Лекция 5. Обход графов. Расстояния в графах

1 Задачи об обходах графа

Для определения количества и состава компонент связности используют алгоритмы обхода графов: поиск в глубину и поиск в ширину.

Поиск в глубину (англ. Depth-first search, DFS)

Идея DFS состоит в том, чтобы идти «вглубь» графа, насколько это возможно.

Рекурсивный вариант алгоритма

1. Все вершины графа отмечаем, как не посещенные. Выбирается первая вершина и помечается как посещённая.
2. Для последней посещённой вершины выбирается первая смежная не посещенная вершина, ей присваивается значение посещённой. Если таких вершин нет, то берётся предыдущая помеченная как посещённая вершина.
3. Предыдущий шаг повторяется до тех пор, пока все вершины не будут помечены как посещённые.

Рассмотрим шаги алгоритма на примере графа, показанного на рисунке 1.

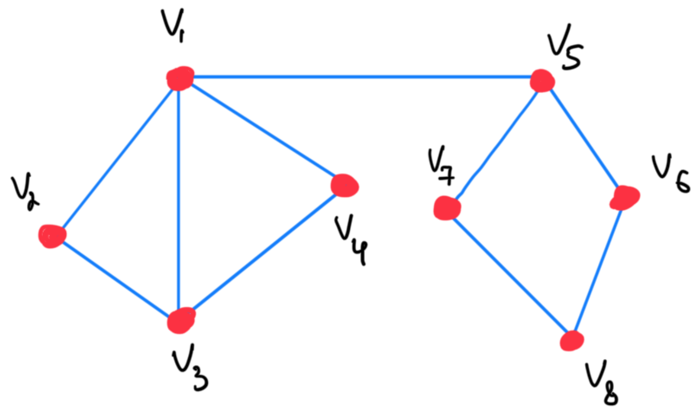

Рис. 1: Пример графа

1. Стартовая вершина v_1, с ней смежны v_2, v_3, v_4 и v_5. Из v_1 переходим в первую смежную непосещённую вершину v_2.
2. Аналогичным образом из v_2 переходим в v_3, а из v_3 в v_4.
3. Больше нет смежных с v_4 вершин, поэтому возвращаемся в v_3.
4. Аналогичным образом из v_3 возвращаемся в v_2, а оттуда в v_1.
5. С v_1 смежна непосещённая вершина v_5, переходим в неё и продолжаем аналогичным образом обходить оставшиеся вершины.
6. Получившийся порядок обхода вершин:

$$v_1, v_2, v_3, v_4, v_5, v_6, v_8, v_7.$$

Нерекурсивный вариант алгоритма

Определение 1: Стек

Стек (англ. stack — стопка) – абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. last in – first out, «последним пришёл – первым вышел»).

Рассмотрим шаги алгоритма:

1. Все вершины графа отмечаем, как не посещённые. Выбирается первая вершина и помечается как посещённая. Эту вершину помещаем в стек.
2. Пока стек не пустой:
   (a) Извлекаем последнюю добавленную вершину.
   (b) Просматриваем все смежные с ней не посещённые вершины, помещаем их в стек и отмечаем как посещённые. Порядок выхода вершин из стека и будет порядком обхода вершин графа.
3. Если после завершения алгоритма не все вершины были рассмотрены, то необходимо запустить алгоритм от одной из нерассмотренных вершин.

Рассмотрим шаги алгоритма на примере графа, показанного на рисунке 1.

1. Изначально все вершины отмечены как не посещённые, а стек пуст.

Посещённость вершин							
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
0	0	0	0	0	0	0	0

Стек				

2. Помещаем вершину v_1 в стек и отмечаем её как посещённую.

Посещённость вершин							
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
1	0	0	0	0	0	0	0

Стек				
v_1				

3. Извлекаем вершину v_1 из стека, добавляем туда смежные с v_1 вершины v_2, v_3, v_4, v_5 и отмечаем их как посещённые.

Посещённость вершин							
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
1	1	1	1	1	0	0	0

Стек				
v_2	v_3	v_4	v_5	

4. Извлекаем вершину v_5 из стека, добавляем туда смежные с v_5 вершины v_6, v_7 и отмечаем их как посещённые.

Посещённость вершин							
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
1	1	1	1	1	1	1	0

Стек				
v_2	v_3	v_4	v_6	v_7

5. Извлекаем вершину v_7 из стека, добавляем туда смежную с v_7 вершину v_8 и отмечаем её как посещённую.

