# Элементы теории графов

# Понятие графа

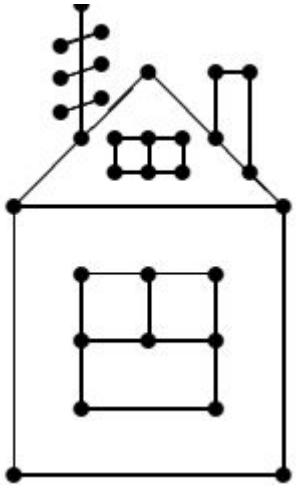
- Графы представляют **объекты и связи между ними**, например: города и дороги, люди и знакомства, атомы и межатомные связи.

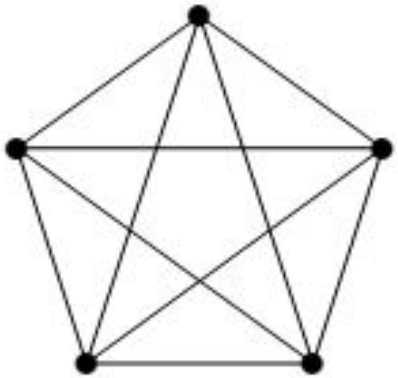
- **Граф** – геометрическая фигура, состоящая из точек и соединяющих их линий.
- Точки называются **вершинами**, а линии — **ребрами**.
- Граф с множеством вершин V и множеством ребер X обозначается через **G (V, X)**.
- Множество всех вершин графа G обозначается через **V(G)**, а множество всех его ребер —**E(G)**.

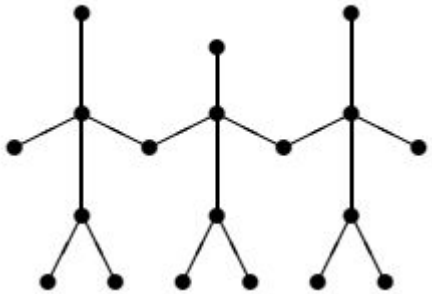
- Множество вершин графа всегда считается непустым, в то время как множество ребер может быть пустым (графы без ребер называются пустыми).
- **Число вершин графа G** обозначается через **n (G )**, а **число его ребер** — через **m (G )**.

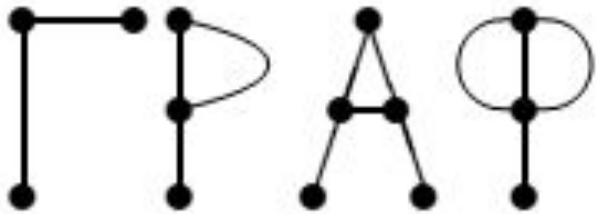
* Мы рассматриваем только графы, содержащие конечное число вершин и конечное число ребер.

# Примеры графов









# Инцидентность и смежность

- Если ребро ***x*** соединяет вершины ***u*** и ***v*** , то говорят, что
  - ✔ вершины ***u*** и ***v*** инцидентны ребру ***x***;
  - ✔ ребро ***x*** инцидентно вершинам ***u*** и ***v***;
  - ✔ вершины ***u*** и ***v*** смежны (при условии $u \neq v$ ).

- Если k ребер (k > 1) инцидентны одной и той же паре вершин, то эти ребра образуют кратное ребро кратности k .

- Ребро, которое связывает вершину саму с собой, называется петлей.

- На рисунке вершины ***u*** и ***v*** соединены ребром кратности 3, на вершине ***w*** имеется петля, а вершина ***z*** изолирована.

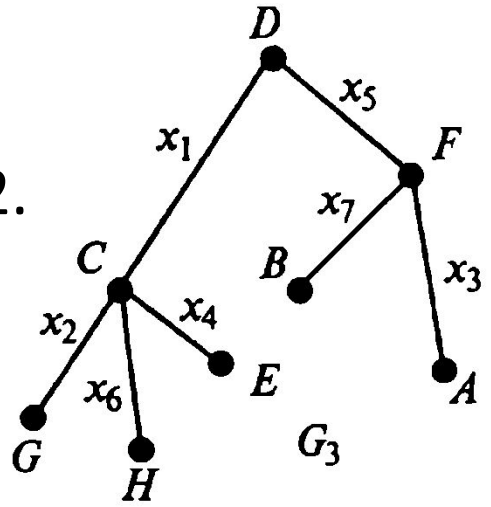

- Граф без кратных ребер и петель называется **обыкновенным**.
- Граф, состоящий из изолированных вершин, называется **нуль-графом.** Для нуль-графа X=0.

# Степень вершины

- Число ребер, инцидентных вершине A, называется степенью вершины A и обозначается deg(A) (от англ. degree — степень).

- Если вершине инцидентна петля, она дает вклад в степень, равный двум, так как оба конца приходят в эту вершину.

- На рисунке deg(A)=1, deg(C)=4, deg(D)=2.
- Вершина графа, имеющая степень, равную 1, называется висячей.



**Теорема 1**.

В графе G(V, X) сумма степеней всех его вершин — число четное, равное удвоенному числу ребер графа:

$$\sum_{i=1}^{n} \deg(V_i) = 2m,$$

- Вершина называется *четной (нечетной)*, если ее степень – *четное (нечетное)* число.

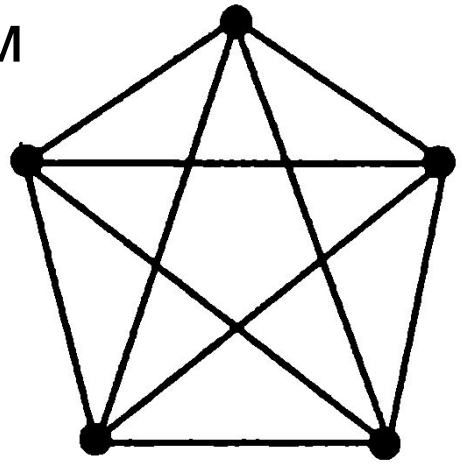
**Теорема 2.**

Число нечетных вершин любого графа — четно.

**Следствие**.

Невозможно начертить граф с нечетным числом нечетных вершин.

- Граф G называется *полным*, если любые две его различные вершины соединены одним и только одним ребром.



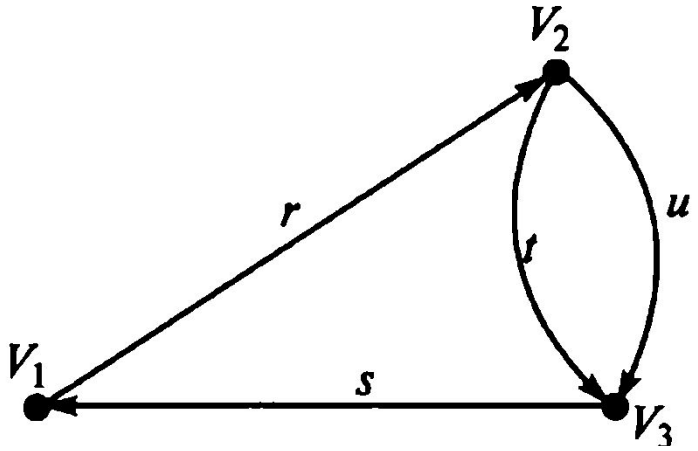
- Таким образом, полный граф определяется только своими вершинами.

