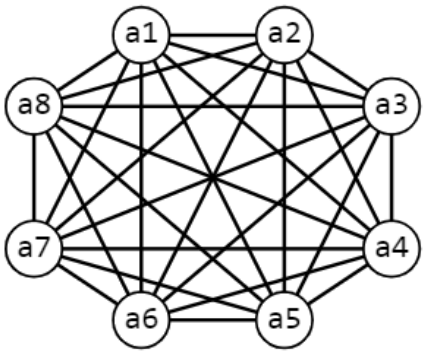

Рисунок 2.3. Пример 2

Программа получила на вход количество вершин, равное 8 и массив связей, который имеет вид: [['a1', 'a2'], ['a1', 'a3'], ['a1', 'a4'], ['a1', 'a5'], ['a1', 'a6'], ['a1', 'a7'], ['a1', 'a8'], ['a2', 'a3'], ['a2', 'a4'], ['a2', 'a5'], ['a2', 'a6'], ['a2', 'a7'], ['a2', 'a8'], ['a3', 'a4'], ['a3', 'a5'], ['a3', 'a6'], ['a3', 'a7'], ['a3', 'a8'], ['a4', 'a5'], ['a4', 'a6'], ['a4', 'a7'], ['a4', 'a8'], ['a5', 'a6'], ['a5', 'a7'], ['a5', 'a8'], ['a6', 'a7'], ['a6', 'a8'], ['a7', 'a8']].

В качестве корня для будущего остовного дерева, которое будет получено из начального графа, выбрана вершина a2.

После выполнения алгоритма, программа получила массив вершин, распределённых по уровням A и массив итоговых связей B:

A = [['a2'], ['a1', 'a3', 'a4', 'a5', 'a6', 'a7', 'a8']];

B = [['a2', 'a1'], ['a2', 'a3'], ['a2', 'a4'], ['a2', 'a5'], ['a2', 'a6'], ['a2', 'a7'], ['a2', 'a8']].

Графическое представление связи между множествами A и B, которое является остовным деревом графа из рисунка 2.3 и решением для заданной условием в примере 2 задачи, также было смоделировано программой и представлено на рисунке 2.4.

Рисунок 2.4. Остовное дерево для примера 2

2.3 Неполный граф

Пример 3

План из условия задачи представлен на рисунке 2.5.

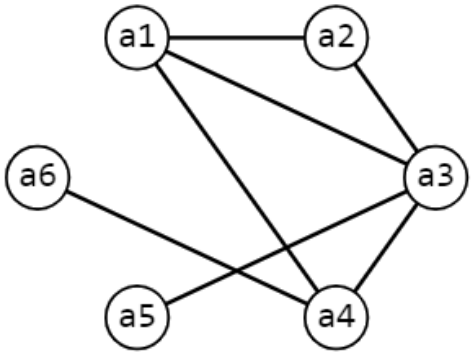

Рисунок 2.5. Пример 3

Программа получила на вход количество вершин, равное 6 и массив связей, который имеет вид: [['a1', 'a2'], ['a1', 'a3'], ['a1', 'a4'], ['a2', 'a3'], ['a3', 'a4'], ['a3', 'a5'], ['a4', 'a6']].

Так как в данном случае план по условию является неполным графом, то в зависимости от выбора вершины, которая станет корнем остовного дерева, решения задачи будут существенно отличаться по своему виду. Рассмотрим два случая, когда в качестве корня выбрана вершина a1 и когда в качестве корня выбрана вершина a2.

1. Корень - вершина a1, тогда после выполнения алгоритма, программа получила массив вершин, распределённых по уровням A и массив итоговых связей B:

A = [['a1'], ['a2', 'a3', 'a4'], ['a5', 'a6']];

B = [['a1', 'a2'], ['a1', 'a3'], ['a1', 'a4'], ['a3', 'a5'], ['a4', 'a6']].

