

Санкт-Петербургский государственный электротехнический
университет имени В. И. Ленина "ЛЭТИ"

билеты

КОМБИНАТОРИКА И ТЕОРИЯ ГРАФОВ

Студент:

Группа:

Лектор:

Придчин В.Е.

2308

Зяблицева Л.

L^AT_EX

Санкт-Петербург
2024

1 Основные определения теории графов. Смежность и инцидентность вершин и ребер графа. Степени вершин в графе и орграфе. Теоремы о сумме степеней вершин в графе и орграфе. Матрицы смежности и инцидентности. Найти матрицы смежности и инцидентности указанного графа.

V – множество вершин.

E – множество пар вида $(u, v) : u, v \in V$ (множество ребер).

Опр: Графом – называют совокупность 2-ух множеств непустого множества E

$$E = \{(u, v) : u, v \in V\}, \quad G(V, E) - \text{обозначение графа}$$

Опр: Петля – пара вида (v, v) в множестве E .

Опр: Кратные ребра – одинаковые пары в множестве E . Количество кратных ребер - кратность ребра.

Существуют следующие виды графов:

1. Псевдограф – в графе могут быть и *кратные ребра*, и *петли*.
2. Мультиграф – в графе есть *кратные ребра*, но нет *петель*.
3. Простой граф – отсутствуют и *кратные ребра* и *петли*.

Опр: Ориентированный граф (орграф) – граф с ориентированными ребрами.

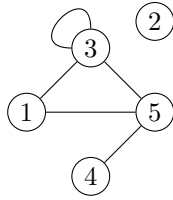
Опр: Если $e = (u, v)$ – ребро неориентированного графа, то u, v – концы ребра.

Опр: Если $e = (u, v)$ – ребро (дуга) ориентированного графа, то u – начало ребра, v – конец ребра.

Опр: a, b смежные $\Leftrightarrow e = (a, b)$.

Опр: u, v инцидентны ребру $e \Leftrightarrow e = (u, v)$.

Опр: Степенью вершины v неориентированного графа называется количество ребер инцидентных данной вершине, $\delta(v)$ (петлю считают два раза).



$$\begin{aligned} \delta(1) &= 2 & \delta(4) &= 1 \\ \delta(2) &= 0 & \delta(5) &= 3 \\ \delta(3) &= 4 & \Rightarrow \text{sum} &= 10 \end{aligned}$$

Теорема: Сумма степеней вершин неориентированного графа равна удвоенному числу ребер

$$\sum_{u \in V} \delta(u) = 2r, \text{ где } r - \text{число ребер}$$

Док-во: Теорема справедлива, так как вклад каждого ребра равен двум. \square

Опр: Если степень вершины равна нулю, то вершина *изолированная*, $\delta(v) = 0$.

Опр: Если степень вершины равна единице, то вершина *висячая*, $\delta(v) = 1$

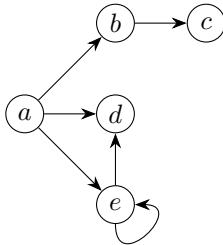
Опр: Полустепенью исхода(захода) вершина v ориентированного графа называют количество ребер исходящих(заходящих) в данную вершину.

$\delta^-(v)$ – полустепень исхода

$\delta^+(v)$ – полустепень захода

Теорема: Для орграфа справедливо равенство

$$\sum_{u \in V} \delta^-(u) = \sum_{u \in V} \delta^+(u) = r, \text{ где } r - \text{число ребер}$$



$$\begin{aligned} \delta^-(a) &= 3 & \delta^+(a) &= 0 \\ \delta^-(b) &= 1 & \delta^+(b) &= 1 \\ \delta^-(c) &= 0 & \delta^+(c) &= 1 \\ \delta^-(d) &= 0 & \delta^+(d) &= 2 \\ \delta^-(e) &= 2 & \delta^+(e) &= 2 \end{aligned}$$

$$\sum \delta^-(v) = 6; \quad \sum \delta^+(v) = 6$$

Опр: Матрицей смежности графа(орграфа) называют квадратную матрицу размерностью n , где $n = |v|$ (мощность множества вершин), в котором $a_{ij} = k$, где k - число ребер (v_i, v_j)

Опр: Пусть $G(V, E)$ – неориентированный граф. Матрицей инцидентности неориентированного графа называется матрица B размером $n * r$, $|v| = n, |E| = r$, где каждый элемент матрицы:

$$b_{ij} = \begin{cases} 1, & \text{если } v_i \text{ инцидентно ребру } e_j \\ 0, & \text{иначе} \end{cases}$$

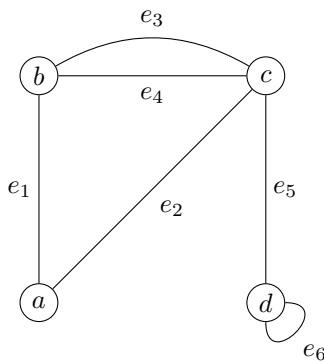
Такая матрица будет симметричной.

Опр: Пусть $G(V, E)$ – ориентированный граф. Матрицей инцидентности ориентированного графа называется матрица B размером $n * r$, $|v| = n, |E| = r$, где каждый элемент матрицы:

$$b_{ij} = \begin{cases} -1, & \text{если ребро } e_j \text{ выходит из } v_i \\ 1, & \text{если ребро } e_j \text{ входит в } v_i \\ 0, & \text{иначе} \end{cases}$$

Если есть петля, то на соответствующее место ставят любое число.

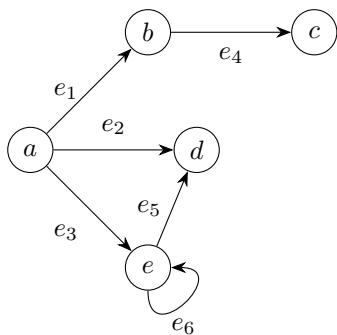
Поиск матрицы смежности A и инцидентности B для неориентированного графа:



$$A = \begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix}$$

$$B = \begin{matrix} & \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix}$$

Поиск матрицы смежности A и инцидентности B для ориентированного графа:

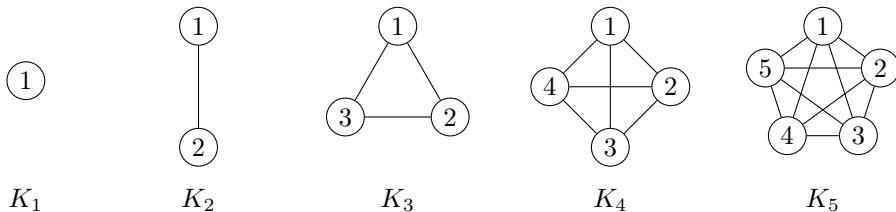


$$A = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{matrix}$$

$$B = \begin{matrix} & \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{pmatrix} \end{matrix}$$

2 Полные и двудольные графы. Число ребер в полном графе с n вершинами и в полном двудольном графе (вывод формул).

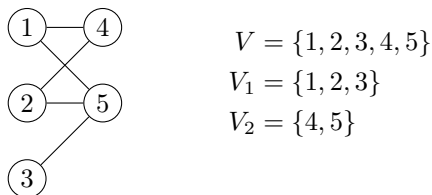
Опр: *Полный граф* – простой неориентированный граф у которого любые две вершины смежны.



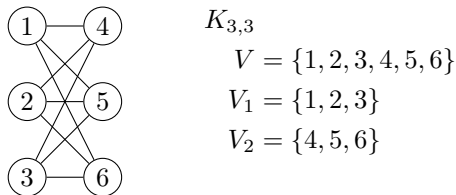
Количество полных ребер в графе $K_n = C_n^2 = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$.

Опр: *Двудольный граф* – граф, если его множество вершин V можно разделить на подмножество V_1 и V_2 такое что: $V_1 \cap V_2 = \emptyset$
 $V_1 \cup V_2 = V$

Смежными *обязаны* быть только вершины из разных долей графа.



Опр: *Полный двудольный граф* – каждая вершина одной доли соединяется с другой. K_{t_1, t_2} , где t_1, t_2 – количество вершин в долях графа.



В полном двудольном графе содержится $t_1 \cdot t_2$ ребер.

3 Изоморфизм и гомеоморфизм графов. Примеры изоморфных и гомеоморфных графов. Способы проверки изоморфизма графов. Инварианты графа. Дополнение графа, проверка изоморфизма графов с помощью дополнений. Выяснить, являются ли графы G_1 и G_2 изоморфными, гомеоморфными.

Опр: *Изоморфизм графов* – биективное отображение $\varphi : V_1 \rightarrow V_2$, такое что:

$$(\forall u, v \in V)((u, v) \in E_1) \Leftrightarrow (\varphi(u), \varphi(v)) \in E_2$$

Изоморфные графы обозначаются: $G_1 \cong G_2$

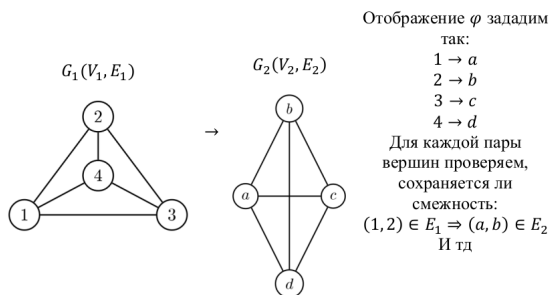
Изоморфные объекты *не различимы* с точки зрения математики. Это экземпляры одного и того же математического объекта. Изоморфные объекты имеют одинаковое число элементов и свойств.

Изоморфизм графов можно определить:

1. По определению (найдя биективное отображение множества вершин, сохраняющее смежность)
2. Перерисовав один из графов так, чтобы изображение совпало с другим графом
3. Сравнив матрицы смежности графов

Теорема: Графы изоморфны тогда и только тогда, когда матрицу смежности одного из них можно получить из матрицы смежности другого путём одновременной перестановки местами i -ой и j -ой строк и столбцов.

Пример изоморфных графов:



Опр: *Инвариантом изоморфизма графов* называется некоторое (обычно числовое) значение или упорядоченный набор значений, характеризующий структуру графа и не зависящий от способа его задания.

Инвариантами графа являются:

1. Количество вершин
2. Количество ребер
3. Набор степеней вершин (упорядоченный)
4. Определитель матрицы смежности
5. Количество компонент связности
6. Хроматическое число и т.д.

Пусть $G(V, E)$ – простой неориентированный граф.

Опр: Дополнением графа G называется граф $\overline{G}(\overline{V}, \overline{E})$, у которого $\overline{V} = V$, в множестве \overline{E} содержатся ребра полного графа, которых нет в множестве E .

Теорема: Простые графы G и H изоморфны тогда и только тогда, когда изоморфны их дополнения.

$$G \cong H \Leftrightarrow \overline{G} \cong \overline{H}$$

Опр: Граф является самодополнительным, если он изоморфен своему дополнению.

Понятие изоморфизма можно дать и для произвольных графов.

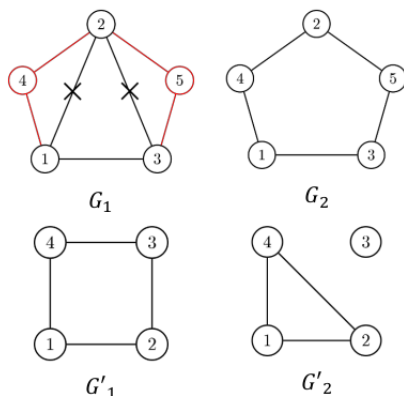
Опр: Граф $G_1(V_1, E_1)$ и $G_2(V_2, E_2)$ *изоморфны*, если \exists биективные отображения $\varphi : V_1 \rightarrow V_2$ и $h : E_1 \rightarrow E_2$ такие, что:

$$e = (u, v) \in E_1 \Leftrightarrow h(e) = (\varphi(u), \varphi(v)) \in E_2$$

Гомеоморфизм графов

Опр: Операция *подразбиения ребра* (u, v) состоит в удалении этого ребра, добавлении вершины ω и рёбер (u, ω) и (ω, v) .

Опр: Граф G_2 называется *подразбиением* графа G_1 , если его можно получить из графа G_1 путём последовательного применения операции подразбиения рёбер (сам граф тоже является своим подразбиением).