Посещённость вершин							
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
1	1	1	1	1	1	1	1

Стек				
v_2	v_3	v_4	v_6	v_8

6. Все вершины отмечены как посещённые, удаляя оставшиеся вершины из стека, запишем получившийся порядок обхода:

$$v_1, v_5, v_7, v_8, v_6, v_4, v_3, v_2.$$

Поиск в ширину (англ. Breadth-first search, BFS)

Идея алгоритма состоит в распределении вершин по уровням, характеризующим удалённость от первой вершины.

Определение 2: Очередь

Очередь – абстрактный тип данных с дисциплиной доступа к элементам «первый пришёл – первый вышел» (FIFO, англ. first in, first out).

Рассмотрим шаги алгоритма:

1. Всем вершинам графа присваивается значение не посещённой. Выбирается первая вершина и помечается как посещённая и заносится в очередь.
2. Посещается первая вершина из очереди (если она не помечена как посещённая). Все её соседние вершины заносятся в очередь и отмечаются как посещённые. После этого она удаляется из очереди. Порядок обхода вершин графа определяется порядком выхода вершин из очереди.
3. Повторяется шаг 2 до тех пор, пока очередь не станет пустой.
4. Если после завершения алгоритма не все вершины были рассмотрены, то необходимо запустить алгоритм от одной из нерассмотренных вершин.

Рассмотрим шаги алгоритма на графе из предыдущего примера.

1. Изначально все вершины отмечены как не посещённые, а очередь пуста.

Посещённость вершин							
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
0	0	0	0	0	0	0	0

Очередь				

2. Помещаем вершину v_1 в очередь и отмечаем её как посещённую.

Посещённость вершин							
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
1	0	0	0	0	0	0	0

Очередь				
v_1				

3. Помещаем смежные с v_1 вершины v_2, v_3, v_4, v_5 в очередь и отмечаем их как посещённые. Удаляем из очереди v_1.

Посещённость вершин							
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
1	1	1	1	1	0	0	0

Очередь				
v_2	v_3	v_4	v_5	

4. У вершин v_2, v_3, v_4 нет смежных не посещённых вершин, удаляем их из очереди и помещаем смежные с v_5 вершины v_6 и v_7 в очередь, отмечаем их как посещённые.

Посещённость вершин							
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
1	1	1	1	1	1	1	0

Очередь				
v_6	v_7			

5. Помещаем вершину смежную с v_6 вершину v_8 в очередь и отмечаем её как посещённую. Удаляем вершину v_6 из очереди.

Посещённость вершин							
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
1	1	1	1	1	1	1	1

Очередь				
v_7	v_8			

6. Все вершины отмечены как посещённые, удаляя оставшиеся вершины из очереди, запишем получившийся порядок обхода:

$$v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8.$$

В ходе выполнения алгоритмов DFS и BFS на выходе мы получаем лес – граф, различные компоненты связности которого являются деревьями. А значит эти алгоритмы можно использовать для исследования неориентированных графов на связность и выделения в ориентированных графах компонент слабой связности.

Для поиска компонент сильной связности используется алгоритм Косарайю (алгоритм Косараджу, Kosaraju's algorithm), в основе которого лежат два прохода DFS.

Алгоритм Косарайю для поиска компонент сильной связности

Рассмотрим шаги алгоритма:

1. Запускаем DFS на исходном графе G, запоминая времена выхода $\tau(v)$ для каждой вершины.
2. Транспонируем граф $G^T = (G)^T$, меняя направление дуг на противоположное.
3. Запускаем DFS на транспонированном графе G^T, выбирая в качестве стартовой непосещённую вершину с максимальным значением $\tau(v)$.
4. Полученные в шаге 3 подграфы и образуют компоненты сильной связности.

Рассмотрим шаги алгоритма на рисунке 2.

