Введение в дискретную математику и математическую логику

•

Лекция №9

Кратчайшие пути между всеми парами вершин

Апанович Зинаида Владимировна

Часть 1
Введение, определения

Введение

В этой лекции мы рассмотрим задачу поиска кратчайших путей между всеми парами вершин в графе.

Такая проблема может возникнуть при составлении таблицы расстояний между всеми парами городов для атласа автодорог.

Как и ранее, дан взвешенный, ориентированный граф

$G = (V, E)$ с функцией веса $w : E \to R$, которая сопоставляет рёбрам вещественные веса.

Мы хотим найти для каждой пары вершин $u, v \in V$ — кратчайший (с наименьшим весом) путь из u в v , где вес пути равен сумме весов его составляющих ребер.

Обычно нам нужен вывод в табличной форме : запись в строке u и столбце v должна быть весом кратчайшего пути из вершины u в вершину v .

Поиск кратчайших путей между всеми парами вершин

Мы можем решить задачу поиска кратчайших путей для всех пар вершин, запустив алгоритм поиска кратчайших путей с одним источником $|V|$ раз, по одному разу из каждой вершины в качестве источника.

Если все веса ребер неотрицательны, мы можем использовать алгоритм Дейкстры .

Грубая оценка времени выполнения составляет менее $O(V^3)$.

Если граф имеет ребра с отрицательным весом , мы не можем использовать алгоритм Дейкстры .

Вместо этого нам придется запустить более медленный алгоритм Беллмана-Форда по одному разу из каждой вершины.

Результирующее время выполнения составляет $O(V^2 E)$, что на плотном графе составляет $O(V^4)$.

Посмотрим, можно ли сделать лучше.

Поиск кратчайших путей между всеми парами вершин

В отличие от алгоритмов с одним источником, которые используют представление графа в виде списка смежности, большинство алгоритмов в этой лекции используют представление графа в виде матрицы смежности .

(Алгоритм Джонсона для разреженных графов использует списки смежности.)

Мы предполагаем, что вершины пронумерованы $1, 2, \ldots, |V|$.

Поиск кратчайших путей между всеми парами вершин

Входные данные : матрица W размером n x n, представляющая веса ребер n- вершинного ориентированного графа $G = (V, E)$.

То есть, $W = \{ w_{ij} \}$, где

w_{ij} = 0, если $i = j$;

= весу ориентированного ребра (i, j), если $i \neq j$ и $(i, j) \in E$; (1)

$\propto$ если $i \neq j$ и $(i, j) \notin$ **E** :

Мы допускаем наличие ребер с отрицательным весом, но предполагаем, что входной граф не содержит циклов с отрицательным весом.

Выходные данные : матрица $D = (d_{ij})$ размером n x n, где элемент d_{ij} содержит вес кратчайшего пути из вершины i в вершину j .

То есть, если $\delta(i , j)$ обозначает вес кратчайшего пути из вершины i в вершину j, то $d_{ij} = \delta(i , j)$ в результатеработы алгоритма.

Матрица предшествования

Чтобы решить задачу поиска кратчайших путей для всех пар вершин на входной матрице смежности, нам необходимо вычислить не только веса кратчайших путей, но и матрицу предшествования Π, *где* π_{ij} *=NIL,* если

либо $i = j$

или нет пути из i в j , и

в противном случае π_{ij} является предшественником вершины j по некоторому кратчайшему пути из i .

Так же, как подграф предшествования G_π является деревом кратчайших путей для данной исходной вершины,

подграф, индуцированный i- й строкой матрицы Π должен быть деревом кратчайших путей с корнем в вершине i .

Подграф предшествования

Для каждой вершины $i \in V$, определим подграф предшествования $G_{\pi, i}$ для i как

$G_{\pi,i} = (V_{\pi,i}, E_{\pi,i})$,

где

$V_{\pi,i} = \{ j \in V : \pi_{ij} \neq \text{NIL} \} \cup \{ i \}$

и

$E_{\pi,i} = \{(\pi_{ij}, j): j \in V_{\pi, i} - \{ i \}\}$.

Некоторые соглашения:

1) Входной граф $G = (V, E)$ имеет n вершин, так что $n = |V|$.
2) Будем обозначать матрицы заглавными буквами, например W, L или D, а их отдельные элементы — строчными буквами с нижним индексом, например w_{ij}, l_{ij} или d_{ij}.
3) Некоторые матрицы будут иметь верхние индексы в скобках, как в $L^{(m)} = l_{ij}^{(m)}$ или $D^{(m)} = d_{ij}^{(m)}$, чтобы указать номер итерации.
4) Для заданной матрицы A размером n x n предположим, что значение n хранится в атрибуте *A.rows.*

Часть 2
Кратчайшие пути и умножение

Кратчайшие пути и умножение матриц

Сначала рассмотрим алгоритм динамического программирования для задачи поиска кратчайших путей для всех пар вершин в ориентированном графе $G = (V, E)$.

Каждый основной цикл динамической программы будет вызывать операцию, очень похожую на умножение матриц , так что алгоритм будет выглядеть как повторное умножение матриц.

Начнем с разработки алгоритма с временем выполнения $\theta(V^4)$ для задачи поиска кратчайших путей для всех пар вершин, а затем улучшим время его выполнения до $(V^3 \lg V)$.

Структура кратчайшего пути

Начнем с характеристики структуры оптимального решения.

Для задачи о кратчайших путях с одним источником на графе $G = (V, E)$ мы доказали (лемма 1 из предыдущей лекции), что все подпути кратчайшего пути являются кратчайшими путями.

Предположим, что мы представляем граф матрицей смежности $W = (w_{ij})$.

Рассмотрим кратчайший путь p *из* вершины i в вершину j и предположим, что p содержит $\leq m$ ребер.

Если предположить, что в графе нет циклов с отрицательным весом, то m конечно.

1. Если $i = j$, то p имеет вес 0 и не имеет ребер.
2. Если вершины i и j различны, то мы разобъем путь p на $i \overset{p'}{\rightsquigarrow} k \to j$, где путь p' теперь содержит не более $m-1$ ребер.

По лемме 1, p' — кратчайший путь из i *в* k, и поэтому

$$\delta(i, j) = \delta(i, k) + w_{kj}.$$

Рекурсивное решение задачи поиска кратчайших путей для всех пар вершин

Теперь пусть $l_{ij}^{(m)}$ будет минимальным весом любого пути из вершины i в вершину j , который содержит $\leq m$ *ребер*.

Когда $m = 0$, существует кратчайший путь от i до j без ребер $\Leftrightarrow i = j$.

Таким образом,

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \ , \\ \infty & \text{if } i \neq j \ . \end{cases}$$

Рекурсивное решение задачи поиска кратчайших путей для всех пар вершин

Для $m \geq 1$, вычисляем $l_{ij}^{(m)}$ как минимум $l_{ij}^{(m-1)}$

(вес кратчайшего пути из i в j, состоящего из $\leq m-1$ ребер) и минимальный вес любого пути из i в j, состоящего не более чем из m ребер, полученный путем просмотра всех возможных предшественников k вершины j .

Таким образом, мы рекурсивно определяем

$$l_{ij}^{(m)} = \min\left(l_{ij}^{(m-1)}, \min_{1 \leq k \leq n}\left\{l_{ik}^{(m-1)} + w_{kj}\right\}\right) \qquad (2)$$

$$= \min_{1 \leq k \leq n}\left\{l_{ik}^{(m-1)} + w_{kj}\right\} .$$

Последнее равенство следует из того, что $w_{jj} = 0$ для всех j .