Графы G_1 и G_2 на рисунке гомеоморфны, потому что их подразбиения изоморфны.

Покажем, что графы G'_1 и G'_2 не гомеоморфны.

При любой операции подразбиения ребер появляются вершины степени 2, у остальных вершин степень не меняется. В графе G'_2 есть вершина степени 0, она останется такой при любом количестве операций подразбиения ребер. В графе G'_1 все вершины имеют степень 2. При любом количестве операций подразбиения ребер появятся вершины степени 2, у имеющихся вершин степень не изменится. Это значит, что любые подразбиения графов не являются изоморфными, поэтому графы G'_1 и G'_2 на рисунке не гомеоморфны.

4 Маршруты, цепи, циклы в графе. Метрические характеристики графа. Найти эксцентриситет вершины указанного графа, радиус, диаметр графа, центральные и периферийные вершины указанного графа.

Опр: *Маршрут* (путь) в графе – это конечная чередующаяся последовательность смежных вершин и ребер, соединяющих эти вершины.

Маршрут – последовательность $v_1 e_1 v_2 e_2 \dots e_{s-1} v_s e_s v_{s+1} \dots v_k$, в которой чередуются все вершины и ребра. $e_s = (v_s, v_{s+1})$.

Опр: *Длина маршрута* – количество ребер в маршруте.

Опр: *Нуль-маршрут* – последовательность из одной вершины v .

Опр: *Замкнутый маршрут (циклический)* – последовательность, в которой совпадает начальная вершина с конечной. Иначе – *незамкнутый (открытый)*.

Опр: *Цепь* – незамкнутый маршрут, где все *ребра* попарно различны.

Опр: *Простая цепь* – цепь, где все *вершины* попарно различны.

Опр: *Цикл* – циклический маршрут, где все *ребра* попарно различны.

Опр: *Простой цикл* – цикл, где все *вершины*, кроме первой и последней попарно различны.

Теорема: Из любого цикла можно выделить простой цикл с той же начальной вершиной.

Док-во: Пусть \exists цикл $v_1 e_1 v_2 e_2 \dots v_s e_s \dots v_j e_j \dots v_n e_n v_1$. Если все вершины, кроме первой и последней, различны, то этот цикл простой.

Допустим в цикле есть совпадающие вершины (не первая и не последняя) $v_s = v_j$. Тогда удаляем часть маршрута между этими вершинами.

$$v_1 e_1 v_2 e_2 \dots v_s e_s \dots v_j e_j \dots v_n e_n v_1 \Rightarrow v_1 e_1 v_2 e_2 \dots v_s e_j \dots v_n e_n v_1$$

Так как $v_s = v_j$, то полученная последовательность является маршрутом. Продолжая получим простой цикл. \square

Теорема: Из любой цепи можно выделить простую цепь с теми же начальными и конечными вершинами (док-во аналогично).

Опр: *Связанные вершины в графе* – \exists маршрут с началом в первой вершине и концом на второй.

Опр: *Связный граф* – две любые вершины являются *связанными*.

Пусть $G(V, E)$ – связный неориентированный граф ($v_i, v_j \in V$).

Опр: *Расстояние между вершинами*, $d(v_i, v_j)$ – длина кратчайшего маршрута между этими вершинами.

Опр: *Эксцентриситет вершины графа* – число, максимальное из расстояний между этой вершиной и всеми остальными вершинами.

$$\varepsilon(v_i) = \max d(v_i, v_j), v_j \in V, i \neq j$$

Опр: *Диаметр графа*, $D(G)$ – максимальный из всех эксцентриситетов вершин данного графа.

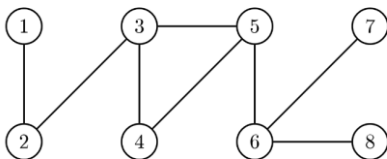
Опр: *Радиус графа*, $R(G)$ – минимальный из всех эксцентриситетов вершин данного графа.

Опр: *Периферическая вершина* – ее эксцентриситет равен диаметру.

Опр: *Центральная вершина* – ее эксцентриситет равен радиусу.

Опр: *Центр графа* – множество всех центральных вершин.

Пример: Найти эксцентриситеты вершин, диаметр и радиус графа, периферийные и центральные вершины.



$$\begin{aligned} \varepsilon(1) = 5, \varepsilon(2) = 4, \varepsilon(3) = 3, \varepsilon(4) = 3, \\ \varepsilon(5) = 3, \varepsilon(6) = 4, \varepsilon(7) = 5, \varepsilon(8) = 5. \\ D(G) = 5, R(G) = 3. \end{aligned}$$

Периферийные: 1, 7, 8
Центральные: 3, 4, 5

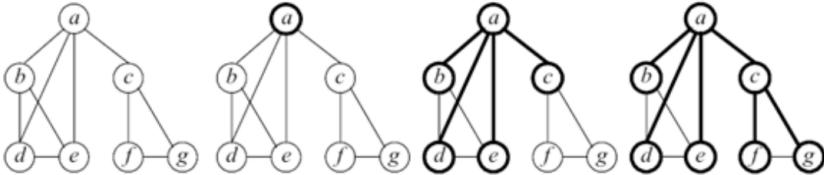
5 Алгоритмы обхода графа в глубину и ширину.

Опр: Обход графа (поиск на графе) — это процесс систематического просмотра всех ребер или вершин графа с целью отыскания ребер или вершин, удовлетворяющих некоторому условию.

Опр: Поиск в ширину (BFS) нужен для нахождения расстояний между вершин в связном графе. Алгоритм по принципу напоминает "пожар".

Алгоритм поиска в ширину:

1. Все вершины окрашиваются в белый цвет.
2. Выбирается первая вершина, раскрашивается в черный цвет и заносится в очередь.
3. Посещается первая вершина из очереди. Все смежные с ней белые вершины раскрашиваются в черный цвет и заносятся в очередь. После этого эта вершина удаляется из очереди.
4. Повторяется шаг 3 до тех пор пока очередь не пуста.



Сложность поиска в ширину при нематричном представлении графа равна $O(n + m)$, так как рассматриваются все n вершин и m ребер. Использование матрицы смежности приводит к оценке $O(n^2)$.

Если граф не является связным, то нужно добавить шаг 5, в котором написать, что нужно повторять 2 шаг до тех пор, пока в графе не останется белых вершин.

Если сначала присвоить первой вершине значение 0, потом смежным с ней вершинам $0+1=1$, и так далее, то в итоге для каждой вершины будет найдено расстояние от первой вершины.

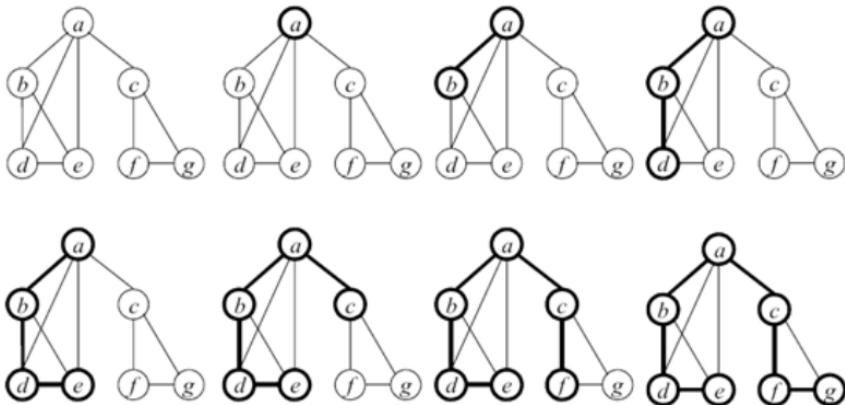
Опр: Поиск в глубину (DFS) исследует конкретную ветку в графе и только потом переходит к другой (если они останутся нерассмотренными).

Вершины графа могут быть раскрашены в три цвета: белый цвет означает, что в вершине еще не были, серый – что в вершине были, но еще вернемся, черный – что были и больше не рассматриваем.

Алгоритм поиска в глубину:

1. Всем вершинам графа присваиваем белый цвет.
2. Выбираем первую вершину и раскрашиваем в серый цвет.
3. Для последней раскрашенной в серый цвет вершины выбираем белую смежную ей вершину (если такая вершина есть), раскрашиваем ее в серый цвет и переходим к шагу 3.
4. Повторяем шаг 3 до тех пор, пока все вершины не будут раскрашены в черный цвет.

Если для рассматриваемой вершины белых смежных с ней вершин нет, то рассматриваемую вершину раскрашиваем в черный цвет и переходим к шагу 3.



Временная сложность алгоритма зависит от представления графа. Если применена матрица смежности, то временная сложность равна $O(n^2)$, а если нематричное представление – $O(n + m)$: рассматриваются все вершины и все ребра.

6 Прямое произведение множеств, мощность прямого произведения конечных множеств. Бинарные отношения между множествами A и B . Граф бинарного отношения Матрица бинарного отношения, область определения и значений бинарного отношения. Отношение, обратное к данному отношению. Найти отношение, изобразить его граф, найти матрицу.

Пусть A и B – не пустые множества.

Опр: *Прямым (декартовым) произведением множеств A и B называют множество всевозможных упорядоченных пар $(a, b), a \in A, b \in B$.*

$$A \times B = \{ (a, b) \mid a \in A, b \in B \}$$

Пример:

$$\begin{aligned} A &= \{ 1, 2 \}, B = \{ a, b, c \} \\ A \times B &= \{ (1, a), (1, b), (1, c), (2, a), (2, b), (2, c) \} \\ |A| &= n, |B| = m, |A \times B| = n \cdot m \end{aligned}$$

Если $A = B$, то пишут $A \times A = A^2$.

Опр: *Кортеж длины s – упорядоченный набор (a_1, a_2, \dots, a_s) , где $a \in A_i$.*

$$\begin{aligned} A_1 \times A_2 \times \dots \times A_s &= \{ (a_1, a_2, \dots, a_s), a_i \in A_i, i \in \overline{1, s} \} \\ |A_i| &= t_i \end{aligned}$$

$$\text{Тогда: } |A_1 \times A_2 \times \dots \times A_s| = t_1 \cdot t_2 \cdot \dots \cdot t_s$$

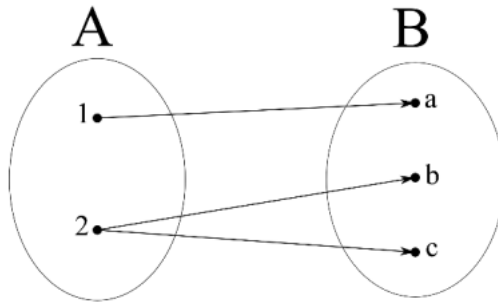
$$\text{Если: } A_1 = A_2 = \dots = A_s = A, \text{ то пишут } A_1 \times A_2 \times \dots \times A_s = A^s$$

Опр: *Бинарным отношением между множествами A и B называется любое подмножество их прямого произведения. Бинарное отношение принято записывать большими буквами или малыми буквами:*

$$\begin{aligned} A &= \{ 1, 2 \}, B = \{ a, b, c \} \\ R &= \{ (1, a), (2, b), (2, c) \} \\ f &= \{ (1, b), (2, a) \} \end{aligned}$$

Если $(a, b) \in R$, то говорят, что элементы a и b связаны отношением R или находятся в отношении R . Обозначают: aRb .

Сам граф бинарного отношения будет иметь вид:



Пусть бинарное отношение $R \subset A \times B, |A| = n, |B| = m$.