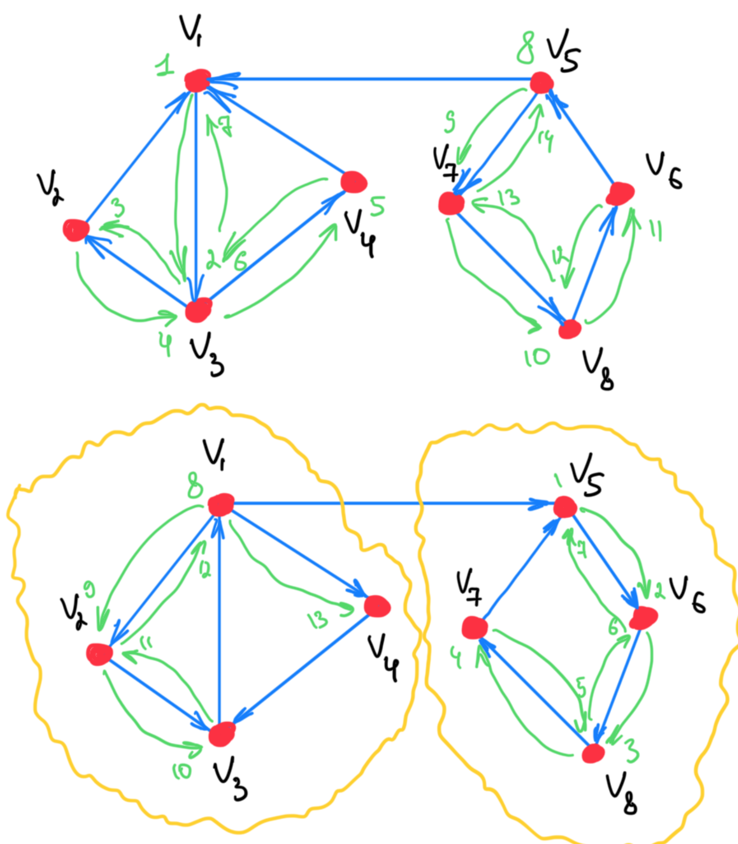

Рис. 2: Выделение компонент сильной связности

Конденсация и база графа

Введём обозначение $R(v)$ для множества вершин, достижимых из вершины v.

Определение 3: База графа

Подмножество вершин $\{v_1, v_2, \dots, v_k\} \subseteq V$ ориентированного графа $G = (V, E)$ называется его **базой**, если из вершин этого множества достижимы все остальные вершины и оно минимально по включению.

База существует в любом ориентированном графе. Любые две вершины базы не достижимы одна из другой.

Для построения базы используется конденсация графа, то есть такой граф G^c, в котором каждая сильно связная компонента исходного графа стянута в одну вершину.

Определение 4: Конденсация графа

Граф конденсации $G^c = (V^c, E^c)$ ориентированного графа $G = (V, E)$ – ориентированный граф с множеством вершин

$$V^c = \{V_1, V_2, \dots, V_k\},$$

где каждая V_i – множество вершин сильно связной компоненты орграфа G.
Вершины V_i и V_j графа конденсаций соединяются дугой, если в исходном ориентированном графе существовала дуга (v, u), где $v \in V_i$, $u \in V_j$.

При построении графа конденсаций можем выделить следующие шаги:

1. Определить для каждой вершины v множество достижимых из неё вершин $R(v)$.
2. Выделить наибольшие по включению классы вершин $V_1, V_2, \dots, V_k$, достижимых друг из друга. Эти классы будут вершинами графа G^c.
3. Построить множество дуг графа конденсации по определению.

Множество вершин, содержащих ровно одну вершину из множества новых вершин $\{V_1, V_2, \dots, V_k\}$, имеющих нулевую полустепень захода, образует базу.

2 Расстояния в графе

Определение 5: Взвешенный граф

Взешенным называется граф, каждому ребру которого поставлено в соответствие некоторое число – вес, иначе говоря задана функция $f : E \to \mathbb{R}$.

Определение 6: Длина пути

Длиной $l(v_1, v_k)$ пути

$$P = (v_1, \{v_1, v_2\}, v_2, \{v_2, v_3\}, v_3, \dots, v_{k-2}, \{v_{k-2}, v_{k-1}\}, v_{k-1}, \{v_{k-1}, v_k\}, v_k)$$

называется суммарный вес рёбер, входящих в путь.

$$l(v_1, v_k) = \sum_i \omega_i.$$

Определение 7: Расстояние между вершинами

Расстоянием $\rho(u, v)$ между двумя вершинами v и u называется длина кратчайшего пути, соединяющего эти вершины.

$$\rho(u, v) = \min_i l(u, v).$$

Определение 8: Экцентриситет вершины

Эксцентриситетом $\varepsilon(v)$ вершины v называют расстояние до максимально удаленной от нее вершины.

$$\varepsilon(v) = \max_i \rho(u_i, v).$$

Для графа, у которого не определен вес его ребер, расстояние определяется в виде числа ребер (иными словами вес каждого ребра считаем равным единице).

