Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Алгоритм поиска кратчайшего пути в графовых базах данных»

Студент ___ИУ7-52Б___ _____ **К.М. Искакова**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ **Т.И. Вишневская**
(Подпись, дата) (И.О.Фамилия)

2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ-7
_____ И. В. Рудаков
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме ____«Алгоритм поиска кратчайшего пути в графовых базах данных»_____

_____

_____

Студент группы ____ИУ7-52Б_____

_____Искакова Карина Муратовна_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

_____учебная_____

Источник тематики (кафедра, предприятие, НИР) ___НИР_____

График выполнения НИР: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Техническое задание: <u>классифицировать существующие алгоритмы поиска пути в графовых базах данных, предложить критерии оценки эффективности.</u>

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на 15-25 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
Презентация на 8-10 слайдах.

Дата выдачи задания «03» сентября 2021 г.

Руководитель НИР _____ ___Т.И. Вишневская___
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____К.М. Искакова___
(Подпись, дата) (И.О.Фамилия)

<u>Примечание</u>: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня людям необходимо иметь доступ к большому объему информации. Для этого были созданы базы данных. Среди средств хранения данных наиболее часто используемыми являются системы управления базами данных (СУБД), основанные на реляционном подходе. Он заключается в хранении данных в таблицах с определенными между ними связями и ограничениями для поддержки целостности, корректности и неизбыточности данных. Однако данный тип не всегда представляется удобным в использовании, так как не обладает гибкостью во внесении изменений. В свою очередь, графовая модель данных обладает этим достоинством, что говорит о ее конкурентоспособности. В основе ее работы лежит аппарат ориентированных графов. Вершины хранят основную информацию об объектах и сгруппированы по видам с помощью специальных меток. Также вершины (сущности) связаны друг с другом дугами (связь между сущностями), в метках которых можно хранить данные. Примером графовой СУБД является Neo4j [1].

Целью данной работы является обзор существующих алгоритмов поиска пути в графовых базах данных, а также анализ их эффективности.

Задачи, которые необходимо решить для достижения поставленной цели:

1) изучить существующие алгоритмы поиска в графовых базах данных;
2) предложить критерии оценки алгоритмов;
3) классифицировать существующие алгоритмы.

1 Анализ предметной области

1.1 Актуальность задачи

Несмотря на то, что реляционные хранилища обеспечивают наилучшее сочетание простоты, устойчивости, гибкости, производительности, масштабируемости и совместимости, их показатели по каждому из этих пунктов не обязательно выше, чем у аналогичных систем, ориентированных на какую-то одну особенность. Однако универсальность реляционных СУБД перевешивала какие-либо другие недостатки [2].

Сегодня ситуация несколько иная. Графовый подход представления баз данных применяется в следующих областях:

1) статический анализ кода;
2) биоинформатика;
3) анализ RDF файлов;
4) управление бизнесом;
5) моделирование социальных графов (социальных сетей);
6) семантическая паутина;
7) искусственный интеллект;
8) маркетинговые исследования.

Одной из веских причин выбора графовой базы данных является большой прирост производительности при работе со взаимосвязанными данными, по сравнению с реляционными базами данных и другими NoSQL-хранилищами – производительность графовых баз данных остается неизменной с увеличением объема хранимых данных [2].

Таким образом, графовые базы данных и алгоритмы поиска путей в них являются актуальной областью исследования, как с точки зрения производительности, так и с точки зрения широты их применения.

1.2 Основные определения

Основная задача алгоритмов поиска пути в графовых базах данных заключается в нахождении кратчайшего пути из одной вершины графа в другую.

Графовые алгоритмы поиска кратчайшего пути по принципу нахождения путей между вершинами можно разделить на:

1) находящие кратчайшие пути от одной вершины графа до всех остальных;
2) находящие кратчайшие пути между всеми парами вершин графа [3].

Система управления базами данных (СУБД) – комплекс программ, позволяющих создать базу данных и взаимодействовать с данными (обновлять, изменять, удалять).

NoSQL-хранилища (Not only SQL) – системы управления базами данных, реализующие модели данных, имеющие существенные отличия от традиционной реляционной модели. Основной их целью является расширение возможностей баз данных в тех областях, где реляционная модель и SQL недостаточно гибки, и не вытеснять их там, где они справляются со своими задачами. Графовые СУБД относятся к данному типу. Также NoSQL включает в себя документоориентированные СУБД и другие [4].

