Лекция 2.

Цепи и пути в графах

Путь в орграфе – последовательность дуг такая, что начало следующей дуги совпадает с концом предыдущей.

Цепь (маршрут) в неориентированном графе – последовательность ребер такая, что введением соответствующей ориентации её можно превратить в путь.

Определение 1. Длина пути (цепи) равна количеству дуг (ребер), содержащихся в нем, причем считаем столько раз, сколько дуга (ребро) встречается в пути (цепи).

Алгоритмы поиска цепей (путей), которые мы рассмотрим

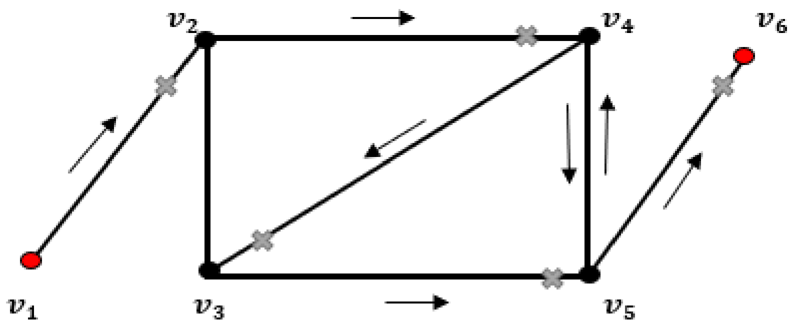
1. Алгоритм Тэрри поиска цепи (маршрута) в неориентированном графе.
2. Алгоритм ’фронта волны’ нахождения кратчайшего пути в орграфе.
3. Алгоритм нахождения минимального пути в нагруженном орграфе.

Алгоритм Тэрри

1. Помечаем направление, в котором проходим ребро.
2. По каждому ребру можно идти не более одного раза в каждом направлении, т.е. не более 2-х раз в разных направлениях.
3. Помечаем ребро q_i, по которому в вершину v_j зашли первый раз.
4. По помеченному ребру q_i можно идти в обратном направлении, если нет непомеченных ребер.

Используя этот алгоритм, можно легко выбраться из лабиринта. Помечая ребро, по которому зашли в вершину в первый раз, мы не имеем возможности идти по этому ребру в противоположном направлении, пока не будут пройдены остальные ребра, инцидентные данной вершине. Это и не дает возможности проходить в лабиринте несколько раз по одному коридору.

Пример 1. Поиск выхода из лабиринта. Маршрут из v_1 в v_6.

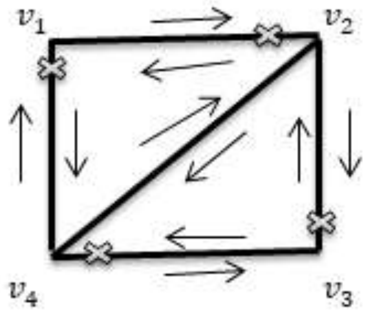


Из v_1 в v_6. $\boldsymbol{v_1 - v_2 - v_4 - v_3 - v_5 - v_4 - v_5 - v_6}$

Из v_5 мы не увидели v_6 и пошли в v_4, но алгоритм вернул нас в v_5, к вершине, имеющей непомеченные инцидентные ребра.

Пример 2. Задача о поливальной машине.

Полить все улицы и вернуться на базу v_1, проходя улицы по разу в каждом направлении.



$$\boldsymbol{v_1\text{-}\ v_2\text{-}\ v_3\text{-}\ v_4\text{-}\ v_2\text{-}\ v_4\text{-}\ v_1\text{-}\ v_4\text{-}\ v_3\text{-}\ v_2\text{-}\ v_1.}$$

Нахождение кратчайшего пути в орграфе

Определение 2. Путь из вершины v_i в v_j называется кратчайшим, если он содержит наименьшее количество дуг по сравнению со всеми путями из v_i в v_j.

Будем рассматривать орграф без петель. Найдем кратчайший путь из вершины v_1 в вершину v_t.

Алгоритм “фронта волны”

0) Помечаем вершину v_1 индексом 0; $v_1 \in w_0(v_1)$ – фронт волны нулевого уровня.

1) Помечаем вершины из $\Gamma w_0(v_1) = \Gamma v_1$ единицей. $\Gamma v_1 = w_1(v_1)$

......

k) Помечаем не помеченные ранее вершины из $\Gamma w_{k-1}(v_1)$ индексом k, они принадлежат $w_k(v_1) \subseteq \Gamma\big(w_{k-1}(v_1)\big)$.