Определение 9: Радиус графа

Радиус $r(G)$ графа $G = (V, E)$ – минимальный эксцентриситет его вершин.

$$r(G) = \min_i \varepsilon_i.$$

Определение 10: Диаметр графа

Диаметр $d(G)$ графа $G = (V, E)$ – максимальный эксцентриситет его вершин.

$$d(G) = \max_i \varepsilon_i.$$

Определение 11: Центр графа

Центром $O(G)$ связного графа $G = (V, E)$ называют множество вершин, у которых эксцентриситет равен радиусу графа.

$$O(G) = \{v | v \in V, \varepsilon(v) = r(G)\}.$$

В качестве примера рассмотрим граф, изображенный на рисунке 3.

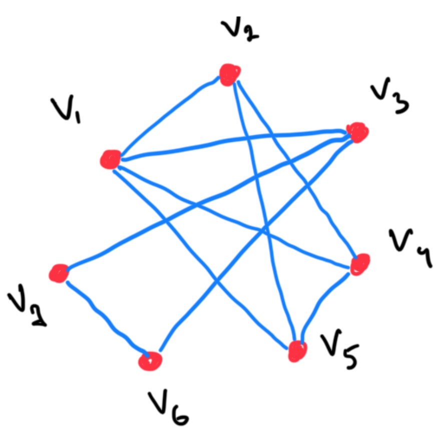

Рис. 3: Граф $G = (V, E)$

Приняв веса всех его рёбер равными единице, составим таблицу расстояний между вершинами и определим эксцентриситеты вершин.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	$\varepsilon(v_i)$
v_1	0	1	1	1	1	2	2	2
v_2	1	0	2	1	1	3	3	3
v_3	1	2	0	2	2	1	1	2
v_4	1	1	2	0	1	3	3	3
v_5	1	1	2	1	0	3	3	3
v_6	2	3	1	3	3	0	1	3
v_7	2	3	1	3	3	1	0	3

По найденным эксцентриситетам можно определить диаметр, радиус и центр графа.

$$d(G) = 3,$$

$$r(G) = 2,$$

$$O(G) = \{v_1, v_3\}.$$

Алгоритм Дейкстры

Алгоритм позволяет построить кратчайшие пути от корневой вершины до остальных вершин графа.

Алгоритм работает на как на неориентированных, так и на ориентированных графах (в этом случае учитывается направление) в том случае, если все веса графа неотрицательны.

Рассмотрим шаги алгоритма:

1. Корневой вершине приписывают метку $\mu_1 = 0$, а остальным вершинам метку ∞.

2. На i-м шаге выбирают вершину v_i, метка которой минимальна. Метку этой вершины называют постоянной.

3. Перебирают все смежные с v_i вершины, метка которых не является постоянной.

4. Если вершина v_j смежна с v_i и имеет метку μ_j, тогда метку вершины v_j изменяют по правилу

$$\mu_j = \min(\mu_i + \rho(v_i, v_j), \mu_j).$$

5. Алгоритм заканчивает работу, когда все вершины имеют постоянные метки.

Постоянную метку будем выделять прямоугольником, а в качестве индекса будем указывать вершину-предка. Так, запись $\boxed{1_{v_1}}$ говорит о том, что значение метки равно единице, а вершина-предок в дереве кратчайших путей – v_1.

Рассмотрим работу алгоритма на примере графа G, показанного на рисунке 4.

Пусть v_1 – корневая вершина, выполним первый шаг алгоритма

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Шаг 1	$\boxed{0}$	∞	∞	∞	∞	∞	∞

Вершина v_1 смежна с v_2, v_3, v_4 и v_5.

$$\mu_2 = \min(\mu_1 + \rho(v_1, v_2), \mu_2) = \min(0 + 1, \infty) = 1,$$

$$\mu_3 = \min(\mu_1 + \rho(v_1, v_3), \mu_3) = \min(0 + 2, \infty) = 2,$$

$$\mu_4 = \min(\mu_1 + \rho(v_1, v_4), \mu_4) = \min(0 + 3, \infty) = 3,$$

$$\mu_5 = \min(\mu_1 + \rho(v_1, v_5), \mu_5) = \min(0 + 2, \infty) = 2.$$

Минимальная метка у вершины v_2, она становится постоянной.

