электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.021

Обзор алгоритмов на графах для решения прикладных задач

***Котович М.А.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Гапанюк Ю.Е., к.т.н, доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
gapyu@bmstu.ru

Введение.

Неформально граф можно представить как множество точек и линий, соединяющих эти точки. Линии могут быть со стрелками, что характерно для ориентированных графов (см. рис.1) , и без стрелок, что характерно для неориентированных (см. рис.2).

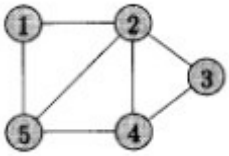

Рис. 1. Неориентированный граф

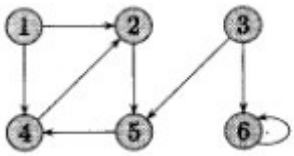

Рис. 2. Ориентированный граф

В 1736 году Эйлер написал статью, в которой рассмотрел задачу о Кёнигсбергских мостах. Эту статью призвано считать первой работой в теории графов как математической дисциплины. В ней Эйлер показал, что нельзя обойти семь городских мостов и вернуться в исходную точку, пройдя по каждому мосту ровно один раз.

Спустя почти 100 лет в связи с развитием исследований по электрическим сетям, кристаллографии, органической химии и другим наукам к теории графов возобновился интерес.

В нашей жизни мы сталкиваемся с графами постоянно. Например, графом является схема линий метрополитена, родословная, возводимая от потомков к предкам, называемая иначе генеалогическим древом.

Графы отнюдь не всегда используют только как иллюстрации. Например, рассматривая граф, изображающий сеть дорог между населёнными пунктами, можно

определить маршрут проезда из пункта *А* до пункта *Б*. Если таких маршрутов окажется несколько, хотелось бы выбрать оптимальный: самый короткий или самый безопасный.

Без методов теории графов невозможно обойтись при анализе и синтезе различных дискретных преобразователей: функциональных блоков компьютеров, комплексов программ и т.д.

Задачи и алгоритмы на графах для их решения.

Одной из задач, в которых удобно применять алгоритмы на графах, является **задача Штейнера**: на плоскости заданы *n* точек; нужно соединить их отрезками прямых таким образом, чтобы суммарная длина отрезков была наименьшей.

Эффективных алгоритмов, дающих точное решение этой задачи, не существует. Алгоритмы Крускала и Прима, однако, находят некоторые приближения точного решения. Алгоритм Крускала вычисляет для заданного взвешенного неориентированного графа дерево с наименьшей суммой весов рёбер: множество рёбер представляет собой лес, состоящий из нескольких связных компонент (деревьев); добавляется ребро минимального веса среди всех рёбер, концы которых лежат в разных компонентах. Существенное отличие только что сформулированной задачи от задачи Штейнера состоит в том, что в новой задаче множество вершин при поиске древа наименьшего веса не меняется.

Алгоритм Прима не предполагает использования связных компонент. В нём к строящемуся дереву на каждом шаге добавляется ребро наименьшего веса среди рёбер, соединяющих вершины этого дерева с вершинами не из дерева.

С помощью поиска в глубину и в ширину в графах решаются **задачи глобального анализа** как неориентированных, так и ориентированных графов. К этим задачам относятся задачи поиска циклов или контуров, вычисление длин путей между парами вершин, перечисление путей с теми или иными свойствами.

Алгоритм поиска в ширину перечисляет все достижимые из корня дерева *s* (если идти по рёбрам) вершины в порядке возрастания расстояния от *s*. Расстоянием считается длина (число рёбер) кратчайшего пути. В процессе поиска просматриваются все соседние вершины, затем соседи соседей и т.д. В процессе поиска из графа выделяется часть, для каждой вершины которой путь из корня в дереве поиска будет одним из кратчайших путей в графе. Алгоритм применим и к ориентированным, и к неориентированным графам.

Стратегия поиска в глубину такова: идти «вглубь», пока это возможно (есть непройденные исходящие рёбра), и возвращаться и искать другой путь, когда таких рёбер

нет. Так делается, пока не обнаружены все вершины, достижимые из исходной (если после этого остаются необнаруженные вершины, можно выбрать одну из них и повторять процесс и делать так до тех пор, пока мы не обнаружим все вершины графа).

Сам по себе поиск в глубину начал использоваться незадолго до 1960 года, в первую очередь в программах, связанных с так называемым «искусственным интеллектом».

Во многих прикладных задачах возникает проблема такого упорядочения вершин сети, при котором вершины, принадлежащие одному уровню, располагаются друг под другом, а души ориентированного графа ведут в его изображении на плоскости от вершин с меньшим уровнем к вершинам с большим уровням слева направо. Подобного рода проблема называется **проблемой топологической сортировки**, поскольку необходимо расположить вершины графа на плоскости так, чтобы отчётливо было видно распределение вершин по уровням (см. рис.3). Само расположение при этом может быть разным, лишь бы оно имело «слоистую» структуру, в которой каждый слой составляют вершины одного уровня.

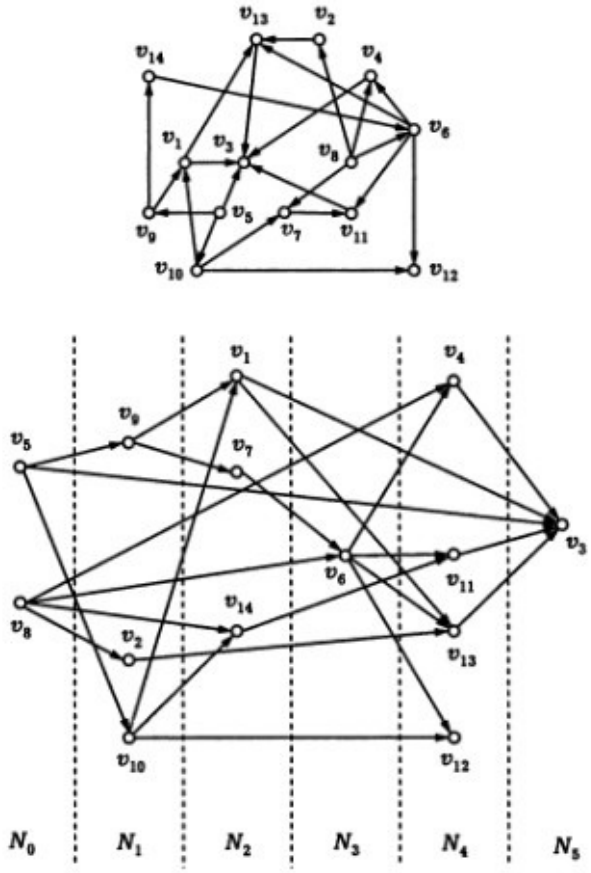

Рис. 3. Топологическая сортировка: сверху показано обычное представление графа, снизу – представление графа после топологической сортировки. N_0, N_1, … , N_5 – уровни

Топологическая сортировка применяется в самых разных ситуациях. Например, при распараллеливании алгоритмов, когда по некоторому описанию алгоритма нужно составить граф зависимостей его операций и, отсортировав его топологически, определить, какие из операций являются независимыми и могут выполняться параллельно (одновременно).

Топологическую сортировку можно организовать с помощью поиска в глубину. Это популярный, наглядный и простой в реализации способ. Алгоритм Демукрона также позволяет осуществить топологическую сортировку, используя матрицу смежности вершин, в которой, если между вершинами есть ребро, то на соответствующем (пересечении строки и столбца) месте стоит единица, а в противном случае – ноль. Матрица смежности для графа с рис.1 показана на рис.4. Данный алгоритм использует дополнительный массив, в котором хранятся суммы элементов по всем столбцам. В нём на каждой итерации определяются нулевые элементы, которые и являются вершинами вычисляемого уровня. Затем строки матрицы, соответствующие этим вершинам, вычитаются из массива.

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Рис. 4. Матрица смежности для ориентированного графа

В жизни часто можно повстречать **задачи о кратчайших путях из одной вершины**. Допустим, дана сеть автомобильных дорог, соединяющих города Московской области. Некоторые дороги односторонние. Нужно найти кратчайшие пути от города Москва до каждого города области, если двигаться можно только по дорогам. Ещё пример: имеется некоторое количество авиарейсов между городами мира, для каждого известна стоимость. Стоимость перелёта из *A* в *B* может быть не равна стоимости перелёта из *B* в *A*. Требуется найти маршрут минимальной стоимости (возможно, с пересадками) от Копенгагена до Барнаула. Для таких задач хорошо применять алгоритм Дейкстры, в котором каждой вершине графа сопоставляется минимально известное расстояние от этой вершины до исходной. Алгоритм работает пошагово: на каждом шаге он «посещает» одну

вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Алгоритм Беллмана-Форда решает задачу о кратчайших путях из одной вершины для случая, когда весам рёбер разрешено быть отрицательными. Этот алгоритм возвращает значение *True*, если в графе нет циклов отрицательного веса, достижимого из исходной вершины, и *False*, если таковой цикл имеется. В первом случае алгоритм находит кратчайшие пути и из веса; во втором – кратчайших путей (по крайней мере, для некоторых вершин) не существует.

Для **построения кратчайших расстояний для всех пар вершин** удобно использовать алгоритм Флойда-Уоршолла, суть которого в том, чтобы найти минимальный среди двух возможных путей: проходящего и не проходящего через промежуточную вершину.

В разреженных графах для решения данной задачи удобно использовать алгоритм Джонсона. В этом случае он эффективнее алгоритма Флойда-Уоршолла. Алгоритм Джонсона основан на идее изменения весов. Если веса всех рёбер неотрицательны, то можно найти кратчайшие пути между всеми парами вершин, применив алгоритм Дейкстры в каждой вершине. Если же в графе имеются рёбра с отрицательным весом, то можно попытаться свести задачу к случаю неотрицательных весов, путём замены весовой функции. В этом алгоритме используется также алгоритм Беллмана-Форда. Данный алгоритм либо возвращает матрицу весов кратчайших путей, либо сообщает, что в графе имеется цикл отрицательного веса.