Если через k шагов мы дошли до вершины v_t (до конца), то длина кратчайшего пути равна k. Если через $(n-1)$ шаг мы не дошли до v_t, то пути из v_1 в v_t не существует.

Предположим, на k шаге мы дошли до вершины v_t. Найдем все вершины кратчайшего пути, начиная с последней v_t: $v_1 = v_{i_0}, v_{i_1}, v_{i_2}, \ldots.., v_{i_k} = v_t$

1. $v_t = v_{i_k}$

2. $v_{i_{k-1}} \in w_{k-1}(v_1) \cap \Gamma^{-1} v_{i_k}$

3. $v_{i_{k-2}} \in w_{k-2}(v_1) \cap \Gamma^{-1} v_{i_{k-1}}$

…………………

$k.\ v_1 = v_{i_0} \in w_0(v_1) \cap \Gamma^{-1} v_{i_1}$

Пример 2. Задана матрица смежности орграфа

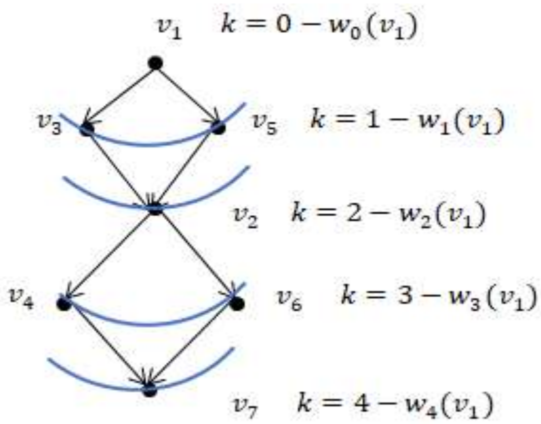
$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Γv_i – вершины, соответствующие единицам в i-й строке матрицы смежности A.

$\Gamma^{-1} v_i$ – вершины, соответствующие единицам в i-м столбце матрицы смежности A.

Найти кратчайшие пути из вершины v_1 в вершину v_7.

Последовательные шаги поиска кратчайшего пути изобразим на подграфе “фронтов волны”.



k =4, следовательно длина кратчайших путей равна 4.

Найдем вершины кратчайших путей, начиная с последней v_7.

1. $v_{i_4} = v_7$

2. $v_{i_3} \in (w_3(v_1) \cap \Gamma^{-1} v_7) = \{v_6, v_4\} \cap \{v_6, v_4\} = \{v_6, v_4\}$

3. $v_{i_2} \in (w_2(v_1) \cap \Gamma^{-1} v_4) = \{v_2\} \cap \{v_2\} = \{v_2\}$

   $v_{i_2} \in (w_2(v_1) \cap \Gamma^{-1} v_6) = \{v_2\} \cap \{v_2\} = \{v_2\}$

4. $v_{i_1} \in (w_1(v_1) \cap \Gamma^{-1}v_2) = \{v_3, v_5\} \cap \{v_3, v_5, v_7\} = \{v_3, v_5\}$

5. $v_{i_0} \in (w_0(v_1) \cap \Gamma^{-1}v_3) = \{v_1\} \cap \{v_1, v_6\} = \{v_1\}$

$v_{i_0} \in (w_0(v_1) \cap \Gamma^{-1}v_5) = \{v_1\} \cap \{v_1, v_3, v_6\} = \{v_1\}$

Кратчайшие пути (длина равна 4):

1. v_1, v_3, v_2, v_4, v_7.
2. v_1, v_3, v_2, v_6, v_7.
3. v_1, v_5, v_2, v_4, v_7.
4. v_1, v_5, v_2, v_6, v_7.

Минимальный путь в нагруженном графе

Определение 3. Нагруженным называется граф, в котором каждой дуге $< v_i, v_j >\in \Gamma$ (каждому ребру) ставится в соответствие число $l_{ij} \geq 0$, называемое весом или длинной дуги (ребра). $l_{ij} = l(v_i, v_j)$.

В некоторых задачах теории графов вводятся и отрицательные веса.

