

# DM E<sub>χ</sub> 02

isagila

@pochtineploho

@DUBSTEPHAVEGUN

Собрано 22.06.2023 в 21:07



# Содержание

<b>1. Теория графов</b>	<b>3</b>
1.1. Основные определения.	3
1.2. Морфизмы графов.	4
1.3. Пути и расстояния.	5
1.4. Эйлеровы графы.	5
1.5. Гамильтоновы графы.	6
1.6. Двудольные графы.	6
1.7. Теорема Холла.	7
1.8. Теорема Татта.	7
1.9. Связность.	8
1.10. Теорема Уитни.	9
1.11. Теорема Менгера.	9
1.12. Деревья.	10
1.13. Алгоритм Дейкстры.	11
1.14. Алгоритм Форда-Белмана.	12
1.15. Алгоритм Флойда-Уоршелла.	13
1.16. Иерархическая кластеризация.	13
<b>2. Комбинаторика</b>	<b>15</b>
2.1. Упорядоченные и неупорядоченные размещения.	15
2.2. Разбиения и композиции.	15
2.3. Принцип включения-исключения.	15
<b>3. Конечные автоматы</b>	<b>16</b>
3.1. Формальные и регулярные языки.	16
3.2. Детерминированный и недетерминированный конечные автоматы.	16
3.3. Преобразование НКА в ДКА.	16
3.4. $\varepsilon$ -НКА. Преобразование $\varepsilon$ -НКА в НКА.	16
3.5. Построение $\varepsilon$ -НКА по регулярному выражению (построение Томпсона).	16
3.6. Теорема Клини.	16
<b>4. Рекуррентные соотношения</b>	<b>17</b>
4.1. Рекуррентные соотношения. Характеристические уравнения.	17
4.2. Асимптотический анализ.	17
4.3. Мастер теорема.	17
4.4. Метод Акра -Бацци.	17
4.5. Производящие функции.	17
4.6. Операторы и аннигиляторы.	17

# 1. Теория графов

## 1.1. Основные определения.

**Def 1.1.1.** Граф это упорядоченная пара вида  $G = \langle V, E \rangle$ , где  $V = \{v_1, \dots, v_n\}$  это множество вершин, а  $E = \{e_1, \dots, e_n\}$  это множество ребер.

**Def 1.1.2.** Порядком графа называется число его вершин  $|V|$ .

**Def 1.1.3.** Размером графа называется количество его ребер  $|E|$ .

У неориентированного графа  $E \subseteq V^{(2)}$ , где  $V^{(2)}$  это множество всех непустых подмножеств размера не более двух. Таким образом каждое ребро обозначается как  $\{u, v\} \in V^{(2)}$  (если  $u \neq v$ ) или  $\{v\} \in V^{(2)}$  (если данное ребро это петля).

У ориентированного графа  $E \subseteq V^2$ , т.е. каждое ребро это упорядоченная пара вида  $\langle u, v \rangle$ . Ребра ориентированного графа также называют дугами.

**Def 1.1.4.** Петля это ребро, которое соединяет вершину саму с собой.

**Def 1.1.5.** Мультиребра это ребра, у которых общее начало и общий конец.

**Def 1.1.6.** Граф называется простым, если в нем нет мультиребер и петель.

**Def 1.1.7.** Мультиграф это граф с мультиребрами.

**Def 1.1.8.** Псевдограф это мультиграф с петлями.

**Def 1.1.9.** Гиперграф это граф, в котором ребро может соединять несколько вершин одновременно.

**Def 1.1.10.** Нулевой граф это граф, который не содержит вершин.

**Def 1.1.11.** Пустой граф это граф, которые не содержит ребер.

**Def 1.1.12.** Граф-синглтон (тривиальный граф) это граф состоящий только из одной вершины.

**Def 1.1.13.** Полный граф  $K_n$  это простой граф, в котором каждая пара различных вершин соединена ребром.

**Def 1.1.14.** Взвешенный граф  $G = \langle V, E, w \rangle$  это граф, в котором каждому ребру сопоставляется некоторое числовое значение (вес), которое определяется весовой функцией  $w: E \rightarrow \text{Num}$ .

**Def 1.1.15.** Подграфом графа  $G = \langle V, E \rangle$  называется граф  $G' = \langle V', E' \rangle$  такой, что  $V' \subseteq V$  и  $E' \subseteq E$ .

**Def 1.1.16.** Подграф называется остовным, если он содержит все вершины исходного графа.

**Def 1.1.17.** Индуцированный подграф это подграф, который получается из некоторого подмножества вершин исходного графа  $V' \subseteq V$  и **всех** ребер исходного графа, соединяющих эти вершины.

**Def 1.1.18.** Две вершины называются смежными (соседними), если между ними есть ребро.

Для иллюстрации отношения смежности обычно используют матрицу смежности. Это квадратная матрица размера  $V \times V$ , ячейки которой содержат:

- 0 или 1 — для простых графов
- $-1, 0, 1$  — для ориентированных графов
- $\mathbb{N}$  — для взвешенных графов

Однако описанные выше правила не строгие: в разных задачах числа в матрице смежности могут обозначать разные вещи.

**Def 1.1.19.** Вершина и ребро называются инцидентными, если вершина является одним из концов ребра.

Для иллюстрации отношения инцидентности обычно используют матрицу инцидентности. Это прямоугольная матрица размера  $V \times E$ , строки которой соответствуют вершинам, а столбцы — ребра. Ячейки этой матрицы могут содержать:

- 1 если вершина и ребро инцидентны и 0 в противном случае — для неориентированных графов.
- $-1$ , если ребро выходит из данной вершин, и 1 если входит, 0 если ребро и вершина неинцидентны — для ориентированных графов.

Петля в матрице инцидентности обычно обозначается двойкой. Но, как и в случае с матрицей смежности, описанные правила не являются строгими.

**Def 1.1.20.** Степенью вершины  $\deg u$  называется количество инцидентных ей ребер (петли учитываются дважды).

Со степенью вершины также связаны такие понятия как:

- Минимальная степень вершины в графе  $\delta(G) = \min_{v \in V} \deg v$ .
- Максимальная степень вершины в графе  $\Delta(G) = \max_{v \in V} \deg v$ .