Рекурсивное решение задачи поиска кратчайших путей для всех пар вершин

Если граф не содержит циклов с отрицательным весом, то для каждой пары вершин *i* и *j* , для которых $\delta(i, j) < \infty$, существует кратчайший путь из *i* в *j* , который является простым и, таким образом, содержит ≤ *n* -1 ребер.

Путь из вершины *i* в вершину *j* с > *n* -1 ребрами не может иметь вес меньше, чем кратчайший путь из *i* в *j* .

Фактические веса кратчайшего пути, таким образом, определяются как

$$\delta(i, j) = l_{ij}^{(n-1)} = l_{ij}^{(n)} = l_{ij}^{(n+1)} = \cdots . \quad (3)$$

Вычисление весов кратчайшего пути снизу вверх

Взяв в качестве входных данных матрицу $W = (w_{ij})$, теперь вычисляем последовательность матриц

$L^{(1)}, L^{(2)}, \ldots, L^{(n-1)}$, где для $m = 1, 2, \ldots, n - 1$, имеем

$L^{(m)} = (l_{ij}^{(m)})$.

Окончательная матрица $L^{(n-1)}$ содержит фактические веса кратчайших путей.

Заметим, что $l_{ij}^{(1)} = w_{ij}$ для всех вершин $i, j \in V$, и поэтому $L^{(1)} = W$.

Сердцем алгоритма является следующая процедура, которая, имея матрицы $L^{(m-1)}$ и W, возвращает

матрицу $L^{(m)}$

То есть он расширяет кратчайшие пути, вычисленные на данный момент, еще на одно ребро.

Вычисление весов кратчайших путей

EXTEND-SHORTEST-PATHS(L, W)
1 $n = L.rows$
2 let $L' = (l_{ij}')$ be a new n x n matrix
3 **for** $i = 1$ **to** n
4 **for** $j = 1$ **to** n
5 $l_{ij}' = \infty$
6 **for** $k = 1$ **to** n
7 $l_{ij}' = \min(l_{ij}', l_{ik} + w_{kj})$
8 return L'

Вычисление весов кратчайших путей

Процедура вычисляет матрицу $L' = (l_{ij}')$, которую возвращает в конце.

Это делается путем вычисления уравнения (2) для всех i и j , используя L для $L^{(m-1)}$ и L' для $L^{(m)}$.

(Она записана без верхних индексов, чтобы сделать ее входные и выходные матрицы независимыми от m .)

Время его выполнения составляет $\Theta(n^3)$ из-за 3 вложенных циклов **for** .

Теперь мы можем увидеть связь с умножением матриц.

Предположим, мы хотим вычислить матричное произведение $C = A \cdot B$ двух матриц A и B *размера* n x n .

Вычисляем для $i, j = 1, 2, \ldots, n$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \ . \qquad (4)$$

Вычисление произведения матриц

SQUARE-MATRIX-MULTIPLY(*A*, *B*)
1 $n = A.rows$
2 let *C* be a new *n* x *n* matrix
3 **for** $i = 1$ **to** n
4 **for** $j = 1$ **to** n
5 $c_{ij} = 0$
6 **for** $k = 1$ **to** n
7 $c_{ij} = c_{ij} + a_{ik} \times b_{kj}$
8 **return** *C*

Вычисление весов кратчайших путей

Обратите внимание, что если мы сделаем замены

$l^{(m-1)}$ -> a,

w -> b,

$l^{(m)}$ -> c,

min -> +,

+ -> ·

в уравнении (2) получаем уравнение (4).

Таким образом, если мы внесем эти изменения в EXTEND-SHORTEST-PATHS, а также заменим $\propto$ (тождество для min) на 0 (тождество для +), мы получим ту же процедуру за $\Theta(n^3)$ времени для умножения квадратных матриц.

Вычисление весов кратчайших путей

Возвращаясь к задаче поиска кратчайших путей для всех пар вершин, мы вычисляем веса кратчайших путей, расширяя кратчайшие пути ребро за ребром.

Пусть $A \cdot B$ обозначает матрицу «произведение», возвращаемую EXTEND-SHORTEST-PATHS(A , B), мы вычисляем последовательность из n -1 матриц

$L^{(1)} = L^{(0)} \cdot W = W$,

$L^{(2)} = L^{(1)} \cdot W = W^2$

$L^{(3)} = L^{(2)} \cdot W = W^3$;

:

:

:

$L^{(n-1)} = L^{(n-2)} \cdot W = W^{n-1}$:

Матрица $L^{(n-1)} = W^{n-1}$ содержит веса кратчайших путей.

:

Вычисление весов кратчайших путей

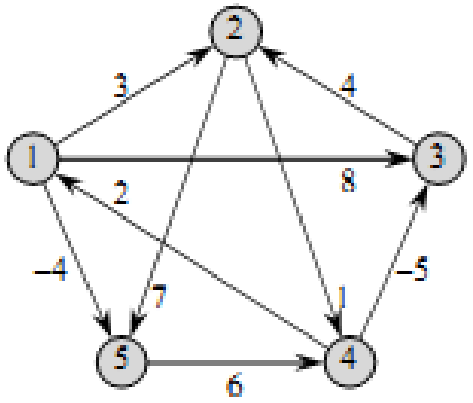
Следующая процедура вычисляет эту последовательность за $\Theta(n^4)$ времени.

```
SLOW-ALL-PAIRS-SHORTEST-PATHS(W)
1 n = W.rows
2 L^(1) = W
3 for m = 2 to n - 1
4     let L^(m) be a new n x n matrix
5     L^(m) = EXTEND-SHORTEST-PATHS(L^(m-1), W)
6 return L^(n-1)
```

На слайдах 21–24 показаны граф и матрицы $L^{(m)}$, вычисленные с помощью процедуры SLOW-ALL-PAIRS-SHORTEST-PATHS.

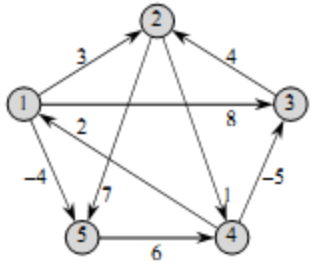
Пример вычисления весов кратчайших путей

$$L^{(1)} = L^{(0)} \cdot W = W$$



$$L^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Пример вычисления весов кратчайших путей $L^{(2)} = L^{(1)} \cdot W = W^2$