2. Корень - вершина a2, тогда после выполнения алгоритма, программа получила массив вершин, распределённых по уровням A и массив итоговых связей B:

A = [['a2'], ['a1', 'a3'], ['a4', 'a5'], ['a6']];

B = [['a2', 'a1'], ['a2', 'a3'], ['a1', 'a4'], ['a3', 'a5'], ['a4', 'a6']].

Графические представления связи между множествами A и B для случаев 1 и 2, которые является вариантами остовных деревьев графа из рисунка 2.5 и решениями для заданной условием в примере 3 задачи, также были смоделированы программой и представлены на рисунке 2.6 справа и слева соответственно.

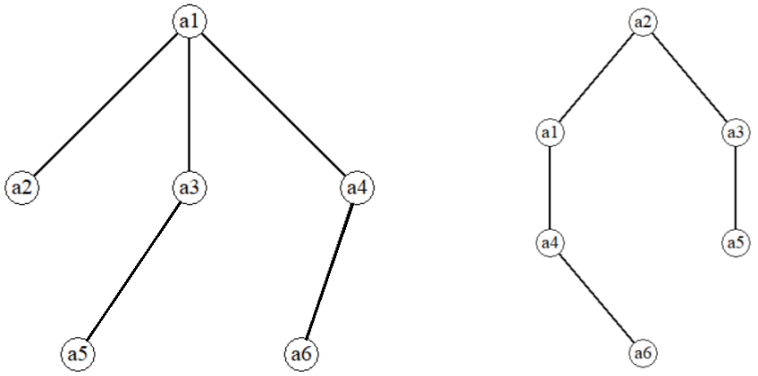

Рисунок 2.6. Остовные деревья для примера 3

Пример 4

План из условия задачи представлен на рисунке 2.7.

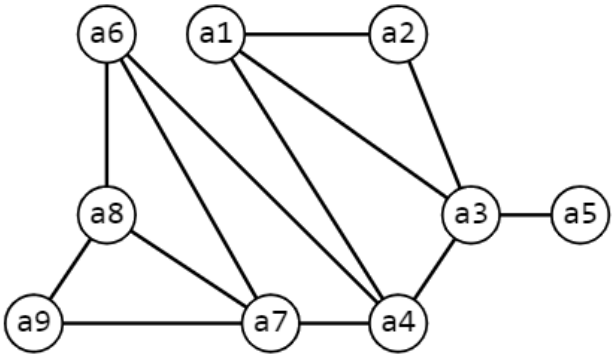

Рисунок 2.7. Пример 4

Программа получила на вход количество вершин, равное 9 и массив связей, который имеет вид: [['a1', 'a2'], ['a1', 'a3'], ['a1', 'a4'], ['a2', 'a3'], ['a3', 'a5'], ['a3', 'a4'], ['a4', 'a6'], ['a4', 'a7'], ['a6', 'a7'], ['a6', 'a8'], ['a7', 'a8'], ['a7', 'a9'], ['a8', 'a9']].

По аналогии с примером 3, рассмотрим несколько вариантов корня для остовного дерева.

1. Корень - вершина a4, тогда после выполнения алгоритма, программа получила массив вершин, распределённых по уровням A и массив итоговых связей B:

A = [['a4'], ['a1', 'a3', 'a6', 'a7'], ['a2', 'a5', 'a8', 'a9']];

B = [['a4', 'a1'], ['a4', 'a3'], ['a4', 'a6'], ['a4', 'a7'], ['a1', 'a2'], ['a3', 'a5'], ['a6', 'a8'], ['a7', 'a9']].

2. Корень - вершина a1, тогда после выполнения алгоритма, программа получила массив вершин, распределённых по уровням A и массив итоговых связей B:

A = [['a1'], ['a2', 'a3', 'a4'], ['a5', 'a6', 'a7'], ['a8', 'a9']];

B = [['a1', 'a2'], ['a1', 'a3'], ['a1', 'a4'], ['a3', 'a5'], ['a4', 'a6'], ['a4', 'a7'], ['a6', 'a8'], ['a7', 'a9']].