**Lm 1.1.21.** Лемма о рукопожатиях.

$$\sum_{v \in V} \deg v = 2|E|$$

*Доказательство.* Т.к. каждое ребро инцидентно ровно двум ребрам, то сложив все степени вершин мы учтем каждое ребро дважды (по одному разу для каждой из его концевых вершин). ■

**Def 1.1.22.** Граф называется  $r$ -регулярным, если степень каждой из его вершин равна  $r$ .

**Def 1.1.23.** Граф называется планарным, если его можно изобразить на плоскости без пересечения ребер.

**Def 1.1.24.** Граф называется плоским, если он *уже* изображен на плоскости без пересечения ребер.

**Теорема 1.1.25.** Теорема Эйлера для графов

Для планарных графов выполняется соотношение

$$|V| - |E| + |F| = 2$$

где  $|V|$  — количество вершин,  $|E|$  — количество ребер, а  $|F|$  — количество граней.

*Доказательство.* Индукция по количеству граней.

**База:**  $F = 2$ . Тогда граф представляет собой многоугольник с  $|V|$  вершинами и при этом  $|V| = |E| \implies$  равенство выполняется.

**Переход:** пусть теорема верна для графа в котором  $F$  граней. Добавим новую грань, для этого проведем по внешней границе некоторую простую цепь, которая будет содержать  $r - 1$  вершин и  $r$  ребер. Подставим эти изменения в формулу и упростим:

$$\begin{aligned} (|V| + r - 1) - (|E| + r) + (|F| + 1) &\stackrel{?}{=} 2 \\ |V| - |E| + |F| &\stackrel{?}{=} 2 \end{aligned}$$

Это верно по предположению индукции.

**TODO:** figure ■

**Def 1.1.26.** Клика это множество вершин в графе, которые индуцируют полный подграф.

**Def 1.1.27.** Независимое (стабильное) множество это множество вершин графа такое, что любые две вершины в нем не смежны.

**Def 1.1.28.** Паросочетание это множество ребер графа такое, что любые два ребра в нем не смежны.

**Def 1.1.29.** Совершенное (идеальное) паросочетание это такое паросочетание, что любая вершина в графе инцидентна некоторому ребру из него.

**Def 1.1.30.** Вершинное покрытие это множество вершин таких, что любое ребро в графе инцидентно хотя бы одной вершине из этого множества.

**Def 1.1.31.** Реберное покрытие это множество ребер таких, что любая вершина инцидентна хотя бы одному ребру из этого множества.

## 1.2. Морфизмы графов.

**Def 1.2.1.** Два графа называются изоморфными, если между их вершинами существует биекция, причем две вершины в первом графе смежны тогда и только тогда, когда смежны соответствующие им (по биекции) вершины второго графа.

**Def 1.2.2.** Автоморфизм графа это отображение множества вершин графа в себя, сохраняющее смежность.

*Замечание 1.2.3.* Граф, для которого единственный возможный автоморфизм это тождественное отображение называется асимметрическим.

**Def 1.2.4.** Гомоморфизм графа это отображение одного графа в другой, которое сохраняет смежность вершин.

**Def 1.2.5.** Подразделением графа  $G$  называется граф  $G'$ , полученный делением ребер графа  $G$  новыми вершинами.

**Def 1.2.6.** Два графа  $G_1$  и  $G_2$  называются гомеоморфными, если существует изоморфизм некоторого подразделения графа  $G_1$  и некоторого подразделения графа  $G_2$

**TODO:** figures as examples

### 1.3. Пути и расстояния.

**Def 1.3.1.** Маршрут (*walk*) это чередующаяся последовательность из вершин и ребер.

**Def 1.3.2.** Путь (*trail*) это маршрут без повторяющихся ребер.

**Def 1.3.3.** Цепь (*path*) это маршрут без повторяющихся вершин.

*Замечание 1.3.4.* Для приведенных выше терминов существуют замкнутые версии (т.е. начальная и конечная вершины совпадают):

- Замкнутый маршрут (*closed walk*)
- Замкнутый путь (*closed trail, circuit*)
- Замкнутая цепь/цикл (*closed path, cycle*)

**Def 1.3.5.** Для маршрута (пути, цепи) это количество ребер в нем.

**Def 1.3.6.** Обхват (*girth*) это длина кратчайшего цикла в графе.

**Def 1.3.7.** Расстояние  $\text{dist}(u, v)$  между двумя вершинами  $u$  и  $v$  это длина кратчайшего пути  $u \rightsquigarrow v$ .

**Def 1.3.8.** Длина наибольшего из кратчайших расстояний от данной вершины  $v$  до остальных вершин называется эксцентриситетом  $\varepsilon(v) = \max_{u \in V} \text{dist}(u, v)$

**Def 1.3.9.** Радиусом графа называется минимальный из эксцентриситетов его вершин  $\text{rad}(G) = \min_{v \in V} \varepsilon(v)$ .

**Def 1.3.10.** Диаметром графа называется максимальный из эксцентриситетов его вершин  $\text{diam}(G) = \max_{v \in V} \varepsilon(v)$ .

**Def 1.3.11.** Множество вершин, эксцентриситет которых равен радиусу называются центром графа  $\text{center}(G) = \{v \mid \varepsilon(v) = \text{rad}(G)\}$ .

### 1.4. Эйлеровы графы.

**Def 1.4.1.** Эйлеров путь это путь, который проходит все ребра графа ровно по одному разу.

**Def 1.4.2.** Эйлеров цикл это эйлеров путь, у которого начальная вершина совпадает с конечной.

**Def 1.4.3.** Граф называется эйлеровым, если в нем есть эйлеров цикл, и полуэйлеровым, если в нем есть только эйлеров путь.

**Теорема 1.4.4.** Условие существования Эйлерова цикла

Эйлеров цикл существует тогда и только тогда, когда степени всех вершин четны.