- Пусть число вершин полного графа n. Тогда степень любой вершины, равна deg (V) = n - 1, а число ребер равно числу сочетаний из n по $m = C_n^2$

# Орфографы

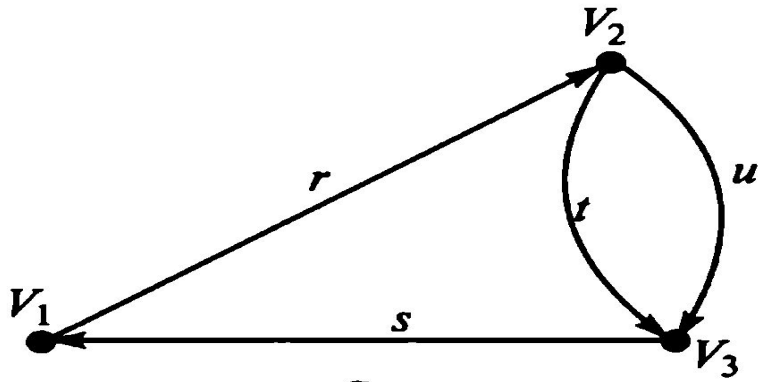
- Если все пары ($V_i$, $V_j$) во множестве X являются упорядоченными, т.е. кортежами длины 2, то граф называется *ориентированным*, *орграфом*, или *направленным*.

- В таком случае ребра принято изображать стрелками.

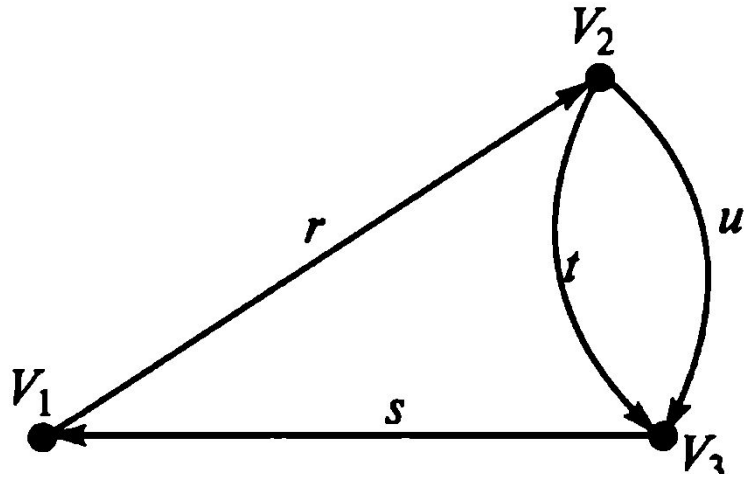


**Степенью** $\frac{\textbf{входа}}{\textbf{выхода}}$ вершины ориентированного графа называется число ребер, для которых эта вершина является $\frac{\text{концом}}{\text{началом}}$.

Степень входа вершины $V$ будем обозначать $\deg_+(V)$, а степень выхода — $\deg_-(V)$. На рис. ... $\deg_+(V_1) = 1$, $\deg_+(V_2) = 1$, $\deg_+(V_3) = 2$, $\deg_-(V_1) = 1$, $\deg_-(V_2) = 2$, $\deg_-(V_3) = 1$.



- Дуги орграфа называются *кратными*, если они имеют одинаковые начальные и конечные вершины, т.е. одинаковые направления. Например, кратны дуги $u(V_2, V_3)$ и $t(V_2, V_3)$



# Маршруты и пути

- Последовательность попарно инцидентных вершин $V_{i1}$ ,$V_{i2}$, ..., $V_{ik}$ неориентированного графа, т. е. последовательность ребер неориентированного графа, в которой вторая вершина предыдущего ребра совпадает с первой вершиной следующего, называется *маршрутом*.

- Число ребер маршрута называется *длиной маршрута.*

- Если начальная вершина маршрута совпадает с конечной, то такой маршрут называется *замкнутым* или *циклом*.

- *Расстоянием* между двумя вершинами называется минимальная длина из всех возможных маршрутов между этими вершинами при условии, что существует хотя бы один такой маршрут
- Обозначается как $d(V_1V_2)$ (от лат. distantio — расстояние)

$$d(V_1V_2) = \min| V_1 \ldots V_2 |.$$

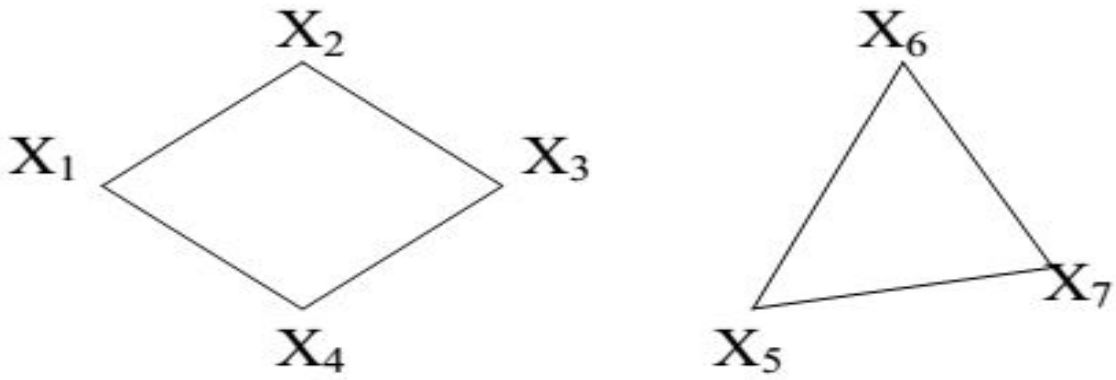
- В маршруте одно и то же ребро может встретиться несколько раз. Если ребро встретилось только один раз, то маршрут называется *цепью*.

- В орграфе маршрут является ориентированным и называется *путем*.
- На путь сразу налагаются важные требования, являющиеся частью определения:
  - ✔ направление каждой дуги должно совпадать с направлением пути;
  - ✔ ни одно ребро пути не должно встречаться дважды.

- Другими словами, *путь* — упорядоченная последовательность ребер ориентированного графа, в которой конец предыдущего ребра

# Связность

- Вершины u и v графа G связаны, если в G существует (u, v )-маршрут.
- Граф, в котором любые две вершины связаны, называется связным.
- На рисунке представлен несвязный граф с двумя компонентами связности.



