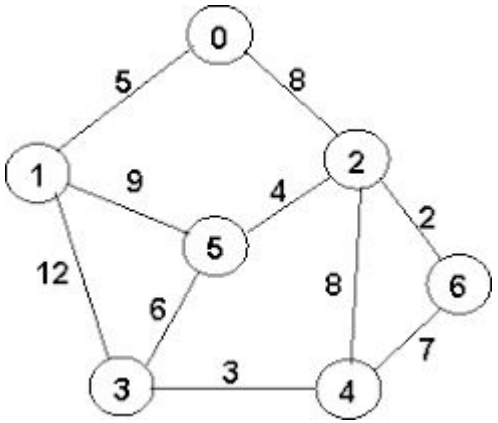
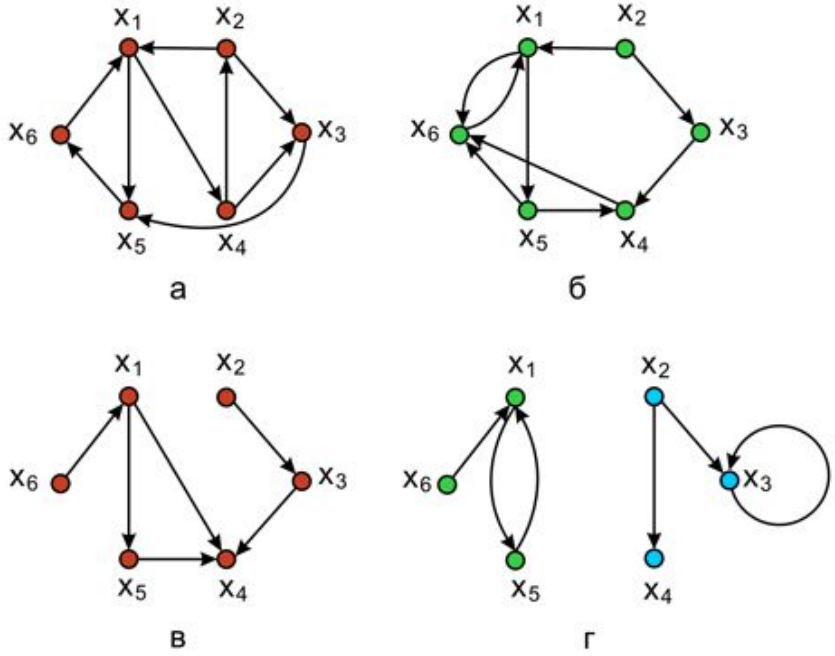
# Графы

# Графы

**Граф** – это совокупность двух конечных множеств: *множества* точек и *множества* линий, попарно соединяющих некоторые из этих точек.



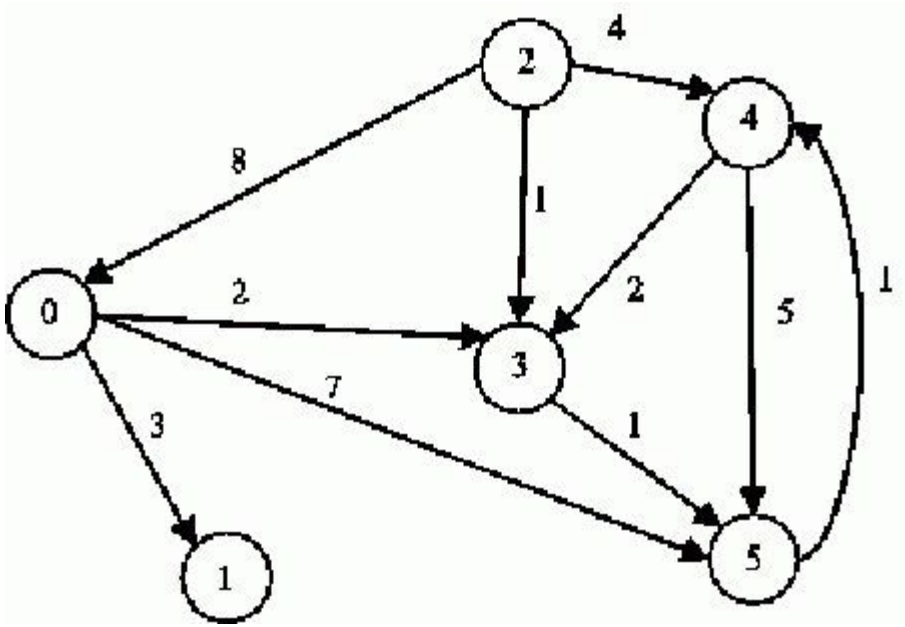



# Графы

- Множество точек называется ***вершинами (узлами) графа.***
- Множество линий, соединяющих *вершины графа,* называются ***ребрами (дугами) графа.***