*Доказательство.* Индукция по количеству ребер

**База:**  $m = 0$ , т.к. ребер, то считаем, что Эйлеров цикл есть

**Переход:** пусть теорема верна для графов с менее чем  $m$  ребрами. Т.к. все степени четны, то войдя в вершины мы всегда сможем выйти из неё (за исключением стартовой вершины)  $\implies$  будем идти по ребрам в случайном порядке пока не вернемся в стартовую вершину. Далее возможны две ситуации:

- Если мы обошли все ребра, то теорема верна
- Иначе удалим из графа все пройденные ребра. Он распадется на компоненты связности, в которых будет меньше ребер, чем в исходном графе  $\implies$  к ним применимо предположение индукции. Найдем в каждой компоненте Эйлеров цикл, после чего 'прикрепим' его к первоначально найденному циклу  $\implies$  получим Эйлеров цикл.

**TODO:** figures

*Замечание 1.4.5.* Для существования Эйлерова пути необходимо и достаточно, чтобы количество вершин с нечетной степенью не превышало двух. В этом случае можно соединить эти вершины дополнительным ребром и построить Эйлеров цикл (т.к. в графе все степени вершин станут четными). После чего достаточно удалить добавленное ребро, чтобы получить Эйлеров путь. ■

## 1.5. Гамильтоновы графы.

**Def 1.5.1.** Гамильтонов путь это путь, который проходит все вершины в графе ровно по одному разу.

**Def 1.5.2.** Гамильтонов цикл это гамильтонов путь, у которого начальная вершина совпадает с конечной.

Несмотря на то, что каждая вершина должна быть пройдена ровно один раз, для стартовой вершины делается исключение.

**Def 1.5.3.** Граф называется гамильтоновым, если в нем есть гамильтонов цикл, и полугамильтоновым, если в нем есть только гамильтонов путь.

**Теорема 1.5.4.** Теорема Оре.

Если  $n \geq 3$  и  $\deg u + \deg v \geq n$  для любых двух **несмежных** вершин неориентированного графа  $G$ , то этот граф гамильтонов.

*Доказательство.* От противного: пусть дан граф, удовлетворяющий условиям теоремы, но не являющийся гамильтоновым. Будем добавлять к нему ребра до тех пор, пока он будет оставаться не гамильтоновым. Т.к. мы только добавляем ребра, то условия теоремы не нарушатся.

Пусть в полученном графе есть две несмежные вершины  $u$  и  $v$  такие, что добавление ребра  $(u, v)$  приводит к появлению гамильтонова цикла. Это значит, что путь  $u \rightsquigarrow v$  гамильтонов. Обозначим вершины на этом пути  $t_j$ . Рассмотрим два множества:

- $S = \{i \mid \exists (u, t_{i+1}) \in E\}$
- $T = \{i \mid \exists (t_i, v) \in E\}$

Тогда  $|S| + |T| = \deg u + \deg v \geq n$ , при этом  $|S \cup T| < n$ . Из этого следует, что  $|S \cap T| = |S| + |T| - |S \cup T| > 0$ , т.е. найдутся две смежные вершины  $t_1$  и  $t_2$  такие, что будут существовать ребра  $(u, t_2)$  и  $(t_1, v)$ .

Построим гамильтонов цикл:  $u \rightsquigarrow t_1 \rightarrow v \rightsquigarrow t_2 \rightarrow u$ . Получаем противоречие: предполагалось, что граф не гамильтонов, однако мы смогли построить гамильтонов цикл. ■

**Теорема 1.5.5.** Теорема Дирака.

Если  $|V| \geq 3$  и  $\delta(G) \geq \frac{n}{2}$ , то неориентированный граф  $G$  — гамильтонов.

*Доказательство.* Возьмем две произвольные несмежные вершины  $u$  и  $v$ , тогда  $\deg u + \deg v \geq \frac{n}{2} + \frac{n}{2} = n$ . Значит по т. Оре (1.5.4) граф  $G$  гамильтонов. ■

*Замечание 1.5.6.* Эти две теоремы являются достаточными, но не необходимыми условиями гамильтоновости графа, т.е. если они не выполняются, то это не значит, что граф не гамильтонов.

**TODO:** figure example

## 1.6. Двудольные графы.

**Def 1.6.1.** Двудольный граф это граф, вершины которого могут быть разбиты на два множества (две доли) так, что между вершинами каждого из множеств не будет ребер.

*Замечание 1.6.2.* Об обозначениях.

Двудольный граф обычно обозначается как  $G = \langle X, Y, E \rangle$ , где  $X$  и  $Y$  это его доли, а  $E$  — множество ребер между этими долями.

Пусть задано некоторое подмножество  $S \subseteq X$ , тогда  $E(S)$  это множество ребер, инцидентных вершинам из множества  $S$ .

Обозначение  $N(S)$  используется что показать 'соседей' для вершин из множества  $S$ , т.е. такие вершины из множества  $Y$ , которые смежны хотя бы с одной из вершин из множества  $S$ .

**Def 1.6.3.** Двудольный граф называется сбалансированным, если размеры его долей совпадают.

**Теорема 1.6.4.**  $r$ -регулярный двудольный граф является сбалансированным.

*Доказательство.* Пусть дан двудольный граф  $G = \langle X, Y, E \rangle$ . Т.к. он двудольный, то множество ребер может быть представлено в виде  $|E| = |X| \cdot r$ . С другой стороны оно может быть представлено в виде  $|E| = |Y| \cdot r$ . Приравниваем и получаем искомое равенство  $|X| = |Y|$ . ■

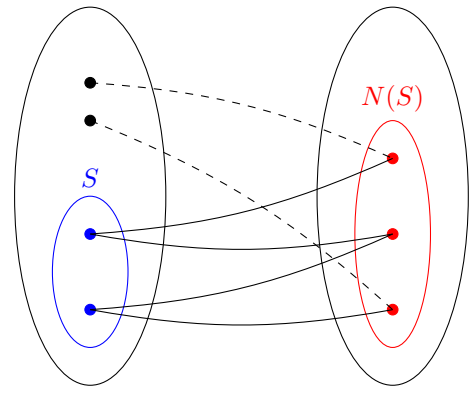
**Теорема 1.6.5.** В любом регулярном двудольном графе существует совершенное паросочетание.