Опр: Матрицей бинарного отношения R называется матрица $C(R) \in M_{n \times m}$, элементы которой находятся по правилу:

$$c_{ij} = \begin{cases} 1, & \text{если } a_i R b_j \\ 0, & \text{иначе} \end{cases}.$$

Пример:

$$\begin{aligned} A &= \{1, 2\}, B = \{a, b, c\} \\ R &= \{(1, a), (2, b), (2, c)\} \\ C(R) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} \end{aligned}$$

Опр: Областью определения бинарного отношения R называется множество первых элементов пар бинарного отношения.

$$D(R) = \{x \in A \mid (\exists y \in B)(xRy)\}$$

Опр: Областью значений бинарного отношения R называется множество вторых элементов пар бинарного отношения.

$$E(R) = \{y \in B \mid (\exists x \in A)(xRy)\}$$

Пример:

$$\begin{aligned}A &= \{ 1, 2 \}, \quad B = \{ a, b, c \} \\R &= \{ (1, b), (2, c), (2, a), (2, b) \} \\D(R) &= \{ 1, 2 \}, \quad E(R) = \{ a, b, c \}\end{aligned}$$

Опр: Обратное бинарное отношение к R (R^{-1}) называется им, если:

$$R^{-1} = \{ (y, x) \mid (x, y) \in R \}$$

Пример:

$$\begin{aligned}R &= \{ (1, b), (2, c), (2, a), (2, b) \} \\C(R) &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \\R^{-1} &= \{ (b, 1), (c, 2), (a, 2), (b, 2) \} \\C(R) &= \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}\end{aligned}$$

Заметим, что матрица $C(R^{-1})$ является транспонированной матрицей $C(R)$.

7 Свойства бинарных отношений: рефлексивность, антирефлексивность, симметричность, антисимметричность, асимметричность, транзитивность. Свойства матриц и графов таких отношений, число таких отношений, заданных на n -элементном множестве.

Будем рассматривать бинарные отношения на множестве A^2 в этом случае говорят что бинарное отношение задано на множестве A .

Опр: Бинарное отношение R , заданное на множестве A называется *рефлексивным*, если для любого $a \in A$ верно aRa .

$$R - \text{рефлексивно} \iff (\forall a \in A) aRa$$

Если R рефлексивное отношение, то квадратная матрица размера n на главной диагонали состоит только из 1. В графе рефлексивное бинарное отношение в каждой вершине имеет *петлю*.

Опр: Бинарное отношение R , заданное на множестве A называется *антирефлексивным*, если для любого $a \in A$ неверно aRa .

$$R - \text{антирефлексивно} \iff (\forall a \in A) \overline{aRa}$$

Если R антирефлексивное отношение, то квадратная матрица размера n на главной диагонали состоит только из 0. В графе антирефлексивное бинарное отношение в каждой вершине не имеет *петель* вообще.

Опр: Бинарное отношение R , заданное на множестве A называется *симметричным* если для любого $a, b \in A$ верно $aRb \Rightarrow bRa$.

$$R - \text{симметрично} \iff (\forall a, b \in A) aRb \Rightarrow bRa$$

$a \text{ может быть } = b$

Матрица симметричного бинарного отношения симметрична относительно главной диагонали. Граф симметричного бинарного отношения *как правило* неориентированный.

Опр: Бинарное отношение R , заданное на множестве A называется *асимметричным*, если для любого $a, b \in A$ верно aRb , то неверно bRa (\overline{bRa}).

$$R - \text{асимметрично} \iff (\forall a, b \in A) aRb \Rightarrow \overline{bRa}$$

В матрице бинарного отношения элементы симметричные главной диаго-

нали *различны*. В графе если есть ребро ab , то нет ребра ba .

Опр: Бинарное отношение R , заданное на множестве A называется *антисимметричным*, если для любого $a, b \in A$, $aRb \wedge bRa \Rightarrow a = b$

$$R - \text{антисимметрично} \iff (\forall a, b \in A) (aRb \wedge bRa) \Rightarrow a = b$$

Элементы матрицы антисимметричного бинарного отношения относительно главной диагонали *различны*. На главной диагонали могут быть 1. В графе при антисимметричном бинарном отношении могут быть *петли*

Опр: Бинарное отношение R , заданное на множестве A называется *транзитивным*, если для любого $a, b, c \in A$, $(aRb \wedge bRc) \Rightarrow aRc$.

$$R - \text{транзитивно} \iff (\forall a, b, c \in A) (aRb \wedge bRc) \Rightarrow aRc$$

Если отношение обладает каким-нибудь свойством, нужно доказать это, если не обладает, то нужно привести контрпример.

Упражнение 1. Найти число рефлексивных отношений n -элементного множества

Упражнение 2. Найти число антирефлексивных отношений n -элементного множества

Упражнение 3. Найти число симметричных отношений n -элементного множества

Упражнение 4. Найти число антисимметричных отношений n -элементного множества

Упражнение 5. Найти число асимметричных отношений n -элементного множества

8 Отношение эквивалентности: определение, примеры. Классы эквивалентности, полная система представителей.

Фактор-множество множества по отношению эквивалентности. Алгоритм выделения классов эквивалентности по графу отношения. Проверить, является ли указанное отношение отношением эквивалентности.

Опр: Бинарное отношение R , заданное на множестве A , называется *отношением эквивалентности*, если оно рефлексивно, симметрично и транзитивно.

Пример:

$$\begin{aligned} A & - \text{множество граждан России} \\ A & = \{ x \mid x - \text{Гражданин РФ} \} \\ R & = \{ (x, y) \mid x, y \in A; x, y - \text{родились в одном месяце} \} \\ R & - \text{рефлексивно, симметрично, транзитивно} \Rightarrow \\ & \Rightarrow R - \text{отношение эквивалентности} \end{aligned}$$

Пусть R – отношение эквивалентности на множестве A и элемент $a \in A$.

Опр: Классом эквивалентности отношения эквивалентности R , порождённым элементом a (обозначается \bar{a}), называется множество всех элементов множества A , которые находятся в отношении R с элементом a .

$$\bar{a} = \{ x \in A \mid xRa \}$$

Опр: Любой элемент класса эквивалентности называется *представителем этого класса*.

Опр: *Полной системой представителей классов эквивалентности* называется множество представителей всех классов, взятых по одному и только по одному из каждого класса эквивалентности.

Опр: Пусть A – непустое множество. Фактор-множеством множества A по отношению эквивалентности R называется множество всех классов эквивалентности.

$$A/R = \{ \bar{a} \mid a \in A \}.$$

Алгоритм выделения классов эквивалентности по графу отношения:

1. Всем вершинам графа приписываем значение 0: $\text{color}(v) := 0$
2. $k := 0$ (количество классов эквивалентности)
3. Рассматриваем вершины графа

Если есть вершина, у которой $\text{color}(v) = 0$, то $k := k + 1$, этой вершине и всем, смежным с ней, присваиваем значение k : $\text{color}(v) := k$, выводим эти вершины с их цветом.

Если вершин, у которых $\text{color}(v) = 0$, нет, то выводим значение k – число классов эквивалентности и stop.

Результатом работы алгоритма является число k – количество классов эквивалентности; каждой вершине графа присвоен номер класса эквивалентности.

Временная сложность алгоритма зависит от представления графа. Если применена матрица смежности, то временная сложность равна $O(n^2)$, а если нематричное представление — $O(n + n) = O(2^n)$: рассматриваются все вершины и часть ребер.

9 Разбиение множества. Доказать теорему о связи фактор-множества множества по отношению эквивалентности и разбиением множества.

Опр: Разбиением множества A называется такое семейство его непустых подмножеств, что их объединение совпадает с множеством A , а пересечение двух различных подмножеств является пустым множеством.

Пример:

$A = \{1, 2, 3, 4, 5\}$, Множество подмножеств множества A
 $\{\{1, 2, 3\}, \{2, 3\}, \{4, 5\}\}$ не является разбиением множества A , так как элементы 2,3 принадлежат одновременно двум подмножествам. Множество подмножеств множества A $\{\{1\}, \{2, 3\}, \{4, 5\}\}$ является разбиением множества A по определению.

Теорема: Пусть R - отношение эквивалентности на множестве A . Фактор-множество множества A по R задает разбиение этого множества. (подмножествами разбиения являются классы эквивалентности).

$$A/R = \{\bar{a} \mid a \in A\}$$

Док-во: Докажем, что классы эквивалентности по отношению R являются подмножествами разбиения множества.

1. Докажем, что каждый класс эквивалентности не является пустым:

$$\bar{a} = \{x \in A \mid xRa\}$$

$$R - \text{рефлексивно} \Rightarrow aRa \Rightarrow a \in \bar{a}$$

2. Докажем: если элемент $c \in \bar{a}$ и $c \in \bar{b}$, то эти классы совпадают: $\bar{a} = \bar{b}$.

Это означает, что различные классы не пересекаются

$$c \in \bar{a} \Rightarrow cRa = aRc$$

$$c \in \bar{b} \Rightarrow cRb = bRc$$

$$aRb \Rightarrow bRa$$

Это означает, что если какой-то элемент принадлежит двум классам эквивалентности, то элементы, порождающие эти классы, находятся в этом отношении.

Докажем теперь, что $\bar{a} = \bar{b}$. Для этого докажем два включения.

1. $\bar{a} \subset \bar{b}$

Докажем: $(\forall x)(x \in \bar{a} \Rightarrow x \in \bar{b})$

$$x \in \bar{a} \Rightarrow xRa \wedge aRb \Rightarrow xRb \Rightarrow x \in \bar{b}$$

2. $\bar{b} \subset \bar{a}$

Докажем: $(\forall x)(x \in \bar{b} \Rightarrow x \in \bar{a})$

$$x \in \bar{b} \Rightarrow xRb \wedge bRa \Rightarrow xRa \Rightarrow x \in \bar{a}$$

Докажем, что $\cup \bar{a} = A$. Так как $(\forall a \in A)a \in \bar{a}$, то $A \subset \cup \bar{a}$, любой класс эквивалентности по определению является подмножеством множества A , поэтому и объединение классов является подмножеством A . По определению получили, что фактор множество является разбиением множества a . □

Из этой теоремы следует следующее:

1. $(\forall a \in A)a \in \bar{a}$
2. $(\forall a, b \in A)(a \in \bar{b} \Leftrightarrow \bar{a} = \bar{b})$
3. $(\forall a, b \in A)(a \notin \bar{b} \Leftrightarrow \bar{a} \cap \bar{b} = \emptyset)$
4. $(\forall a \in A) \cup \bar{a} = A$.

10 Отношение нестрогого и строгого, линейного порядка: определение, важнейшие примеры. Определить, является ли отношение отношением порядка.

Пусть A – множество, R – отношение на множестве A , т.е. $R \subseteq A^2$.

Опр: Отношение R называется отношением *нестрогого частичного порядка* на множестве A , если R : рефлексивное, антисимметричное, транзитивное. (Отношение $a \leq b$)

Опр: Если R – порядок на множестве A и $a, b \in R$ или aRb , то элементы a и b называются *сравнимыми*.

Порядок называют частичным порядком, так как не обязательно все элементы являются сравнимыми.

Опр: Отношение R называется отношением *строгого частичного порядка* на множестве A , если оно является антирефлексивным, асимметричным и транзитивным.

В *строгом частичном порядке* можно ограничиться двумя свойствами: антирефлексивность, транзитивность. (доказывается через предположение отсутствия асимметрии, а это противоречит антирефлексивности).

Опр: Порядок R называется линейным порядком, если выполняются условия:

$$(\forall a, b \in A)(a = b \vee aRb \vee bRa).$$

Другими словами, все различные элементы в линейном порядке обязательно сравнимы. Поэтому в случае, когда порядок линейный, слово частичный опускается.

На множестве Z отношение “меньше или равно” является нестрогим линейным порядком.

11 Частично упорядоченные множества. Минимальные элементы, линейно упорядоченные множества. Диаграмма Хассе. Построить диаграмму Хассе указанного множества.

Опр: Множество M с заданным на нем отношением частичного порядка называется *частично упорядоченным*. (для записи частично упорядоченных множеств часто употребляют \leq)

Пусть в множестве M задана частичная упорядоченность. Элементы a и b называются *сравнимыми*, если $a \leq b$ или $b \leq a$. Далеко не всегда два элемента из M обязаны быть сравнимыми – именно по этой причине говорят о "частичной упорядоченности".

Опр: *Минимальным элементом частично упорядоченного множества* называется элемент, для которого не существует элемента, меньшего его. Минимальных элементов может быть несколько.

Опр: Частично упорядоченное множество, в котором любые два элемента сравнимы, называется *линейно упорядоченным множеством* или *цепью*.

Опр: *Диаграмма Хассе* – граф частично упорядоченного множества. Не совпадает с графом отношения (нет некоторых ребер).

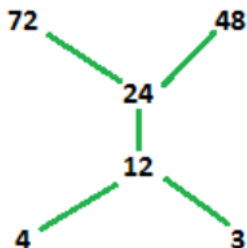
Однако не любой ориентированный граф является представлением частично упорядоченного множества.