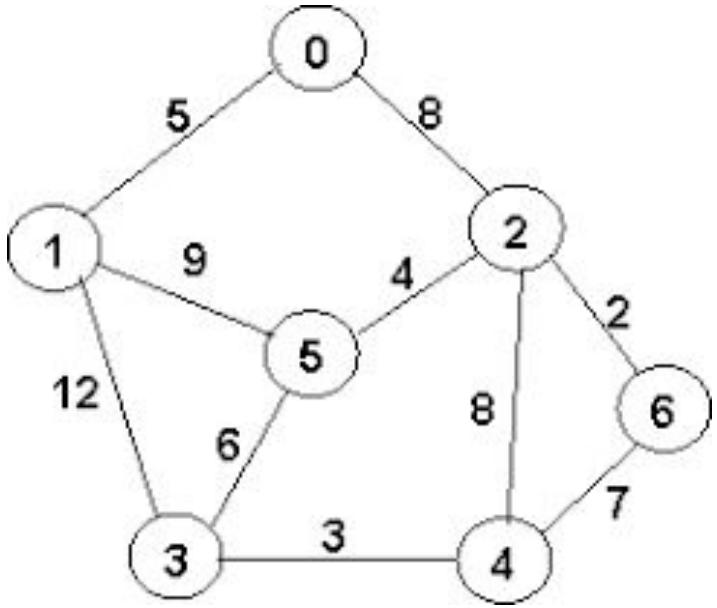
# Графы

***Ориентированный граф*** *(орграф) – граф,* у которого все *ребра* ориентированы, т.е. *ребрам* которого присвоено направление.



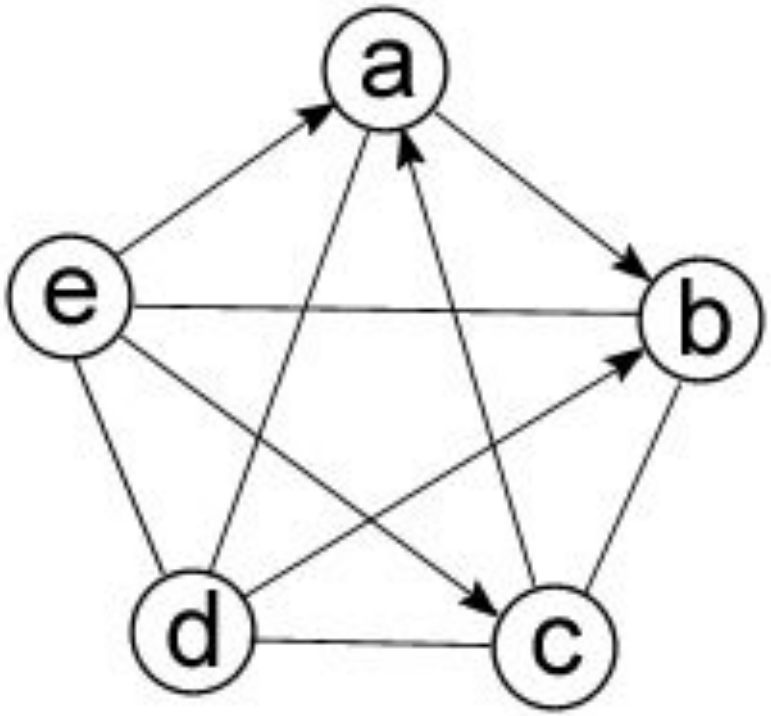
# Графы

***Неориентированный граф*** *(неорграф)* – *граф*, у которого все *ребра* неориентированы, т.е. *ребрам* которого не задано направление.



# Графы

***Смешанный граф*** – *граф*, содержащий как ориентированные, так и неориен ра.



# Графы

***Петлей*** называется *ребро*, соединяющее вершину саму с собой. Две вершины называются ***смежными***, если существует соединяющее их *ребро*. *Ребра*, соединяющие одну и ту же пару вершин, называются ***кратными***.