*Доказательство.* Рассмотрим двудольный граф  $G = \langle X, Y, E \rangle$ . Выберем произвольное  $S \subseteq X$ . Заметим, что  $E(N(S)) \geq E(S)$ , причем т.к. граф регулярный, то  $E(N(S)) = |N(S)| \cdot r$ , а  $E(S) = |S| \cdot r$ .

Подставляя это в полученное ранее неравенство, получаем, что  $|N(S)| \geq |S|$  причем это выполняется для любого  $S \subseteq X$ . Значит по т. Холла (1.7.1) в графе существует  $X$ -совершенное паросочетание.

Т.к.  $G$  — регулярный двудольный граф, то по 1.6.4 он сбалансирован, значит  $|X| = |Y|$ . Таким образом  $X$ -совершенное паросочетание является совершенным паросочетанием. ■

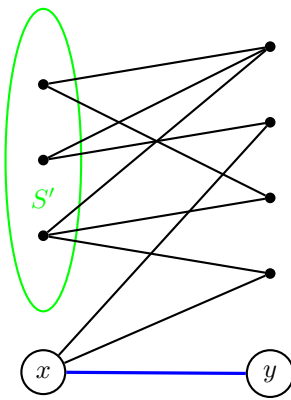


## 1.7. Теорема Холла.

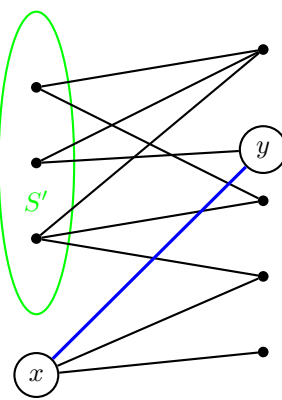
**Теорема 1.7.1.** Теорема Холла.

Пусть дан двудольный граф  $G = \langle X, Y, E \rangle$ . В нем существует  $X$ -совершенное паросочетание тогда и только тогда, когда

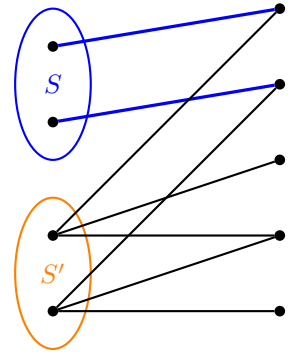
$$\forall S \subseteq X: |N(S)| \geq |S|$$



(a) 1ый случай (a)



(b) 1ый случай (b)



(c) 2ой случай

*Доказательство.*  $\Rightarrow$  Если в графе существует  $X$ -совершенное паросочетание, то для любого  $x \in X$  существует уникальный сосед в  $Y$ , значит количество соседей у любого  $S \subseteq X$  будет как минимум  $|x|$ .

$\Leftarrow$  Индукция по  $|X|$ .

**База:**  $|X| = 1$ . Очевидно, что  $X$ -совершенное паросочетание существует тогда и только тогда, когда у единственной вершине в  $X$  будет как минимум один сосед из  $Y$ .

**Переход:** пусть теорема верна при  $|X| < n$ . Рассмотрим граф, удовлетворяющий условиям теоремы, в котором  $|X| = n$ . Возможны два случая:

1.  $\forall S \subseteq X: |N(S)| > |S|$

Выберем произвольный  $x \in S$  и поставим ему в пару любого из его соседей. Обозначим этого соседа  $y$ , а оставшиеся в  $S$  вершины  $S'$ .

Т.к. до выбора соседа для  $x$  выполнялось неравенство  $|N(S')| > |S'|$ , то после выбора соседа выполняется неравенство  $|N(S')| \geq |S'|$ , т.к. количество свободных соседей  $S'$  могло уменьшиться не более чем на один (оно уменьшилось на один в случае, если  $y \in N(S')$ ).

Пары для оставшихся вершин мы можем найти по предположению индукции.

2.  $\exists S \subseteq X: |N(S)| = |S|$

Каждому  $x \in S$  по предположению индукции поставим в пару одного из его соседей. Рассмотрим оставшиеся вершины  $S' = X \setminus S$ . Покажем, что для них тоже можно найти пару.

Пусть найдется такое  $P \subseteq S'$ , что  $|N(P)| < |P|$ . Тогда рассмотрим множество  $S \cup P$ . Т.к.  $|N(S)| = |S|$  и  $|N(P)| < |P|$ , то  $|N(S \cup P)| < |S \cup P|$ . Это противоречит условию теоремы, значит  $\forall P \subseteq S': |N(P)| \geq |P|$ , поэтому по предположению индукции мы можем найти пары для всех  $x \in S'$ . ■

## 1.8. Теорема Татта.

**TODO:**

## 1.9. Связность.

**Def 1.9.1.** Две вершины неориентированного графа  $u$  и  $v$  называются связными, если существует путь из  $u$  в  $v$ .

*Замечание 1.9.2.* Связность это отношение эквивалентности, а классы эквивалентности этого отношения называются компонентами связности графа.

**Def 1.9.3.** Неориентированный граф называется связным, если он состоит из одной компоненты связности.

**Def 1.9.4.** Две вершины  $u$  и  $v$  ориентированного графа называются слабо связными, если существует путь из  $u$  в  $v$  без учета ориентации ребер.

*Замечание 1.9.5.* Слабая связность это отношение эквивалентности, а классы эквивалентности этого отношения называются компонентами слабой связности.

**Def 1.9.6.** Ориентированный граф называется слабо связным, если он состоит из одной компоненты слабой связности.

**Def 1.9.7.** Две вершины  $u$  и  $v$  ориентированного графа называются сильно связными, если существует путь из  $u$  в  $v$  и путь из  $v$  в  $u$ .

*Замечание 1.9.8.* Сильная связность это отношение эквивалентности, а классы эквивалентности этого отношения называются компонентами сильной связности.

**Def 1.9.9.** Ориентированный граф называется сильно связным, если он состоит из одной компоненты сильной связности.

**Def 1.9.10.** Конденсацией ориентированного графа  $G$  называется граф  $G'$ , вершины которого соответствуют компонентам сильной связности графа  $G$ , а ребра между вершинами существуют лишь в том случае, если между соответствующими компонентами сильной связности есть ребро.