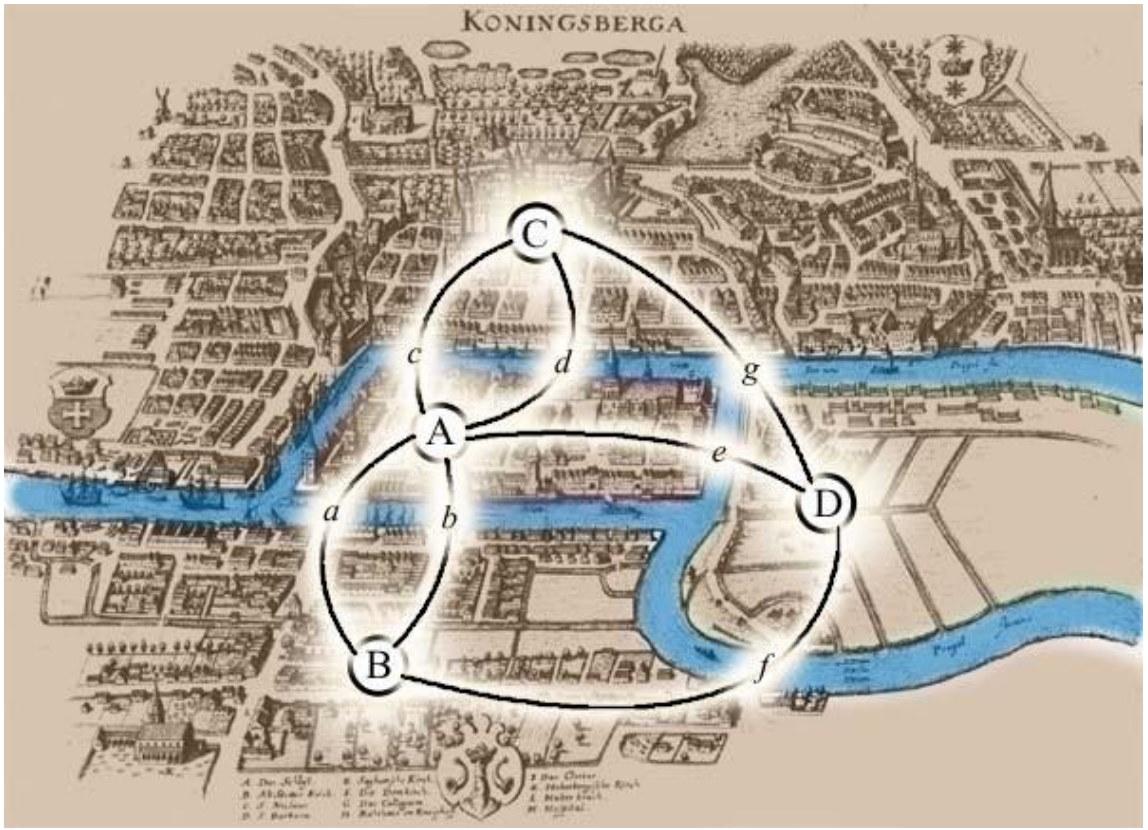
# Эйлеров цикл

- Цикл, содержащий все ребра графа, называется эйлеровым. Граф называется эйлеровым, если в нем существует эйлеров цикл.

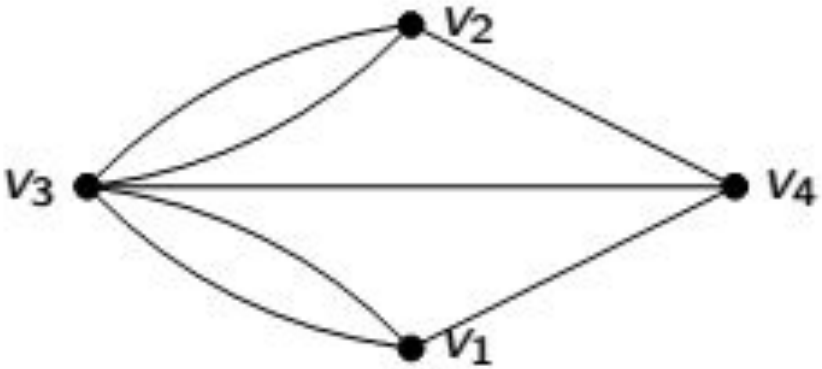
- Теорема Эйлера о циклах: граф без изолированных вершин является эйлеровым тогда и только тогда, когда он связен и степени всех его вершин четны.

- Замечание: для решения некоторых задач вместо эйлерова цикла удобно использовать родственное понятие эйлеровой цепи, отказываясь от условия «вернуться в исходную точку».

# Задача о 7 мостах Кёнигсберга




- Как можно пройти по всем семи мостам Кёнигсберга, не проходя ни по одному из них дважды?




- Степени всех вершин этого графа нечетны.
- Из теоремы Эйлера о циклах вытекает, что обойти Кенигсберг, пройдя по каждому мосту ровно один раз, и вернуться в исходную точку невозможно.

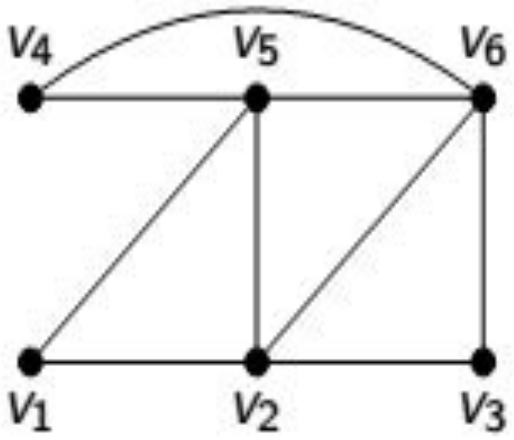
# Алгоритмы для построения эйлеровой цепи

## Алгоритм Флёри

*Вход*: эйлеров граф $G$.
*Выход*: список ребер графа $G$ в той последовательности, в которой они образуют эйлеров цикл.

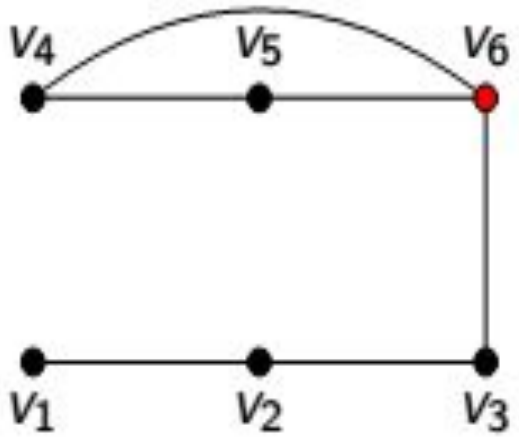
1. Положить текущий граф равным $G$, а текущую вершину — равной произвольной вершине $v \in V(G)$.
2. Выбрать произвольное, с учетом ограничения (см. ниже) ребро $e$ текущего графа, инцидентное текущей вершине.
3. Назначить текущей вторую вершину, инцидентную $e$.
4. Удалить $e$ из текущего графа и внести в список.
5. Если в текущем графе еще остались ребра, вернуться на шаг 2.