***Простой граф*** – это *граф*, в котором нет ни петель, ни кратных ребер.

***Мультиграф*** – это *граф*, у которого любые две вершины соединены более чем одним *ребром*.

## Графы

**Маршрутом** в графе называется конечная чередующаяся последовательность смежных вершин и ребер, соединяющих эти вершины.

*Маршрут* называется ***открытым***, если его начальная и конечная вершины различны, в противном случае он называется **замкнутым**.

# Графы

*Маршрут* называется **цепью**, если все его *ребра* различны. Открытая цепь называется **путем**, если все ее вершины различны.

Замкнутая цепь называется **циклом**, если различны все ее вершины, за исключением концевых.

*Граф* называется ***связным***, если для любой пары вершин существует соединяющий их *путь*.

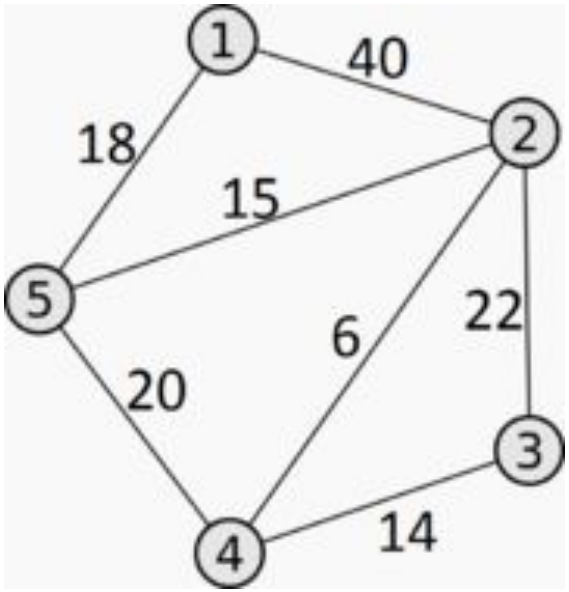
# Графы

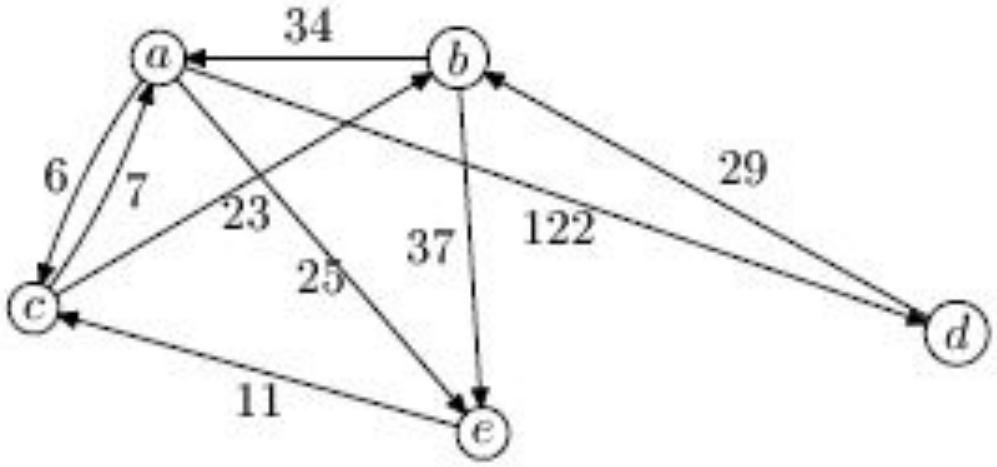
***Вес вершины*** – число (действительное, целое или рациональное), поставленное в соответствие данной вершине (интерпретируется как *стоимость, пропускная способность* и т. д.).