**Def 1.9.11.** Вершинная связность  $\kappa(G)$  это минимальное число вершин, которое необходимо удалить для того, чтобы граф стал несвязным или тривиальным.

**Def 1.9.12.** Вершинная двусвязность это бинарное отношение на ребрах. Два ребра называются двусвязными, если существует два вершинно-независимых пути между концами этих ребер.

**Def 1.9.13.** Вершинная двусвязность это отношение эквивалентности, а классы эквивалентности этого отношения называются вершинно-двусвязными компонентами или *блоками*.

**Def 1.9.14.** Граф называется  $k$ -вершинно связным, если после удаления менее чем  $k$  вершин он остается связным, т.е.  $\kappa(G) \geq k$ .

**Def 1.9.15.** Реберная связность  $\lambda(G)$  это минимальное число ребер, которое необходимо удалить для того, чтобы граф стал несвязным или тривиальным.

**Def 1.9.16.** Реберная двусвязность это бинарное отношение на вершинах. Две вершины называются реберно двусвязными, если между ними существует два реберно-независимых пути.

**Def 1.9.17.** Реберная двусвязность это отношение эквивалентности, а классы эквивалентности этого отношения называются реберно-двусвязными компонентами.

**Def 1.9.18.** Граф называется  $k$ -реберно связным, если после удаления менее чем  $k$  ребер он остается связным, т.е.  $\lambda(G) \geq k$ .

*Замечание 1.9.19.* Крайние случаи:

- $K_1$  не является 1-вершинно связным, но считается связным.
- $K_1$  не является 2-реберно связным, но считается реберно-двусвязным. Т.е. одна вершина может считаться компонентой реберной двусвязности.
- $K_2$  не является 2-вершинно связным, но считается вершинно-двусвязным, поэтому  $K_2$  может быть блоком.

**Def 1.9.20.** Вершина называется точкой сочленения (шарниром), если её удаление увеличивает число компонент связности графа.

**Def 1.9.21.** Ребро называется мостом, если его удаление увеличивает число компонент связности графа.

*Замечание 1.9.22.* Таким образом точки сочленения являются 'границами' между блоками (но при этом в блоки они входят), а мосты — между компонентами реберной-двусвязности (но в сами компоненты они не входят).

**Def 1.9.23.** Дерево блоков-точек сочленения это двудольный граф, в котором в одной доле находятся вершины соответствующие блокам, а в другой — точки сочленения.

Если точка сочленения принадлежит блоку, то между ними будет ребро, в противном случае — нет.

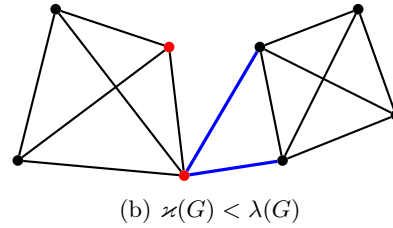
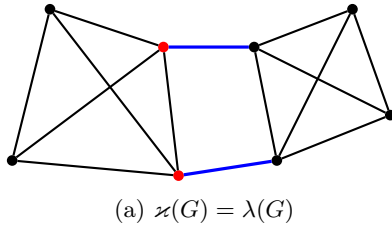


## 1.10. Теорема Уитни.

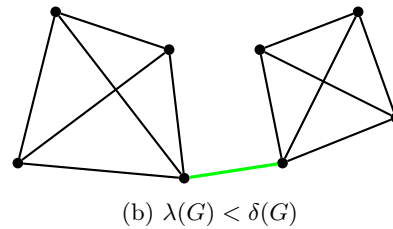
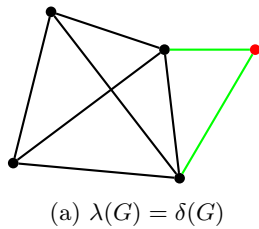
### Теорема 1.10.1.

$$\kappa(G) \leq \lambda(G) \leq \delta(G)$$

*Доказательство.* Сначала докажем левую часть этого неравенства. Рассмотрим минимальное по включению множество ребер  $E_d$ , которые необходимо удалить для того, чтобы граф перестал быть связным.



В худшем случае придется удалить  $|E_d| = \lambda(G)$  вершин (по одной вершине на каждое ребро) чтобы граф перестал быть связным. Однако если существует вершины, при удалении которых удалится несколько ребер из  $E_d$ , то можно будет удалить меньше вершин для достижения несвязности графа. Значит  $\kappa(G) \leq \lambda(G)$ .



Далее докажем правую часть этого неравенства. В худшем случае, чтобы получить несвязный граф, потребуется удалить все ребра инцидентные вершине с наименьшей степенью. В этом случае  $\lambda(G) = \delta(G)$ . Однако бывают графы, в которых  $\lambda(G) < \delta(G)$ . ■

## 1.11. Теорема Менгера.

**Def 1.11.1.**  $uv$ -отделяющее множество это множество вершин, которые необходимо удалить, чтобы между  $u$  и  $v$  не было пути.

**Def 1.11.2.**  $uv$ -разделяющее множество это множество ребер, которые необходимо удалить, чтобы между  $u$  и  $v$  не было пути.

**Теорема 1.11.3.** Для любой пары несмежных вершин  $u$  и  $v$  в неориентированном графе максимальное число внутренне вершинно-независимых путей из  $u$  в  $v$  равно минимальной мощности  $uv$ -отделяющего множества.

*Доказательство.* В ходе доказательства будем использовать следующие сокращения:

- ВНП := внутренне вершинно независимый путь.
- ОМ :=  $uv$ -отделяющее множество минимальной мощности.

Сразу оговоримся, что если  $u$  и  $v$  несвязны, то мощность ОМ равна нулю  $\implies$  теорема верна. Далее будем считать, что  $u$  и  $v$  связны. Обозначим  $S = \{s_1, \dots, s_n\}$  — ОМ и  $P_1, \dots, P_n$  — множество ВНП.