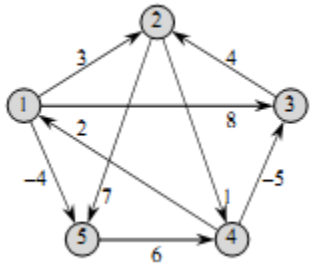


$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$l_{11}^{(2)}$ = min (0+0, 3+∞, 8+∞, ∞+2,-4+∞) = 0
$l_{12}^{(2)}$ = min (0+3, 3+0, 8+4, ∞+∞, -4+∞) = 3
$l_{13}^{(2)}$ = min(0+8, 3+∞, 8+0, ∞ -5, -4+∞) = 8
$l_{14}^{(2)}$ = min (0+2, 3+1, 8+5, ∞+0, -4+6) = 2
$l_{15}^{(2)}$ = min (0-4, 3+7,8+11,∞-2,-4+0) = -4
…
…
…
…

$$L^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

Пример вычисления весов кратчайших путей $L^{(3)} = L^{(2)} \cdot W = W^3$

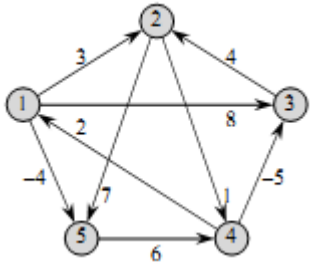


$$\begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$l_{11}^{(3)}$ = min (0+0, 3+∞, 8+∞, 2+2, -4+∞) = 0
$l_{12}^{(3)}$ = min (0+3, 3+0, 8+4, 2+∞, -4+∞) = 3
$l_{13}^{(3)}$ = min(0+8, 3+∞, 8+0, 2 -5, -4+∞) = -3
$l_{14}^{(3)}$ = min (0+∞, 3+1, 8+∞, 2+0, -4+6) = 2
$l_{15}^{(3)}$ = min (0-4, 3+7, 8+∞, 2+∞, -4+0) = -4
…
…

$$L^{(3)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Пример вычисления весов кратчайших путей $L^{(4)} = L^{(3)} \cdot W = W^4$



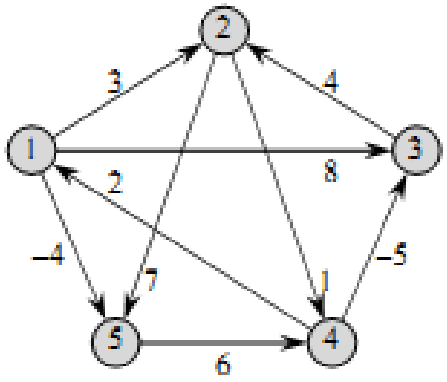
$$\begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$l_{11}^{(4)}$ = min (0+0, 3+∞, -3+∞, 2+2, -4+∞) = 0
$l_{12}^{(4)}$ = min (0+3, 3+0, -3+4, 2+∞, -4+∞) = 1
$l_{13}^{(4)}$ = min(0+8, 3+∞, -3+0, 2 -5, -4+∞) = -3
$l_{14}^{(4)}$ = min (0+∞, 3+1, -3+∞, 2+0, -4+6) = 2
$l_{15}^{(4)}$ = min (0-4, 3+7, -3+∞, 2+∞, -4+0) = - …

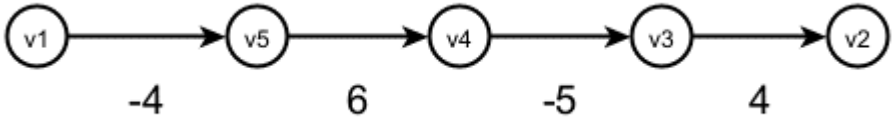
$$L^{(4)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Дерево кратчайших путей из v_1



$$L^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$L^{(4)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$



Улучшение времени работы

Однако наша цель не состоит в вычислении *всех* матриц $L^{(m)}$: нас интересует только матрица $L^{(n-1)}$.

Напомним, что при отсутствии циклов с отрицательным весом уравнение (3) подразумевает

$L^{(m)} = L^{(n-1)}$ для всех целых чисел $m \geq n - 1$.

Так же, как традиционное умножение матриц является ассоциативным, так и умножение матриц, определяемое процедурой EXTEND-SHORTEST-PATHS, тоже является ассоциативным.

Ускорение вычисления весов кратчайших путей

Следовательно, мы можем вычислить $L^{(n-1)}$ всего лишь с помощью $\lceil \lg(n-1) \rceil$ матричных произведений, вычислив последовательность

$L^{(1)} = W$;

$L^{(2)} = W^2 = W \cdot W$;

$L^{(4)} = W^4 = W^2 \cdot W^2$;

$L^{(8)} = W^8 = W^4 \cdot W^4$;

:

:

:

$L^{(2\lceil \lg(n-1) \rceil)} = W^{2\lceil \lg(n-1) \rceil} = W^{2\lceil \lg(n-1) \rceil - 1} \cdot W^{2\lceil \lg(n-1) \rceil - 1}$.

Так как $2^{\lceil \lg(n-1) \rceil} \geq n - 1$, конечное произведение $L^{(2\lceil \lg(n-1) \rceil)}$ равно $L^{(n-1)}$.

Пример вычисления весов кратчайших путей

Следующая процедура вычисляет указанную выше последовательность матриц, используя этот метод повторного возведения в квадрат.

FASTER-ALL-PAIRS-SHORTEST-PATHS(W)

1 $n = W.rows$

2 $L^{(1)} = W$

3 $m = 1$

4 **while** $m < n - 1$

5 let $L^{(2m)}$ be a new n x n matrix

6 $L^{(2m)}$ = EXTEND-SHORTEST-PATHS($L^{(m)}$, $L^{(m)}$)

7 $m = 2m$

8 **return** $L^{(m)}$

Ускорение вычисления весов кратчайших путей

На каждой итерации цикла **while** строк 4–7 мы вычисляем $L^{(2m)} = (L^{(m)})^2$,
начиная с $m = 1$.

В конце каждой итерации мы удваиваем значение m .

Последняя итерация вычисляет $L^{(n-1)}$ путем фактического вычисления $L^{(2m)}$ для некоторого

$n - 1 \leq 2m < 2n - 2$.

По уравнению (3) $L^{(2m)} = L^{(n-1)}$.

В следующий раз, когда будет выполнен тест в строке 4, m удвоится,

так что теперь $m \geq n - 1$, тест не пройден, и процедура возвращает последнюю вычисленную ею матрицу.

Поскольку каждое из произведений матриц $\lceil \lg(n-1) \rceil$ занимает $\Theta(n^3)$ времени,

FASTER-ALL-PAIRS-SHORTEST-PATHS выполняется за $\Theta(n^3 \lg n)$ операций.

Часть 3. Алгоритм Флойда-Уоршелла

Алгоритм Флойда-Уоршелла

Далее мы воспользуемся другой формулировкой динамического программирования для решения задачи поиска кратчайших путей для всех пар вершин в ориентированном графе $G = (V, E)$.

Полученный алгоритм, известный как алгоритм Флойда-Уоршелла , выполняется за время $\Theta(V^3)$.

Как и прежде, в графе могут присутствовать ребра с отрицательным весом , но мы предполагаем, что циклы с отрицательным весом отсутствуют .

Для разработки алгоритма мы следуем процессу динамического программирования.

Структура кратчайшего пути

Алгоритм Флойда-Уоршелла рассматривает промежуточные вершины кратчайшего пути, где промежуточной вершиной простого пути $p = < v_1, v_2, \ldots, v_l >$ является любая вершина p кроме v_1 или v_l, то есть, любая вершина во множестве

$\{ v_2, v_3, \ldots, v_{l-1} \}$.

Алгоритм Флойда-Уоршелла основан на следующем наблюдении.

При нашем предположении, что вершины графа G равны $V = \{1, 2, \ldots, n\}$, рассмотрим подмножество $\{1, 2, \ldots, k\}$ вершин для некоторого k .

Для любой пары вершин $i, j \in V$ рассмотрим все пути от i до j, все промежуточные вершины которых взяты из $\{ 1, 2, \ldots, k \}$, и пусть p будет путем минимального веса. (Путь p простой.)

Алгоритм Флойда-Уоршелла использует связь между путем p и кратчайшими путями от i до j со всеми промежуточными вершинами во множестве $\{1, 2, \ldots, k - 1\}$.

