Алгоритм поиска кратчайшего пути в графовых базах данных

Студент: Искакова Карина Муратовна

Группа: ИУ7-52Б

Руководитель: Вишневская Татьяна Ивановна

Цели и задачи работы

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.

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 ребра 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 Дейкстры

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 непосещенные.

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

Алгоритм Дейкстры

Рассматриваются всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут ребра из u, называются соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещенные, 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 всех соседей, вершина u помечается как посещенная и повторяется шаг алгоритма.

Алгоритм Дейкстры

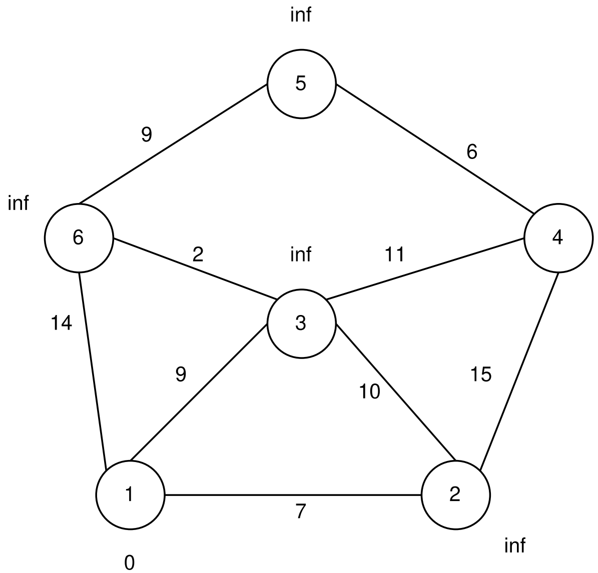

Рис. 1. Инициализация

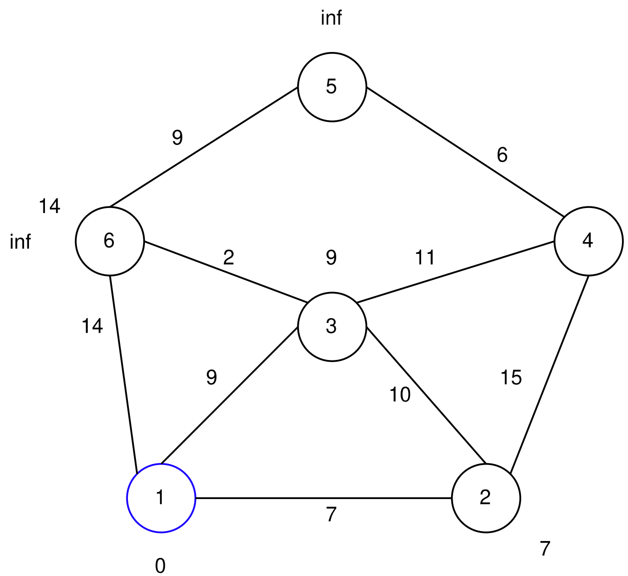

Рис. 2. Все соседи вершины 1 проверены

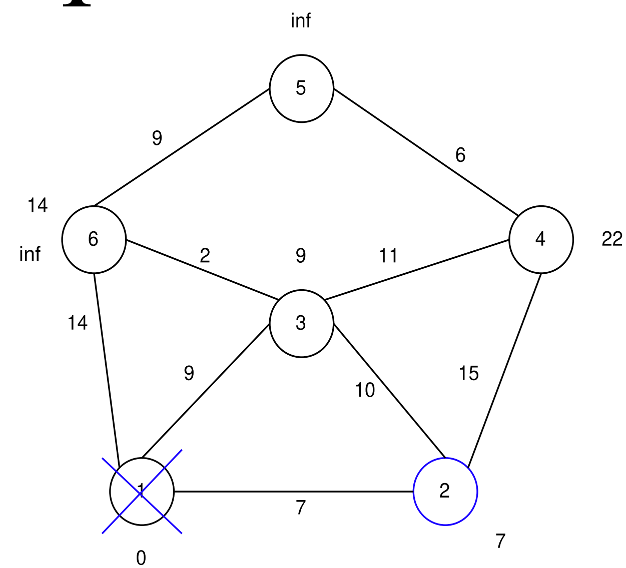

Рис. 3. Все соседи вершины 2 проверены

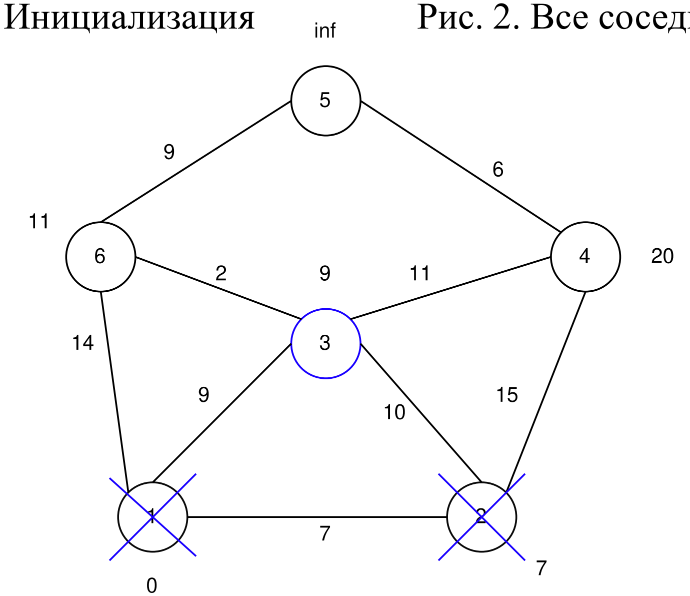

Рис. 4. Все соседи вершины 3 проверены

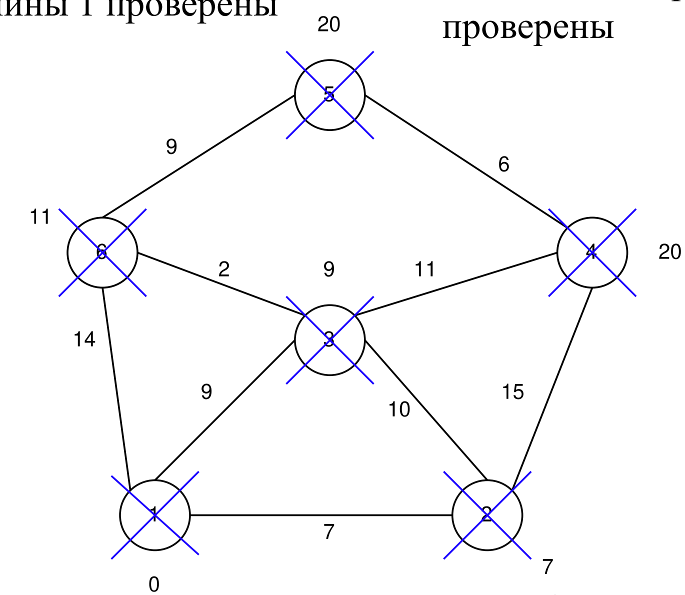

Рис. 5. Алгоритм завершил работу

Алгоритм Беллмана-Форда

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) инициализия 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.

Алгоритм Беллмана-Форда

2) Вычисление самых коротких расстояний. Следующие шаги нужно выполнять V-1 раз, повторение шага для ребер между каждой парой вершин u-v:

   если dist[v]>dist[u] + вес ребра uv, то обновить dist[v], 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:

   если dist[v]>dist[u] + вес ребра uv, то в графе присутствует цикл отрицательного веса.

Идея шага 3 заключается в том, что шаг 2 гарантирует кратчайшее расстояние, если граф не содержит цикла отрицательного веса. Если мы снова переберем все ребра и получим более короткий путь для любой из вершин, это будет сигналом присутствия цикла отрицательного веса.

Алгоритм Флойда-Уоршелла

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

Перед k-ой фазой k=(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,…,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.

Алгоритм Флойда-Уоршелла

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,…,k}, совпадает с кратчайшим путём, которому разрешено проходить через вершины множества {1,2,…,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.

Алгоритм Флойда-Уоршелла

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, вся 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.

Классификация и оценка методов

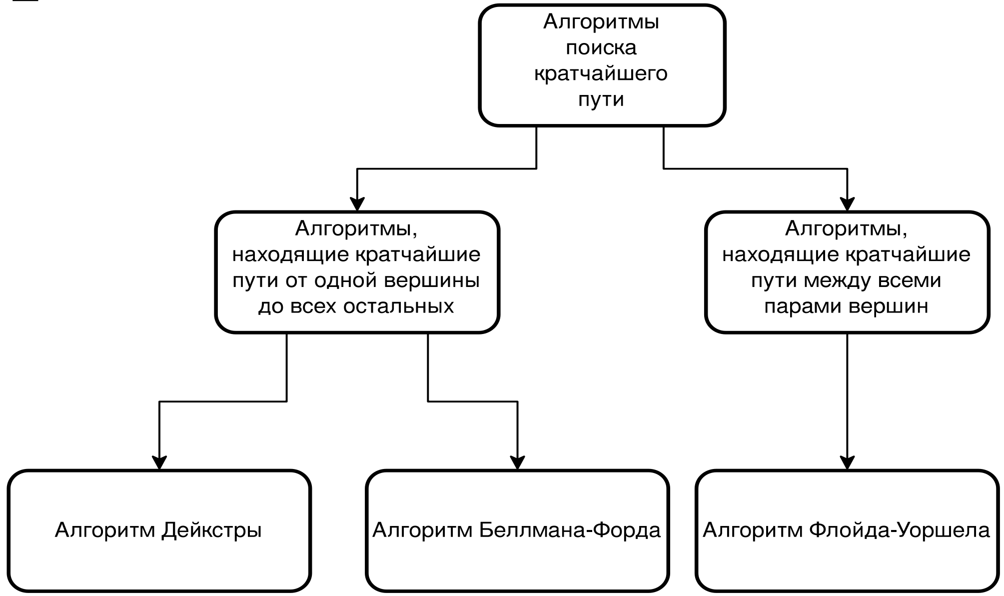

Рис. 6. Классификация алгоритмов поиска кратчайшего пути

Таблица 1. Сравнение алгоритмов поиска кратчайшего пути

Название	Сложность, лучший вариант	Сложность, худший вариант	Возможность работы с ребрами 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

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.