*Ограничение*: если степень текущей вершины в текущем графе больше 1, нельзя выбирать ребро, удаление которого из текущего графа увеличит число компонент связности в нем.




- Пусть на шаге 1 выбрана вершина v1.
- На шаге 2 ограничение никак не сказывается; пусть выбрано ребро (v1, v5).
- На двух следующих итерациях ограничений на выбор по-прежнему не возникает; пусть выбраны ребра (v5, v2) и (v2, v6).

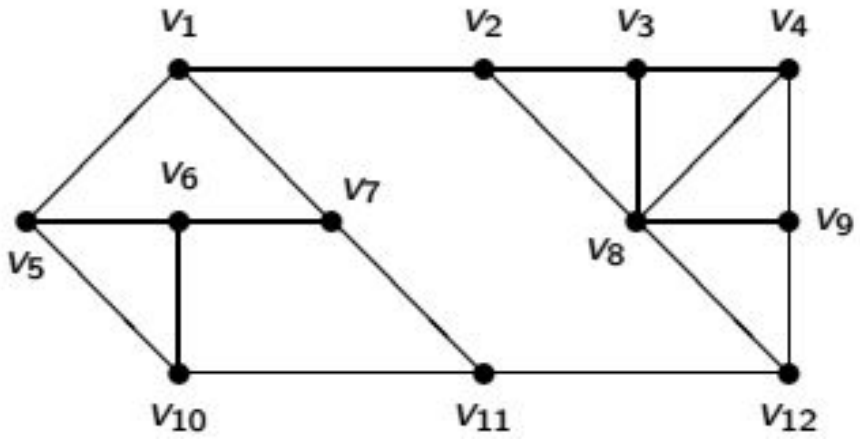
- Текущим графом становится граф, изображенный на рисунке (текущая вершина — v6).
- Далее выбрать ребро (v6, v3) нельзя из-за ограничения; пусть выбрано ребро (v6, v5).
- Дальнейший выбор ребер определен однозначно (текущая вершина всегда будет иметь степень 1), так что в итоге будет построен следующий эйлеров цикл:




$$v_1 \to v_5 \to v_2 \to v_6 \to v_5 \to v_4 \to v_6 \to v_3 \to v_2 \to v_1.$$

# Гамильтонов цикл

- Цикл, проходящий через каждую вершину графа ровно один раз, называется гамильтоновым.
- Граф называется гамильтоновым, если в нем существует гамильтонов цикл.




$$v_1 \to v_2 \to v_3 \to v_8 \to v_4 \to v_9 \to v_{12} \to v_{11} \to v_7 \to v_6 \to v_{10} \to v_5 \to v_1.$$

# Задача коммивояжера

- Дан список городов, соединенных дорогами, длины которых известны. Коммивояжер должен объехать все города, побывав в каждом по одному разу, и вернуться в свой город. Требуется найти кратчайший маршрут коммивояжера.

- Задача коммивояжера разрешима тогда и только тогда, когда граф этой задачи гамильтонов.

- Ни критерия гамильтоновости графа, ни эффективного алгоритма нахождения гамильтонова цикла в произвольном гамильтоновом графе, не известно (и

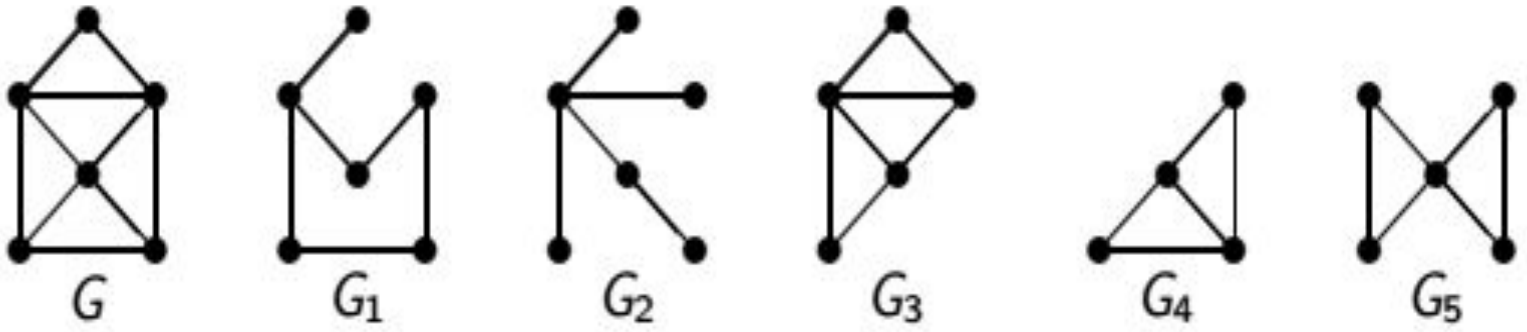
# Операции над графами

- Подграфом графа G = (V , X) называется граф G′=(V′, X′) такой, что V′⊆ V и X′⊆ X .

- Суграф графа G получается из G удалением некоторых ребер (с сохранением всех вершин).

- Вершинно-порожденный подграф получается удалением некоторых вершин и всех инцидентных им ребер (с сохранением всех остальных ребер).

- подграфы $G_1$ и $G_2$ являются суграфами в $G$;
- $G_3$ и $G_4$ — вершинно-порожденные подграфы в $G$;
- подграф $G_5$ не является ни суграфом, ни вершинно-порожденным подграфом.