Чтобы ориентированный граф представлял частично упорядоченное множество, необходимо и достаточно чтобы в нем не было циклов. В математической литературе частично упорядоченные множества часто изображаются в виде неориентированных графов, при этом подразумевается, что предшествующие элементы расположены ниже последующих. Поэтому, если в этих схемах правильно заменить ребра на ориентированные, то все они окажутся направленными снизу вверх (иногда наоборот сверху вниз).

Пример диаграммы Хассе для множества $R = \{ 3, 4, 12, 24, 48, 72 \mid \vdots \}$

Отношение состоит из следующих пар. $R = (3,12), (3,24), (3,48), (3,72), (4,12), (4,24), (4,48), (4,72), (12,24), (12,48), (12,72), (24,48), (24,72), (3,3), (4,4), (12,12), (24,24), (48,48), (72,72)$

Диаграмма Хассе будет иметь вид:



На приведенной выше диаграмме элементы 3 и 4 находятся на одном уровне, поскольку они не связаны друг с другом и меньше других элементов в наборе. Следующий последующий элемент для 3 и 4 — это 12, т. е. 12 делится и на 3, и на 4. Тогда 24 делится на 3, 4 и 12. Следовательно, оно помещается выше 12. 24 делит и 48, и 72, но 48 — не делит 72. Следовательно, 48 и 72 не соединяются.

На нашей диаграмме мы можем видеть транзитивность по мере увеличения уровня.

12 Алгоритм топологической сортировки. Применить алгоритм топологической сортировки.

Если для двух вершин v и u есть маршрут из v в u , но нет маршрута из u в v , то говорят, что v предшествует u , а u - последующее для v .

Алгоритм располагает вершины в последовательность таким образом, чтобы если в построенной последовательности вершина v находится правее вершины u , то u - последующее для v в исходном графе или же v и u не сравнимы в исходном бинарном отношении.

При работе алгоритма могут получиться различные последовательности.

Алгоритм работает на графах, в которых нет циклов, содержащих более одного ребра. (то есть в графе могут быть циклы вида vev , это значит, что вершина может иметь петлю).

Алгоритмы, которые рассмотрены ниже, работают на ациклических графах. Если же в графах есть петли, а при удалении петель они становятся ациклическими, то предварительно перед применением алгоритма их нужно удалить.

Алгоритм топологической сортировки:

1. $i = 1$
2. В графе находим вершину, в которую не заходит ни одно ребро. Присваиваем ей номер i . Удаляем эту вершину и инцидентные ей рёбра.
3. $i = i + 1$
4. Переходим к шагу 2 до тех пор, пока не пронумеруем все вершины.

Выше графический способ

Алгоритм Кана (формализация графического метода)

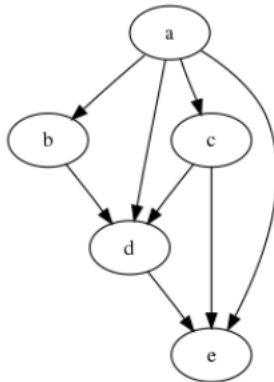
Пусть дан ациклический ориентированный простой граф. Для каждой вершины графа v находим $A(v)$ - множество таких вершин u графа, что в графе есть ребро (u, v) . То есть $A(v)$ — множество всех вершин, из которых есть ребро в вершину v .

1. Для всех вершин находим $A(v)$
2. $i = 1$
3. В графе находим вершину, для которой $A(v) = \emptyset$. Присваиваем этой вершине номер i . Удаляем эту вершину из всех множеств $A(v)$.
4. $i = i + 1$
5. Переходим к шагу 3 до тех пор, пока не пронумеруем все вершины.

Опишем матричный способ:

1. Рассмотрим матрицу смежности A графа. Найдём в этой матрице нулевые столбцы. Вершины, соответствующие этим столбцам, нумеруем. Вычёркиваем столбцы и строки, соответствующие этим вершинам.
2. Переходим к шагу 1 до тех пор, пока не пронумеруем все вершины.

Пример:

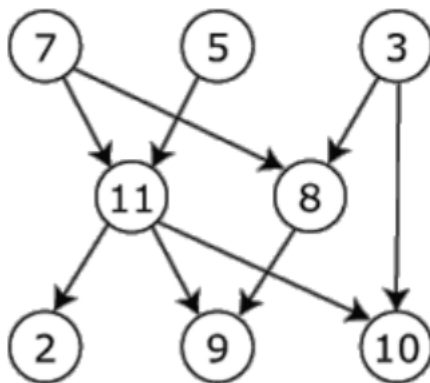


Применим алгоритм Кана к графу:

iter	i	$A(a)$	$A(b)$	$A(c)$	$A(d)$	$A(e)$	set
0		\emptyset	$\{a\}$	$\{a\}$	$\{a, b, c\}$	$\{a, c, d\}$	\emptyset
1	1	\emptyset	\emptyset	\emptyset	$\{b, c\}$	$\{c, d\}$	a
2	2	\emptyset	\emptyset	\emptyset	\emptyset	$\{d\}$	a, b
3	3	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	a, b, c
4	4	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	a, b, c, d
5	5	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	a, b, c, d, e

Результат работы алгоритма: a, b, c, d, e .

Пример:



Для графа существует несколько согласованных последовательностей его вершин, которые могут быть получены при помощи топологической сортировки, например: 7, 5, 3, 11, 8, 2, 9, 10; 3, 7, 5, 8, 11, 10, 9, 2.

Видно, что в последовательности могут быть переставлены любые две стоящие рядом вершины, которые не входят в отношение частичного порядка (несравнимы)

13 Связность в неориентированном графе. Теорема о разложении графа в объединение связных подграфов. Компоненты связности графа. Алгоритм нахождения связных компонент графа. Найти компоненты связности указанного графа.

Опр: Две вершины в графе называются *связанными*, если существует маршрут с началом в первой вершине и концом во второй.

Опр: Граф называется *связным*, если любые две его вершины являются связанными.

Теорема: В неориентированном графе отношение связности на множестве вершин является отношением эквивалентности.

$$V, \rho = \{ (v_1, v_2), v_1, v_2 \in V, v_1 \text{ и } v_2 - \text{связанные} \}$$

1. Рефлексивность – \exists нуль-маршрут $(\forall v \in V, vrv)$
2. Симметричность – \exists маршрут $ue_1 \dots e_s v (urv) + \exists$ маршрут $ve_s \dots e_1 u (vru)$
3. Транзитивность – $\forall u, v, w \in V \text{ } urv \wedge vrw \Rightarrow urw$. \exists маршрут $u \dots v$ и $v \dots w \Rightarrow \exists$ маршрут $u \dots w$

Опр: Пусть $G(V, E)$ – граф, то граф $G(V_1, E_1)$ – подграф графа G , если $V_1 \subset V$ и $E_1 \subset E$.

Теорема: Любой неориентированный граф распадается в объединение своих связных подграфов.

Док-во: Доказано, что в неориентированном графе отношение связности на множестве вершин является отношением эквивалентности.

Известно, что отношение эквивалентности разбивает множество на непересекающиеся подмножества - классы эквивалентности, поэтому всё множество вершин разбивается на попарно непересекающиеся подмножества $V_1, V_2, \dots, V_s, V = V_1 \cup V_2 \cup \dots \cup V_s$.

В каждом таком подмножестве V_i вершины являются связанными. Вершины из разных подмножеств не являются связанными. Это, в частности, означает, что в графе нет ребер, инцидентных вершинам из разных компонент связности. Это и доказывает справедливость утверждения. \square

Связные подграфы также называются *связными компонентами графа*.

Задача 1. Доказать, что любой конечный граф имеет чётное количество вершин нечётной степени.

Задача 2. Доказать, что если в графе только две вершины имеют нечётную степень, то они связанные.

Док-во: Докажем, что две вершины, которые имеют нечетную степень, принадлежат одной компоненте связности. Предположим противное. Пусть эти вершины принадлежат разным компонентам связности. Тогда в подграфах, содержащих эти вершины, содержится только одна вершина нечётной степени, что противоречит задаче 1.

Поэтому эти вершины принадлежат одной компоненте связности, тогда они являются связанными. \square

Для нахождения количество связных компонент графа существует алгоритм: Применяем алгоритм обхода графа в ширину.

1. Всем вершинам графа приписываем значение 0: $\text{color}(v) := 0$
2. $k := 0$ (количество компонент связности)
3. Рассматриваем вершины графа. Если есть вершина, у которых $\text{color}(v) = 0$, то $k := k + 1$, присваиваем этой вершине $\text{color}(v) := k$, заносим вершину в очередь.
4. Посещается первая вершина из очереди. Всем смежным с ней вершинам, у которых $\text{color}(v) = 0$, присваиваем $\text{color}(v) := k$ и заносим в очередь. После этого первая вершина в очереди удаляется из очереди.
5. Переходим к шагу 4 до тех пор, пока очередь не пуста.
6. Переходим к шагу 3 до тех пор, пока есть вершины, у которых $\text{color}(v) = 0$.
7. Если вершин, у которых $\text{color}(v) = 0$, нет, то выводим значение k – число компонент связности и *stop*.

Результатом работы алгоритма является число k – количество компонент связности; каждой вершине графа присвоен номер компоненты связности

Иначе говоря: количество запусков алгоритмов в ширину для обхода всего графа = количеству компонент связности.

14 Реберная и вершинная двусвязность графа. Блоки графа, граф блоков – точек сочленения. Найти компоненты реберной и вершинной двусвязности, граф блоков – точек сочленения графа.

Часто при решении прикладных задач теории графов важно, чтобы граф был “как можно более связным”, то есть при удалении какого-то числа ребер или вершин он также бы оставался связным. Поэтому кроме понятий связности существуют также понятия двусвязности, k -связности.

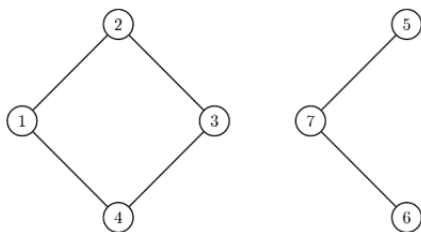
Опр: Две вершины в графе называются реберно двусвязными, если существует два маршрута с началом в первой вершине и концом во второй, в которых нет общих ребер (общие вершины могут быть).

Опр: Граф называется реберно двусвязным, если любые две вершины являются реберно двусвязными.

Теорема: В неориентированном графе отношение реберной двусвязности на множестве вершин является отношением эквивалентности.

Опр: Компонентами реберной двусвязности называют его подграфы, множества вершин которых – классы эквивалентности реберной двусвязности, а множества ребер – множества ребер, соединяющих вершины соответствующих классов эквивалентности.

У графа две компоненты связности: $V_1 = \{1, 2, 3, 4\}$ и $V_2 = \{5, 6, 7\}$:



Все вершины первой компоненты реберно двусвязны, поэтому эта компонента связности является и компонентой реберной двусвязности.

Во второй компоненте никакие две вершины не являются реберно двусвязными, поэтому каждая из них является отдельной компонентой реберной двусвязности.

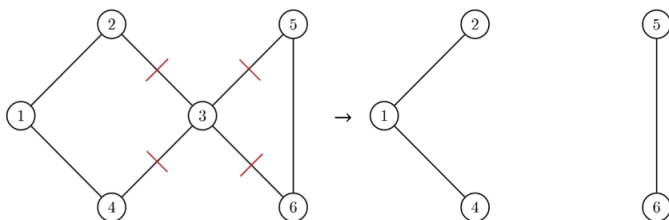
Объединение компонент реберной двусвязности не совпадает с графом в случае, когда в графе есть мосты.

Теорема: Компоненты реберной двусвязности графа $G(V, E)$ – это компоненты связности в графе $G(V, E')$, где E' получено из E удалением всех мостов.

То есть для того, чтобы найти компоненты реберной двусвязности в графе нужно найти все мосты, удалить их из множества ребер графа и в полученном графе найти все компоненты связности.