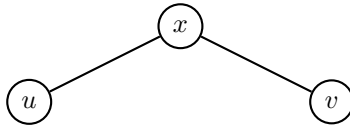
Индукция по количеству вершин в графе.

**База:**  $n = 3$ , тогда мощность ОМ равна единице и существует один ВНП.

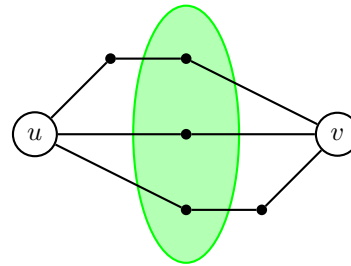
**Переход:** пусть утверждение теоремы верно для графов, у которых менее  $n$  вершин. Рассмотрим граф с  $n$  вершинами. Возможны три случая:

1. В графе есть вершина  $x$ , которая смежна и с  $u$ , и с  $v$ .

Для графа без этой вершины будет выполняться предположение индукции. Если же добавить эту вершину в граф, то мощность ОМ увеличится на единицу и при этом появится еще один ВНП  $u \rightarrow x \rightarrow v$ .



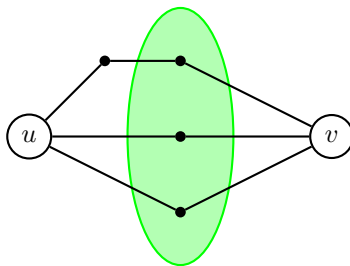
(a) 1ый случай



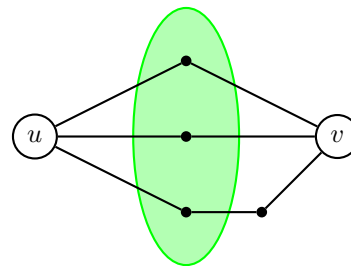
(b) 2ой случай

2. Существует ОМ, которое содержит вершину не смежную с  $u$  и вершину, не смежную с  $v$ . Удаление всех вершин из ОМ разобьет его на подграфа  $A$  и  $B$ . Из графа  $A$  построим граф  $G_1$  по следующим правилам:

- Восстановим все вершины из ОМ
- Восстановим все ребра, которые соединяли подграф  $A$  и ОМ
- Восстановим вершину  $v$
- Добавим ребра из  $v$  в каждую из вершин ОМ



(a) Граф  $G_1$

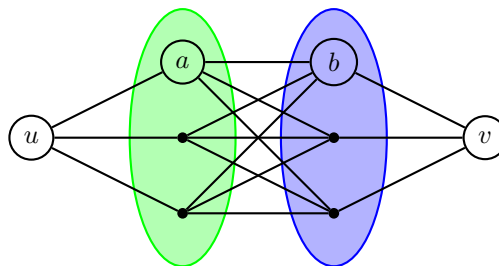


(b) Граф  $G_2$

Т.к. по условиям пункта в ОМ существовала вершина, не смежная с  $v$ , то в полученном графе  $G_1$  будет меньше вершин, чем в изначальном. Применив к нему предположение индукции получим  $k$  ВНП (по количеству вершин в ОМ) вида  $u \rightsquigarrow s_i \rightarrow v$ .

Аналогично из подграфа  $B$  можно построить граф  $G_2$  и найти в нем  $k$  ВНП вида  $u \rightarrow s_i \rightsquigarrow v$ . Из двух полученных множеств путей очевидным образом составим искомые ВНП вида  $u \rightsquigarrow s_i \rightsquigarrow v$ .

3. В любой ОМ все вершины смежны с  $u$  и не смежны с  $v$  или наоборот.



Если какая либо вершина не входит ни в одно ОМ, то её удаление не повлияет на количество ВНП. Значит её можно удалить и применить предположение индукции, поэтому далее будет считать, что каждая вершина принадлежит некоторому ОМ.

Среди вершин, смежных с  $u$  найдется вершина  $a$ , которая будет смежна с вершиной  $b \neq u$ , причем  $b$  будет не смежна с  $u$ . Если  $b$  не смежна с  $u$ , то она смежна с  $v$  по условиям пункта. Получем путь  $u \rightarrow a \rightarrow b \rightarrow v$ .

Если удалить вершины  $a$  и  $b$ , то к полученному графу будет применимо предположение индукции. Возвращая вершины  $a$ ,  $b$  в граф мы добавляем один ВНП  $u \rightarrow a \rightarrow b \rightarrow v$  и увеличиваем мощность ОМ на один (в него нужно будет взять либо  $a$ , либо  $b$ ).

■

*Замечание 1.11.4.* Приведенная формулировка и доказательство являются 'вершинной' формой теоремы Менгера. Существует также и реберная форма теоремы Менгера: для любой пары несмежных вершин  $u$  и  $v$  количество реберно-независимых путей из  $u$  в  $v$  равно минимальной мощности  $uv$ -разделяющего множества.

## 1.12. Деревья.

**Def 1.12.1.** Дерево это связный ациклический граф.

**Def 1.12.2.** Лес это ациклический граф, т.е. объединение непересекающихся деревьев.

**Def 1.12.3.** Укорененное дерево это дерево, в котором одна из вершин обозначена корнем.

**Def 1.12.4.** Предком вершины называется следующая вершина на кратчайшем пути к корню.

**Def 1.12.5.** Если  $u$  — предок  $v$ , то  $v$  называется ребенком  $u$ .

**Def 1.12.6.** Если у двух вершин один и тот же родитель, то они называются братьями.

**Def 1.12.7.** Лист это вершина, у которой нет детей.

**Def 1.12.8.** Вершины, не являющиеся листьями, называются внутренними.

**Def 1.12.9.** Дерево называется  $k$ -арным, если все его вершины имеют не более  $k$  детей. Если  $k = 2$ , то такое дерево называют бинарным.

**Def 1.12.10.** Веткой вершины в дереве называется некоторый индуцированный подграф такой, что данная вершина является в нем корнем.

**Def 1.12.11.** Вес ветки это сумма весов всех входящих в него ребер. Если дерево невзвешенное, то считаем вес каждого ребра равным единице.