Соотношение зависит от того, является ли k промежуточной вершиной пути p .

Алгоритм Флойда-Уоршелла

Если k не является промежуточной вершиной пути p, то все промежуточные вершины пути p находятся в множестве

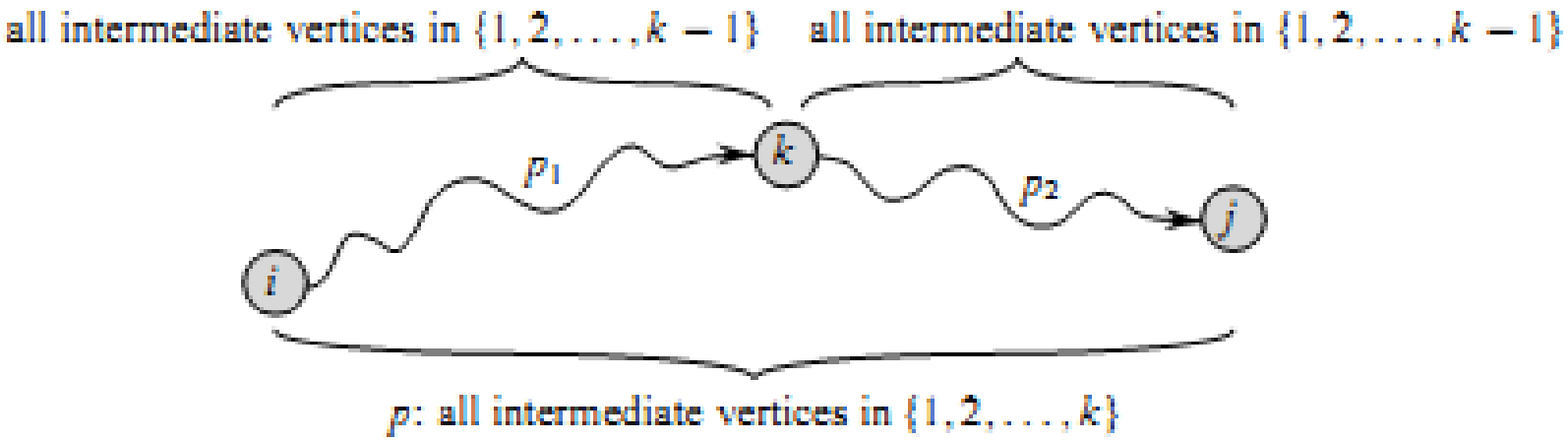
{1, 2, …, k - 1}.

Таким образом, кратчайший путь из вершины i в вершину j со всеми промежуточными вершинами из множества

{1, 2, …, k - 1 } также является кратчайшим путем из i в j со всеми промежуточными вершинами из множества { 1, 2, …, k }.

Если k — промежуточная вершина пути p, то мы разлагаем путь p на два подпути $i^{(p1)} \rightsquigarrow k^{(p2)} \rightsquigarrow j$, как показано на следующем слайде.

Алгоритм Флойда-Уоршелла



Путь p — кратчайший путь из вершины i *в* вершину j , а k — промежуточная вершина с наибольшим номером для p .

Путь p_1 , часть пути p из вершины i в вершину k , имеет все промежуточные вершины во множестве {1, 2, …, k - 1}.

То же самое справедливо для пути p_2 из вершины k в вершину j .

Алгоритм Флойда-Уоршелла

По лемме 1 из предыдущей лекции, путь p_1 — кратчайший путь из i в k со всеми промежуточными вершинами во множестве

{1, 2, …, k }.

На самом деле, мы можем сделать немного более сильное заявление.

Поскольку вершина k не является промежуточной вершиной пути p_1 , все промежуточные вершины пути p_1 находятся во множестве {1, 2, …, k - 1 }.

Следовательно, p_1 — кратчайший путь из i *в* k со всеми промежуточными вершинами во множестве

{ 1, 2, …, k - 1 }.

Аналогично, p_2 — кратчайший путь из вершины k в вершину j со всеми промежуточными вершинами во множестве

{ 1, 2 …, k - 1 }.

Алгоритм Флойда-Уоршелла

На основе вышеизложенных наблюдений мы определяем еще одну рекурсивную формулировку оценок длины кратчайших путей.

Пусть $d_{ij}^{(k)}$ — вес кратчайшего пути из вершины i в вершину j, для которого все промежуточные вершины находятся во множестве $\{1, 2, \ldots, k\}$.

При $k = 0$ путь из вершины i вершину j без промежуточных вершин с номером больше 0 вообще не имеет промежуточных вершин.

Такой путь имеет ≤ 1 ребро, и, следовательно,

$d_{ij}^{(0)} = w_{ij}$.

Алгоритм Флойда-Уоршелла

Следуя вышеизложенному обсуждению, мы определяем $d_{ij}^{(k)}$ рекурсивно по следующему правилу

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0, \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1. \end{cases} \qquad (5)$$

Поскольку для любого пути все промежуточные вершины находятся во множестве {1, 2, …, n }, матрица $D^{(n)} = d_{ij}^{(n)}$ дает окончательный ответ: $d_{ij}^{(n)} = \delta(i, j)$ для всех $i, j \in V$.

Вычисление весов кратчайшего пути снизу вверх

На основе рекуррентного соотношения (5) мы можем использовать следующую процедуру для вычисления значений $d_{ij}^{(k)}$ в порядке возрастания значений k .

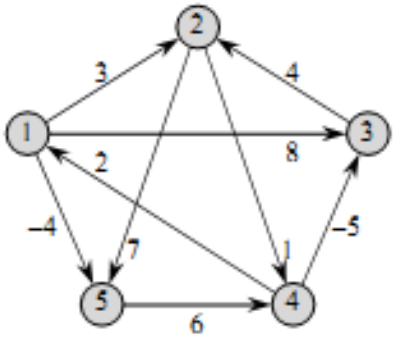
Вход: матрица W *размером* n x n, определенная как в уравнении (1).

Выходные данные: матрица $D^{(n)}$ весов кратчайших путей.

Алгоритм Флойда-Уоршелла

FLOYD-WARSHALL(W)
1 $n = W.rows$
2 $D^{(0)} = W$
3 **for** $k = 1$ **to** n
4 let $D^{(k)} = d_{ij}^{(k)}$ be a new n x n matrix
5 **for** $i = 1$ **to** n
6 **for** $j = 1$ **to** n
7 $d_{ij}^{(k)} = \min (d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
8 **return** $D^{(n)}$

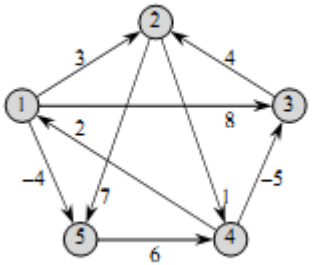
Алгоритм Флойда-Уоршелла. Пример.



$D^{(0)} = W$

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Алгоритм Флойда-Уоршелла. Пример.



Промежуточные вершины = {1}

Все ребра, входящие в вершину 1, показаны в столбце 1.

Все ребра, исходящие из вершины 1, показаны в строке 1.