Графовая база данных моделирует сущности в виде графа в том виде, как это определено в теории графов.

Структуры данных – это вершины и рёбра. Атрибуты – это свойства вершин и рёбер. Связь – это соединение вершин.

Обход графа можно выполнять либо по определенным типам рёбер, либо по всему графу. Обход соединений или взаимосвязей в графовых базах данных выполняется очень быстро, поскольку взаимосвязи между узлами не вычисляются во время выполнения запроса, а хранятся в базе данных.

Взвешенный граф – граф, каждому ребру которого соответствует некоторое значение, то есть вес ребра.

Ориентированный граф – граф, рёбра которого имеют направление.

Связный граф – граф, между парой вершин которого имеется как минимум один путь. Несвязный же граф может не иметь связи между какими-либо вершинами.

Для оценки сложности были введены обозначения:

1) V – количество вершин в графе;
2) E – количество рёбер в графе.

2 Классификация существующих решений

2.1 Алгоритм Дейкстры

Данный алгоритм относится к алгоритмам, находящим кратчайшие пути от одной вершины до всех остальных вершин графа. Недостаток данного алгоритма в том, что он будет некорректно работать, если в графе имеются рёбра с отрицательным весом.

Дан взвешенный ориентированный граф $G(V, E)$ без дуг отрицательного веса. Необходимо найти кратчайшие пути от некоторой вершины s графа G до остальных вершин этого графа. Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до s.

Алгоритм работает пошагово – на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Инициализация заключается в том, что метка самой вершины s полагается равной 0, метки остальных вершин – бесконечности. Это отражает то, что расстояния от s до других вершин пока неизвестны. Все вершины графа помечаются как не посещённые.

Если все вершины посещены, алгоритм завершается. В противном случае, из еще не посещённых вершин выбирается вершина u, имеющая минимальную метку.

Рассматриваются всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, называются соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещённые, рассматривается новая длина пути, равная сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом.

Если полученное значение длины меньше значения метки соседа, значение метки будет заменено полученным значением длины. Рассмотрев

каждого из соседей, вершина u помечается как посещённая и повторяется шаг алгоритма [5].

Можно рассмотреть пример алгоритма.

На рис. 1 рядом с каждой вершиной обозначена метка – длина кратчайшего пути в эту вершину из вершины 1.

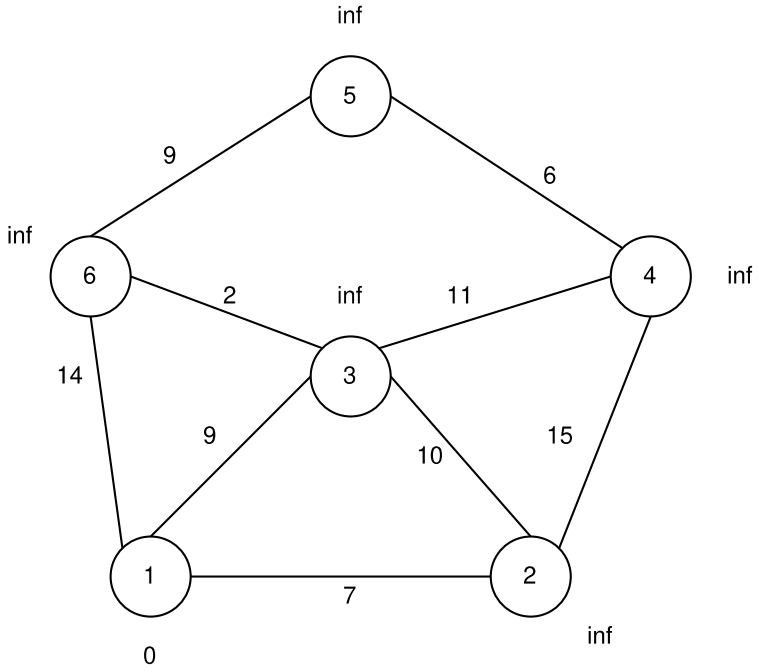

Рис. 1. Инициализация

Минимальную метку имеет вершина 1. Ее соседи – вершины 2, 3 и 6. Первый по очереди сосед вершины 1 – вершина 2, потому что длина пути до нее минимальна. Длина пути в нее через вершину 1 равна сумме значения метки вершины 1 и длины ребра, идущего из 1-й в 2-ю, то есть $0 + 7 = 7$. Это меньше текущей метки вершины 2, бесконечности, поэтому новая метка 2-й вершины равна 7. Аналогичную операцию нужно проделать с двумя другими соседями 1-й вершины – 3-й и 6-й.