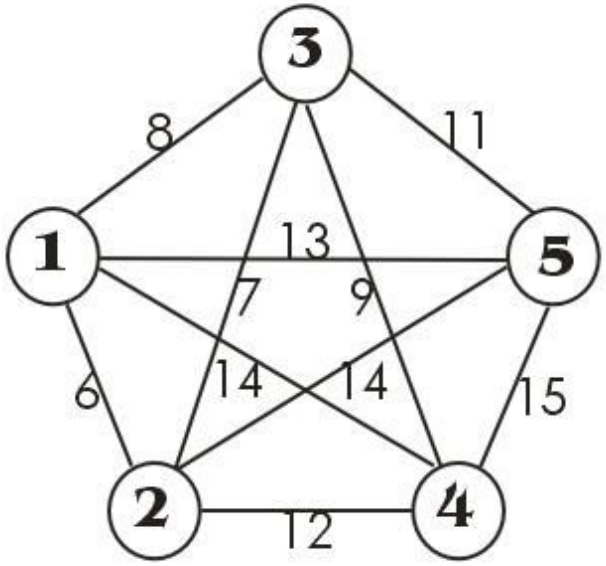
***Вес (длина) ребра*** – число или несколько чисел, которые интерпретируются по отношению к ребру как *длина, пропускная способность* и т. д.