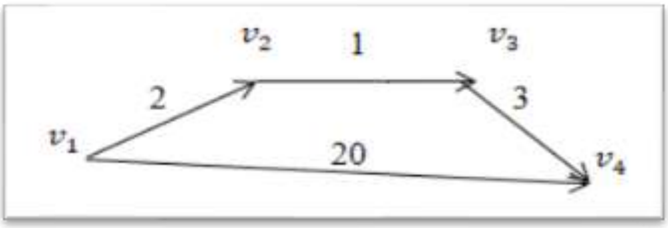
Определение 4. Длина пути (цепи) в нагруженном графе равна сумме длин дуг (ребер), входящих в этот путь (цепь).

$$L = \sum_{\substack{\text{по всем} \\ \text{дугам пути}}} l_{ij}$$

Определение 5. Матрица весов нагруженного графа – квадратная матрица порядка n (n – число вершин) с элементами $C = ||c_{ij}||$.

$$c_{ij} = \begin{cases} l_{ij}, \text{если } \exists \text{ дуга } < v_i, v_j > \\ \infty \text{ в противном случае} \end{cases}$$

Определение 6. Путь из v_i в v_j называется минимальным, если его длина наименьшая по сравнению со всеми путями из v_i в v_j.



Кратчайший путь $v_1 - v_4$ содержит одну дугу, её длина $L_1 = 20$.

Минимальный путь содержит три дуги, он не является кратчайшим:

$v_1 - v_2 - v_3 - v_4$. Его длина $L_2 = 2 + 1 + 3 = 6$.

Алгоритм поиска минимального пути в нагруженном графе

Алгоритм Форда-Беллмана итерационный. Максимальное число итераций $n -$ число вершин орграфа.

$\lambda_i^{(k)} -$ длина минимального пути из вершины v_1 в v_i, содержащего не более k дуг.

Найдем $\lambda_i^{(0)}, \ldots, \lambda_i^{(n-1)}$, $i = 1, \ldots, n$.

0) Положим $\lambda_i^{(0)} = \infty, \quad i = 2, \ldots, n$

$\lambda_1^{(j)} = 0, \quad j = 0, \ldots, n-1$. Длина минимального пути из v_1 в v_1 всегда равна нулю.

………..

$$k) \ \lambda_i^{(k)} = \min_{1 \le j \le n} (\lambda_j^{(k-1)} + c_{ji})$$

$\lambda_i^{(n-1)}$ – длина минимального пути из v_1 в v_i, , $i = 1, \ldots, n$.

Если две итерации полностью совпали, то следующие искать не надо, они не изменятся.

Пусть $\lambda_t^{(n-1)} = \lambda_t^{(k)}$ и на k-м шаге мы впервые нашли длину минимального пути из вершины v_1 в v_t. Найдем вершины минимального пути $v_1 = v_{i_0}, v_{i_1}, \ldots, v_{i_k} = v_t$, выписав соотношения для всех итераций, начиная с последней, полученной на k-м шаге, где мы впервые нашли длину минимального пути:

$$\boldsymbol{\lambda_{i_{k-1}}^{(k-1)} + c_{i_{k-1}i_k} = \lambda_{i_k}^{(k)}}$$

$$\boldsymbol{\lambda_{i_{k-2}}^{(k-2)} + c_{i_{k-2}i_{k-1}} = \lambda_{i_{k-1}}^{(k-1)}}$$

$$\ldots \ldots \ldots \ldots \ldots \ldots .$$

$$\boldsymbol{\lambda_{i_0}^{(0)} + c_{i_0 i_1} = \lambda_{i_1}^{(1)}}$$

Индексы по столбцам соответствуют номерам вершин минимального пути.

Может существовать несколько путей, тогда перебором находим все так же, как и в алгоритме «фронта волны».

Пример 3. **Поиск минимального пути.**

$$\boldsymbol{C} = \begin{pmatrix} \infty & \infty & \mathbf{1} & \mathbf{5} & \mathbf{4} & \mathbf{20} \\ \infty & \infty & \infty & \infty & \infty & \mathbf{2} \\ \infty & \mathbf{7} & \infty & \infty & \mathbf{2} & \infty \\ \infty & \mathbf{3} & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \mathbf{1} & \infty & \mathbf{8} \\ \infty & \infty & \infty & \infty & \mathbf{10} & \infty \end{pmatrix}$$

$\lambda_i^{(0)}$	$\lambda_i^{(1)}$	$\lambda_i^{(2)}$	$\lambda_i^{(3)}$	$\lambda_i^{(4)}$	$\lambda_i^{(5)}$
0	0	0	0	0	0 в v_1
∞	∞	8	8	7	7 в v_2
∞	1	1	1	1	1 в v_3
∞	5	5	4	4	4 в v_4
∞	4	3	3	3	3 в v_5
∞	20	12	10	10	9 в v_6