Все соседи вершины 1 проверены, что показано на рис. 2.

Текущее минимальное расстояние до вершины 1 считается окончательным и пересмотру не подлежит. Необходимо вычеркнуть ее из графа, чтобы отметить, что эта вершина посещена.

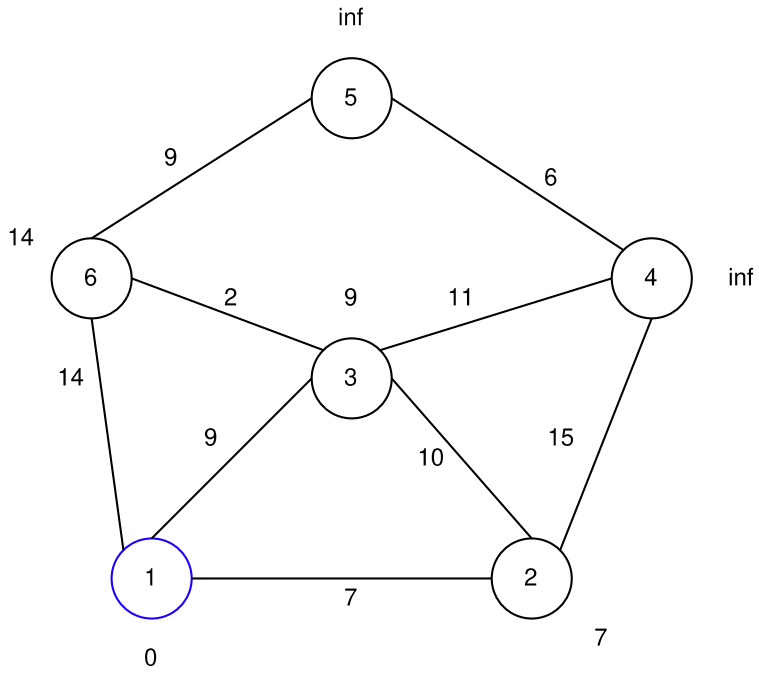

Рис. 2. Все соседи вершины 1 проверены

Далее снова необходимо найти «ближайшую» из не посещённых вершин. Это вершина 2 с меткой 7.

Затем нужно попытаться уменьшить метки соседей выбранной вершины, пытаясь пройти в них через 2-ю вершину. Соседями вершины 2 являются вершины 1, 3 и 4. Первый (по порядку) сосед вершины 2 – вершина 1. Но она уже посещена, поэтому с 1-й вершиной ничего делать не нужно.

Следующий сосед – вершина 3, так как имеет минимальную метку. Если идти в нее через 2, то длина такого пути будет равна $17\,(7 + 10 = 17)$. Но текущая метка третьей вершины равна 9, а это меньше 17, поэтому метка не меняется.

Еще один сосед вершины 2 – вершина 4. Если идти в нее через 2-ю, то длина такого пути будет равна сумме кратчайшего расстояния до 2-й вершины и расстояния между вершинами 2 и 4, то есть $22\,(7 + 15 = 22)$. Таким образом, нужно установить метку вершины 4 равной 22.

Все соседи вершины 2 проверены, значит, она считается просмотренной, что показано на рис. 3.