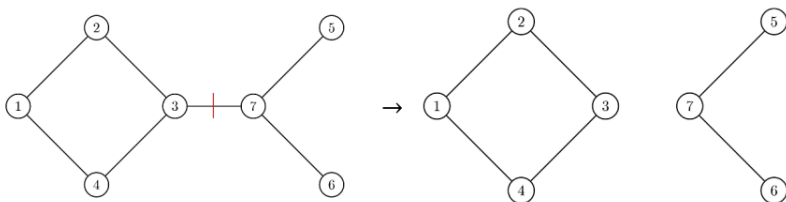
15 Шарниры и мосты в графе. Алгоритм нахождения мостов в графе. Применить его для графа.

Опр: Вершина в графе называется *разделяющей вершиной* (или *точкой сочленения*, или *шарниром*), если её удаление вместе с рёбрами, инцидентными ей, приводит к увеличению числа компонент связности.



В данном графе вершина 3 является *шарниром*.

Опр: *Мостом* в графе называется ребро, удаление которого приводит к увеличению числа компонент связности.

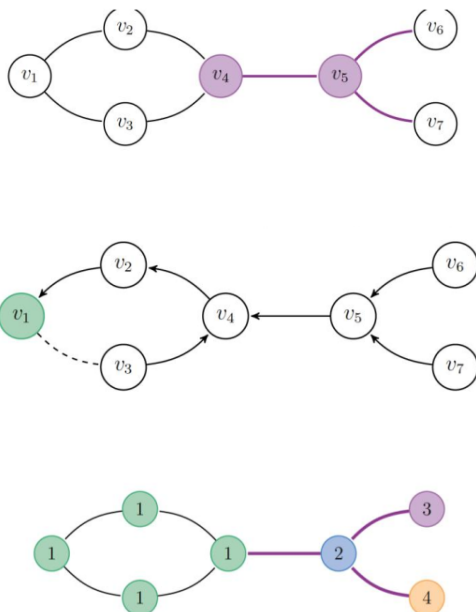


Ребро $(3, 7)$ является *мостом*.

Алгоритм нахождения мостов графа:

1. Проводим поиск в графе в глубину. При появлении новой вершины записываем ее в последовательность L . При прохождении ребра (u, v) делаем ребро ориентированным, а именно, запрещаем прохождение при втором поиске в глубину по этому ребру в направлении от вершины u к вершине v .
2. Проводим второй поиск в глубину с учетом ориентированности ребер. (если по ребру не проходили, то можно по нему идти в любую сторону). Вершины, из которых начинаем обход, берем в порядке из последовательности L , полученной в шаге 1. Вершинам из одной компоненты приписываем один цвет.
3. Ищем ребра графа, вершины которого раскрашены в разный цвет. Это и будут мосты.

Пример алгоритма на графе:



Находим ребра, вершины которых раскрашены в разный цвет – это и будут мосты: $(4, 5)$, $(5, 6)$, $(5, 7)$.

16 Связность в орграфе: различные виды связности, примеры. Компоненты сильной связности. Алгоритм Косарайю нахождения компонент сильной связности. Граф конденсации Применить алгоритм к орграфу, найти граф конденсации.

Опр: Ориентированный граф называется *сильно связным*, если для любых двух вершин v, u графа есть маршрут из v в u и наоборот.

Опр: Ориентированный граф G называется *односторонне связным*, если для любых двух вершин v, u существует хотя бы один из маршрутов из одной вершины в другую.

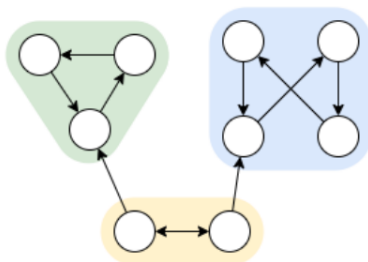
Опр: Ориентированный граф называется *слабо связным*, если соответствующий ему неориентированный граф, в котором нет ориентации рёбер, является связным.

Если задать каждому из этих трех определений бинарное отношение, то только один из заданных отношений на множестве вершин ориентированного графа является отношением эквивалентности.

Теорема: Отношение сильной связности на множестве вершин ориентированного графа является отношением эквивалентности.

Поэтому все вершины распадаются на компоненты сильной связности – такие подмножества, что внутри одной компоненты все вершины сильно связаны, а между вершинами разных компонент сильной связности нет.

Пример графа с тремя компонентами сильной связности.



Самый простой пример сильно-связной компоненты – это цикл. Но это может быть и полный граф, или сложное пересечение нескольких циклов.

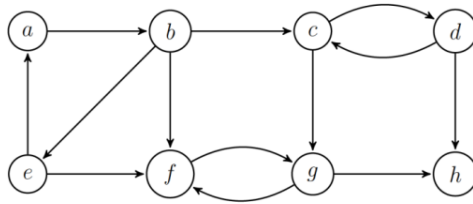
Опр: Транспонирование (инвертирование) графа – смена направлений всех ребер в графе на противоположные.

Заметим, что инвертирование в графе не меняет компонент сильной связности графа.

Алгоритм Косарайю для нахождения компонент сильной связности

1. Запускаем алгоритм DFS (обхода графа в глубину). Фиксируем время для вершины, когда она становилась черной (тупиковой). Обозначаем это время T_{out} . Получаем последовательность вершин в том порядке, в котором они становились черными.
2. Записываем последовательность вершин, в порядке убывания времени T_{out} .
3. Инвертируем граф.
4. Запускаем DFS к инвертированному графу для последовательности вершин в порядке, полученном в шаге 2. Всем достижимым из текущей вершины вершинам присваиваем ее номер (это будут компоненты сильной связности)

Пример алгоритма:

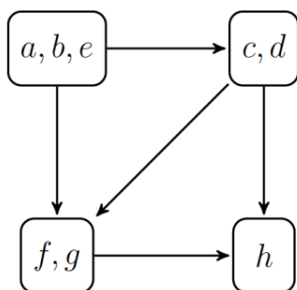


Имеем следующие компоненты сильной связности:
 $\{a, e, b\}, \{f, g\}, \{c, d\}, \{h\}$

Часто рассматривают граф, составленный из самих компонент сильной связности, а не индивидуальных вершин. Очевидно, такой граф уже будет ациклическим, и с ним проще работать. Задачу о сжатии каждой компоненты сильной связности в одну вершину называют *конденсацией графа*.

Графом конденсации или графом Герца или конденсацией ориентированного графа G называют такой ориентированный граф G' вершинами которого служат компоненты сильной связности G , а дуга в G' между двумя вершинами проводится только если в графе G существует хотя бы одна дуга между вершинами, входящими в соответствующие компоненты связности.

Для графа предыдущего примера графом конденсации является следующий граф:



17 Теорема о числе маршрутов длины k из одной вершины в другую. Формулировка, доказательство. Применить теорему к графу.

Пусть у графа $G(V, E)$, $|V| = n$, A – матрица смежности. Рассмотрим матрицы A^2, \dots, A^n , обозначим символом $a_{ij}^{(k)}$ – элементы матрицы A^k .

Теорема: Количество маршрутов из v_i в v_j длины k равно числу $a_{ij}^{(k)}$.

Док-во: С помощью метода математической индукции:

1. $k = 1, A = A^1$
 $a_{ij}^{(1)} = s$, где s – количество ребер, соединяющих вершины v_i, v_j . Поэтому количество маршрутов из v_i в v_j длины 1 равно s .
2. Пусть утверждение верно для A^k и число маршрутов длины k равно из v_i в v_j равно $a_{ij}^{(k)}$. Докажем справедливость для числа маршрутов длины $k + 1$.

$$A^{k+1} = A^k \cdot A = \begin{pmatrix} a_{11}^{(k)} & \dots & a_{1n}^{(k)} \\ \dots & \dots & \dots \\ a_{n1}^{(k)} & \dots & a_{nn}^{(k)} \end{pmatrix} \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$$

$$a_{ij}^{(k+1)} = \sum_{t=1}^n a_{it}^{(k)} a_{tj} = a_{i1}^{(k)} a_{1j} + a_{i2}^{(k)} a_{2j} + \dots + a_{in}^{(k)} a_{nj}$$

$a_{i1}^{(k)}$ – количество маршрутов длины k из v_i в v_1 , a_{1j} – количество маршрутов из v_1 в v_j длины 1 $\Rightarrow a_{i1}^{(k)} a_{1j}$ – количество маршрутов длины $k + 1$ из v_i в v_j , где последнее ребро (v_1, v_j) .

Поскольку последнее ребро может быть либо (v_1, v_j) , либо (v_2, v_j) , либо (v_n, v_j) , то по правилу суммы в комбинаторике, получаем справедливость утверждения для $k + 1$.

3. По методу математической индукции заключаем, что теорема верна для любого $n \in \mathbb{N}$.

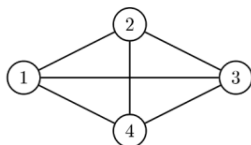
□

В любом графе с n вершинами расстояние от любой вершины до любой другой не более $n-1$. Простой цикл в таком графе имеет длину не больше n . Поэтому справедливы утверждения.

В графе (орграфе) G с n вершинами существует незамкнутый маршрут из вершины v_i в вершину v_j ($v_i \neq v_j$) тогда и только тогда, когда элемент b_{ij} матрицы $A + A^2 + \dots + A^{n-1}$ не равен 0.

В графе (орграфе) G с n вершинами тогда и только тогда существует цикл, проходящий через вершину v_i , когда элемент b_{ii} матрицы $A + A^2 + \dots + A^n$ не равен нулю.

Пример теоремы к графу:



$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, A^2 = \begin{pmatrix} 3 & 2 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 2 & 3 \end{pmatrix}, A^3 = \begin{pmatrix} 6 & 7 & 7 & 7 \\ 7 & 6 & 7 & 7 \\ 7 & 7 & 6 & 7 \\ 7 & 7 & 7 & 6 \end{pmatrix}$$

Так как $a_{11}^{(2)} = 3$, то существует ровно три маршрута из вершины 1 в 1 длиной 2.

18 Матрицы связности и достижимости в графе. Найти матрицу достижимости для указанного орграфа, с ее помощью выделить компоненты сильной связности графа.

Пусть A – матрица смежности G , $|V| = n$. Найдем $B = E + A + A^2 + \dots + A^n$.

Определим матрицу D размерности $n \times n$, $d_{ij} = \text{sign}(b_{ij})$.

Эта матрица показывает, есть ли путь из вершины v_i в вершину v_j (в этом случае $d_{ij} = 1$).

Опр: В случае если граф G – неориентированный, матрица D называется матрицей связности. В случае, если граф G – ориентированный, матрица D называется матрицей достижимости.

Если G – неориентированный граф, $v_i \in V$, то в одну компоненту связности с вершиной v_i входят такие вершины v_j , для которых $d_{ij} = 1$.

Можно также ввести матрицу, по которой можно найти компоненты сильной связности ориентированного графа.

Пусть D – матрица достижимости орграфа G , $L = D^T$,

$$l_{ij} = \begin{cases} 1, & \text{если есть маршрут из } v_j \text{ в } v_i, \\ 0, & \text{если нет.} \end{cases}$$

$$F = D \times L, f_{ij} = d_{ij} * l_{ij}.$$

Если $f_{ij} = 1$, то вершины v_i и v_j принадлежат одной компоненте сильной связности.

Пример. Рассмотрим ориентированный граф $G(V, E)$

$$V = \{1, 2, 3\}, E = \{(1, 2), (2, 1), (1, 3)\}$$

$$\begin{aligned} A &= \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, A^2 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}, A^3 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ B &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 2 \\ 2 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix} \\ D &= \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, L = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}, F = D * L = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

Поэтому у графа 2 компоненты сильной связности $\{1, 2\}$ и $\{3\}$. Это один из способов нахождения компонент сильной связности. Временная сложность $O(n^4)$.

19 Транзитивное замыкание графа. Алгоритм Уоршелла. Применить алгоритм к заданному бинарному отношению.

Опр: Транзитивным замыканием бинарного отношения R на множестве M называется наименьшее по числу элементов транзитивное отношение на множестве M , включающее R .

Если изобразить граф бинарного отношения, то транзитивность отношения означает, что если есть ребра $(u, v), (v, w)$, то есть и ребро (u, w) .