**Def 1.12.12.** Весом вершины называется максимальный из весов веток этой вершины.

**Def 1.12.13.** Центроид дерева это множество вершин с минимальным весом.

**Lm 1.12.14.** Центроид дерева содержит либо одну, либо две вершины.

*Доказательство.* Рассмотрим невзвешенное дерево. На каждом шаге будем убирать все листья из текущего дерева, при этом степени оставшихся вершин будут уменьшаться на единицу. Будем так повторять, пока существует вершина со степенью более единицы.

В итоге останется либо одна вершина со степенью ноль, либо две вершины со степенью один. Значит в исходном графе именно у этой вершины (или у этих двух вершин) была наименьшая степень. ■

**Def 1.12.15.** Дерево называется помеченным, если каждой из его вершин соответствует уникальная метка от 1 до  $n$ .

**Def 1.12.16.** Код Прюфера это уникальная последовательность меток от 1 до  $n$  длины  $n - 2$ , которая соответствует уникальному помеченному дереву на  $n$  вершинах.

**Lm 1.12.17.** Всего существует  $n^{n-2}$  помеченных деревьев на  $n$  вершинах.

*Доказательство.* Между деревьями на  $n$  вершинах и кодами Прюфера длина  $n - 2$  действует биекция. Существует  $n^{n-2}$  кодов Прюфера длины  $n - 2$ . ■

Кодирование [визуализация]:

1. Выбираем лист с наименьшей меткой.
2. Добавляем в ответ его родителя, после чего удаляем этот лист.
3. Повторяем шаги 1 – 2  $n - 2$  раза.

Декодирование [визуализация]:

1. Выписываем в строку все метки от 1 до  $n$ .
2. Среди выписанных меток ищем первую, которой нет в коде Прюфера.
3. Добавляем в граф ребро между найденной меткой и первой меткой в коде Прюфера.
4. Удаляем найденную метку из последовательности меток, а также удаляем первый элемент в коде Прюфера.
5. Повторяем шаги 2 – 4 пока не закончится код Прюфера.
6. В последовательности останется 2 метки, добавляем ребро между ними.

*Замечание 1.12.18.* Кодировать и декодировать код Прюфера можно за  $O(n)$ . Алгоритм описан [здесь](#).

### 1.13. Алгоритм Дейкстры.

Цель: ищем кратчайшие пути от одной вершины до всех остальных в графе с неотрицательными весами.

Оценка по времени зависит от реализации приоритетной очереди:

- Наивная  $O(V^2 + E)$
- На бинарной куче  $O(V + E \log V)$
- На Фибоначчиевой куче  $O(V \log V + E)$

Алгоритм:

1. Будем поддерживать множество вершин  $A$ , расстояние до которых минимально.
2. Изначально  $A = \emptyset$ ,  $\forall v \neq s \in V: d_v = +\infty$ ,  $d_s = 0$ , где  $d_x$  это расстояние от стартовой вершины до вершины  $x$ .
3. Из непосещенных вершин выберем вершину  $u$ , у для которой  $d_u$  минимально.
4. Добавим  $u$  в  $A$ , причем  $d_u$  будет являться кратчайшим расстоянием до вершины  $u$ .
5. Далее будем обновлять расстояние до непосещенных вершин, смежных с выбранной:  
 $\forall (u, p) \in E: d_p = \min(d_p, d_u + w_{up})$ , где  $w_{up}$  это вес ребра между вершинами  $u$  и  $p$ .
6. Будем повторять шаги 3 – 5 пока в  $A$  не попадут все достижимые вершины (либо в  $A$  не попадет финишная вершина).

О корректности:

Корректность покажем по индукции.

**База:** согласно шагу 2 расстояние до стартовой вершины  $s$  равно нулю.

**Переход:** пусть  $\forall a \in A: d_a = \text{dist}(a)$  это действительно кратчайшие расстояния до вершин в  $A$  (кратчайшие расстояния будем обозначать  $\text{dist}(a)$ , а расстояния, найденные в ходе работы алгоритма —  $d_a$ ). Покажем, что  $d_u = \min_{a \in A} (d_a + w_{au})$  это кратчайшее расстояние до  $u$ .

Рассмотрим кратчайший путь  $s \rightsquigarrow u$ . Пусть  $a \in A$  это последняя вершина на этом пути, которая принадлежит  $A$ , а  $b$  — это следующая за ней вершина на пути в  $u$ . Тогда  $\text{dist}(u) \geq d_a + w_{ab}$ .

Но т.к.  $d_u = \min_{a \in A} (d_a + w_{au})$ , то  $d_a + w_{ab} \geq d_u$ , значит  $\text{dist}(u) \geq d_u$ . Итого  $d_u$  это кратчайшее расстояние до  $u$ .

#### 1.14. Алгоритм Форда-Белмана.

Цель: ищем кратчайшие пути от одной вершины до всех остальных (допускаются отрицательные ребра).

Оценка по времени:  $O(VE)$

Алгоритм:

1. Сделаем  $|V| - 1$  итерацию (именно столько, т.к. в кратчайшем пути не более  $|V| - 1$  ребра).
2. На каждой итерации будем проходить по всем ребрам и релаксировать их.
3. Релаксация ребра выглядит так: пусть дано ребро  $(u, v)$ . Тогда обновим расстояние до  $v$  следующим образом:  
 $d_v = \min(d_v, d_u + w_{uv})$ .

О корректности:

Построим следующую динамику: пусть  $d[k, v]$  это длина кратчайшего пути из стартовой вершины  $s$  в вершину  $v$ , содержащего не более  $k$  ребер. Тогда получаем:

- Изначально:
  - $\forall v \neq s \in V: d[0, v] = +\infty$
  - $d[0, s] = 0$
  - $\forall v \in V, \forall i > 0: d[i, v] = +\infty$
- Переход  $d[k + 1, v] = \min \left( d[k, v], \min_{u \in V} (d[k, u] + w_{uv}) \right)$

Он вытекает из следующих соображений: пусть мы хотим получить кратчайший путь  $s \rightsquigarrow v$ , который состоит из не более чем  $k + 1$  ребра. Возможно два случая:

- Можно взять длины кратчайшего пути, который состоит из не более чем  $k$  ребер.
- Можно найти путь в некоторую смежную вершину  $u$ , который содержит не более  $k$  ребер и присоединить к нему ребро  $(u, v)$ . Среди всех таких вершин  $u$  берем такую, чтобы длина итогового пути была наименьшей.