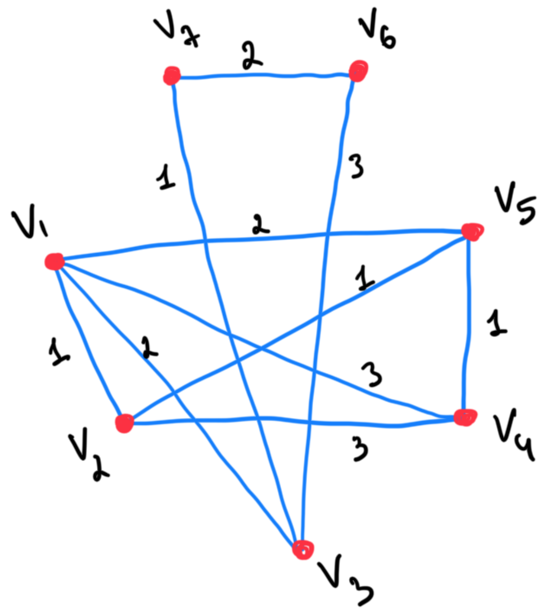

Рис. 4: Взвешенный граф

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Шаг 1	$\boxed{0}$	∞	∞	∞	∞	∞	∞
Шаг 2		$\boxed{1_{v_1}}$	2_{v_1}	3_{v_1}	2_{v_1}	∞	∞

Вершина v_2 смежна с v_4 и v_5.

$$\mu_4 = \min(\mu_2 + \rho(v_2, v_4), \mu_4) = \min(1 + 3, 3) = 3,$$

$$\mu_5 = \min(\mu_2 + \rho(v_2, v_5), \mu_5) = \min(1 + 1, 2) = 2.$$

Минимальная метка у вершины v_3, она становится постоянной.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Шаг 1	$\boxed{0}$	∞	∞	∞	∞	∞	∞
Шаг 2		$\boxed{1_{v_1}}$	2_{v_1}	3_{v_1}	2_{v_1}	∞	∞
Шаг 3			$\boxed{2_{v_1}}$	3_{v_1}	2_{v_1}	∞	∞

Вершина v_3 смежна с v_6 и v_7.

$$\mu_6 = \min(\mu_3 + \rho(v_3, v_6), \mu_6) = \min(2 + 3, \infty) = 5,$$

$$\mu_7 = \min(\mu_3 + \rho(v_3, v_7), \mu_7) = \min(2 + 1, \infty) = 3.$$

Минимальная метка у вершины v_5, она становится постоянной.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Шаг 1	$\boxed{0}$	∞	∞	∞	∞	∞	∞
Шаг 2		$\boxed{1_{v_1}}$	2_{v_1}	3_{v_1}	2_{v_1}	∞	∞
Шаг 3			$\boxed{2_{v_1}}$	3_{v_1}	2_{v_1}	∞	∞
Шаг 4				3_{v_1}	$\boxed{2_{v_1}}$	5_{v_3}	3_{v_3}

Вершина v_5 смежна с v_4.

$$\mu_4 = \min(\mu_5 + \rho(v_5, v_4), \mu_4) = \min(2 + 1, 3) = 3,$$

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Шаг 1	$\boxed{0}$	∞	∞	∞	∞	∞	∞
Шаг 2		$\boxed{1_{v_1}}$	2_{v_1}	3_{v_1}	2_{v_1}	∞	∞
Шаг 3			$\boxed{2_{v_1}}$	3_{v_1}	2_{v_1}	∞	∞
Шаг 4				3_{v_1}	$\boxed{2_{v_1}}$	5_{v_3}	3_{v_3}
Шаг 5				$\boxed{3_{v_1}}$		5_{v_3}	3_{v_3}

Вершина v_4 не смежна ни с одной вершиной с непостоянной меткой. Минимальная метка у вершины v_7, она становится постоянной.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Шаг 1	$\boxed{0}$	∞	∞	∞	∞	∞	∞
Шаг 2		$\boxed{1_{v_1}}$	2_{v_1}	3_{v_1}	2_{v_1}	∞	∞
Шаг 3			$\boxed{2_{v_1}}$	3_{v_1}	2_{v_1}	∞	∞
Шаг 4				3_{v_1}	$\boxed{2_{v_1}}$	5_{v_3}	3_{v_3}
Шаг 5				$\boxed{3_{v_1}}$		5_{v_3}	3_{v_3}
Шаг 6						5_{v_3}	$\boxed{3_{v_3}}$