На этом может быть основан алгоритм нахождения транзитивного замыкания бинарного отношения. Для графа можно найти матрицу $T = A + A^2 + \dots + A^n$, здесь элементы матриц складываем дизъюнктивно:

$$0 + 0 = 0 \vee 0 = 0; 0 + 1 = 0 \vee 1 = 1; 1 + 0 = 1 \vee 0 = 1; 1 + 1 = 1 \vee 1 = 1.$$

Если элемент этой матрицы $t_{ij} = 1$, то это означает, что существует маршрут от вершины v_i до v_j . Отсюда следует, что для транзитивности отношения эти вершины в графе должны соединяться ребром.

Поэтому соответствующее этой матрице бинарное отношение и является его транзитивным замыканием.

Минус этого метода в его временной сложности: временная сложность нахождения произведения матриц размерности n равна $O(n^3)$, матрицу смежности нужно умножать на себя n раз, поэтому временная сложность этого алгоритма $O(n^4)$.

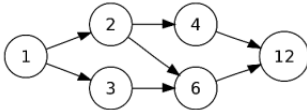
Есть алгоритмы, сложность которых меньше. Опишем один из таких алгоритмов.

Алгоритм нахождения транзитивного замыкания Уоршелла.

```
T := A (матрица отношения)
for k = 1 to n do
  (Внешний цикл идет по промежуточным вершинам.)
  for i = 1 to n do
    (Циклы по i и j перебирают всевозможные пары.)
    for j = 1 to n do
      T(i; j) := max {T(i; j), T(i; k) * T(k; j)}
      (Благодаря операции max сохраняются те связи,
       которые были)
    end for
  end for
end for
```

Трудоемкость такого алгоритма будет $O(n^3)$. Цикл по k нельзя менять с другими циклами.

Пример. Рассмотрим алгоритм Уоршелла на примере бинарного отношения, которому соответствует граф на рисунке



1. Вершины расположим так: 1(1),2(2),3(3),4(4),6(5),12(6) (в скобках указан номер вершины). Найдём матрицу бинарного отношения.

В итоге видим, что нужно добавить 5 ребер: (1,4), (1,6), (1,12), (2,12), (3,12).

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

2. Сделаем пересчет по вершине 1. Если $t_{ij} = 1$, то этот элемент пересчитывать не нужно. Если $t_{ij} = 0$, то смотрим, можно ли пройти из вершины с номером i в вершину с номером j через вершину 1. Так как в вершину 1 не заходит ни одно ребро, то нет. Поэтому матрица не меняется.

3. Сделаем пересчет по вершине 2.

$$T(1; 4) := \max \{T(1; 4), T(1; 2) \cdot T(2; 4)\} = \max \{0, 1 \cdot 1\} = 1$$

$$T(1; 5) := \max \{T(1; 5), T(1; 2) \cdot T(2; 5)\} = \max \{0, 1 \cdot 1\} = 1$$

Другие элементы матрицы не изменятся.

$$T = \begin{pmatrix} 0 & 1 & 1 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

4. Сделаем пересчет по вершине 3. Матрица не меняется.

5. Сделаем пересчет по вершине 4.

$$T(1; 6) := \max \{T(1; 6), T(1; 4) \cdot T(4; 6)\} = \max \{0, 1 \cdot 1\} = 1$$

$$T(2; 6) := \max \{T(2; 6), T(2; 4) \cdot T(4; 6)\} = \max \{0, 1 \cdot 1\} = 1$$

Другие элементы матрицы не изменятся.

$$T = \begin{pmatrix} 0 & 1 & 1 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

6. Сделаем пересчет по вершине 6.

$$T(3; 6) := \max \{T(3; 6), T(3; 5) \cdot T(5; 6)\} = \max \{0, 1 \cdot 1\} = 1$$

Другие элементы матрицы не изменятся.

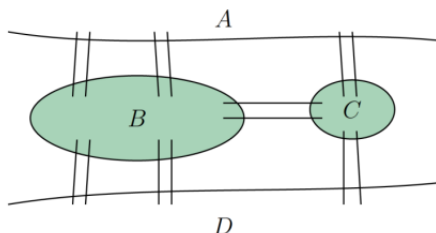
$$T = \begin{pmatrix} 0 & 1 & 1 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 1 & 1 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 1 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

7. Сделаем пересчет по вершине 12. Так как из вершины 12 не выходит ребер, то матрица не меняется.

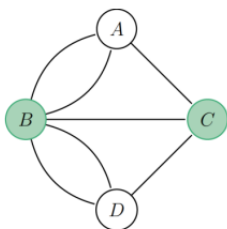
20 Эйлеровы циклы и цепи в графе. Теорема об эйлеровом цикле. Выяснить, существует ли в графе Эйлеров цикл и цепь.

Задача о Кёнигсбергских мостах послужила началом математической теории графов.

Изобразим расположение мостов. Задача: выйти из любой точки, пройти по всем мостам по одному разу и вернуться обратно.



Изобразим граф, в котором вершинами является суша, а рёбрами – мосты.



Опр: Если в графе есть цикл, проходящий по всем рёбрам графа, то такой цикл называется *эйлеровым циклом*, а граф называется *эйлеровым графом*.

Теорема: Конечный граф без изолированных вершин является эйлеровым графом если он является связным и степени всех его вершин четные числа.

$$G - \text{Эйлеров граф} \Leftrightarrow \begin{array}{l} 1. G - \text{связный граф} \\ 2. \text{для любой вершины } \delta(v) - \text{чётное число} \end{array}$$

Док-во: $\Rightarrow G$ – эйлеров граф. По определению, в графе есть эйлеров граф. Значит граф связный.

Каждый раз, проходя через вершину, цикл заходит в эту вершину по одному ребру, а выходит по другому ребру. То есть при прохождении через вершину задействуется два различных ребра. Так как в эйлеров цикл входят все рёбра графа, то степень каждой вершины – четное число.

\Leftarrow Пусть граф является связным и степени всех его вершин четные числа. Построим в графе G эйлеров цикл. Начнём цикл в произвольной вершине a . Из вершины a будем строить цикл, проходя всё время по разным рёбрам. Так как степень каждой вершины – чётное число, то цикл может завершиться только в вершине a .

Обозначим полученный цикл за P . Если в P входят все рёбра графа G , то получили эйлеров цикл. Пусть в P входят не все рёбра графа G , тогда удалим из графа G все рёбра, принадлежащие циклу P . Полученный граф обозначим графом G_1 . Так как в графе G степени всех вершин чётные, при прохождении по циклу P задействуется чётное число рёбер, инцидентных каждой вершине, то в графе G_1 степень каждой вершины также чётна. Так как G – связный граф, то среди вершин цикла P найдётся вершина b , инцидентная какому то рёбру графа G_1 .

Из вершины b в графе G_1 снова строим цикл. Получим цикл, который завершится в точке b . Назовём его P' . Рассмотрим цикл, полученный объединением маршрутов $P(a, b), P', P(b, a)$. Получили цикл, содержащий больше рёбер, чем в первоначальном. Этот цикл возможно будет эйлеровым. Если нет, то процесс продолжим. Так как граф конечный, то этот процесс завершится на некотором шаге.

В итоге получим эйлеров цикл. □

21 Алгоритм Флери нахождения эйлерова цикла. Применить к указанному графу.

Рассмотрим один из алгоритмов нахождения эйлерова цикла – *алгоритм Флёрри*. Он отличается от приведенного алгоритма в доказательстве теоремы Эйлера тем, что каждый раз при добавлении нового ребра в маршрут нужно проверить, не является ли это ребро мостом.

В алгоритме нумеруются все ребра графа числами $1, 2, \dots, |E|$, так, чтобы номер, присвоенный данному ребру, указывал, каким по счету это ребро будет в эйлеровом цикле.

1. Выбираем произвольную вершину u , присваиваем произвольному ребру (u, v) номер $k = 1$. Вычеркиваем это ребро из множества ребер графа и переходим в вершину v .
2. Пусть v – вершина, в которую мы перешли в результате выполнения предыдущего шага, k – номер, присвоенный ребру на этом шаге. Выбираем любое ребро, инцидентное вершине v . При этом мост выбираем только в случае, когда ребер, инцидентных вершине и не являющихся мостами нет. Этому ребру присваиваем номер $k + 1$, проходим по нему в следующую вершину и вычеркиваем это ребро.
3. Если в графе есть не вычеркнутые ребра, то переходим к шагу 2, иначе stop.

Опр: Граф называется *полуэйлеровым*, если в нём существует цепь, проходящая по всем рёбрам графа.

Теорема: Граф без изолированных вершин называется *полуэйлеровым графом* тогда и только тогда, когда выполняется два условия:

1. Граф G является связным
2. Только две вершины в графе имеют нечетную степень.

Опр: *Полустепенью исхода вершины v* орграфа $\delta^-(v)$ называется количество рёбер, исходящих из данной вершины.

Опр: *Полустепенью захода вершины v* орграфа $\delta^+(v)$ называют количество рёбер, заходящих в данную вершину.

Критерий эйлера и полуэйлера графа можно дать и для ориентированного графа без изолированных вершин: для случая эйлера графа: полустепени исхода всех вершин должны равняться ее полустепени захода, граф должен быть сильносвязным; для случая полуэйлера графа: полустепени исхода всех вершин, кроме двух должны равняться полустепени захода, для одной из оставшихся вершин полустепень исхода на единицу больше полустепени захода, для другой вершины наоборот.

22 Графы де Брюина: определение, свойства, алгоритм построения. Применение для построения слова наименьшей длины, которое содержит все указанные под слова.

Опр: Конечное множество символов $A = \{a_1, a_2, \dots, a_n\}$ называют алфавитом, символы алфавита – буквами, последовательность символов алфавита – словами. Число символов в слове называют длиной слова.

Опр: Пусть задан алфавит, состоящий из n букв. Графом де Брюина (обозначается $B(n, k)$) для алфавита A называется ориентированный граф $G(V, E)$, множеством вершин которого является множество слов длины k в алфавите A . Ребро (дуга) из вершины u в вершину v проводится, если

$$u = \{x_1x_2 \dots x_k\}, v = \{y_1y_2 \dots y_k\} \\ x_2 = y_1, x_3 = y_2, \dots, x_k = y_{k-1}.$$

Это означает, что если вершина $u = \{x_1x_2 \dots x_k\}$, то вершина $v = \{x_2x_3 \dots x_ky_k\}$, при этом ребру (u, v) сопоставляется последовательность из $k + 1$ символа: $e = (u, v) = x_1x_2 \dots x_ky_k$.

Вершина u называется префиксом, а вершина v – суффиксом слова e .

Справдливы следующие теоремы.

Теорема: Граф де Брюина является эйлеровым графом.

Теорема: Граф де Брюина содержит n^k вершин и n^{k+1} ребер.

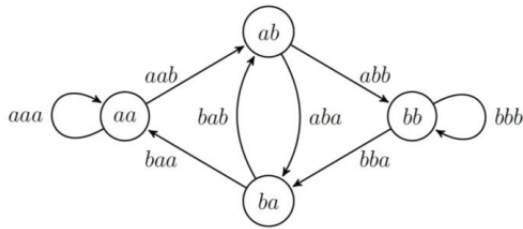
Алгоритм построения графа де Брюина

1. Генерируем в любом порядке все слова длины $k + 1$ в алфавите A .
2. Для каждого слова(ребра) $\{x_1x_2 \dots x_kx_{k+1}\}$ определяем вершины (узлы): $u = x_1x_2 \dots x_k, v = x_2x_3 \dots x_kx_{k+1}$.

Пример: Пусть $A = \{a, b\}, k = 2$. Получаем следующие ребра и вершины графа:

Слово	Узлы	Слово	Узлы
{aaa}	{aa} → {aa}	{baa}	{ba} → {aa}
{aab}	{aa} → {ab}	{bab}	{ba} → {ab}
{aba}	{ab} → {ba}	{bba}	{bb} → {ba}
{abb}	{ab} → {bb}	{bbb}	{bb} → {bb}

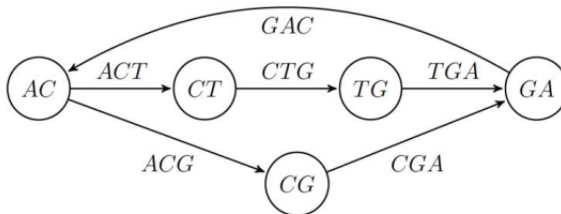
Изобразим полученный граф:



Пример: При помощи графа де Брюина найти слово наименьшей длины, которое содержит все подслова из множества $R = \{ ACT, CTG, TGA, GAC, ACG, CGA \}$.