Последний столбец таблицы – длины минимальных путей из вершины v_1 во все достижимые вершины графа. Минимальный путь из вершины v_1 в v_6 единственный:

$$v_1 - v_3 - v_5 - v_4 - v_2 - v_6.$$

Восстановим последовательность вершин этого пути, выписав следующие соотношения:

$$\lambda_2^{(4)} + c_{26} = \lambda_6^{(5)}$$

$$\lambda_4^{(3)} + c_{42} = \lambda_2^{(4)}$$

$$\lambda_5^{(2)} + c_{54} = \lambda_4^{(3)}$$

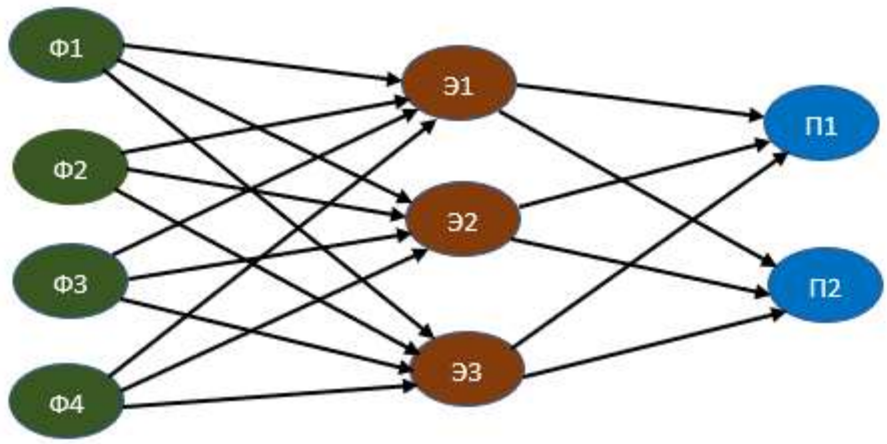
$$\lambda_3^{(1)} + c_{35} = \lambda_5^{(2)}$$

$$\lambda_1^{(0)} + c_{13} = \lambda_3^{(1)}$$

Индексы по столбцам соответствуют номерам вершин минимального пути.

Прикладная задача.
Нахождение минимального пути при перевозке урожая кукурузы.

Пусть четыре фермерских хозяйства выращивают кукурузу для продажи в другую страну (Иран). Перевозку осуществляют два порта. Но прежде, чем отправить на перевозку кукурузу сушат на трех элеваторах.



Упорядочим вершины Ф1, Ф2, Ф3, Ф4, Э1, Э2, Э3, П1, П2. Матрица длин дуг (расстояний между пунктами-вершинами) имеет вид:

$$C = \begin{pmatrix} \infty & \infty & \infty & \infty & 15 & 18 & 22 & \infty & \infty \\ \infty & \infty & \infty & \infty & 20 & 16 & 14 & \infty & \infty \\ \infty & \infty & \infty & \infty & 17 & 19 & 10 & \infty & \infty \\ \infty & \infty & \infty & \infty & 12 & 20 & 18 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 55 & 67 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 48 & 70 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 60 & 56 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

Рассмотрим задачи, решение которых позволит уменьшить транспортные расходы на перевозку кукурузы.

1. Найти минимальные расстояния для каждой фермы в один из портов через любой элеватор.
2. Выбрать одну или несколько ферм для закупки кукурузы, находящихся на минимальном расстоянии от какого-либо порта.

Задача 1. Найдем минимальное расстояние от фермы 1 до любого порта, используя алгоритм Форда-Беллмана. Составим таблицу итераций.

$\lambda_i^{(0)}$	$\lambda_i^{(1)}$	$\lambda_i^{(2)}$	**Номер вершины**
0	**0**	**0**	**1-Ф1**
∞	∞	∞	**2-Ф2**
∞	∞	∞	**3-Ф3**
∞	∞	∞	**4-Ф4**
∞	**15**	**15**	**5-Э1**
∞	**18**	**18**	**6-Э2**
∞	**22**	**22**	**7-Э3**
∞	∞	**66⇒П1**	**8-П1**
∞	∞	**76⇒П2**	**9-П2**

Минимальный путь от фермы Ф1 в порт П1 - 66 км.

$$\lambda_6^{(1)} + c_{68} = 18 + 48 = 66 = \lambda_8^{(2)}$$
$$\lambda_1^{(0)} + c_{16} = 0 + 18 = 18 = \lambda_3^{(1)}$$