Вершина v_7 смежна с v_6.

$$\mu_6 = \min(\mu_7 + \rho(v_7, v_6), \mu_6) = \min(3 + 2, 5) = 5.$$

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Шаг 1	$\boxed{0}$	∞	∞	∞	∞	∞	∞
Шаг 2		$\boxed{1_{v_1}}$	2_{v_1}	3_{v_1}	2_{v_1}	∞	∞
Шаг 3			$\boxed{2_{v_1}}$	3_{v_1}	2_{v_1}	∞	∞
Шаг 4				3_{v_1}	$\boxed{2_{v_1}}$	5_{v_3}	3_{v_3}
Шаг 5				$\boxed{3_{v_1}}$		5_{v_3}	3_{v_3}
Шаг 6						5_{v_3}	$\boxed{3_{v_3}}$
Шаг 7						$\boxed{5_{v_3}}$	

Таким образом, значения постоянных меток указывают на кратчайшее расстояние от корневой вершины, а двигаясь в обратном направлении по меткам от потомков к предкам, можно восстановить дерево кратчайших путей.

Для вершины v_7 постоянная метка равна 3_{v_3}, следовательно, кратчайший путь из v_1 в v_7 имеет длину 3. Так как для вершины v_7 предком является v_3, а для v_3 предком является v_1, кратчайший путь имеет вид

$$P = (v_1, \{v_1, v_3\}, v_3, \{v_3, v_7\}, v_7),$$
$$\rho(v_1, v_3) + \rho(v_3, v_7) = 2 + 1 = 3.$$

Алгоритм Флойда-Уоршелла

Определение 12: Ранг пути

Путь $v_{i0} \to v_{i1} \to \dots \to v_{im}$ длины m называют путём ранга k при $m > 1$, если k – наибольшее из чисел $i_1, \dots, i_{m-1}$, и путём ранга 0 при $m = 1$. Путь нулевой длины также считают путём ранга 0. Таким образом, **ранг пути** – максимальный номер вершины, в которую разрешено заходить по пути из v_i в v_j (исключая вершины v_i и v_j).

Обозначим через $C^{(k)}$ матрицу стоимостей прохождения между различными парами вершин по всем путям ранга, не превосходящего k. Элемент $c_{ij}^{(k)}$ содержит стоимость прохождения из вершины v_i в v_j по всем путям рангов $0, 1, \dots, k-1, k$.

Алгоритм работает на как на неориентированных, так и на ориентированных графах (в этом случае учитывается направление) в том случае, если граф не содержит циклов отрицательного веса.

Идея алгоритма заключается в сравнении стоимостей путей:

- из вершины v_i в вершину v_j по пути ранга не превосходящего $k-1$, т.е. минуя вершину v_k;
- из вершины v_i в вершину v_k, после чего по пути из вершины v_k в вершину v_j.

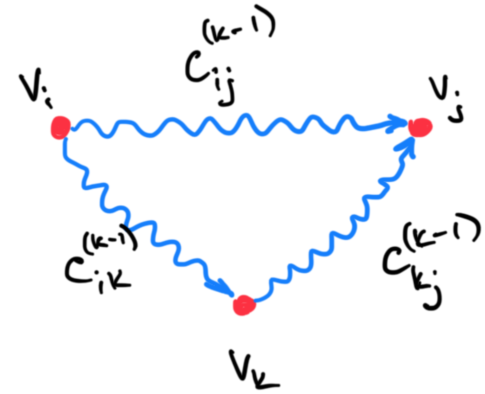

Рис. 5: Схема путей

Тогда матрицу стоимостей можно найти последовательно вычисляя $C^{(k)}$, $k=\overline{0,k}$ с помощью рекуррентной формулы

$$c_{ij}^{(k)} = \min\left(c_{ij}^{(k-1)}, c_{ik}^{(k-1)} + c_{kj}^{(k-1)}\right).$$

Рассмотрим шаги алгоритма:

1. Инициализируют матрицу стоимостей $C^{(0)}$ с помощью весов рёбер графа.
2. Рекурсивно вычисляют матрицу стоимостей $C^{(k)}$, $k=\overline{0,k}$.
3. Алгоритм заканчивает работу, ранг рассмотренных путей равен числу вершин $k=|V|$.