$d_{41}^{(1)} = \min(d_{41}^{(0)} + d_{11}^{(0)}, d_{41}^{(0)}) = \min(0+2, 2) = 2$

$d_{42}^{(1)} = \min(d_{41}^{(0)} + d_{12}^{(0)}, d_{42}^{(0)}) = \min(3+2, \infty) \Rightarrow d_{42} = 5,\ \pi_{42} = 1$

$d_{43}^{(1)} = \min(d_{41}^{(0)} + d_{13}^{(0)}, d_{43}^{(0)}) = \min(8+2, -5) = -5$

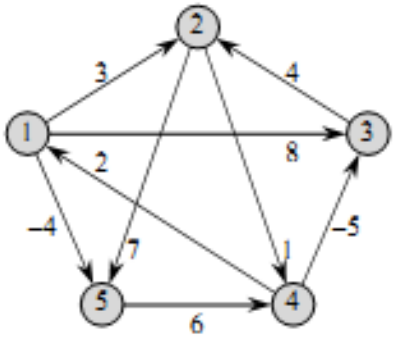
$d_{44}^{(1)} = \min(d_{41}^{(0)} + d_{14}^{(0)}, d_{44}^{(0)}) = \min(\infty+2, 0) = 0$

$d_{45}^{(1)} = \min(d_{41}^{(0)} + d_{15}^{(0)}, d_{45}^{(0)}) = \min(-4+2 = -2, \infty) \Rightarrow d_{45} = -2,\ \pi_{45} = 1$

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Алгоритм Флойда-Уоршелла. Пример.



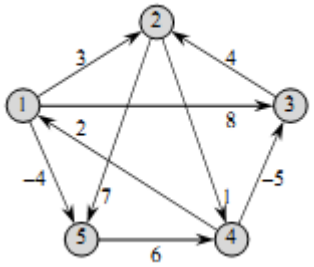
Промежуточные вершины = {1, 2}
Все ребра, входящие в вершину 2, показаны в столбце 2.
Все ребра, исходящие из вершины 2, показаны в строке 2.

$d_{14}^{(2)} = d_{12}^{(1)} + d_{24}^{(1)} = 1 + 3 = 4,$
$d_{34}^{(2)} = d_{32}^{(1)} + d_{24}^{(1)} = 1 + 4 = 5,$
$d_{35}^{(2)} = d_{32}^{(1)} + d_{25}^{(1)} = 4 + 7 = 11.$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Алгоритм Флойда-Уоршелла. Пример



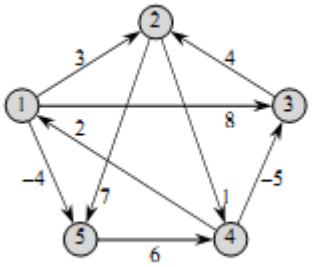
Промежуточные вершины = {1, 2, 3}

$d_{42}^{(3)} = d_{43}^{(2)} + d_{32}^{(2)} = 4 - 5 = -1.$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Алгоритм Флойда-Уоршелла. Пример



Промежуточные вершины = {1, 2, 3, 4}

$d_{13}^{(4)} = d_{43}^{(3)}\, d_{43}^{(3)} = 4 - 5 = -1,$

$d_{21}^{(4)} = d_{24}^{(3)} + d_{41}^{(3)} = 1 + 2 = 3$

$d_{23}^{(4)} = d_{24}^{(3)} + d_{43}^{(3)} = 1 - 5 = -4,$

$d_{25}^{(4)} = d_{24}^{(3)} + d_{45}^{(3)} = 1 - 2 = -1,$

$d_{31}^{(4)} = d_{34}^{(3)} + d_{41}^{(3)} = 5 + 2 = 7,$

$d_{35}^{(4)} = d_{34}^{(3)} + d_{45}^{(3)} = 5 - 2 = 3,$

$d_{51}^{(4)} = d_{54}^{(3)} + d_{41}^{(3)} = 6 + 2 = 8$

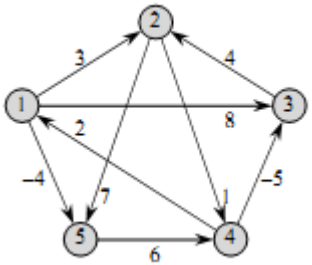
$d_{52}^{(4)} = d_{54}^{(3)} + d_{42}^{(3)} = 6 - 1 = 5$

$d_{53}^{(4)} = d_{54}^{(3)} + d_{43}^{(3)} = 6 - 5 = 1$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Алгоритм Флойда-Уоршелла. Пример



Промежуточные вершины = {1, 2, 3, 4, 5}

$d_{12}^{(5)} = d_{15}^{(4)} + d_{52}^{(4)} = -4 + 5 = 1,$

$d_{13}^{(5)} = d_{15}^{(4)} + d_{53}^{(4)} = -4 + 1 = -3$

$d_{14}^{(5)} = d_{15}^{(4)} + d_{54}^{(4)} = -4 + 6 = 2.$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Алгоритм Флойда-Уоршелла

Время выполнения алгоритма Флойда-Уоршелла определяется тройным вложенным циклом **for** строк 3–7.

Поскольку каждое выполнение строки 7 занимает время $O(1)$, алгоритм выполняется за время $\Theta(n^3)$.

Построение кратчайших путей в алгоритме Флойда-Уоршелла.

Существует множество различных методов построения кратчайших путей в алгоритме Флойда-Уоршелла.

Один из способов — вычислить матрицу *D* весов кратчайших путей, а затем построить матрицу предшествования Π *по* матрице *D.*

Построение кратчайших путей в в алгоритме Флойда-Уоршелла.

В качестве альтернативы мы можем вычислить матриц-предшествования Π, пока алгоритм вычисляет матрицы $D^{(k)}$

В частности , мы вычисляем последовательность матриц $\Pi^{(0)}$, $\Pi^{(1)}$, $\Pi^{(n)}$, *где* $\Pi = \Pi^{(n)}$.

Мы определяем $\pi_{ij}^{(k)}$ как предшественника вершины *j* на кратчайшем пути из вершины *i* со всеми промежуточными вершинами во множестве {1, 2, …, *k* }.

Мы можем дать рекурсивную формулировку $\pi_{ij}^{(k)}$.

Когда $k = 0$, кратчайший путь из вершины *i* в *j* вообще не имеет промежуточных вершин.

Таким образом,

$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty , \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty . \end{cases}$$

Построение кратчайших путей в в алгоритме Флойда-Уоршелла.

Для $k \geq 1$, если мы пойдем по пути $i \rightsquigarrow k \rightsquigarrow j$, где $k \neq j$, то предшественник j , которого мы выбираем, совпадает с предшественником j, которого мы выбираем на кратчайшем пути из k со всеми промежуточными вершинами во множестве $\{1, 2, \ldots, k - 1\}$.

В противном случае, мы выбираем того же предшественника j , которого мы выбрали на кратчайшем пути из i со всеми промежуточными вершинами во множестве $\{1, 2, \ldots, k - 1\}$.

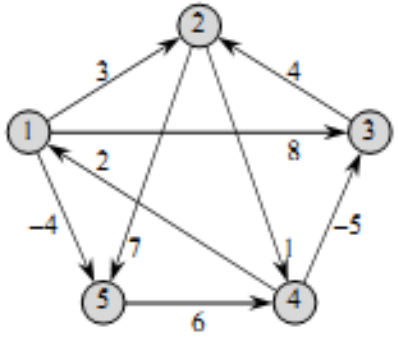
Формально, для $k \geq 1$,

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}, \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}. \end{cases}$$

Построение кратчайших путей в в алгоритме Флойда-Уоршелла.