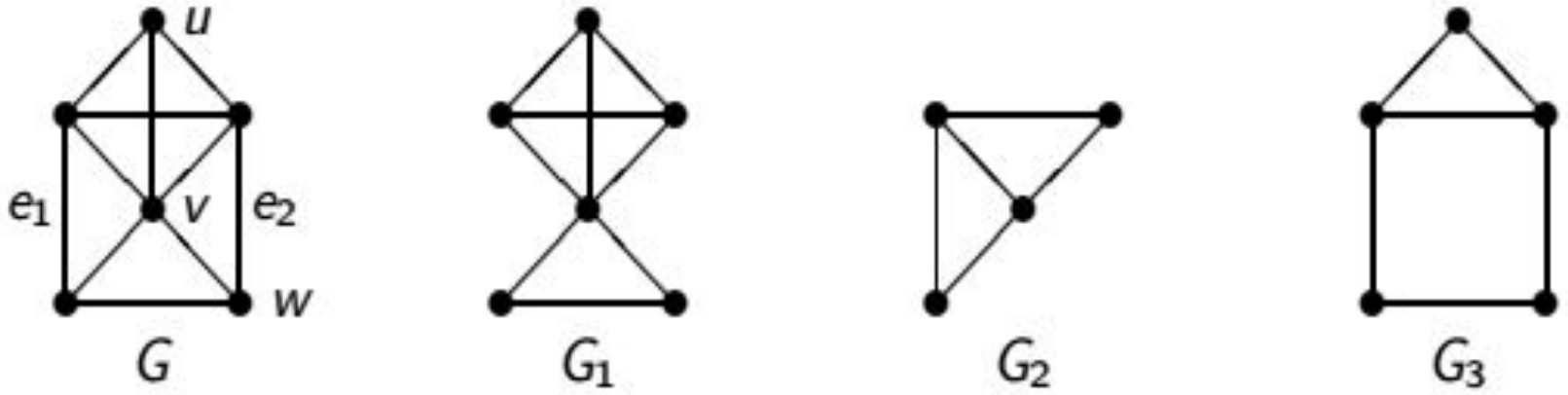


# Операции удаления вершин и ребер

- Если x — произвольное ребро графа G , то удаление ребра x — это операция, результатом которой является граф с множеством вершин V (G ) и множеством ребер X (G ) \ {x}. Этот граф обозначается через G − e .

- Если v — произвольная вершина графа G , то удаление вершины v — это операция, результатом которой является граф с множеством вершин V(G)\{v} и множеством ребер X (G )\ {$x_1$, $x_2$, . . . $x_k$}, где $x_1$, $x_2$, . . . , $x_k$ — все ребра графа G , инцидентные вершине v . Этот граф обозначается через G − v .

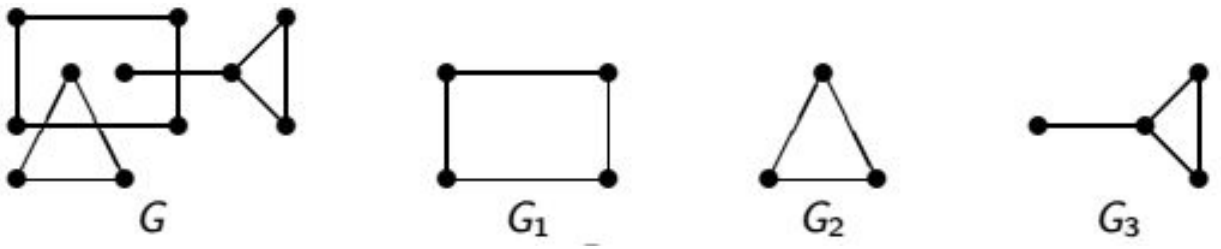
# Примеры удаления вершин и ребер графа



# Операция объединения графов

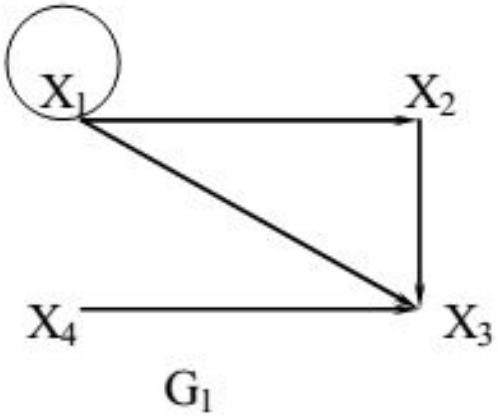
- Пусть $G_1 = (V_1, X_1)$, $G_2 = (V_2, X_2)$, . . . , $G_k = (V_k, X_k)$ — набор графов, никакие два из которых не имеют общих вершин. Объединением графов $G_1, G_2, \ldots, G_k$ называется граф $G = (V, X)$ такой, что $V = V_1 \cup V_2 \cup \cdots \cup V_k$ и $X = X_1 \cup X_2 \cup \cdots \cup X_k$.

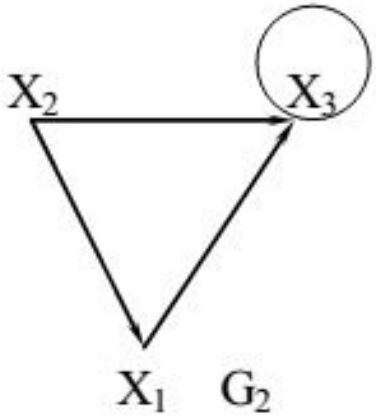
- Объединение графов $G_1, G_2, \ldots, G_k$ обозначается через $G_1 \cup G_2 \cup \cdots \cup G_k$.

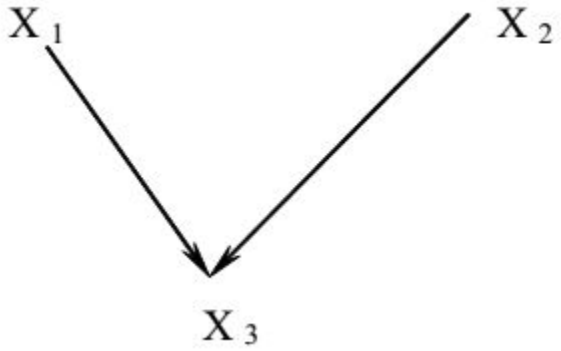


# Операция пересечения графов

- Пусть $G_1 = (V_1, X_1)$, $G_2 = (V_2, X_2)$ - графы, не имеющие общих вершин. Пересечением графов $G_1$, $G_2$ называется граф $G = (V, X)$ такой, что $V = V_1 \cap V_2$ и $X = X_1 \cap X_2$
- Пересечение графов $G_1$, $G_2$ обозначается через $G_1 \cap G_2$.

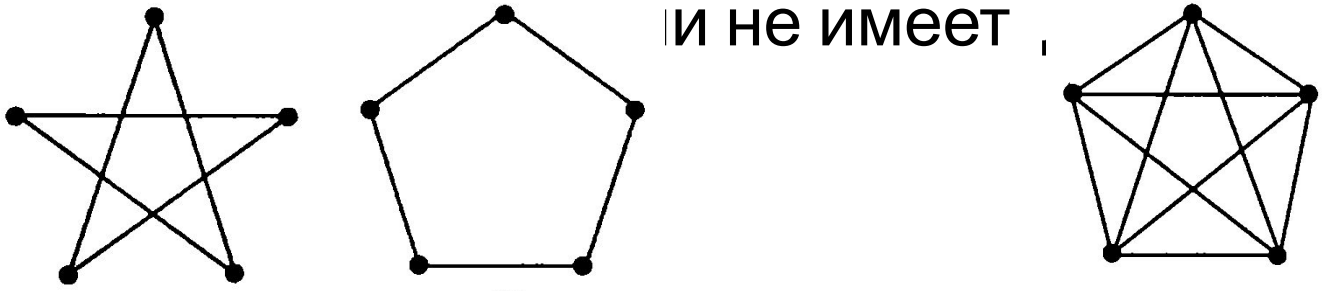


- **Дополнением** графа G (V, X) называется гр $\overline{G}(V, X')$ с теми же вершинами V, что и граф G, и имеющий те и только те ребра X', которые необходимо добавить к графу G, чтобы он стал полным.