Путь: Ф1-Э2-П1

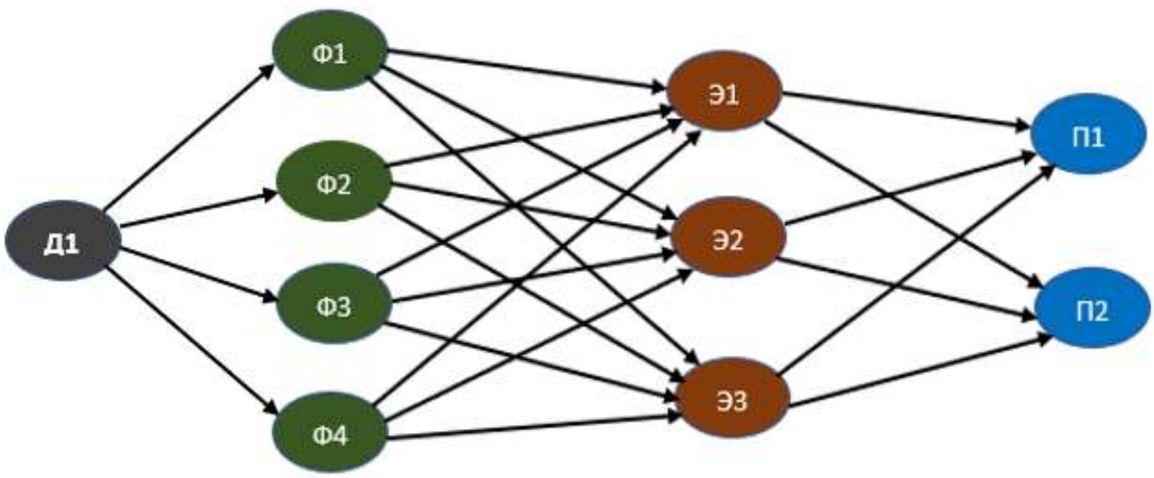
Аналогично можно вычислить минимальные пути из остальных ферм в порт.

Путь: Ф2-Э2-П1, его длина 64 км.

Путь: Ф3-Э3-П2, его длина 66 км.

Путь: Ф4-Э1-П1, его длина 67 км.

Задача 2. Чтобы выбрать одну ферму с самым коротким путем в порт через любой элеватор, можно вычислить таблицу итераций только один раз, а не четыре. Введем дополнительную вершину Д1. Из Д1 проведем дуги в Ф1, Ф2, Ф3, Ф4 весом нуль.



$$C_0 = \begin{pmatrix} \infty & 0 & 0 & 0 & 0 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 15 & 18 & 22 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 20 & 16 & 14 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 17 & 19 & 10 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 12 & 20 & 18 & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 55 & 67 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 48 & 70 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 60 & 56 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{pmatrix}$$

Составим таблицу итераций.

$\lambda_i^{(0)}$	$\lambda_i^{(1)}$	$\lambda_i^{(2)}$	$\lambda_i^{(3)}$	**Номер вершины**
0	**0**	**0**	**0**	**1-Д1**
∞	**0**	**0**	**0**	**2-Ф1**
∞	**0**	**0**	**0**	**3-Ф2**
∞	**0**	**0**	**0**	**4-Ф3**
∞	**0**	**0**	**0**	**5-Ф4**
∞	∞	**12**	**12**	**6-Э1**
∞	∞	**16**	**16**	**7-Э2**
∞	∞	**10**	**10**	**8-Э3**
∞	∞	∞	**64⇒П1**	**9-П1**
∞	∞	∞	**66⇒П2**	**10-П2**

Минимальный путь от фермы в порт 64 км (если искать один самый выгодный путь, т.е. покупать кукурузу у одного фермера)

Найдем минимальный путь до порта П1 и определим, из какой фермы он идет:

$\lambda_7^{(2)} + c_{79} = \lambda_9^{(3)}$

$\lambda_3^{(1)} + c_{37} = \lambda_7^{(2)}$

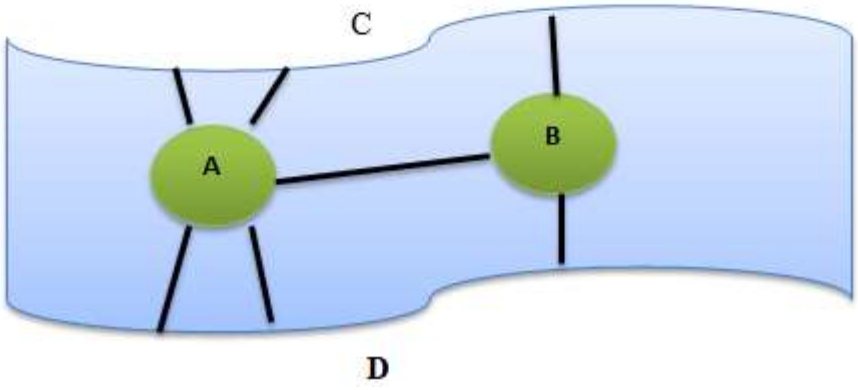
$\lambda_1^{(0)} + c_{13} = \lambda_3^{(1)}$