Слайды 50–55 показывают последовательность матриц $\Pi^{(k)}$, которую результирующий алгоритм вычисляет для графа на слайдах 38–43.

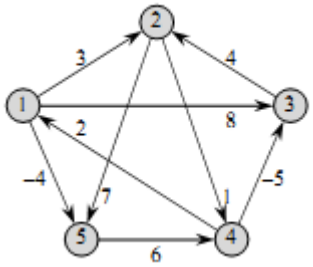
Последовательность матриц $D^{(k)}$ и $\Pi^{(k)}$, вычисленная алгоритмом Флойда- Уоршелла.



$D^{(0)} = W$

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

Последовательность матриц $D^{(k)}$ и $\Pi^{(k)}$, вычисленная алгоритмом Флойда- Уоршелла



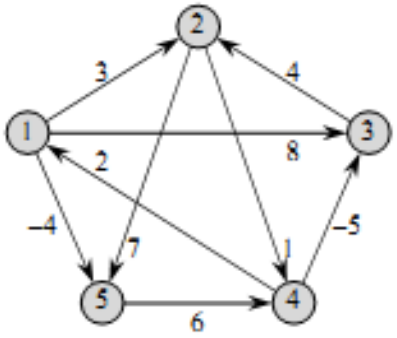
$d_{42}^{(1)} = d_{41}^{(0)} + d_{12}^{(0)}$ => $\pi_{42}^{(1)} = \pi_{12}^{(0)} = 1$

$d_{45}^{(1)} = (d_{41}^{(0)} + d_{15}^{(0)}$ => $\pi_{45}^{(1)} = \pi_{15}^{(0)} = 1$

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

Последовательность матриц $D^{(k)}$ и $\Pi^{(k)}$, вычисленная алгоритмом Флойда-Уоршелла



$d_{14}^{(2)} = d_{12}^{(1)} + d_{24}^{(1)} \Rightarrow \pi_{14}^{(2)} = \pi_{24}^{(1)} = 2$

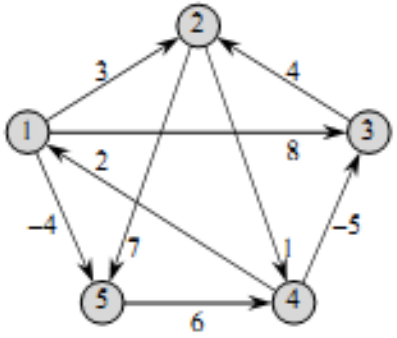
$d_{34}^{(2)} = d_{32}^{(1)} + d_{24}^{(1)} \Rightarrow \pi_{34}^{(2)} = \pi_{24}^{(1)} = 2$

$d_{35}^{(2)} = d_{32}^{(1)} + d_{25}^{(1)} \Rightarrow \pi_{42}^{(1)} = \pi_{12}^{(0)} = 2$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

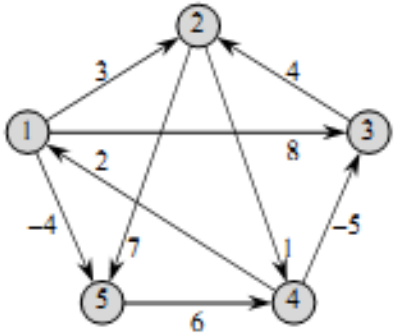
Последовательность матриц $D^{(k)}$ и $\Pi^{(k)}$, вычисленная алгоритмом Флойда-Уоршелла



$d_{42}^{(3)} = d_{43}^{(2)} + d_{32}^{(2)} \Rightarrow \pi_{42}^{(3)} = \pi_{32}^{(2)} = 3$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$



$d_{13}^{(4)} = d_{43}^{(3)} + d_{43}^{(3)} \Rightarrow \pi_{13}^{(4)} = \pi_{43}^{(3)} = 4$

$d_{21}^{(4)} = d_{24}^{(3)} + d_{41}^{(3)} \Rightarrow \pi_{21}^{(4)} = \pi_{41}^{(3)} = 4$

$d_{23}^{(4)} = d_{24}^{(3)} + d_{43}^{(3)} \Rightarrow \pi_{23}^{(4)} = \pi_{43}^{(3)} = 4$

$d_{25}^{(4)} = d_{24}^{(3)} + d_{45}^{(3)} \Rightarrow \pi_{25}^{(4)} = \pi_{45}^{(3)} = 1$

$d_{31}^{(4)} = d_{34}^{(3)} + d_{41}^{(3)} \Rightarrow \pi_{31}^{(4)} = \pi_{41}^{(3)} = 4$

$d_{35}^{(4)} = d_{34}^{(3)} + d_{45}^{(3)} \Rightarrow \pi_{35}^{(4)} = \pi_{45}^{(3)} = 1$

$d_{51}^{(4)} = d_{54}^{(3)} + d_{41}^{(3)} \Rightarrow \pi_{51}^{(4)} = \pi_{41}^{(3)} = 4$

$d_{52}^{(4)} = d_{54}^{(3)} + d_{42}^{(3)} \Rightarrow \pi_{52}^{(4)} = \pi_{42}^{(3)} = 3$

$d_{53}^{(4)} = d_{54}^{(3)} + d_{43}^{(3)} \Rightarrow \pi_{53}^{(4)} = \pi_{43}^{(3)} = 4$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

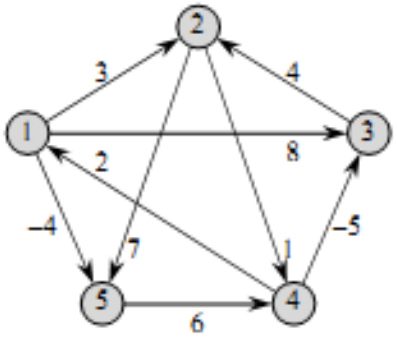
$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Промежуточные вершины = {1, 2, 3, 4, 5}

$d_{12}^{(5)} = d_{15}^{(4)} + d_{52}^{(4)} \Rightarrow \pi_{12}^{(5)} = \pi_{52}^{(4)} = 3$

$d_{13}^{(5)} = d_{15}^{(4)} + d_{53}^{(4)} \Rightarrow \pi_{13}^{(5)} = \pi_{53}^{(4)} = 4$

$d_{14}^{(5)} = d_{15}^{(4)} + d_{54}^{(4)} \Rightarrow \pi_{14}^{(5)} = \pi_{54}^{(4)} = 5$



$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Часть 4. Алгоритм Джонсона для разреженных графов

Алгоритм Джонсона для разреженных графов

Алгоритм Джонсона находит кратчайшие пути между всеми парами вершин за время $O(V^2 \lg V + VE)$.

Для разреженных графов он асимптотически быстрее, чем повторное возведение матриц в квадрат или алгоритм Флойда-Уоршелла.

Алгоритм либо возвращает матрицу весов кратчайших путей для всех пар вершин, либо сообщает, что входной граф содержит цикл с отрицательным весом.

Алгоритм Джонсона для разреженных графов

Алгоритм Джонсона использует в качестве подпрограмм как алгоритм Дейкстры, так и алгоритм Беллмана-Форда .

Алгоритм Джонсона использует технику изменения весов, которая работает следующим образом.

Если все веса w ребер в графе $G = (V, E)$ неотрицательны , мы можем найти кратчайшие пути между всеми парами вершин, запустив алгоритм Дейкстры один раз из каждой вершины.