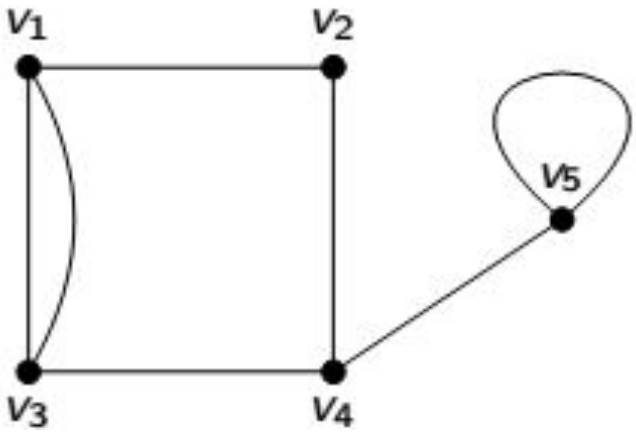
- Гра и не имеет ния.



- Дополнением полного графа будет нуль-граф, и

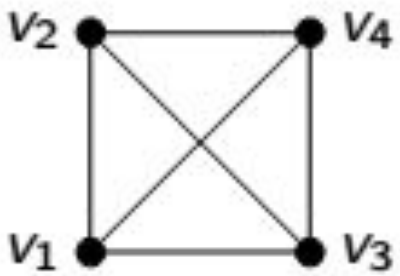
# Матрица смежности

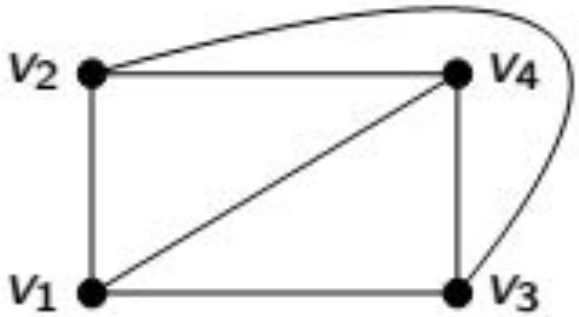
- Пусть G — граф, а $v_1, v_2, \ldots, v_n$ — множество всех его вершин. Матрицей смежности графа G называется квадратная матрица $A = (a_{ij})$ порядка n, в которой для каждых $i, j = 1, 2, \ldots, n$ элемент $a_{ij}$ равен числу ребер, инцидентных вершинам $v_i$ и $v_j$, при этом каждая петля считается дважды.

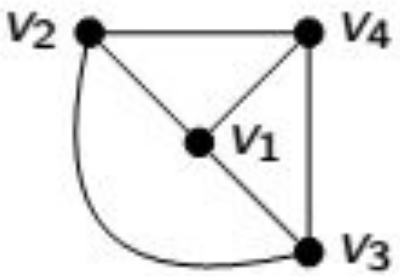



$$\begin{pmatrix} 0 & 1 & 2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{pmatrix}$$

- Два графа равны (алгебраически), если равны их матрицы смежности.

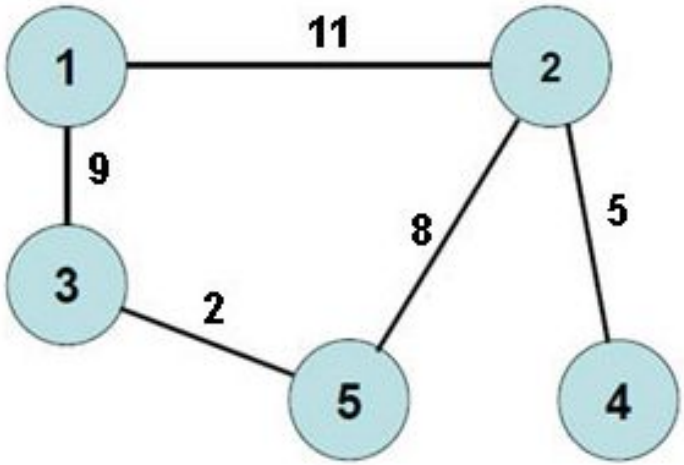


- Взвешенный граф - граф (орграф), ребрам (дугам) которого приписаны веса.
- Иногда изображается в виде пары , где  — весовая функция, определенная на множестве ребер графа и отображающая множество ребер в некоторую соответствующую ему область значений. Такая функция иногда называется функцией стоимости, длины и др.
- В случае взвешенного графа элемент матрицы смежности $a_{ij}$ равен числу $w$, если существует ребро между вершинами $v_i$ и $v_j$ с весом $w$. Элемент $a_{ij}$ равен нулю, если рёбер между вершинами $v_i$ и $v_j$ не существует.

# Пример:



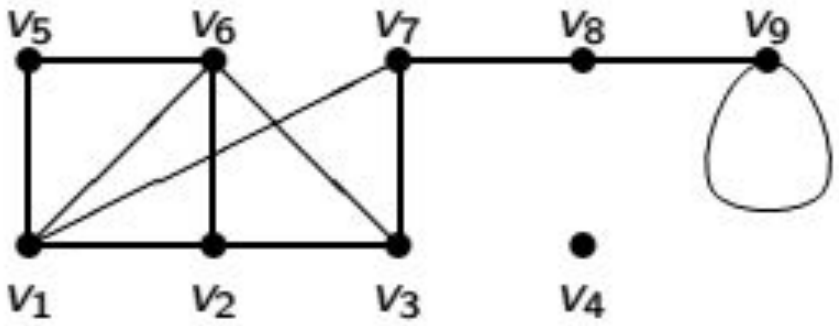

| V | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 11 | 9 | 0 | 0 |
| 2 | 11 | 0 | 0 | 5 | 8 |
| 3 | 9 | 0 | 0 | 0 | 2 |
| 4 | 0 | 5 | 0 | 0 | 0 |
| 5 | 0 | 8 | 2 | 0 | 0 |

- Главное удобство матрицы смежности — в том, что ответ на вопрос, существует ли ребро, инцидентное двум данным вершинам, можно получить за один шаг (заглянув в одну ячейку матрицы).

- Основной недостаток матрицы смежности — большой объем требуемой памяти (матрица смежности n-вершинного графа имеет $n^2$ элементов).