Путь: Ф2-Э2-П1. Выгоднее покупать кукурузу на ферме Ф2 и вести её через элеватор Э2 в порт П1.

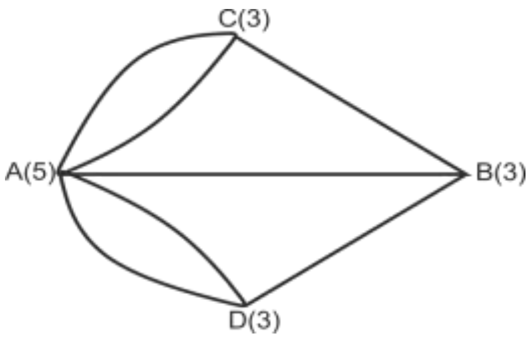
Эйлеровы и Гамильтоновы пути в графе.

Эйлеровы пути

В городе Кёнигсберге протекала река. На ней было два острова, соединенные мостами. Река Преголя.



Прогуливаясь по мостам, Эйлер подумал: можно ли обойти все мосты и ровно по одному разу? Нарисовал граф и, обобщив, доказал теорему.



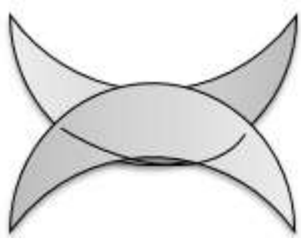
Определение 1. Эйлеров путь (цикл) – это путь (цикл), содержащий все ребра графа, причем ровно по одному разу.

Определение 2. Степень вершины – число инцидентных ей ребер (дуг).

Теорема 1. Эйлеров путь существует тогда и только тогда, когда количество вершин с нечетной степенью 0 или 2. Эйлеров цикл существует тогда и только тогда, когда вершин с нечетной степенью нет.

У графа, представленного на рисунке, четыре вершины и все – нечетной степени. Следовательно, обойти все мосты по одному разу нельзя.

Сабли Магомеда. Можно ли обвести контур по одному разу, не отрывая руки? (Да)

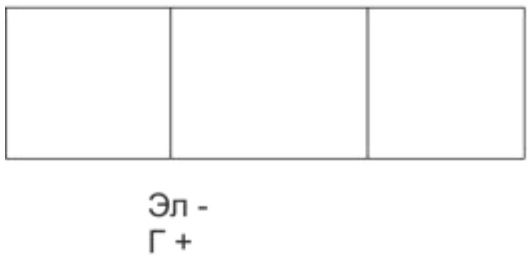


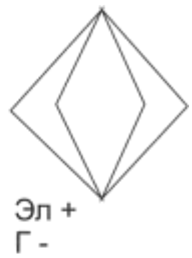
Гамильтоновы пути

Определение 3. Гамильтонов путь (цикл) – это путь (цикл), содержащий все вершины графа, причем ровно один раз. Если цикл, то кроме последней вершины.

Задача 1. Коммивояжер должен проехать все города побывав в каждом ровно один раз и пройдя минимальный путь. В этой задаче среди всех Гамильтоновых путей выбирают путь с минимальной длиной.

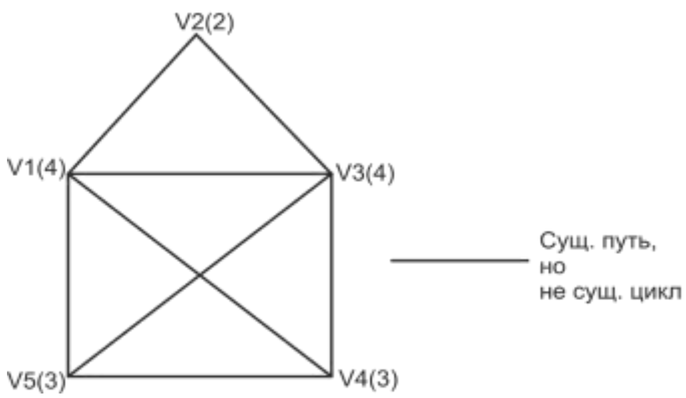
Примеры.