Графические представления связи между множествами A и B для случаев 1 и 2, которые является вариантами остовных деревьев графа из рисунка 2.7 и решениями для заданной условием в примере 4 задачи, также были смоделированы программой и представлены на рисунке 2.8 справа и слева соответственно.

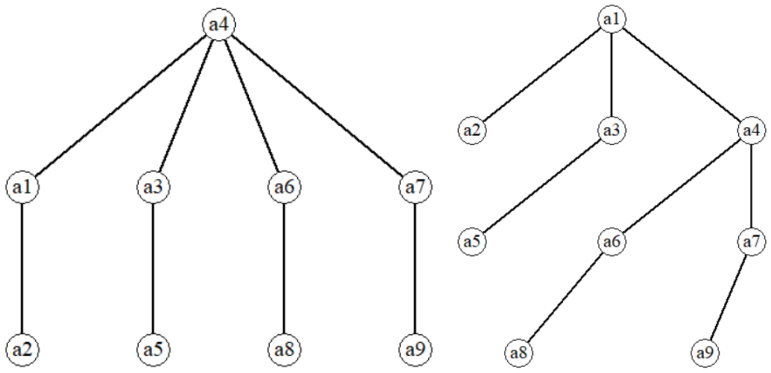

Рисунок 2.8. Остовные деревья для примера 4

Заключение

При написании курсовой работы по представленной теме была изучена специальная литература, включавшая справочные материалы по теории графов, а также учебные пособия и ресурсы сети Интернет посвящённые программированию на языке Python.

В теоретической части курсовой работы раскрыты основные определения и понятия, необходимые для понимания термина “остовное дерево графа”, а также особенности такой структуры, которые непосредственно упоминаются и используются в практической части работы. На основе изученных материалов был разработан алгоритм, с помощью которого можно выделять остовные деревья из любых связных графов, а программная реализация алгоритма позволяет моделировать и выводить их графическое представление, тем самым моделируя древовидные структуры на основе введённых данных. Практическая часть работы состоит из описания разработанного алгоритма и примеров его практического применения в программной реализации, для решения конкретных задач теории графов.

Таким образом, было проведено исследование по теме курсовой работы, а также был изучен материал, необходимый для проведения разработки по заданной теме. Был разработан и программно реализован алгоритм, который использовался для моделирования древовидных структур, на примере остовных деревьев графа.

Список литературы

1. Зыков, А. А. Основы теории графов / А. А. Зыков. – М.: Вузовская книга, 2004. — 664 с.

2. Алгоритмы. Построение и анализ / К. Томас, Л. Чарльз, Р. Рональд, Ш. Клиффорд.: 2015. — 1328 с.

3. Марк Лутц Изучаем Python. Том 1 / Лутц Марк. — 5-е изд. — М.: Вильямс, 2019. — 832 с.