# Списки смежности

- Пусть G — граф, а $v_1, v_2, \ldots, v_n$ — множество всех его вершин. Списками смежности графа G называется массив из n списков, в котором, для любого i = 1, . . . , n, i -й список содержит в точности все вершины, смежные с $v_i$



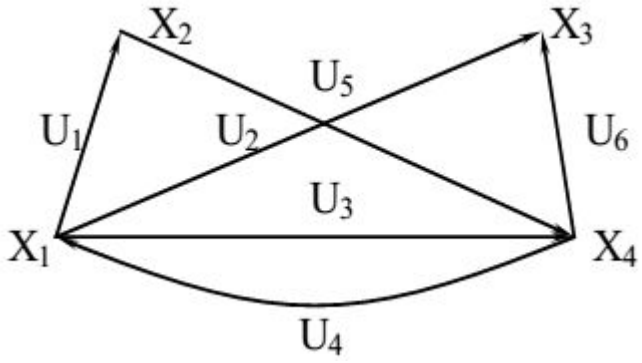

| | |
|---|---|
| 1 | $v_2, v_5, v_6, v_7$ |
| 2 | $v_1, v_3, v_6$ |
| 3 | $v_2, v_6, v_7$ |
| 4 | |
| 5 | $v_1, v_6$ |
| 6 | $v_1, v_2, v_3, v_5$ |
| 7 | $v_1, v_3, v_8$ |
| 8 | $v_7, v_9$ |
| 9 | $v_8, v_9$ |

# Матрица инциденций (инцидентности)

- Матрицей инциденций ориентированного графа G (X, U) называется прямоугольная матрица порядка[n * m], где n – мощность множества X, m – мощность множества U, каждый элемент которого $a_{ij}$ определяется следующим образом:

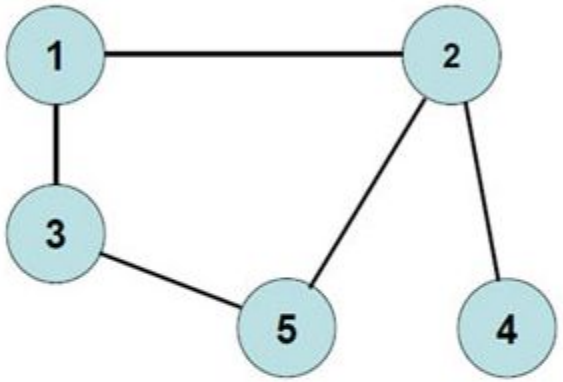
$$\alpha_{ij} = \begin{cases} 1, & \text{если } x_i - \text{начало дуги } u_i \\ -1, & \text{если } x_i - \text{конец дуги } u_i \\ 0, & \text{если } x_i - \text{не инцидентна дуге } u_i \end{cases}$$

# Пример:




| | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ |
|---|---|---|---|---|---|---|
| $x_1$ | 1 | 1 | 1 | −1 | 0 | 0 |
| $x_2$ | −1 | 0 | 0 | 0 | 1 | 0 |
| $x_3$ | 0 | −1 | 0 | 0 | 0 | −1 |
| $x_4$ | 0 | 0 | −1 | 1 | −1 | 1 |

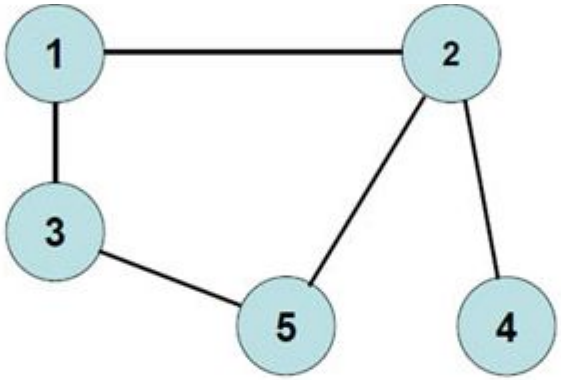
- Для неориентированного графа $a_{ij}$ определяется следующим образом:
  - ✔равен 1, если вершина $x_i$ инцидентна ребру $u_j$;
  - ✔равен 0, если вершина $x_i$ не инцидентна ребру $u_j$.




| V | 1-2 | 1-3 | 2-4 | 2-5 | 3-5 |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 0 |
| 3 | 0 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 | 1 |

# Списки инцидентности

- Графы значительного объёма целесообразно хранить в памяти компьютера в форме ***списков инцидентности***.
- Список инцидентности одной вершины графа включает номера вершин, смежных с ней.
- Ссылки на начало этих списков образуют одномерный массив, индексами которого служат ном




1:2→3
2:1→4→5
3:1→5
4:2
5:2→3

# Деревья

- Деревом называется связный граф без циклов.

- Теорема. (свойства деревьев):
  1. G связен и не содержит циклов;
  2. G не содержит циклов и имеет(n-1) ребро;
  3. G связен и имеет(n-1) ребро;
  4. G не содержит циклов, но добавление ребра между любыми его вершинами приводит к образованию цикла;
  5. G связен, и все его ребра являются перешейками;
  6. Всякая пара вершин G соединена только одной цепью.

- **Остовное дерево** графа состоит из минимального подмножества рёбер графа, таких, что из любой вершины графа можно попасть в любую другую вершину, двигаясь по этим рёбрам.

- Практическое применение:

  Предположим, что имеется n городов, которые нужно соединить нефтепроводом(электролинией, газопроводом). Стоимость строительства нефтепровода между городами $x_i$, $x_j$ задана.

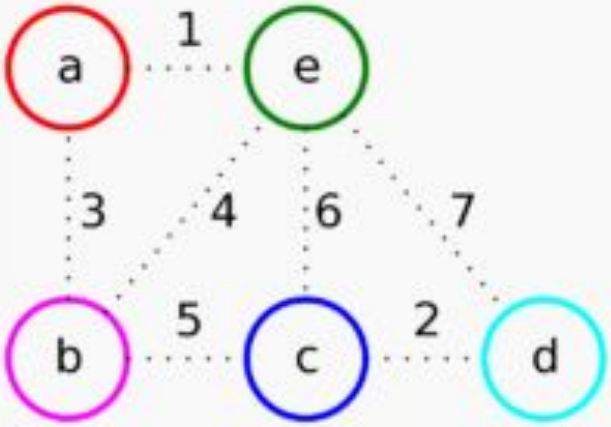
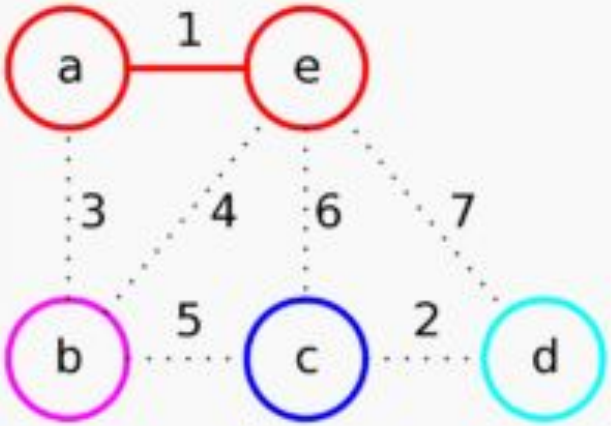
  Как построить самый дешевый нефтепровод, связывающий все города?