В случае, если необходимо не только знать кратчайшие расстояния, но и иметь возможность восстановить кратчайший путь, вводится вспомогательная матрица $D^{(k)}$ (см. пример).

Рассмотрим алгоритм на примере графа $G=(V,E)$ (рисунок 4).

1. При $k=0$ инициализируем матрицу стоимостей $C^{(0)}$ с помощью весов рёбер.

   Для восстановления кратчайших путей введём вспомогательную матрицу $D^{(k)}$ той же размерности, что и $C^{(k)}$.

2. В ячейку $d_{ij}^{(k)}$ на каждой итерации будем помещать значение k, если $c_{ij}^{(k)}$ изменилось.

$$C^{(0)} = \begin{pmatrix} 0 & 1 & 2 & 3 & 2 & \infty & \infty \\ 1 & 0 & \infty & 3 & 1 & \infty & \infty \\ 2 & \infty & 0 & \infty & \infty & 3 & 1 \\ 3 & 3 & \infty & 0 & 1 & \infty & \infty \\ 2 & 1 & \infty & 1 & 0 & \infty & \infty \\ \infty & \infty & 3 & \infty & \infty & 0 & 2 \\ \infty & \infty & 1 & \infty & \infty & 2 & 0 \end{pmatrix} \qquad D^{(0)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

- При $k=1$ пересчитаем коэффициенты матрицы
  $c_{23}^{(1)} = \min(c_{23}^{(0)}, c_{21}^{(0)} + c_{13}^{(0)}) = \min(\infty, 1+2) = 3,$
  $c_{32}^{(1)} = \min(c_{32}^{(0)}, c_{31}^{(0)} + c_{12}^{(0)}) = \min(\infty, 2+1) = 3,$

$c_{34}^{(1)} = \min(c_{34}^{(0)}, c_{31}^{(0)} + c_{14}^{(0)}) = \min(\infty, 2+3) = 5,$
$c_{43}^{(1)} = \min(c_{43}^{(0)}, c_{41}^{(0)} + c_{13}^{(0)}) = \min(\infty, 3+2) = 5,$
$c_{35}^{(1)} = \min(c_{35}^{(0)}, c_{31}^{(0)} + c_{15}^{(0)}) = \min(\infty, 2+2) = 4,$
$c_{53}^{(1)} = \min(c_{53}^{(0)}, c_{51}^{(0)} + c_{13}^{(0)}) = \min(\infty, 2+2) = 4,$

$$C^{(1)} = \begin{pmatrix} 0 & 1 & 2 & 3 & 2 & \infty & \infty \\ 1 & 0 & 3 & 3 & 1 & \infty & \infty \\ 2 & 3 & 0 & 5 & 4 & 3 & 1 \\ 3 & 3 & 5 & 0 & 1 & \infty & \infty \\ 2 & 1 & 4 & 1 & 0 & \infty & \infty \\ \infty & \infty & 3 & \infty & \infty & 0 & 2 \\ \infty & \infty & 1 & \infty & \infty & 2 & 0 \end{pmatrix} \qquad D^{(1)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

- При $k = 2$ коэффициенты матрицы не меняются

$$C^{(2)} = \begin{pmatrix} 0 & 1 & 2 & 3 & 2 & \infty & \infty \\ 1 & 0 & 3 & 3 & 1 & \infty & \infty \\ 2 & 3 & 0 & 5 & 4 & 3 & 1 \\ 3 & 3 & 5 & 0 & 1 & \infty & \infty \\ 2 & 1 & 4 & 1 & 0 & \infty & \infty \\ \infty & \infty & 3 & \infty & \infty & 0 & 2 \\ \infty & \infty & 1 & \infty & \infty & 2 & 0 \end{pmatrix} \qquad D^{(2)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