Если для графа требуется определить, существует ли в графе путь из вершины *i* в вершину *j*, т.е. решить **задачу построения транзитивного замыкания ориентированного графа**, то для этого удобно использовать алгоритм Флойда-Уоршолла, в котором арифметические операции заменены логическими, так как компьютер их выполняет быстрее.

Часто требуется найти, по каким трубам в сети труб нужно пропустить поток вещества, чтобы этот поток был максимальным. Вместо сети труб можно рассматривать также провода, конвейеры, линии связи и т.д.

Поиск **максимального потока** методом Форда-Фалкерсона проводится по шагам: вначале поток нулевой (и величина его равна нулю). На каждом шаге значение потока увеличивается. Для этого находится «дополняющий путь», по которому можно пропустить ещё немного вещества. «Дополняющий путь» используется для увеличения потока. Этот шаг повторяется, пока есть дополняющие пути.

Реализаций метода Форда-Фалкерсона в алгоритмах существует много. Одним из них является алгоритм Эдмондса-Карпа, который использует поиск в ширину.

Самые быстрые из известных алгоритмов для задачи о максимальном потоке используют алгоритм проталкивания предпотока. Он применим и к другим задачам, например, к задаче о потоке наименьшей стоимости. В отличие от алгоритма Эдмондса-Карпа, мы не просматриваем всю остаточную сеть на каждом шаге, а действуем локально в окрестности одной вершины. В алгоритме Форда-Фалкерсона требовалось выполнение закона сохранения потока, так как жидкость по дороге никуда не сливалась. В алгоритмах проталкивания предпотока избыток жидкости в каждой вершине (месте соединения труб) сливается. Здесь также важен параметр, обозначающий высоту вершины, которая определяет, куда мы стараемся направить избыток жидкости. Высота истока всегда равна количеству вершин, а стока – нулю. Изначально все остальные вершины находятся на высоте 0, и со временем поднимаются. Для начала мы отправляем из истока вниз столько жидкости, сколько нам позволяют пропускные способности выходящих из истока труб. Возникающий (в соседних с истоком вершинах) избыток жидкости сперва просто выливается, но затем он будет направлен дальше. Может оказаться, что в какой-либо вершине есть избыток жидкости, но все трубы, по которым ещё модно отправить жидкость из неё куда-то, ведут в вершины той же или большей высоты. В этом случае мы можем выполнить подъём этой вершины. После этого она становится на единицу выше самого низкого соседа, и есть труба, ведущая вниз. В конце концов, мы добьёмся того, что в сток приходит максимально возможное количество жидкости (для данных пропускных способностей труб).

Алгоритм «поднять-в-начало» улучшает алгоритм проталкивания предпотока и уменьшает время его работы. Данный алгоритм хранит все вершины в виде списка. Алгоритм просматривает этот список, начиная с головы, и находит в нём переполненную вершину. Затем алгоритм обслуживает эту вершину, применяя к ней операции подъёма и проталкивания до тех пор, пока избыток не станет равным нулю. Если для этого вершину пришлось поднять, её перемещают в начало списка, и просмотр списка начинается вновь.

Связи между алгоритмами.

Очень часто одни алгоритмы на графах используются в других. Это имеет место быть, так как более простые алгоритмы целесообразно использовать для решения сложных задач. Это может значительно ускорить время работы алгоритма и повысить его эффективность.

Например, алгоритм Эдмондса-Карпа использует поиск в ширину. А алгоритм Джонсона для разреженных графов использует сразу два алгоритма: Дейкстры и Беллмана-Форда. Связь между вышеперечисленными алгоритмами изображена на рис.5.

Рис. 5. Связь между алгоритмами

Список литературы

1. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ / Пер. с англ. под ред. А. Шеня. М.: МЦНМО, 2002. 960 с.: 263 ил.
2. Белоусов А.И., Ткачёв С.Б. Дискретная математика: Учеб. для вузов / Под ред. В.С. Щарубина, А.П. Крищенко. М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. 744 с. (Сер. Математика в техническом университете; Вып. XIX).
3. Дудов М.Х. Эффективные алгоритмы на кодированных графах и их применение: дис. … канд. физ.-мат. наук. Черкесск, 1999. 134 с.
4. Алгоритмы на графах – Часть 0: базовые понятия // habrahabr.ru: Хабрахабр. Режим доступа: http://habrahabr.ru/post/65367 (дата обращения: 01.02.2014).
5. Алгоритм Дейкстры // wikipedia.org: Википедия. Режим доступ: http://ru.wikipedia.org/wiki/Алгоритм_Дейкстры (дата обращения: 01.03.2014).