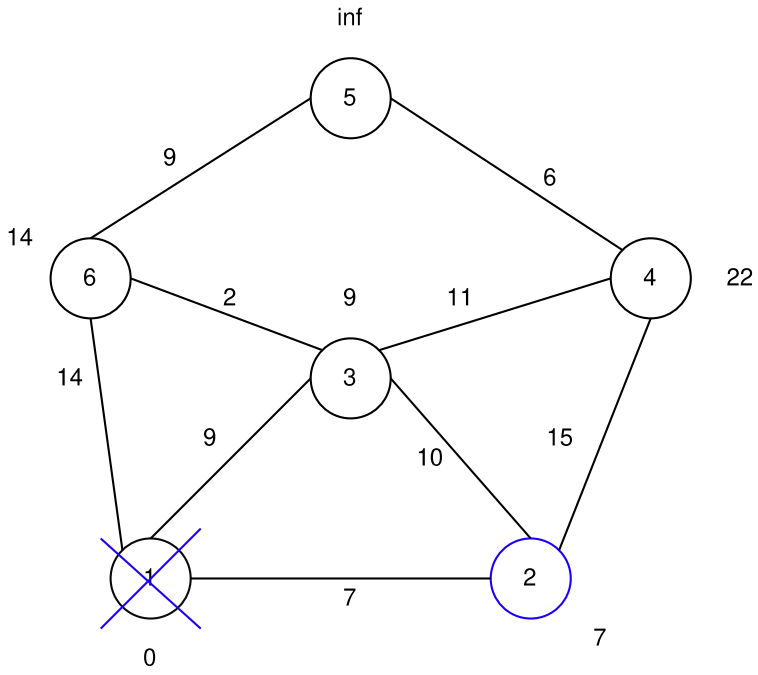

Рис. 3. Все соседи вершины 2 проверены

Далее необходимо повторить шаг алгоритма, выбрав вершину 3. После ее «обработки» получатся результаты, представленные на рис. 4.

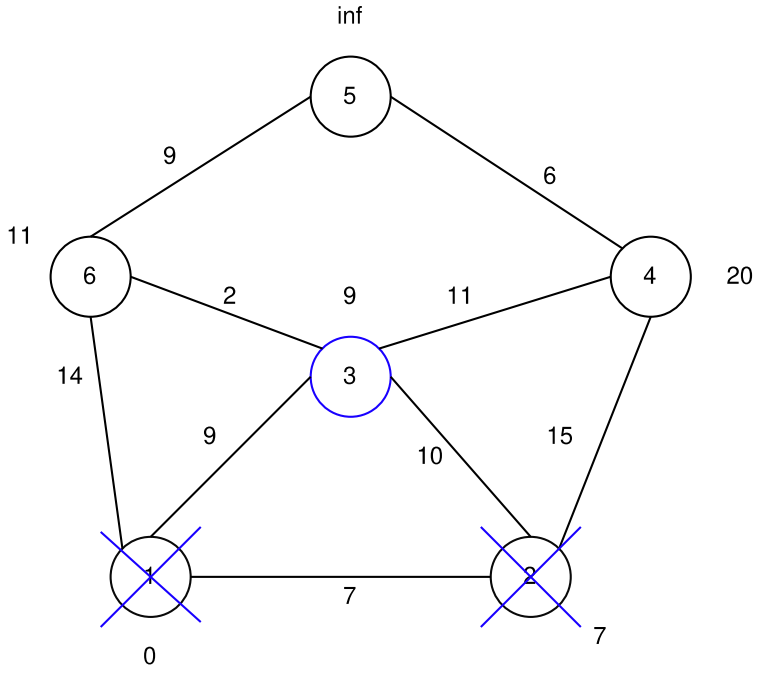

Рис. 4. Все соседи вершины 3 проверены

Далее идет повторение шага алгоритма для оставшихся вершин. Это будут вершины 6, 4 и 5, соответственно порядку.

Алгоритм заканчивает работу, когда все вершины посещены.

Результат работы алгоритма можно увидеть на рис. 5: кратчайший путь от вершины 1 до 2-й составляет 7, до 3-й – 9, до 4-й – 20, до 5-й – 20, до 6-й – 11.