Из этих двух случаев мы выбираем наименьший.

Динамика корректна по построению, а алгоритм Форда-Белмана это следствие из этой динамики: мы убираем индекс  $k$ , чтобы оптимизировать затрачиваемую память.

### 1.15. Алгоритм Флойда-Уоршелла.

Цель: ищем кратчайшие пути от всех вершин до всех.

Оценка по времени:  $O(V^3)$

Алгоритм:

1. Рассматриваем каждую из вершин  $k \in V$ .
2. На каждой итерации проходим по всем возможным парам вершин  $\langle i, j \rangle$  и пытаемся прорелаксировать путь  $i \rightsquigarrow j$  через вершину  $k$ .
3. Релаксация пути через вершину  $k$  выглядит так: пусть дана пара вершин  $\langle i, j \rangle$ . Тогда обновим расстояние от  $i$  до  $j$  следующим образом:  $d_{ij} = \min(d_{ij}, d_{ik} + d_{kj})$ .

О корректности:

Построим следующую динамику: пусть  $d[k, i, j]$  это длина кратчайшего пути из вершины  $i$  в вершину  $j$ , содержащего только вершины с индексами меньше  $k$ . Тогда получаем:

- Изначально:
  - $\forall (i, j) \in E: d[0, i, j] = w_{ij}$
  - $\forall i \in V: d[0, i, i] = 0$
  - Во всех остальных случаях  $d[k, i, j] = +\infty$
- Переход  $d[k+1, i, j] = \min(d[k, i, j], d[k, i, k] + d[k, k, j])$

Он вытекает из следующих соображений:

- Путь  $i \rightsquigarrow j$  нельзя улучшить через вершину  $k$ , тогда  $d[k+1, i, j] = d[k, i, j]$ .
- Путь  $i \rightsquigarrow j$  можно улучшить через вершину  $k$ , тогда  $d[k+1, i, j] = d[k, i, k] + d[k, k, j]$ .

Из этих двух случаев мы выбираем наименьший.

Динамика корректна по построению, а алгоритм Флойда-Уоршелла это следствие из этой динамики: мы убираем индекс  $k$ , чтобы оптимизировать затрачиваемую память.

### 1.16. Иерархическая кластеризация.

**Def 1.16.1.** Иерархическая кластеризация это построение иерархии (дерева) кластеров.

Для визуализации результатов используется дендрограмма — дерево, построенное по матрице расстояний между кластерами. В узлах дерева находятся подмножества объектов из обучающей выборки. Объединения узлов между ярусами дендрограммы соответствует объединению кластеров.

**Алгоритмы WPGMA/UPGMA:**

Задача: построить дерево по матрице расстояний между листьями.

**Def 1.16.2.** Ультраметричность это свойство, согласно которому из трех расстояний между любыми тремя точками два всегда равны и не меньше третьего.

*Замечание 1.16.3.* Полученное дерево уникальное и обладает свойством ультраметричности.

Алгоритм:

1. Ищем две ближайшие вершины  $x$  и  $y$ .
2. В результирующее дерево добавляем вершину  $u$ , которая родителем для двух этих вершин.
3. Вычисляем веса ребер:  $\text{dist}(u, x) = \text{dist}(u, y) = \frac{1}{2} \cdot \text{dist}(x, y)$ .
4. Обновляем матрицу расстояний: убираем из неё  $x, y$  и добавляем  $u$ .
5. Теперь необходимо пересчитать расстояния. В зависимости от алгоритма будут разные формулы пересчета:

(a) UPGMA

Расстояние от кластера до вершины равно среднему арифметическому расстояний от каждой компоненты кластера до этой вершины:

$$u = (x, y)$$

$$\text{dist}(u, z) = \frac{1}{2} \cdot (\text{dist}(u, x) + \text{dist}(u, y))$$

(b) WPGMA

Чтобы найти расстояние от кластера до вершины нужно умножить расстояние от вершины до каждой из компонент кластера на размер этой компоненты. Результаты нужно сложить, после чего поделить на количество вершин в кластере

$$u = (xy, z)$$

$$\text{dist}(u, w) = \frac{1}{2+1} (\text{dist}(xy, w) \cdot 2 + \text{dist}(z, w) \cdot 1)$$

6. Будем повторять шаги 1 – 5 пока матрица не сожмется в одну ячейку.

*Замечание 1.16.4.* В алгоритме WPGMA можно пересчитывать расстояния от вершин до кластера по-другому: можно взять среднее из расстояний от рассматриваемой вершины до каждой из вершин кластера:

$$u = (xy, z) \implies \text{dist}(u, w) = \frac{1}{3}(\text{dist}(x, w) + \text{dist}(y, w) + \text{dist}(z, w))$$

Результат будет такой же, но потребуются дополнительно хранить исходную матрицу расстояний.

**TODO:** Накидать картинок + визуализации (ну может сейчас-то хватит времени на manim?)



## 2. Комбинаторика

2.1. Упорядоченные и неупорядоченные размещения.

2.2. Разбиения и композиции.

2.3. Принцип включения-исключения.

### 3. Конечные автоматы

3.1. Формальные и регулярные языки.

3.2. Детерминированный и недетерминированный конечные автоматы.

3.3. Преобразование НКА в ДКА.

3.4.  $\varepsilon$ -НКА. Преобразование  $\varepsilon$ -НКА в НКА.

3.5. Построение  $\varepsilon$ -НКА по регулярному выражению (построение Томпсона).

3.6. Теорема Клини.

## 4. Рекуррентные соотношения

4.1. Рекуррентные соотношения. Характеристические уравнения.

4.2. Асимптотический анализ.

4.3. Мастер теорема.

4.4. Метод Акра -Бацци.

4.5. Производящие функции.

4.6. Операторы и аннигиляторы.