Если G имеет ребра с отрицательным весом, но не имеет циклов с отрицательным весом, мы просто вычисляем новое множество неотрицательных весов ребер , который позволяет нам использовать тот же метод.

Алгоритм Джонсона для разреженных графов

Новое множество весов ребер w' должно удовлетворять двум важным свойствам:

1. Для всех пар вершин $u, v \in V$, путь p — это кратчайший путь из u *в* v с использованием функции весов w ⇔ p также является кратчайшим путем из u *в* v с использованием функции весов w' .
2. Для всех ребер (u, v) новый вес $w'(u, v)$ неотрицателен .

Мы можем предварительно обработать G, так чтобы определить новую функцию весов w' за время $O(VE)$.

Сохранение кратчайших путей путем изменения весов ребер

Следующая лемма показывает, как легко мы можем изменить веса ребер, чтобы удовлетворить первому свойству выше.

Мы используем δ для обозначения весов кратчайшего пути, полученных из функции весов *w,* и δ' для обозначения весов кратчайшего пути, полученных из функции весов *w'*.

Лемма 1 (Изменение весов не изменяет кратчайшие пути)

Дан взвешенный ориентированный граф $G = (V, E)$ с функцией весов

$w : E \to R$,

пусть $h : V \to R$ — *любая* функция, отображающая вершины в вещественные числа.

Для каждого ребра $(u, v) \in E$, определим

$$w'(u, v) = w(u, v) + h(u) - h(v) \qquad (9)$$

Сохранение кратчайших путей методом изменения весов ребер

Пусть $p = \langle 0, 1, \ldots, k \rangle$ — любой путь из вершины 0 в вершину k .

Тогда p — кратчайший путь из вершины 0 в вершину k с функцией веса w ⇔ это кратчайший путь с функцией веса w'.

То есть,

$w(p) = \delta(0, k) \Leftrightarrow w'(p) = \delta'(0, k).$

Более того, G имеет цикл с отрицательным весом, использующий функцию весов w

⇔ G имеет цикл с отрицательным весом, использующий функцию весов w'.

Сохранение кратчайших путей методом изменения весов

Доказательство Начнем с демонстрации того, что

$w'(p) = w(p) + h(v_0) - h(v_k)$ (10)

У нас есть

$$
\begin{aligned}
\hat{w}(p) &= \sum_{i=1}^{k} \hat{w}(v_{i-1}, v_i) \\
&= \sum_{i=1}^{k} (w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i)) \\
&= \sum_{i=1}^{k} w(v_{i-1}, v_i) + h(v_0) - h(v_k) \qquad \text{(because the sum telescopes)} \\
&= w(p) + h(v_0) - h(v_k) \ .
\end{aligned}
$$

Сохранение кратчайших путей методом изменения весов

Следовательно, любой путь p из v_0 в v_k имеет вес

$w'(p) = w(p) + h(v_0) - h(v_k)$.

Поскольку $h(v_0)$ и $h(v_k)$ не зависят от пути, если один путь из v_0 в v_k короче другого с использованием функции весов w, то он также короче с использованием w'.

Таким образом, $w(p) = \delta(v_0, v_k) \Leftrightarrow w'(p) = \delta'(v_0, v_k)$.

Сохранение кратчайших путей методом изменения весов

Наконец, мы показываем, что G имеет цикл с отрицательным весом, используя функцию весов w ⇔

G имеет цикл с отрицательным весом, используя функцию весов w'.

Рассмотрим любой цикл $c = < v_0 , v_1 ,\ldots, v_k >$, где $v_0 = v_k$.

По уравнению (10),

$w'(c) = w(c) + h(v_0) - h(v_k) = w(c)$;

и таким образом c имеет отрицательный вес,

при использовании w ⇔ c имеет отрицательный вес при использовании w'.

Получение неотрицательных весов методом изменения весов

Мы хотим, чтобы $w'(u, v) \geq 0$ для всех ребер $(u, v) \in E$.

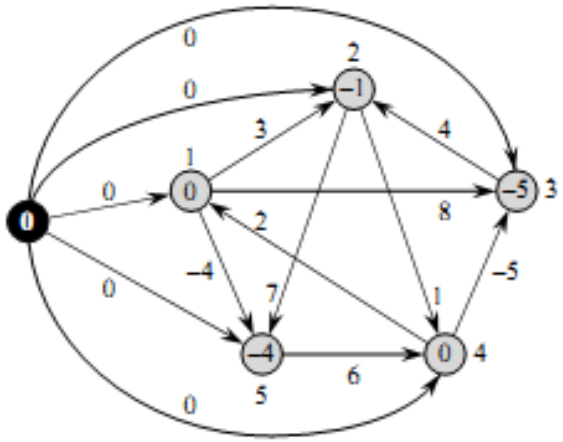
Дан взвешенный ориентированный граф $G = (V, E)$ с функцией весов $w : E \to R$,

мы создаем новый граф $G' = (V', E')$, где $V' = V \cup \{s\}$ для некоторой новой вершины $s \in V$ и $E' = E \cup \{(s, v) : v \in V\}$.

Расширим функцию весов w так, чтобы $w(s, v) = 0$ для всех $v \in V$.

а) поскольку s не имеет входящих в нет ребер, то никакие кратчайшие пути в G', кроме тех, которые имеют истоком s, не содержат s.

б) G' не имеет циклов с отрицательным весом $\Leftrightarrow$ G не имеет циклов с отрицательным весом.



(а)

Получение неотрицательных весов путем перевешивания

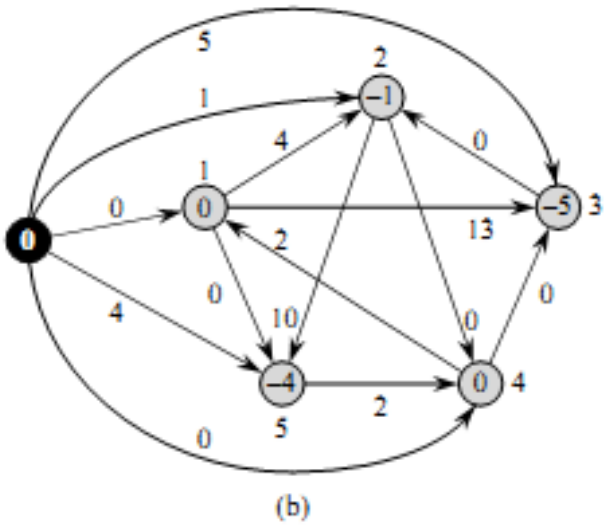
Теперь предположим, что G и G' не имеют циклов с отрицательным весом.

Определим $h(v) = \delta(s, v)$ для всех $v \in V'$.

По неравенству треугольника (лемма 24.10) имеем $h(v) \leq h(u) + w(u, v)$ для всех ребер $(u, v) \in E'$.

Таким образом, если мы определим новые веса w методом измененя веса в соответствии с уравнением (9), то получим $w'(u, v) = w(u, v) + h(u) - h(v) \geq 0$, и мы удовлетворим второму свойству.

Получение неотрицательных весов методом изменения весов



$w'(s, 1) = 0 + 0 - 0 = 0$
$w'(s, 2) = 0 + 0 - (-1) = 1$
$w'(s, 3) = 0 + 0 - (-5) = 5$
$w'(s, 4) = 0 + 0 - 0 = 0$
$w'(s, 5) = 0 + 0 –(-4) = 4$
$w'(1, 2) = 3 + 0 – (-1) = 4$
$w'(1, 3) = 8 + 0 – (-5) = 13$
$w'(1, 5) = -4 + 0 – (-4) = 0$
$w'(2, 4) = 1+ (-1) - 0 = 0$
$w'(2, 5) = 7+ (-1) – (-4) = 10$
$w'(3, 2) = 4 +(-5) – (-1) = 0$
$w'(4, 3) = -5 + 0 – (-5) = 0$
$w'((4, 1) : 2 + 0 - 0 = 2$
$w'((5, 4): 6 – 4 + 0 = 2$

Вычисление кратчайших путей для всех пар вершин

Алгоритм Джонсона для вычисления кратчайших путей для всех пар вершин использует алгоритм Беллмана-Форда и алгоритм Дейкстры в качестве подпрограмм.

Он неявно предполагает, что ребра хранятся в *списках смежности* .

Алгоритм выдает обычную матрицу $D = d_{ij}$ размерности $|V| \times |V|$, где $d_{ij} = \delta(i, j)$, или сообщает, что входной граф содержит цикл с отрицательным весом.

Мы предполагаем, что вершины пронумерованы от 1 до $|V|$.

JOHNSON(G, w)

1 compute G', where $G'.V = G.V \cup \{s\}$,
  $G'.E = G.E \cup \{(s, v) : v \in G.V\}$,
  and $w(s, v) = 0$ for all $v \in G.V$

2 **if** BELLMAN-FORD(G', w, s) == FALSE

3 print "the input graph contains a negative-weight cycle"

4 **else for** each vertex $v \in G'.V$

5 set $h(v)$ to the value of $\delta(s, v)$ computed by the Bellman-Ford algorithm

6 **for** each edge $(u, v) \in G'.E$

7 $w'(u, v) = w(u, v) + h(u) - h(v)$

8 let $D = (d_{uv})$ be a new n x n matrix

9 **for** each vertex $u \in G.V$

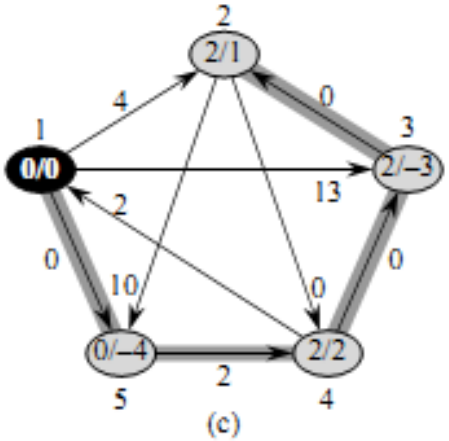
10 run DIJKSTRA(G, w', u) to compute $\delta'(u, v)$ for all $v \in G.V$

11 **for** each vertex $v \in G.V$

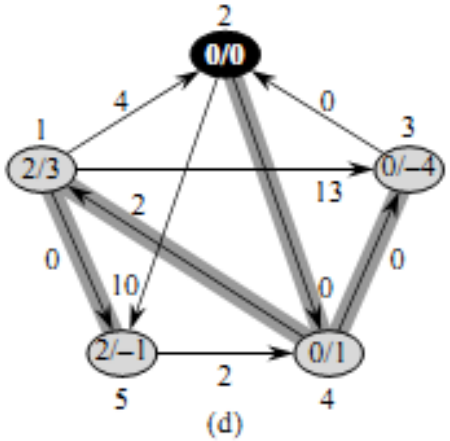
12 $d_{uv} = \delta'(u, v) + h(v) - h(u)$

13 **return** D

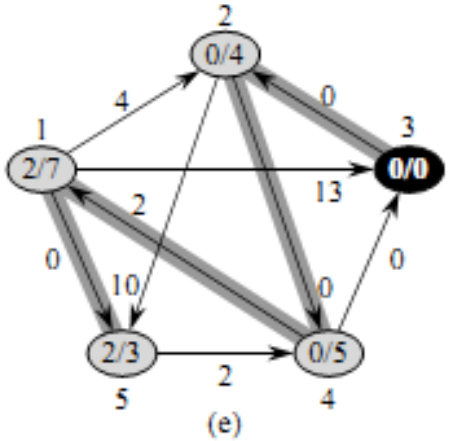
Результат выполнения алгоритма Дейкстры для каждой вершины графа *G* с использованием весовой функции *w*'.



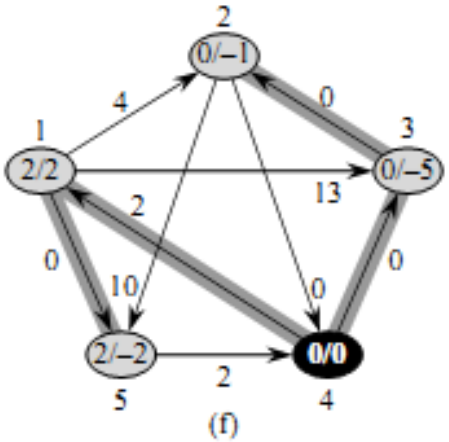
(c)



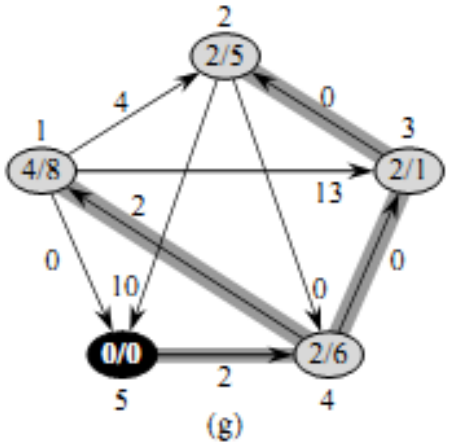
(d)



(e)



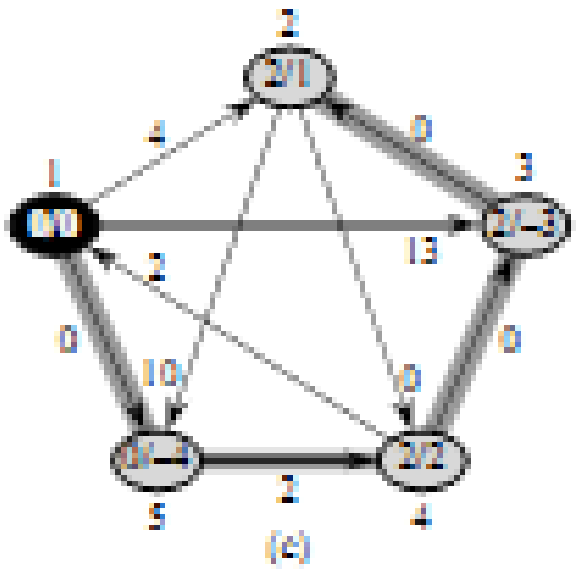
(f)



(g)

Внутри каждой вершины находятся значения $\delta'(u, v)$ и $\delta(u, v)$, разделенные косой чертой.

Возвращение к исходным весам



$\delta(u, v) = \delta'(u, v) + h(v) - h(u)$

$\delta(1, 1) = 0 + 0 - 0 = 0$

$\delta(1, 2) = 2 + (-1) - 0 = 1$

$\delta(1, 3) = 2 + (-5) - 0 = -3$

$\delta(1, 4) = 2 + 0 - 0 = 2$

$\delta(1, 5) = 0 + (-4) - 0 = -4$

- Ваши вопросы?
- Контакты лектора:
  apanovich_09@mail.ru