Приложение 1

```
# Блок 1. ВВОД НАЧАЛЬНОГО ГРАФА

# ввод графа через указание вершин и связей
inpMass = []
n = int(input('Введите количество вершин графа: '))
print()
for i in range(n):
    bufMass = []
    x = i + 1
    name = 'a' + str(x)
    print('Введите связи для элемента: ', name)
    bufMass.append(name)
    op = str(input('Связь: '))
    while op != "":
        bufMass.append(op)
        op = str(input('Связь: '))
    inpMass.append(bufMass)

# начальные связи
MasSvyazBuf = []
for each in inpMass:
    vershina = each[0]
    for el in each:
        if el != vershina:
            svz = [str(vershina), str(el)]
            MasSvyazBuf.append(svz)
for mass in MasSvyazBuf:
    MasSvyazBuf.remove([mass[1], mass[0]])
print()
print(f'Массив начальных связей: {MasSvyazBuf}')



# Блок 2. РАЗМЕЩЕНИЕ ВЕРШИН ПО УРОВНЯМ В ДЕРЕВЕ

# (с вспомогательным элементом)
# Список всех вершин
vershini_mass = []
mass_v = []
mass_for_graph_build = []
for i in range(n):
    x = i + 1
    name = 'a' + str(x)
    vershini_mass.append(name)
    mass_v.append(name)
    mass_for_graph_build.append(name)

# Работа с вершиной для дерева(нулевой уровень)
versh = str(input('Выберите вершину для дерева: '))

levels_mass = [[versh]]
vershini_mass.remove(versh)
```

```
# Создание первого уровня вершин
num = versh[1]
num = int(num)
buffer = []
for elem in inpMass[num-1]:
    if elem == inpMass[num-1][0]:
        buffer.append(elem)
    if elem in vershini_mass:
        buffer.append(elem)
        vershini_mass.remove(elem)
levels_mass.append(buffer)

# второй уровень и выше
while vershini_mass != []:
    for level in levels_mass:
        if level != levels_mass[0]:
            for i in range(len(level)):
                if i != 0:
                    num = level[i]
                    num = num[1]
                    num = int(num)
                    buffer = []
                    for element in inpMass[num - 1]:
                        if element == inpMass[num - 1][0]:
                            buffer.append(element)
                        if element in vershini_mass:
                            buffer.append(element)
                            vershini_mass.remove(element)
                    if len(buffer) > 1:
                        levels_mass.append(buffer)

#(без вспомогательного элемента)
# массив массивов по уровням (0 и 1 уровни)
levels = [[versh]]
mass_v.remove(versh)
num = versh[1]
num = int(num)
buffer = []
for elem in inpMass[num-1]:
    if elem != inpMass[num-1][0]:
        if elem in mass_v:
            buffer.append(elem)
            mass_v.remove(elem)
levels.append(buffer)

# остальные уровни
while mass_v != []:
    for some in levels:
        buf = []
        if some != levels[0]:
            for i in range(len(some)):
                num = some[i]
                num = num[1]
                num = int(num)
                for element in inpMass[num-1]:
                    if element in mass_v:
                        buf.append(element)
                        mass_v.remove(element)
            if len(buf) > 0:
                levels.append(buf)
```

```
print(f'Массив вершин по уровням: {levels}')


# Блок 3. МАССИВ ФИНАЛЬНЫХ СВЯЗЕЙ

# новый(Итоговый) массив связей
MasSvyaz = []
for mass in levels_mass:
    if mass != levels_mass[0]:
        for i in range(len(mass)):
            if i != 0:
                MasSvyaz.append([mass[0], mass[i]])
print(f'Массив итоговых связей: {MasSvyaz}')


# Блок 4. ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ

# задание координат вершин
hei = 600
wid = 600
height_count = len(levels)
delta_y = hei / (height_count + 1)
hei_for_build = 0
coords = {}
for i in range(len(levels)):
    hei_for_build += delta_y
    wid_count = len(levels[i])
    delta_x = wid / (wid_count + 1)
    x_build = 0
    for j in range(len(levels[i])):
        x_build += delta_x
        coords[levels[i][j]] = [x_build, hei_for_build]


# графическое представление дерева
from tkinter import *
tk = Tk()
tk.title('Остовное дерево')

canvas = Canvas(tk, width=wid, height=hei, bg='white')
canvas.pack()

for k in range(len(MasSvyaz)):
    x1 = coords[MasSvyaz[k][0]][0]
    y1 = coords[MasSvyaz[k][0]][1]
    x2 = coords[MasSvyaz[k][1]][0]
    y2 = coords[MasSvyaz[k][1]][1]
    canvas.create_line(x1, y1, x2, y2, fill='black', width=2)

for k in range(len(mass_for_graph_build)):
    x = coords[mass_for_graph_build[k]][0]
    y = coords[mass_for_graph_build[k]][1]
    canvas.create_oval(x-15, y-15, x+15, y+15, outline='black', fill='white')
    canvas.create_text(x, y, text=mass_for_graph_build[k], font='Times 15',
fill='black')

tk.mainloop()
```