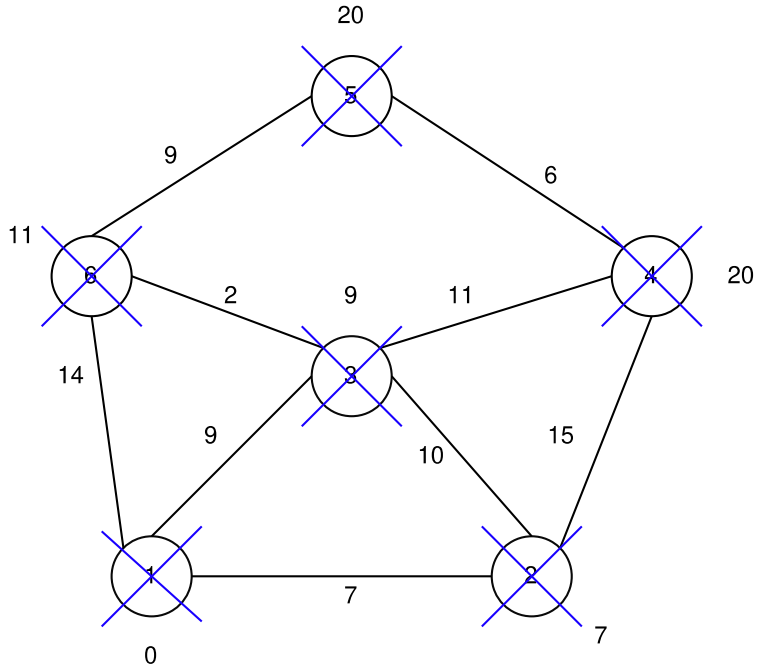

Рис. 5. Алгоритм завершил работу

Если в какой-то момент все не посещённые вершины помечены бесконечностью, то это значит, что до этих вершин нельзя добраться, то есть граф несвязный. Тогда алгоритм может быть завершён досрочно.

2.2 Алгоритм Беллмана-Форда

Алгоритм Беллмана-Форда относится к алгоритмам, которые находят кратчайшие пути от одной исходной вершины до всех остальных. В отличие от алгоритма Дейкстры, в алгоритме Беллмана-Форда могут быть рёбра с отрицательным весом.

Входными данными данного алгоритма являются сам граф и начальная вершина s, от которой и будет производиться поиск кратчайших путей до остальных вершин алгоритма.

В данном алгоритме при наличии цикла отрицательного веса самые короткие расстояния не вычисляются, выводится сообщение о наличии такого цикла.

Далее можно рассмотреть непосредственно шаги алгоритма:

1) инициализация расстояний от исходной вершины до всех остальных вершин, как бесконечные, а расстояние до исходной вершины s принимается равным 0. Создается массив $dist[]$ размера V со всеми значениями равными бесконечности, за исключением элемента $dist[s]$, где s – исходная вершина;
2) вычисление самых коротких расстояний. Следующие шаги нужно выполнять $V - 1$ раз, повторение шага для рёбер между каждой парой вершин $u - v$:
    1) если $dist[v] > dist[u]$ + вес ребра uv, то обновить $dist[v]$;
    2) $dist[v] = dist[u]$ + вес ребра uv;
3) на этом шаге сообщается, присутствует ли в графе цикл отрицательного веса. Для каждого ребра uv необходимо выполнить следующее:
    1) если $dist[v] > dist[u]$ + вес ребра uv, то в графе присутствует цикл отрицательного веса.

Смысл шага 3 заключается в том, что шаг 2 гарантирует кратчайшее расстояние, если граф не содержит цикла отрицательного веса. Если мы снова

переберём все рёбра и получим более короткий путь для любой из вершин, это будет сигналом присутствия цикла отрицательного веса.

Как и в других задачах динамического программирования, алгоритм вычисляет кратчайшие пути снизу вверх. Сначала он вычисляет самые короткие расстояния, то есть пути длиной не более, чем в одно ребро. Затем он вычисляет кратчайшие пути длиной не более двух рёбер и так далее. После i-й итерации внешнего цикла вычисляются кратчайшие пути длиной не более i рёбер. В любом простом пути может быть максимум $V - 1$ рёбер, поэтому внешний цикл выполняется именно $V - 1$ раз.

Идея заключается в том, что если вычислить кратчайший путь с не более чем i рёбрами, то итерация по всем рёбрам гарантирует получение кратчайшего пути с не более чем $i + 1$ рёбрами [6].

2.3 Алгоритм Флойда-Уоршелла

Алгоритм Флойда-Уоршелла относится к алгоритмам, которые находят кратчайшие пути между всеми парами вершин графа. В отличие от алгоритма Беллмана-Форда, в алгоритме Флойда-Уоршелла также могут быть рёбра с отрицательным весом, но не могут быть отрицательные циклы.

Ключевая идея алгоритма – разбиение процесса поиска кратчайших путей на фазы.