Существует Гамильтонов путь и цикл, но не существует Эйлеровых.

Существует Эйлеровы путь и цикл, но Гамильтоновых путей нет.



Существует Гамильтоновы путь и цикл, существует Эйлеров путь, но не существует Эйлерова цикла.

Центр графа. Эксцентриситет, радиус, диаметр

Прикладные задачи логистики. Социальный граф.

Дадим определения этих понятий. Определения справедливы как для неориентированного графа, так и для орграфа, и как для ненагруженного, так и нагруженного графа. Различие состоит только в определении расстояния между вершинами.

Определения 1-5.

Эксцентриситетом вершины называется расстояние до самой дальней вершины графа (ε).

Радиусом графа называется минимальный эксцентриситет среди всех вершин графа (r).

Диаметром графа — это наибольшее расстояние между всеми парами вершин графа (d)

Центр графа (центральная вершина графа) – вершина, чей эксцентриситет равен радиусу графа. Таких вершин может быть несколько.

Периферийная вершина графа – вершина, чей эксцентриситет равен диаметру графа. Таких вершин может быть несколько.

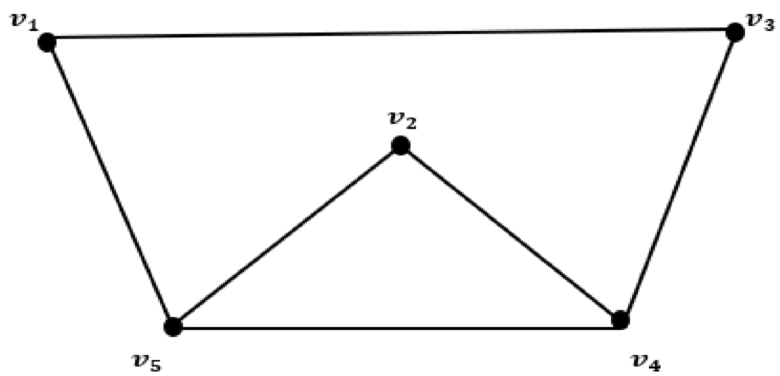


Таблица расстояний содержит величины кратчайших цепей между вершинами. Так как граф неориентированный, таблица симметрична относительно главной диагонали.

Эксцентриситет вершины v_i вычисляется по формуле

$$\varepsilon(v_i) = \max_{j=1,\dots,n} d(v_i, v_j),$$

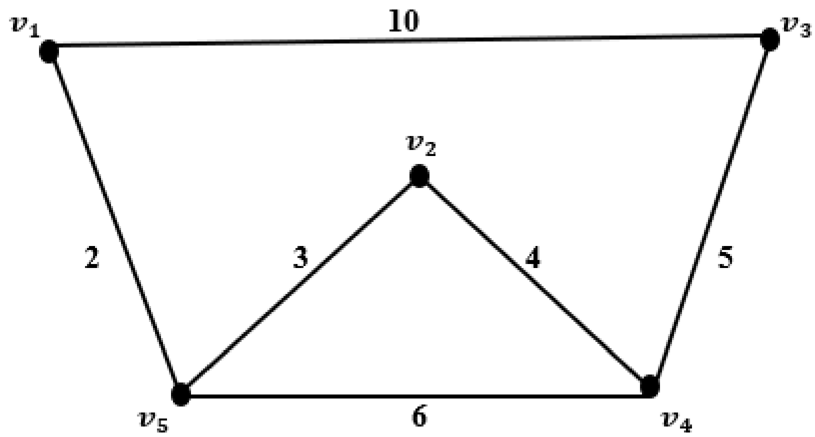
где $d(v_i, v_j)$ – расстояние между вершинами v_i и v_j.

Заполним таблицу

	v_1	v_2	v_3	v_4	v_5	ε
v_1	**0**	**2**	**1**	**2**	**1**	**2**
v_2	**2**	**0**	**2**	**1**	**1**	**2**
v_3	**1**	**2**	**0**	**1**	**2**	**2**
v_4	**2**	**1**	**1**	**0**	**1**	**2**
v_5	**1**	**1**	**2**	**1**	**0**	**2**

В данном примере **эксцентриситеты всех** вершин равны 2. Следовательно, $r = 2$ и все вершины центральные.