По алгоритму:

1. $V = \{ AC, CT, TG, GA, CG \}$ – вершины графа.
2. Строим граф.



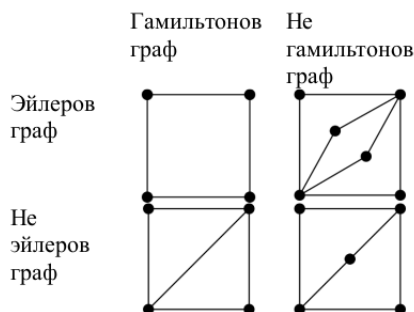
3. Строим эйлерову цепь: $AC \rightarrow CT \rightarrow TG \rightarrow GA \rightarrow AC \rightarrow CG \rightarrow GA$
Замечание. Эйлерову цепь начинаем в вершине, из которой выходит на одно ребро больше, чем заходит в вершину.
4. Построенной эйлеровой цепи соответствует слово минимальной длины, которое содержит все подслова из R : $ACTGACGA$ (Строим по правилу: $AC + T + G + A + C + G + A$)

23 Гамильтоновы циклы и цепи в графе. Теоремы Дирака и Оре. Выяснить, существует ли в указанном графе Гамильтонов цикл.

Опр: Граф называется *гамильтоновым*, если существует простой цикл, проходящий по всем вершинам графа.

Опр: Граф называется *полугамильтоновым*, если существует простая цепь, проходящая через все вершины графа.

Покажем, что понятия эйлерова и гамильтонова графа не связаны между собой, то есть у графов могут встретиться всевозможные комбинации этих понятий.



Критерия гамильтонова графа нет. Но есть достаточные условия гамильтоновости графа. Теоремы Дирака и Оре связывают понятие гамильтоновости со степенями $\delta(v)$ вершин графа.

Теорема: Дирака. Если $n \geq 3$ и $\delta(v) \geq n/2$ для любой вершины v неориентированного графа G , то G – гамильтонов граф.

Теорема: Оре. Если $n \geq 3$ и $\delta(u) + \delta(v) \geq n$ для любых двух различных несмежных вершин u, v неориентированного графа G , то G – гамильтонов граф.

24 Деревья. Теорема о связи вершин и ребер в дереве.

Опр: Граф называется *деревом*, если он является связным и не содержит циклов.

Опр: Граф, все компоненты связности которого являются деревьями, называется *лесом*.

Теорема: Пусть граф G – дерево, у которого n вершин и r ребер. Тогда справедливо равенство $r = n - 1$.

Док-во: Методом математической индукции:

1. G – дерево. $n = 1$. Так как петель в дереве не может быть, то $r = 0$.
2. Пусть равенство выполняется для дерева с $n = k$ вершинами. То есть у такого дерева $r = k - 1$ ребер. Докажем, что равенство будет выполняться и для дерева G с $k + 1$ вершиной.
3. В дереве G существует ребро, поэтому есть и висячая вершина. Эта вершина не является шарниром, поэтому после ее удаления вместе с инцидентным ему ребром граф останется связным. Очевидно также, что в полученном после удаления вершины и ребра графе не будет циклов (так как их нет в G). Поэтому полученный граф является деревом с k вершинами. Известно, по предположению индукции, что у него $k - 1$ ребро. У графа G на одну вершину и на одно ребро больше, поэтому у него $k + 1$ вершина и k ребер, то есть число ребер на единицу меньше числа вершин.
4. По методу математической индукции заключаем, что утверждение справедливо для любого дерева.

□

Задача №1. В любом дереве, в котором есть хотя бы одно ребро, есть висячие вершины.

Задача №2. Висячая вершина не является шарниром.

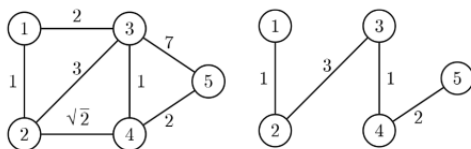
25 Остовное дерево связного графа. Минимальное остовное дерево. Алгоритмы Прима и Краскала, их применение.

Опр: *Остовным (стягивающим) деревом* связного графа называется его подграф, содержащий все вершины графа и являющийся деревом.

Опр: Если каждому ребру графа сопоставляется какое-то число (называют вес или длина или стоимость ребра), то такой граф называется *нагруженным (взвешенным) графом*.

Обозначение: $l(e)$ – длина ребра, $w(e)$ – вес ребра.

Пример нагруженного графа и его остовного дерева



Пусть G – нагруженный граф.

Опр: *Минимальным остовным деревом нагруженного графа* называется остовное дерево с минимальной суммой длин входящих в него ребер.

Опишем алгоритмы Прима и Краскала, которые позволяют находить минимальное остовное дерево нагруженного графа.

Алгоритмы Прима и Краскала называются также *жадными алгоритмами*, то есть алгоритмами, при которых на каждом шаге происходит поиск оптимального выбора.

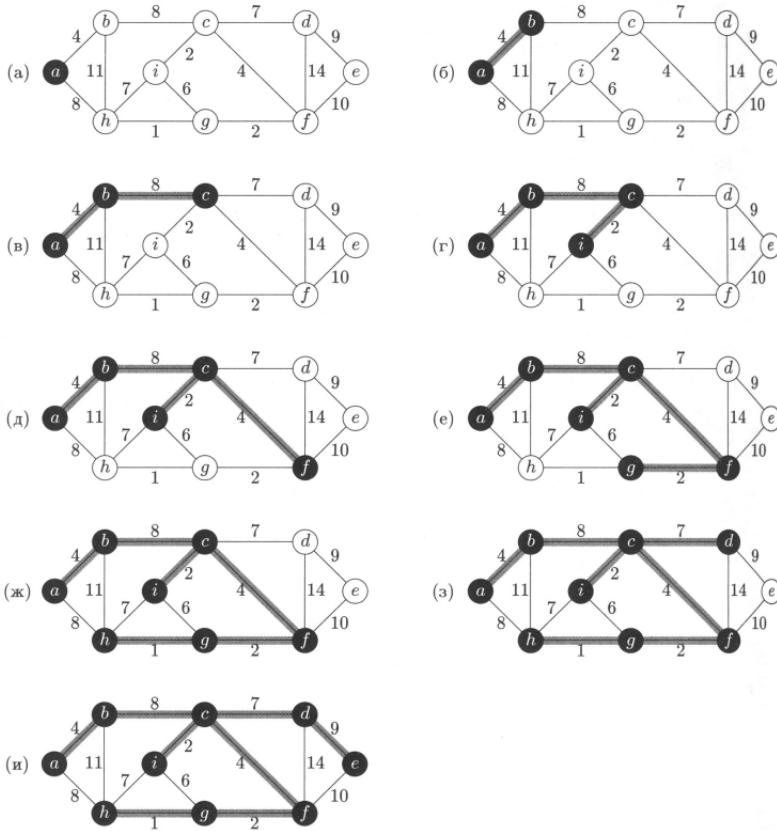
Алгоритм Прима нахождения минимального остовного дерева.

Пусть в нагруженном графе G число вершин равно n .

1. Берем произвольную вершину графа. Находим в графе G ребро минимальной длины, инцидентное этой вершине. Получаем подграф, состоящий из 2 вершин и ребра, соединяющего эти вершины. Обозначим этот подграф G_2 , $i := 2$.
2. Если $i = n$, то останавливаемся. Если нет, то переходим к шагу 3.
3. Рассмотрим рёбра графа G , одна из вершин которых принадлежит графу G_i , а вторая не принадлежит. Из всех таких рёбер выбираем

ребро минимальной длины и добавляем его к графу G_i . Получаем граф G_{i+1} , $i := i + 1$. Переходим к шагу 2.

Пример. Найдем минимальное остовное дерево с помощью алгоритма Прима:



Найдем длину(вес) полученного остовного дерева:

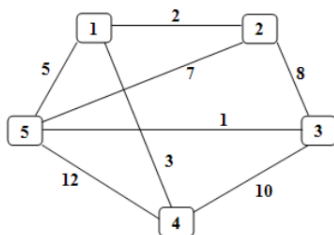
$$4 + 8 + 7 + 9 + 2 + 4 + 1 + 2 = 37$$

Алгоритм Краскала (Крускала) нахождения минимального остовного дерева.

Пусть в нагруженном графе G число вершин равно n .

1. Упорядочиваем все ребра графа G в порядке неубывания – от ребер минимальной длины к максимальной. Множество ребер остовного дерева пусто.
2. Первое ребро в остовном дереве - первое из упорядоченного списка (то есть ребро минимальной длины).
3. Если в множестве ребер содержится $n - 1$ ребер, то останавливаемся. Если нет, то переходим к шагу 4.
4. Добавим в множество ребер остовного дерева такое ребро минимальной длины из оставшихся ребер, при добавлении которого не появятся циклы. Переходим к шагу 3.

Пример. Найдем минимальное остовное дерево с помощью алгоритма Краскала:



1. Упорядочиваем ребра: $(5, 3), (1, 2), (1, 4), (1, 5), (5, 2), (2, 3), (3, 4), (5, 4)$.
2. V – множество вершин, E – множество ребер остовного дерева. $V = \emptyset, E = \emptyset$.
3. $E = \{ (5, 3) \}, V = \{ 5, 3 \}$
4. $E = \{ (1, 2), (5, 3) \}, V = \{ 1, 2, 5, 3 \}$
5. $E = \{ (1, 2), (5, 3), (1, 4) \}, V = \{ 1, 2, 5, 3, 4 \}$
6. $E = \{ (1, 2), (5, 3), (1, 4), (1, 5) \}, V = \{ 1, 2, 5, 3, 4 \}$
7. $n - 1$ ребер, stop.

Найдем длину (вес) полученного остовного дерева: $1 + 2 + 3 + 5 = 11$.

26 Матрица Кирхгофа, ее применение для нахождения числа остовных деревьев. Корневые деревья: основные определения. Найти число остовных деревьев с помощью матрицы Кирхгофа.

Пусть G – связный неориентированный граф. $|V| = n$.

Опр: Матрицей Кирхгофа графа G называется квадратная матрица размерности $n \times n$, в которой

$$b_{ij} = \begin{cases} k, & \text{где } \delta(v_i) = k, i = j, \\ -1, & \text{если } i \neq j, v_i \text{ и } v_j \text{ смежны,} \\ 0, & \text{если } v_i \text{ и } v_j \text{ не смежны.} \end{cases}$$

Теорема: Число остовных деревьев в связном графе G , равно алгебраическому дополнению любого элемента матрицы Кирхгофа.

Пример. Изобразить граф, заданный с помощью матрицы Кирхгофа, найти число остовных деревьев графа и их изобразить.

$$\begin{matrix} & a & b & c & d & e & f \\ \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \end{matrix} & \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ -1 & 0 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & -1 & 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix} \end{matrix}.$$

Найдем алгебраическое дополнение элемента матрицы с индексом 44.

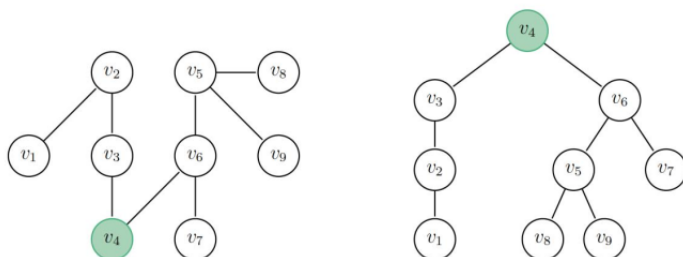
$A_{44} = 1$. Это означает, что существует лишь одно остовное дерево этого графа. Это остовное дерево является самим графом.

Корневые деревья

Опр: *Корневым деревом* называют дерево с выделенной вершиной – корнем.

Опр: Вершину корневого дерева называют *листом*. *Высотой корневого дерева* называют расстояние от корня до самого удаленного листа. Если в корневом дереве маршрут, соединяющий вершину u с корнем, проходит через вершину v , то говорят, что вершина u – *потомок вершины v* , вершина v – *предок вершины u* . Множество всех потомков вершины v является деревом с корнем v , оно называется *ветвью дерева* в вершине v . Если предок и потомок соединены ребром, то они называются *отцом* и *сыном*.