Перед k-ой фазой $k = \overline{1,n}$ считается, что в матрице расстояний $dist[][]$ сохранены длины таких кратчайших путей, которые содержат в качестве внутренних вершин только вершины из множества $\{1, 2, \dots, k-1\}$.

Иными словами, перед k-ой фазой величина $dist[i][j]$ равна длине кратчайшего пути из вершины i в вершину j, если этому пути разрешается заходить только в вершины с номерами, меньшими k (начало и конец пути не считаются).

Для того, чтобы убедиться, что это свойство выполнилось для первой фазы, достаточно в матрицу расстояний $dist[][]$ записать матрицу смежности графа: $dist[i][j] = g[i][j]$ – вес ребра из вершины i в вершину j. При этом, если между какими-то вершинами ребра нет, то записать следует ∞. Из вершины в саму себя всегда следует записывать 0, это критично для алгоритма.

Пусть теперь, находясь на k-ой фазе, нужно пересчитать матрицу $dist[][]$ таким образом, чтобы она соответствовала требованиям уже для $k+1$-ой фазы. Нужно зафиксировать какие-то вершины i и j. Тогда возникает два разных случая:

1) кратчайший путь из вершины i в вершину j, которому разрешено дополнительно проходить через вершины $\{1, 2, \dots, k\}$, совпадает с кратчайшим путём, которому разрешено проходить через вершины множества $\{1, 2, \dots, k-1\}$. В этом случае величина $dist[i][j]$ не изменится при переходе с k-ой на $(k+1)$-ую фазу;

2) новый кратчайший путь стал лучше старого пути. Это означает, что новый кратчайший путь проходит через вершину k. Заметив, что если разбить новый путь вершиной k на две половины (одна идущая $i \Rightarrow k$, а другая — $k \Rightarrow j$), то каждая из этих половин уже не заходит в вершину k. Но тогда получается, что длина каждой из этих половин была посчитана ещё на $k-1$-ой фазе или ещё раньше, что означает, что достаточно взять сумму $dist[i][k] + dist[k][j]$, она и даст длину нового кратчайшего пути.

   Объединяя эти два случая, получается, что на k-ой фазе требуется пересчитать длины кратчайших путей между всеми парами вершин i и j следующим образом:

$$new\_dist[i][j] = min\,(dist[i][j], dist[i][k] + dist[k][j]).$$

Таким образом, работа, которую требуется произвести на k-ой фазе – это перебрать все пары вершин и пересчитать длину кратчайшего пути между ними. В результате после выполнения n-ой фазы в матрице расстояний $dist[i][j]$ будет записана длина кратчайшего пути между i и j, либо ∞, если пути между этими вершинами не существует.

2.4 Классификация алгоритмов

На рис. 6 приведена классификация рассмотренных в данной работе алгоритмов поиска кратчайшего пути в графовых базах данных.