Если взять нагруженный граф, то расстояния и эксцентриситеты вершин будут различаться.



	v_1	v_2	v_3	v_4	v_5	ε
v_1	**0**	**5**	**10**	**8**	**2**	**10**
v_2	**5**	**0**	**9**	**4**	**3**	**9**
v_3	**10**	**9**	**0**	**5**	**11**	**11**
v_4	**8**	**4**	**5**	**0**	**6**	**8**
v_5	**2**	**3**	**11**	**6**	**0**	**11**

Радиус графа: $\mathrm{r} = \min\limits_{i=1,\dots,n} \varepsilon(v_i) = 8$ и центр графа – вершина v_4.

Диаметр графа: $\mathrm{d} = \max\limits_{i=1,\dots,n} \varepsilon(v_i) = 11.$

Для орграфа таблицы расстояний не будут симметричны, а в остальном вычисления такие же. Для построения таблицы расстояний удобно воспользоваться алгоритмом Флойда – поиска кратчайших путей в графе.

Центр графа широко используется в прикладных задачах, в частности в логистике. Центр графа – вершины, минимально удаленные от остальных вершин графа.

1. Если вершины – населенные пункты, то в центре графа удобно разместить жизненно важные объекты или склад с товарами. Если есть улицы с односторонним движением, граф будет ориентированный.
2. Граф – карта метро – можно получить центральные станции. Такую задачу решали для ненагруженного графа. Центральными оказались несколько станций-вершин на Кольцевой линии. Граф нагружали временем проезда между станциями. Тогда центр графа зависел от времени суток.

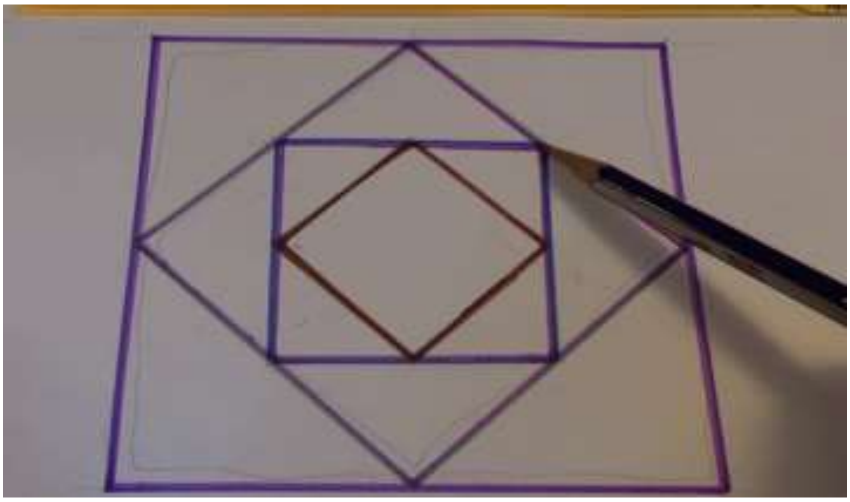
Рассмотренные понятия также используются **при анализе социальных сетей**.

Социальный граф – граф, вершины которого представлены социальными объектами, такими как пользовательские профили с различными атрибутами (например: имя, день рождения, родной город), сообщества, медиаконтент и так далее, а рёбра — социальными связями между ними.

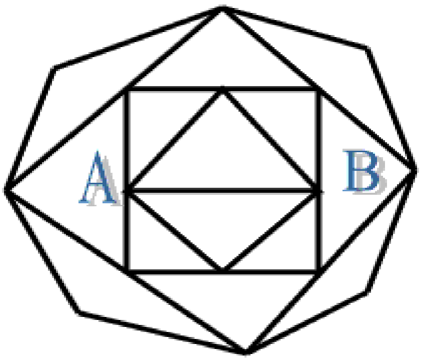
Дуги (ребра) показывают в каких отношениях состоят разные социальные объекты. Пользователь **Петя** находится в дружеских отношениях с пользователями **Аней** и **Катей**, при этом **Аня** и **Катя** не являются друзьями друг другу. Центр графа поможет найти наиболее влиятельных людей в коллективе.

Особенности социального графа характеризуются такими метриками, как: метрики взаимоотношений, метрики связей. С обширными связями можно стать не только центром социального графа, но и богатым!

Примеры головоломок «Обвести, не отрывая руки»



На следующем рисунке две вершины нечетной степени, следовательно, Эйлерова цикла не существует, но существует Эйлеров путь.



А это головоломка уже с 22 мостами.