На рисунке пример дерева, в котором выделили вершину v_4 и оно стало корневым деревом. Обычно корень принято изображать сверху (как в генеалогическом древе). Иногда его изображают внизу.



У графа на рисунке высота корневого дерева равна трем.

Для вершин v_1 и v_3 , вершина v_1 – *потомок* вершины v_3 , вершина v_3 *предок* вершины v_1 , вершина v_2 – *отец* вершины v_1 .

Множество вершин $\{v_1, v_2, v_3\}$ всех потомков вершины v_3 является деревом с корнем v_3 , оно является ветвью дерева в вершине v_3 .

Опр: Ориентированным деревом называют ориентированный граф без циклов, в котором в каждую вершину, кроме одной, называемой корнем, входит ровно одно ребро. В корень ориентированного дерева не входит ни одного ребра.

Иногда ориентированные деревья изображают с вершиной – корнем внизу, ребра также могут быть изображены направленными в сторону корня.

27 Теорема Кэли о числе помеченных деревьев. Код Прюфера для дерева: алгоритмы кодирования и декодирования, его применение.

Опр: Граф называется *помеченным* (*пронумерованным*), если каждой вершине графа сопоставляется некая метка (если у графа n вершин, то метки – это числа от 1 до n).

При изоморфизме помеченных графов вводится дополнительное условие: пары вершин первого и второго графов с одинаковыми метками должны быть смежны одновременно.

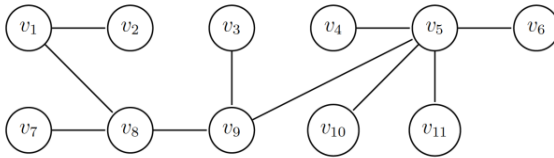
Теорема: Кэли: Число различных помеченных деревьев с n вершинами равно n^{n-2} .

Укажем алгоритм, по которому можно для помеченных деревьев с n вершинами ($n \geq 3$) указать его код, состоящий из $n - 2$ чисел (числа могут повторяться).

Алгоритм нахождения **кода Прюфера** помеченного дерева:

1. Находим в дереве висячую вершину с минимальным номером. Удаляем эту вершину вместе с инцидентным ей ребром из графа. Номер смежной ей вершины записываем в код.
2. Если в графе осталось 1 ребро, то останавливаем алгоритм. Если нет, то переходим к шагу 1.

Найдем код Прюфера для следующего дерева:



i	Вершина	p_i	i	Вершина	p_i	i	Вершина	p_i
1	v_2	1	4	v_4	5	7	v_8	9
2	v_1	8	5	v_6	5	8	v_9	5
3	v_3	9	6	v_7	8	9	v_{10}	5

Получаем следующий код Прюфера: $P = \{ 1, 8, 9, 5, 5, 8, 9, 5, 5 \}$

В код Прюфера входят только вершины, не являющиеся висячими. При этом они участвуют в коде $\delta(v) - 1$ раз.

Алгоритм восстановления дерева:

1. Записываем код дерева в первой строке. Находим количество вершин в дереве (число чисел в коде плюс 2) Во второй строке выписываем вершины, которых нет в коде (висячие вершины).
2. Берём вершину u – первую вершину в первой строке и вершину v – вершину с минимальным номером из второй строки. Записываем ребро (u, v) и вычёркиваем u и v из соответствующих строк. Если вершины u больше нет в первой строке, то записываем её во вторую строку.
3. Если вершин в первой строке не осталось, то из оставшихся двух вершин нижней строки составляем ребро и останавливаем алгоритм, иначе переходим к шагу 2.

Пример. Восстановим дерево по коду Прюфера из примера выше.

- | | |
|--|---|
| 1. Записываем код в первой строке, висячие вершины во второй:
1,8,9,5,5,8,9,5,5 ($n=11$)
2,3,4,6,7,10,11 | 6. Записываем ребро (5,6)
8,9,5,5
7,10,11 |
| 2. Записываем ребро (1,2)
8,9,5,5,8,9,5,5
3,4,6,7,10,11,1 | 7. Записываем ребро (8,7)
9,5,5
10,11,8 |
| 3. Записываем ребро (8,1)
9,5,5,8,9,5,5
3,4,6,7,10,11 | 8. Записываем ребро (9,8)
5,5
10,11,9 |
| 4. Записываем ребро (9,3)
5,5,8,9,5,5
4,6,7,10,11 | 9. Записываем ребро (5,9)
5
10,11 |
| 5. Записываем ребро (5,4)
5,8,9,5,5
6,7,10,11 | 10. Записываем ребро (5,10)
11,5 |
| | 11. Записываем ребро (5,11) |

Получили граф, соответствующий графу, код которого получали в предыдущем примере.

28 Цикломатическое число графа, теорема о связи цикломатического числа с количеством вершин, ребер и компонент связности.

Опр: *Цикломатическим числом графа* называется минимальное число рёбер, которое нужно удалить, чтобы в графе не было циклов.

Если граф G – связный, то нужно удалить рёбра так, чтобы получилось остовное дерево графа.

Теорема: Цикломатическое число графа, у которого n вершин, r рёбер и k компонент связности, равно $r - n + k$.

Док-во: Рассмотрим сначала случай, когда граф G – связный, у него n вершин и r рёбер, одна компонента связности. Для того, чтобы у подграфа этого графа не было циклов, нужно удалить рёбра так, чтобы получилось остовное дерево графа.

Пусть s – количество рёбер, которые нужно удалить, чтобы получить остовное дерево. Из общего числа ребер r вычитаем число ребер в остовном дереве. Известно, что их $n - 1$.

$$s = r - (n - 1) = r - n + 1 - \text{цикломатическое число связного графа.}$$

Обозначим цикломатическое число графа $\mu(G)$.

Пусть теперь $G(V, E)$ – граф, у которого n вершин, r рёбер и k компонент связности. Тогда нам нужно получить остовной лес. Для этого в каждой компоненте связности ищем его остовное дерево. Обозначим n_i и r_i – количество вершин и ребер в компоненте связности с номером i . Тогда цикломатическое число равно:

$$\begin{aligned}\mu(G) &= (r_1 - n_1 + 1) + (r_2 - n_2 + 1) + \cdots + (r_k - n_k + 1) = \\ &= (r_1 + r_2 + \cdots + r_k) - (n_1 + n_2 + \cdots + n_k) + k = r - n + k.\end{aligned}$$

□

29 Главные циклы графа, Теорема о разложении цикла в сумму главных циклов. Применение для указанного графа.

Опр: Пусть $G(V, E)$ – связный граф с n вершинами и r ребрами, $T(V, E_1)$ – остовное дерево графа G . Ребра $e \in E_1$ называются *ветвями*, ребра $y \in E \setminus E_1$ называются *хордами* графа.

Если к остовному дереву добавить хорду $y = (u, v)$, то y и T образуют граф с циклом, который определяется единственным образом цепью, соединяющей вершины u и v . Этот цикл называют *главным циклом*, *определяемым хордой y* .

Множество всех главных циклов называют *фундаментальной системой циклов*.

Ветвей в графе $n - 1$, тогда хорд $r - (n - 1) = r - n + 1 = \mu(G)$ – цикломатическое число графа.

Опр: Введем операцию сложения по модулю 2 над множествами ребер графа M_1 и M_2 :

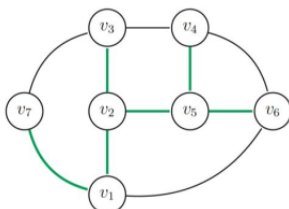
$$M_1 \oplus M_2 = \{ e \in E \mid (e \in M_1 \text{ и } e \notin M_2) \text{ или } (e \notin M_1 \text{ и } e \in M_2) \}.$$

Эту операцию можно задать и для произвольного конечного числа подмножеств ребер графа: ребро будет принадлежать сумме подмножеств ребер, если оно принадлежит нечетному числу подмножеств.

Теорема: Любой цикл Z графа является суммой его некоторых главных циклов:

$$Z = C_{i_1} \oplus \dots \oplus C_{i_k}.$$

Пример. Рассмотрим граф на рисунке. Ребра остовного дерева обозначены зеленым цветом. Построим главные циклы графа.



Рассматриваем все хорды графа. Добавляем их по очереди в остовное дерево, смотрим, какой цикл получился, записываем его.

$$C_1 - (v_3, v_4) : v_3 \rightarrow v_4 \rightarrow v_5 \rightarrow v_2 \rightarrow v_3$$

$$C_2 - (v_4, v_6) : v_4 \rightarrow v_6 \rightarrow v_5 \rightarrow v_4$$

$$C_3 - (v_1, v_6) : v_1 \rightarrow v_6 \rightarrow v_5 \rightarrow v_2 \rightarrow v_1$$

$$C_4 - (v_3, v_7) : v_3 \rightarrow v_2 \rightarrow v_1 \rightarrow v_7 \rightarrow v_3$$

Получили четыре главных цикла. C_1, C_2, C_3, C_4 - фундаментальная система циклов. Через нее можно выразить любой цикл.

Рассмотрим цикл $Z = \{ v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_6 \rightarrow v_1 \}$. Тогда

$$Z = C_1 \oplus C_2 \oplus C_3.$$

Для того, чтобы найти разложение цикла Z в сумму главных циклов, нужно взять те циклы, которым принадлежат хорды, входящие в цикл Z .

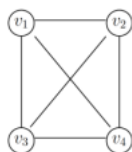
30 Укладка графа. Планарные и плоские графы. Теорема Эйлера о связи числа вершин, ребер и граней в плоском графе.

Опр: Граф называется *укладываемым на поверхности S* , если его можно изобразить на S так, что никакие два ребра не пересекутся во внутренних точках, само такое изображение графа называется *укладкой графа на поверхности S* .

Опр: Если граф является укладываемым на плоскости, то он называется *планарным*.

Опр: Любая укладка планарного графа на плоскости называется его *плоской укладкой (плоским графом)*.

Пример. Граф на рисунке а) является планарным, но не плоским. На рисунке б) граф из пункта а) изображен по другому, без пересечений во внутренних точках ребер, он является планарным и плоским.



а)



б)

Задача. Любое дерево является планарным графом.

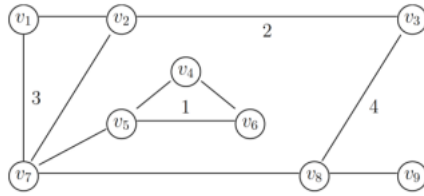
Опр: Пусть дана плоская укладка графа G . *Гранью графа G* называется множество точек плоскости, каждую пару которых можно соединить кривой, не проходящей через вершины и ребра графа.

Опр: Грань, являющаяся неограниченной областью плоскости, называется *внешней гранью*, остальные грани называются *внутренними*.

Любая грань ограничена некоторым замкнутым маршрутом (не обязательно циклом, в таком маршруте могут повторяться вершины и ребра).

Пример. На рисунке ниже отмечены грани плоского графа: 1,2,3 – внутренние, 4 – внешняя.

Внешняя грань ограничена замкнутым маршрутом $v_1 v_2 v_3 v_8 v_9 v_8 v_7 v_1$



Теорема: (Эйлера) Если связный плоский граф имеет n вершин, r ребер и q граней (включая внешнюю), то справедливо равенство

$$n - r + q = 2$$

Док-во: Рассмотрим $G(V, E)$ – граф, у которого n вершин, r рёбер, q граней.

Найдем его остовное дерево. В нем будет n вершин, $n - 1$ рёбер, 1 грань (внешняя). Для полученного остовного дерева равенство выполняется:

$$n - r + q = n - (n - 1) + 1 = 2$$

Будем добавлять в остовное дерево по одному ребру. Так как граф плоский, то при этом каждый раз будет добавляться один замкнутый маршрут, а значит, одна грань будет отделяться, то есть на одну грань каждый раз будет становиться больше. Так как число граней и ребер стоят в равенстве с разными знаками, то сумма меняться не будет. Так можно дополнить остовное дерево до самого графа, при этом равенство $n - r + q = 2$ сохранится. \square