- При $k = 3$ пересчитаем коэффициенты матрицы

$c_{16}^{(3)} = \min(c_{16}^{(2)}, c_{13}^{(2)} + c_{36}^{(2)}) = \min(\infty, 2+3) = 5,$
$c_{61}^{(3)} = \min(c_{61}^{(2)}, c_{63}^{(2)} + c_{31}^{(2)}) = \min(\infty, 3+2) = 5,$
$c_{17}^{(3)} = \min(c_{17}^{(2)}, c_{13}^{(2)} + c_{37}^{(2)}) = \min(\infty, 2+1) = 3,$
$c_{71}^{(3)} = \min(c_{71}^{(2)}, c_{73}^{(2)} + c_{31}^{(2)}) = \min(\infty, 1+2) = 3,$
$c_{26}^{(3)} = \min(c_{26}^{(2)}, c_{23}^{(2)} + c_{36}^{(2)}) = \min(\infty, 3+3) = 6,$
$c_{62}^{(3)} = \min(c_{62}^{(2)}, c_{63}^{(2)} + c_{32}^{(2)}) = \min(\infty, 3+3) = 6,$
$c_{27}^{(3)} = \min(c_{27}^{(2)}, c_{23}^{(2)} + c_{37}^{(2)}) = \min(\infty, 3+1) = 4,$
$c_{72}^{(3)} = \min(c_{72}^{(2)}, c_{73}^{(2)} + c_{32}^{(2)}) = \min(\infty, 1+3) = 4,$
$c_{46}^{(3)} = \min(c_{46}^{(2)}, c_{43}^{(2)} + c_{36}^{(2)}) = \min(\infty, 5+3) = 8,$
$c_{64}^{(3)} = \min(c_{64}^{(2)}, c_{63}^{(2)} + c_{34}^{(2)}) = \min(\infty, 3+5) = 8,$
$c_{47}^{(3)} = \min(c_{47}^{(2)}, c_{43}^{(2)} + c_{37}^{(2)}) = \min(\infty, 5+1) = 6,$
$c_{74}^{(3)} = \min(c_{74}^{(2)}, c_{73}^{(2)} + c_{34}^{(2)}) = \min(\infty, 1+5) = 6,$
$c_{56}^{(3)} = \min(c_{56}^{(2)}, c_{53}^{(2)} + c_{36}^{(2)}) = \min(\infty, 4+3) = 7,$
$c_{65}^{(3)} = \min(c_{65}^{(2)}, c_{63}^{(2)} + c_{35}^{(2)}) = \min(\infty, 3+4) = 7,$
$c_{57}^{(3)} = \min(c_{57}^{(2)}, c_{53}^{(2)} + c_{37}^{(2)}) = \min(\infty, 4+1) = 5,$
$c_{75}^{(3)} = \min(c_{75}^{(2)}, c_{73}^{(2)} + c_{35}^{(2)}) = \min(\infty, 1+4) = 5,$

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

- При $k = 4$ коэффициенты матрицы не меняются

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

- При $k = 5$ пересчитаем коэффициенты матрицы
  $c^{(5)}_{24} = \min(c^{(4)}_{24}, c^{(4)}_{25} + c^{(4)}_{54}) = \min(3, 1 + 1) = 2,$
  $c^{(5)}_{42} = \min(c^{(4)}_{42}, c^{(4)}_{45} + c^{(4)}_{52}) = \min(3, 1 + 1) = 2,$

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

- При $k = 6$ коэффициенты матрицы не меняются

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

- При $k = 7$ коэффициенты матрицы не меняются

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

- Итоговая матрица кратчайших расстояний C и вспомогательная матрица D имеют вид

$$C = \begin{pmatrix} 0 & 1 & 2 & 3 & 2 & 5 & 3 \\ 1 & 0 & 3 & 2 & 1 & 6 & 4 \\ 2 & 3 & 0 & 5 & 4 & 3 & 1 \\ 3 & 2 & 5 & 0 & 1 & 8 & 6 \\ 2 & 1 & 4 & 1 & 0 & 7 & 5 \\ 5 & 6 & 3 & 8 & 7 & 0 & 2 \\ 3 & 4 & 1 & 6 & 5 & 2 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 1 & 5 & 0 & 3 & 3 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 5 & 1 & 0 & 0 & 3 & 3 \\ 0 & 0 & 1 & 0 & 0 & 3 & 3 \\ 3 & 3 & 0 & 3 & 3 & 0 & 0 \\ 3 & 3 & 0 & 3 & 3 & 0 & 0 \end{pmatrix}.$$

3. Восстановим путь из v_4 в v_6

$$v_4 \to \cdots \to v_6,$$

так как $d_{46} = 3 \neq 0$, добавляем в путь вершину v_3,

$$v_4 \to \cdots \to v_3 \to v_6,$$

так как $d_{43} = 1 \neq 0$, добавляем в путь вершину v_1, а так как $d_{41} = 0$, восстановление закончено, полученный путь

$$v_4 \to v_1 \to v_3 \to v_6.$$