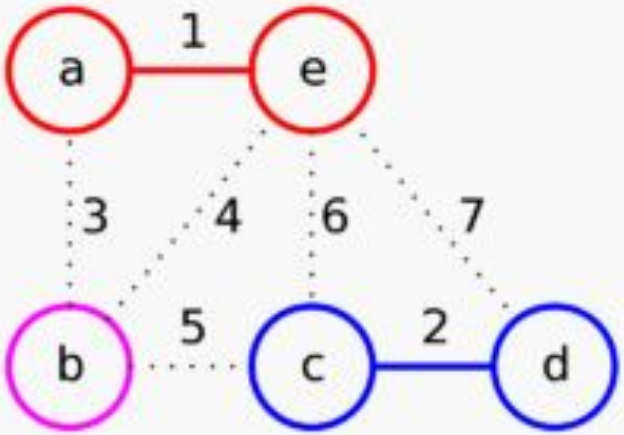
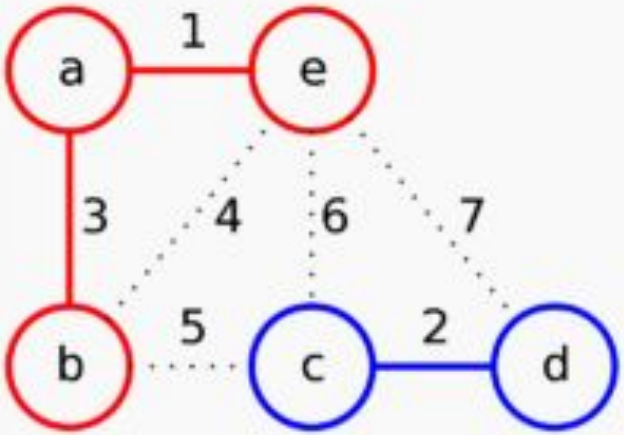
# Алгоритм Краскала (построение кратчайшего остовного дерева)

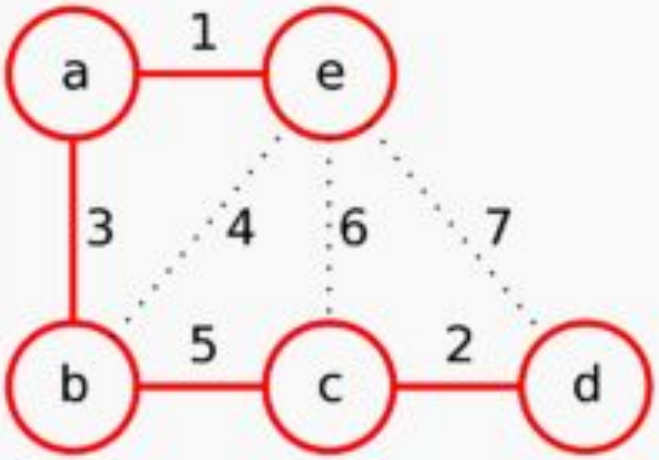
1. Выбираем самое короткое ребро графа $x_1$, затем ребро $x_2$, самое короткое из оставшихся;
2. Из оставшихся ребер выбираем самое короткое ребро $x_3$ так, чтобы оно не образовывало цикла с выбранными ребрами;
3. Продолжаем эту процедуру. На k-м шаге к выбранным ребрам $x_1,\ldots, x_{k-1}$ добавляем самое короткое ребро из оставшихся $|X|-(k-1)$ ребер так, чтобы оно не образовывало цикла с выбранными ребрами;
4. При $k=n-1$ процесс заканчивается. Получим граф без циклов с $(n-1)$-м ребром.

- Пример: построить минимальное остовное дерево.

| Рёбра *(в порядке их просмотра)* | ae | cd | ab | be | bc | ec | ed |
|---|---|---|---|---|---|---|---|
| Веса рёбер | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

| Изображение | Описание |
| --- | --- |
|  | Первое ребро, которое будет рассмотрено — **ae**, так как его вес минимальный.<br>Добавим его к ответу, так как его концы соединяют вершины из разных множеств (**a** — красное и **e** — зелёное).<br>Объединим красное и зелёное множество в одно (красное), так как теперь они соединены ребром. |
|  | Рассмотрим следующие ребро — **cd**.<br>Добавим его к ответу, так как его концы соединяют вершины из разных множеств (**c** — синее и **d** — голубое).<br>Объединим синее и голубое множество в одно (синее), так как теперь они соединены ребром. |

| | |
|---|---|
|  | Дальше рассмотрим ребро **ab**.<br>Добавим его к ответу, так как его концы соединяют вершины из разных множеств (**a** — красное и **b** — розовое).<br>Объединим красное и розовое множество в одно (красное), так как теперь они соединены ребром. |
|  | Рассмотрим следующие ребро — **be**.<br>Оно соединяет вершины из одного множества, поэтому перейдём к следующему ребру **bc**<br>Добавим его к ответу, так как его концы соединяют вершины из разных множеств (**b** — красное и **c** — синее).<br>Объединим красное и синее множество в одно (красное), так как теперь они соединены ребром. |




Рёбра **ec** и **ed** соединяют вершины из одного множества,

поэтому после их просмотра они не будут добавлены в ответ

Всё рёбра были рассмотрены, поэтому алгоритм завершает работу.

Полученный граф — минимальное остовное дерево

# Алгоритм Дейкстры для нахождения минимального пути в графе

- Дан взвешенный орфограф G(V,E) без дуг отрицательного веса. Найти кратчайшие пути от некоторой вершины графа до всех остальных вершин этого графа.
- Каждой вершине из ***V*** сопоставим метку — минимальное известное расстояние от этой вершины до ***a***.
- Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

**Инициализация**.

- Метка самой вершины ***a*** полагается равной 0, метки остальных вершин — бесконечности.
- Это отражает  то, что расстояния от ***a*** до других вершин пока неизвестны.
- Все вершины графа помечаются как непосещённые.

## **Шаг алгоритма**.

- Если все вершины посещены, алгоритм завершается.
- В противном случае, из ещё не посещённых вершин выбирается вершина ***u***, имеющая минимальную метку. Рассматривают всевозможные маршруты, в которых ***u*** является предпоследним пунктом.
- Вершины, в которые ведут рёбра из ***u*** - *соседи* этой вершины. Для каждого соседа вершины ***u***, кроме отмеченных как посещённые, рассматривается новая длина пути, равная сумме значений текущей метки ***u*** и длины ребра, соединяющего ***u*** с этим соседом.
- Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным