Алгоритм Флойда поиска кратчайших путей

ВАРИАНТЫ ПОСТАНОВКИ ЗАДАЧИ

Задача о кратчайшем пути в заданный пункт назначения. Требуется найти кратчайший путь в заданную вершину назначения t, который начинается в каждой из вершин графа (кроме t).

Задача о кратчайшем пути между заданной парой вершин. Требуется найти кратчайший путь из заданной вершины u в заданную вершину v.

Задача о кратчайшем пути между всеми парами вершин. Требуется найти кратчайший путь из каждой вершины u в каждую вершину v.

Алгоритм Флойда-Уоршелла

Строится последовательность матриц $A^{(0)} \to A^{(1)} \to \ldots \to A^{(k)} \to \ldots \to A^{(n)}$.
Элемент $a_{ij}^{(k)}$ матрицы $A^{(k)}$ равен длине кратчайшего пути из вершины V_i в вершину V_j, с номерами промежуточных вершин, не превосходящими k.
Рекуррентное соотношение:

$$a_{ij}^{(k+1)} = \min\left(a_{ij}^{(k)}, a_{ik+1}^{(k)} + a_{k+1j}^{(k)}\right).$$

Действительно, на k+1-м шаге либо минимальный путь не меняется, либо он проходит через вершину V_{k+1}. Матрица $A^{(n)}$ определет результат.

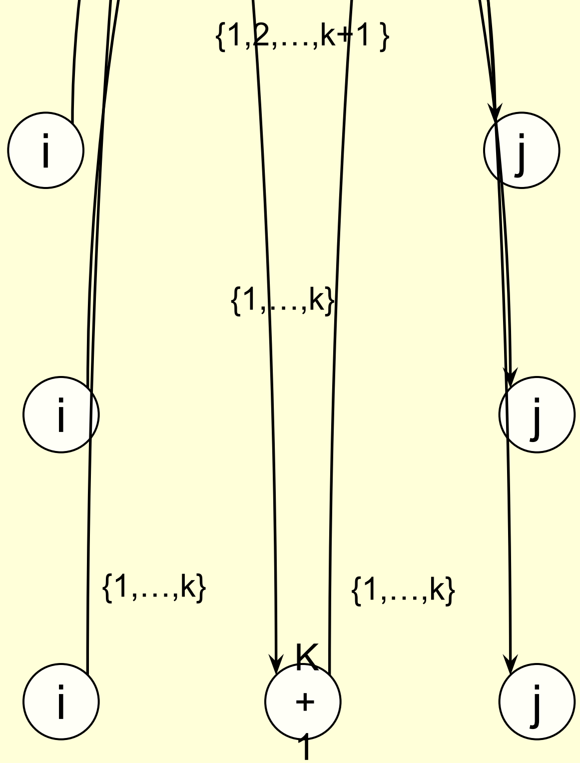
Для нахождения самих кратчайших путей строится последовательность матриц $B^{(0)} \to B^{(1)} \to \ldots \to B^{(k)} \to \ldots \to B^{(n)}$. Элемент $b_{i\,j}^{(k)}$ матрицы $B^{(k)}$ равен номеру второй вершины на кратчайшем пути из V_i в V_j с номерами промежуточных вершин, не превосходящими k, либо 0, если путей нет.

Элемент $b_{i\,j}^{(k+1)}$ не меняется, если в формуле минимум достигается на первом значении, и полагается равным $b_{i\,k+1}^{(k)}$, если минимально второе выражение, так как в этом случае кратчайший путь проходит через V_{k+1}.
Если $s=b_{i\,j}^{(n)}$ дает вторую вершину на кратчайшем пути из V_i в V_j, то $t=b_{s\,j}^{(n)}$ третью, $w=b_{t\,j}^{(n)}$ четвертую и т. д.

Алгоритм Флойда-Уоршелла

Рекуррентная формула:

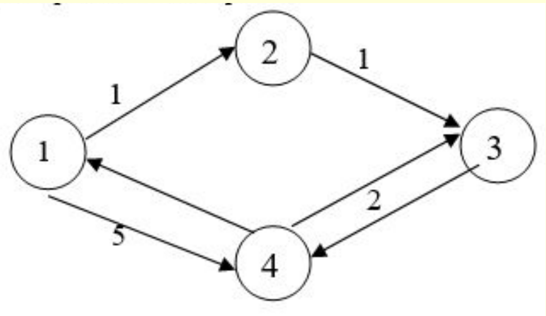
$$a_{i\,j}^{(k+1)} = \min\left(a_{i\,j}^{(k)},\ a_{i\,k+1}^{(k)} + a_{k+1\,j}^{(k)}\right)$$



1) $a_{i\,j}^{(k+1)}$ и $b_{i\,j}^{(k+1)}$ не меняются

2) $a_{i\,j}^{(k+1)} = a_{i\,k+1}^{(k)} + a_{k+1\,j}^{(k)}$, $b_{i\,j}^{(k+1)} = b_{i\,k+1}^{(k)}$

Пример по алгоритму Флойда-Уоршелла



Матрицы $A^{(0)}$ и $B^{(0)}$:

0	1	∞	5
∞	0	1	∞
∞	∞	0	2
5	∞	2	0

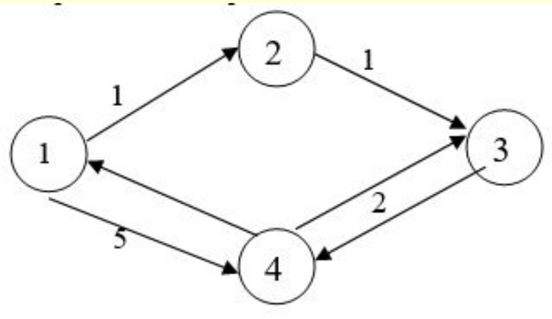
1	2	0	4
0	2	3	0
0	0	3	4
1	0	3	4

Матрицы $A^{(1)}$ и $B^{(1)}$. Через вершину 1 путь 4-2-1, изменения в a_{42} и b_{42}.

0	1	∞	5
∞	0	1	∞
∞	∞	0	2
5	6	2	0

1	2	0	4
0	2	3	0
0	0	3	4
1	1	3	4

Пример по алгоритму Флойда-Уоршелла



Матрицы $A^{(2)}$ и $B^{(2)}$.
Новые пути, проходящие через вершины 1 и 2:
1-2-3 и 4-1-2-3. Изменения в a_{13} и b_{13}, но a_{43} и b_{43}, не меняются!

0	1	2	5
∞	0	1	∞
∞	∞	0	2
5	6	2	0

1	2	2	4
0	2	3	0
0	0	3	4
1	1	3	4

Матрицы $A^{(3)}$ и $B^{(3)}$. Новые пути через вершины 1, 2 и 3: 2-3-4 и 1-2-3-4. Изменения в a_{24} и b_{24}, а также в a_{14} и b_{14}.

0	1	2	4
∞	0	1	3
∞	∞	0	2
5	6	2	0

1	2	2	2
0	2	3	3
0	0	3	4
1	1	3	4

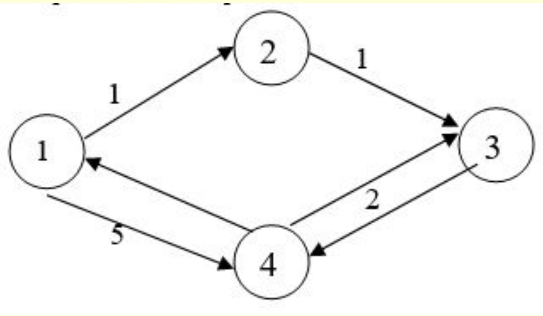
Пример по алгоритму Флойда-Уоршелла

Итог: матрицы $A^{(4)}$ и $B^{(4)}$.

Новые пути, проходящие через вершины 1, 2, 3, 4:

2-3-4-1, 3-4-1 и 3-4-1-2.

Изменения в a_{21} и b_{21}, a_{31} и b_{31}, a_{32} и b_{32}.



0	1	2	4
8	0	1	3
7	8	0	2
5	6	2	0

1	2	2	2
3	2	3	3
4	4	3	4
1	1	3	4

Кратчайший путь из вершины 1 в вершину 4. Его длина a_{14}=4. Для нахождения самого пути просматриваем четвертый столбец матрицы B.

Вторая вершина после вершины 1: b_{14} =2. Переходим в вершину 2.

Вторая вершина на пути 2-4: b_{24}=3. Переходим в вершину 3.

Вторая вершина на пути 3-4: b_{34}=4. Пришли в конец, найден путь 1-2-3-4.
Трудоемкость алгоритма Флойда-Уоршелла O(N^3).

Возможны отрицательные веса, но не должно быть циклов суммарной отрицательной длины.

Максимальный груз

Имеется сеть автомобильных дорог. По некоторым дорогам можно проехать только в одном направлении. Матрица стоимостей A определяет для каждой дороги ее пропускную способность – максимальную массу груза, которую можно провезти по этой дороге. Найти маршрут и максимальную массу груза для его доставки из начального города S в конечный T.

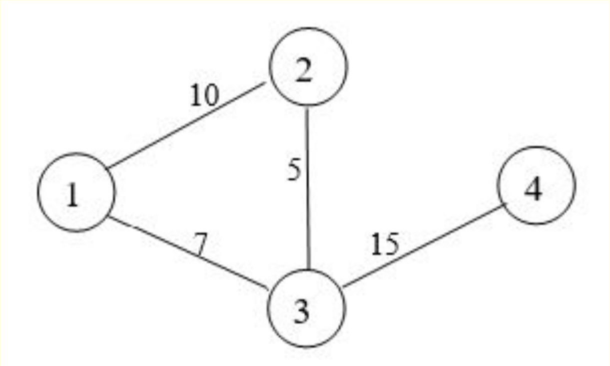
Решение – модификация алгоритма Дейкстры. Временные метки D_i и C_i.

1. Вершине S присваивается окончательная метка ∞, остальным вершинам – временные метки -1.
2. Пусть i – номер последней вершины, которой присвоена окончательная метка C_i. Для каждой вершины j с временной меткой D_j находится $M_j = \min(C_i, a_{ij})$, а затем $D_j = \max(M_j, D_j)$. Если значение D_j увеличивается, то вместе с ним сохраняется номер предыдущей вершины i.
3. Наибольшая из временных меток объявляется окончательной. Если k – номер этой вершины, то $C_k = D_k$.
4. Если новой окончательной метки не появилось, то пути из S в T нет.
5. Если T не получила окончательной метки, то $i = k$ и переход к 2.
6. Конец.

Максимальный груз

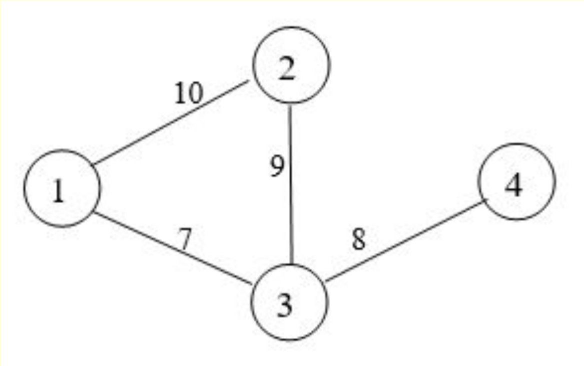
Максимальный груз из 1 в 4. Окончательные метки выделены и подчеркнуты. Формула пересчета меток: $D_j = \max(\min(C_i, a_{ij}), D_j)$. Путь в обратном направлении.

4(3) – 3(1) – 1 или 1 – 3 – 4, вес 7



Шаг	1	2	3	4
1	∞	-1	-1	-1
2	∞	10(1)	7(1)	-1
3	∞	**10(1)**	7(1)	-1
4	∞	**10(1)**	7(1)	-1
5	∞	**10(1)**	**7(1)**	-1
6	∞	**10(1)**	**7(1)**	**7(3)**
7	∞	**10(1)**	**7(1)**	**7(3)**

4(3) – 3(2) – 2(1) – 1 или 1 – 2 – 3 – 4 , вес 8



Шаг	1	2	3	4
1	∞	-1	-1	-1
2	∞	10(1)	7(1)	-1
3	∞	**10(1)**	7(1)	-1
4	∞	**10(1)**	9(2)	-1
5	∞	**10(1)**	**9(2)**	-1
6	∞	**10(1)**	**9(2)**	**8(3)**
7	∞	**10(1)**	**9(2)**	**8(3)**

Поиск циклов

Многие практические задачи сводятся к поиску циклов – путей, начинающихся и заканчивающихся в одной и той же вершине. Элементарные циклы не содержат циклов внутри себя.

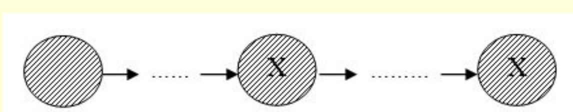
Проверка ацикличности: обход dfs. При входе в вершину красим её в серый цвет, а при выходе - в чёрный. Если при поиске в глубину встретили дугу в серую вершину X, то эта вершина уже была – цикл, находится по предыдущим вершинам. В черные вершины не заходим. Из вершины Y ранее циклов не найдено. Сложность O(*M*+ *N*).

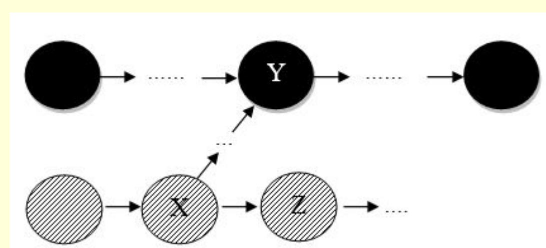
Поиск всех циклов из заданной вершины: проще всего поиск путей на основе dfs. Сложность O(*N!*).

Поиск всех циклов: перебор начальных вершин. Проблема: каждый цикл из *K* вершин порождает еще *K*-1 цикл путем выбора другой начальной вершины. Например, если путь из вершин *a – b – c – a* образует цикл, то циклами будут и пути *b – c – a – b*, *c – a – b – c*.

Прием: чтобы не было повторов, можно считать, что цикл начинает вершина с минимальным номером. Тогда при обходе не заходим в вершину с номером, меньшим начального.

Пример: при обходе из вершины 3 после нахождения начала 3 – 7 не рассматриваем продолжение 3 – 7 – 2, т. к. возможный цикл 3 – 7 – 2 – 3 должен быть найден при обходе из вершины 2 как 2 – 3 – 7 – 2.





Алгоритмы поиска кратчайших путей

- Алгоритм Беллмана-Форда находит кратчайшие пути от начальной вершины до всех остальных вершин и позволяет найти цикл отрицательной длины, достижимый из начальной вершины, если такой имеется.
- Алгоритм Джонсона находит кратчайшие пути между всеми парами вершин и более эффективен для разреженных графов.
- Алгоритм А* использует эвристические оценочные функции, определяющие перспективные направления поиска кратчайшего пути. Вид оценочных функций зависит от предметной области.
- Волновой алгоритм Ли предназначен для планарных графов.
- Топологическая сортировка вершин графа позволяет получить рациональный алгоритм поиска максимальных путей сложности O(M+N) в ациклических ориентированных графах. В общем случае – только переборное решение.
- Поиск k кратчайших непересекающихся путей реализуется путем запрета всех вершин найденных путей и повторения поиска кратчайшего пути.
- Алгоритм Йена определяет k кратчайших простых путей, отличающихся хотя бы одной дугой.
- Алгоритм Эпштейна находит k кратчайших путей, которые могут содержать циклы.
- Как для одного, так и для k кратчайших путей, имеются эвристические и вероятностные алгоритмы поиска.

Благодарю за внимание!