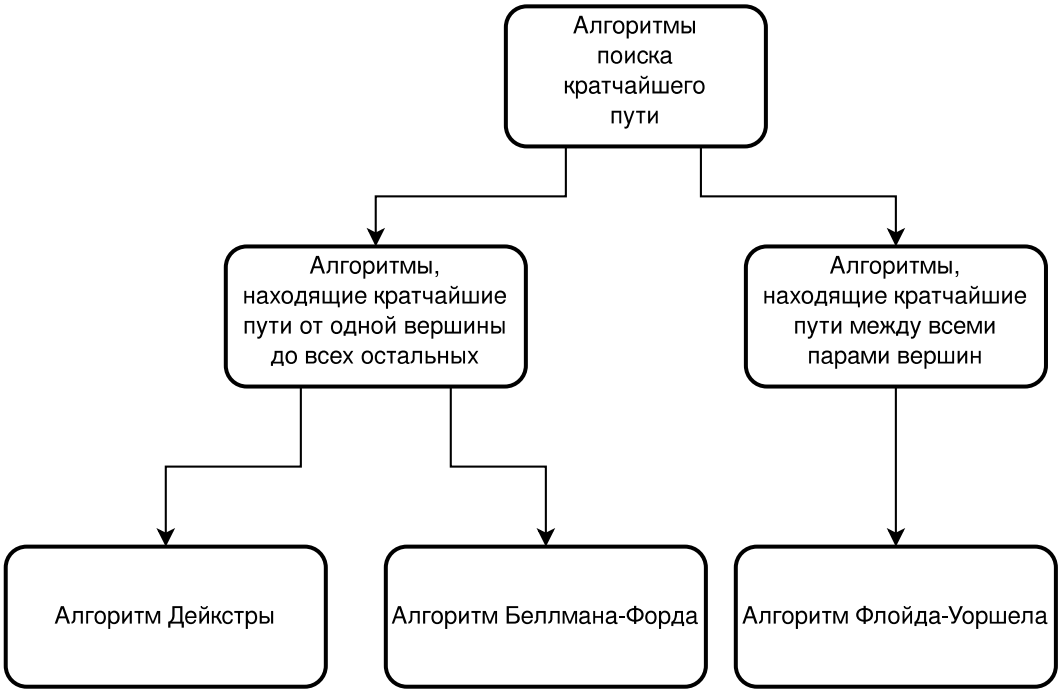

Рис. 6. Классификация алгоритмов поиска кратчайшего пути

2.5 Оценка алгоритмов

На основании описанных выше алгоритмов поиска кратчайшего пути в графовых базах данных для сравнения алгоритмов были выбраны такие критерии как сложность, лучший и худший варианты (так как она в первую очередь важна для подобных алгоритмов), а также возможность работы с рёбрами с отрицательным весом (так как это позволяет увеличить применение и пользу алгоритма).

Результаты представлены в таблице 1.

Таблица 1. Сравнение алгоритмов поиска кратчайшего пути

Название	Сложность, лучший вариант	Сложность, худший вариант	Возможность работы с рёбрами с отрицательным весом
Алгоритм Дейкстры	$O(V \log V)$	$O(V^2)$	Нет
Алгоритм Беллмана-Форда	$O(E)$	$O(EV)$	Да
Алгоритм Флойда-Уоршелла	$O(V^3)$	$O(V^3)$	Да

Для наиболее рационального поиска пути из рассмотренных рекомендуется выбрать алгоритм Беллмана-Форда, так как он имеет наименьшую сложность, обладает возможностью работы с рёбрами с отрицательным весом, а также не ищет кратчайшие пути между остальными парами, как это производит алгоритм Флойда-Уоршелла.

2.6 Вывод

В данной части были рассмотрены алгоритмы поиска кратчайшего пути в графовых базах данных, приведена их классификация, а также сравнительная таблица, был выбран алгоритм, который лучше всего решает задачу поиска пути в графовых базах данных, из рассмотренных.

ЗАКЛЮЧЕНИЕ

В ходе данной работы были проанализированы некоторые алгоритмы поиска кратчайшего пути в графовых базах данных; для поиска кратчайшего пути было предложено использовать алгоритм Беллмана-Форда.

Задачи, решенные для достижения поставленной цели:

1) изучены алгоритмы поиска кратчайшего пути в графовых базах данных;
2) предложены критерии оценки алгоритмов;
3) выбран алгоритм, предположительно наиболее эффективно решающий задачу.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Сравнительный анализ использования реляционных и графовых баз данных в разработке цифровых образовательных систем / Абрамский М.М., Тимерханов Т.И. – Вестник НГУ. Серия: Информационные технологии, 2018, том 16, №4 – 5-12 с.

[2] Использование графовых баз данных в целях оптимизации анализа биллинговой информации / Бартенев М.В., Вишняков И.Э., – Инженерный журнал: наука и инновации, 2013, вып. 11 – 1-8 с.

[3] Обзор алгоритмов поиска кратчайшего пути в графе / Изотова Т.Ю. – Новые информационные технологии в автоматизированных системах, 2016, № 19 – 341-344 с.

[4] Об использовании NoSQL-хранилищ данных / Шарипова Н.Н. – Восточно-европейский научный журнал, 2016, №9 – 73-75 с.

[5] Необходимое и достаточное условие применимости алгоритма Дейкстры / Лебедев С.С., Новиков Ф.А. – Компьютерные инструменты в образовании, 2017, № 4 – 5-13 с.

[6] Автоматизация выбора кратчайших маршрутов судов на основе модифицированного алгоритма Беллмана-Форда / Чертков А.А. – Вестник государственного университета морского и речного флота им. адмирала С. О. Макарова, 2017, том 9, № 5 – 1113-1122 с.