Информатика3.

Алгоритм Флойда-Уоршалла.

Алгоритм очень прост - для его описания даже не нужна картинка. Будем строить последовательность матриц A1ij…Anij методом динамического программирования. Асимптотика - за куб. Что это за матрицы? Эти матрицы - кратчайшее расстояние от i до j вершины. При этом как промежуточными пользуемся от первой до k-ой. Те по сути это просто весовая матрица, не более того. В чем фишка кода? На каждом шаге мы проверяем новое количество вершин, те условно на первом шаге у нас только прямой путь, на втором шаге мы уже смотрим, а можно ли дойти через 2 вершины короче, потом через 3 и так далее. Те мы постепенно исследуем все пути из вершины в вершину через постепенно растущее число вершин.

Топологическая сортировка

Это упорядочивание вершин графа согласно нумерации его вершин.

Если граф не содержит циклов, то его вершины можно пронумеровать так, что любое ребро идет от вершины с меньшим номером к вершине с большим номером.

Применяется в различных прикладных задачах.

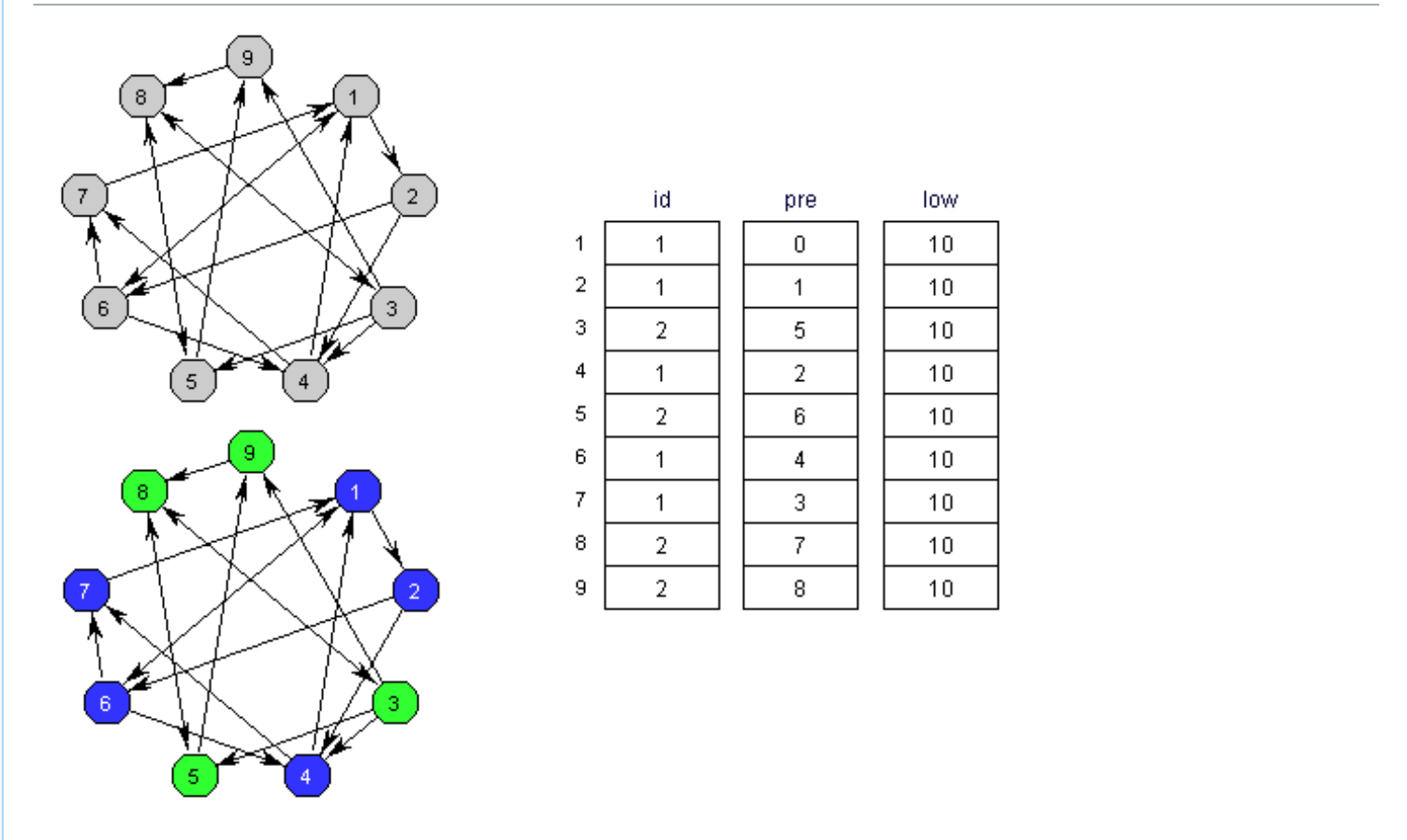
Алгоритм Кана:

Найдем вершину, которая ни от кого не зависит, те такую, что в неё не приходит ни одно ребро. / A(0)=пустое/Изначально каждой вершине соответствует множество вершин, от которых она «зависит»/ Вычеркиваем эту вершину из всех зависимостей.

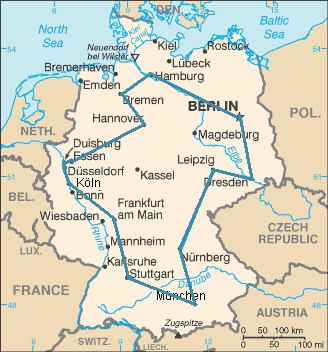
Код громоздкий , много тратит времени и памяти. Поэтому используют другой код.

Алгоритм Тарьяна.

Подход: DFS с покраской вершин белая/серая/черная. С любой вершины, которая не used, запускаем DFS. Когда в вершину зашли-она серая. Вышли уже - черная. Не зашли - белая. #Замечание Если осуществляется попытка входа в вершину, в которой был DFS, то на таком графе не возможна топологическая сортировка, тк тогда есть циклы в графе.

При выходе из вершины (те при покраске вершин в черный цвет) мы её просто добавляем в список вершин. Но добавлять её надо в начало списка. Таким образом, мы сначала заходим в вершины, а потом, при выходе, закидываем их в черные вершины. Реализацию мы советуем читателю написать самому.

Рассмотрим случаи, когда построить эффективный код на графе не построить. Это задачи коммивояжера и о китайском почтальоне.

1. Задача коммивояжера.   
   В графе есть некоторые вершины. Коммивояжер захотел посетить все эти вершины и вернутся домой. Гамильтонов цикл точно есть в этой системе. /когда мы нашли кратчайшие расстояния между всеми вершинами/,т.к. иначе задачу было бы не выполнить. То есть, она сводится к тому, чтобы найти минимальный по весу Гамильтонов цикл. Но эта задача реально плохо реализуется при большом числе вершин, так как основа решения - перебор. К примеру, это оптимальный маршрут коммивояжёра через 15 крупнейших городов Германии. Указанный маршрут является самым коротким из всех возможных 43 589 145 600 вариантов:

То есть оптимизировать эту задачу просто нереально.

2) Задача о китайском почтальоне.

Пройти по каждому ребру минимум 1 раз. То есть, нам необходимо найти цикл минимального суммарного веса, такой, что он проходит по ребру хотя бы 1 раз. Важное замечание-если граф Эйлеров, то собственно Эйлеров цикл и есть решение этой задачи. А вот если его нет, то начинаются сложности - решение идет только полным перебором. То есть, код тоже неэффективен и его скорость тоже очень быстро растет с ростом числа рёбер.