***Взвешенный граф*** – *граф,* каждому ребру которого поставлено в соответствие некое *значение* (*вес ребра*).

# Взвешенный граф



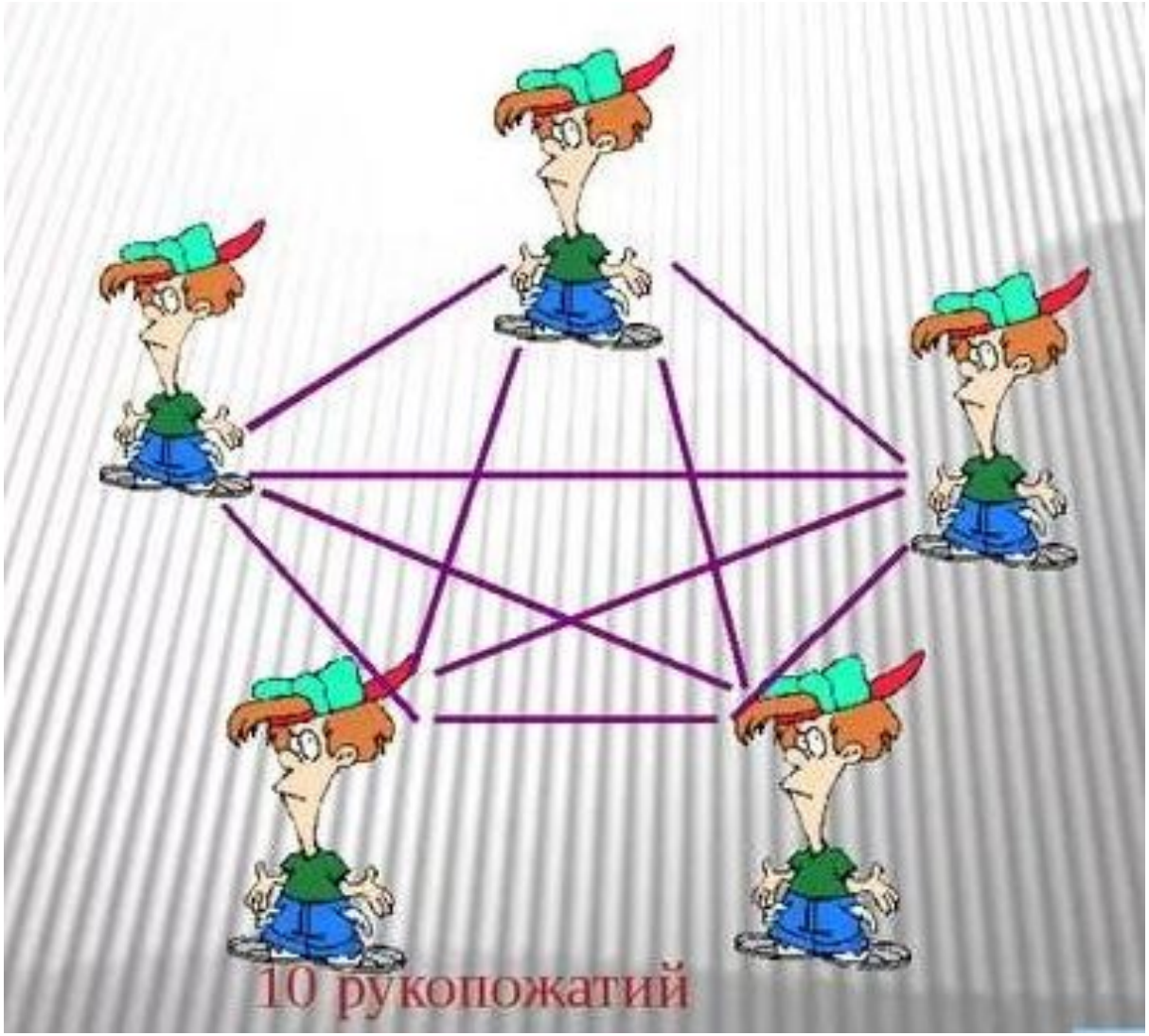



# Задача

- ***Пять человек обменялись рукопожатиями. Сколько было рукопожатий?***

- ***1.*** Каждый из 5 человек пожал руки своим коллегам. Однако произведение 5 * 4 = 20 дает удвоенное число рукопожатий (так как в этом расчете учтено, что первый пожал руку второму, а затем второй первому, на самом же деле было **одно** рукопожатие).
  Итак, число рукопожатий равно: (5 * 4) : 2 = 10.


10 рукопожатий

# Задача

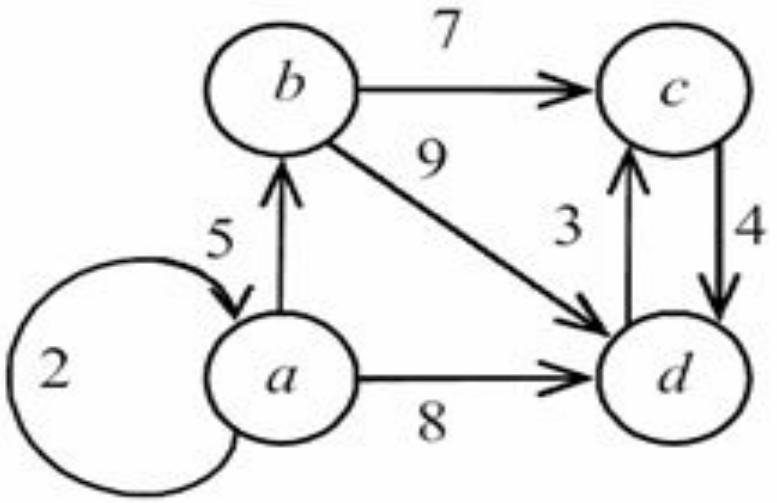
- Колледж. Фойе. Началась первая пара. Обменялись рукопожатиями несколько человек. Всего рукопожатий было 36. Сколько учащихся опоздало на первую пару.

# Способы представления графов

Выбор структуры данных для хранения графа в памяти компьютера имеет принципиальное *значение* при разработке *эффективных алгоритмов.*

**Способы представления графов**

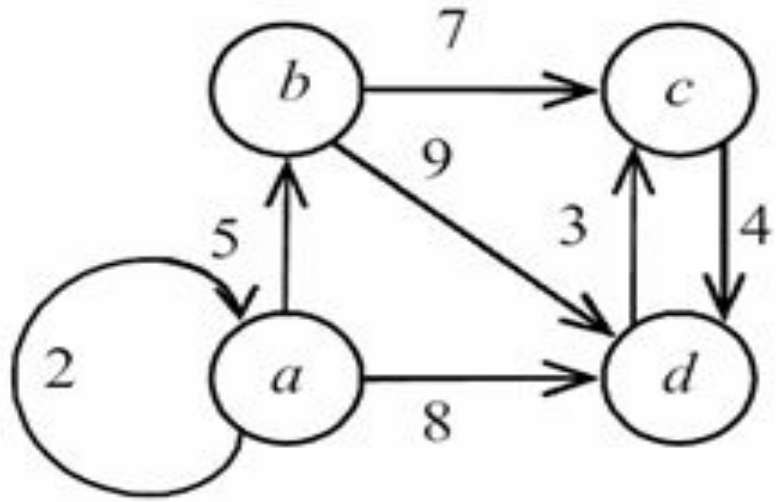
Пусть задан *граф,* у которого количество вершин равно n, а количество ребер – m. Каждое *ребро* и каждая *вершина* имеют *вес* – целое положительное число. Если *граф* не является помеченным, то считается, что *вес* равен единице



**Способы представления графов. Список рёбер**

***1. Список ребер*** – это множество, образованное парами смежных вершин. Для его хранения обычно используют *одномерный массив* размером m, содержащий список пар вершин, смежных с одним *ребром* графа.

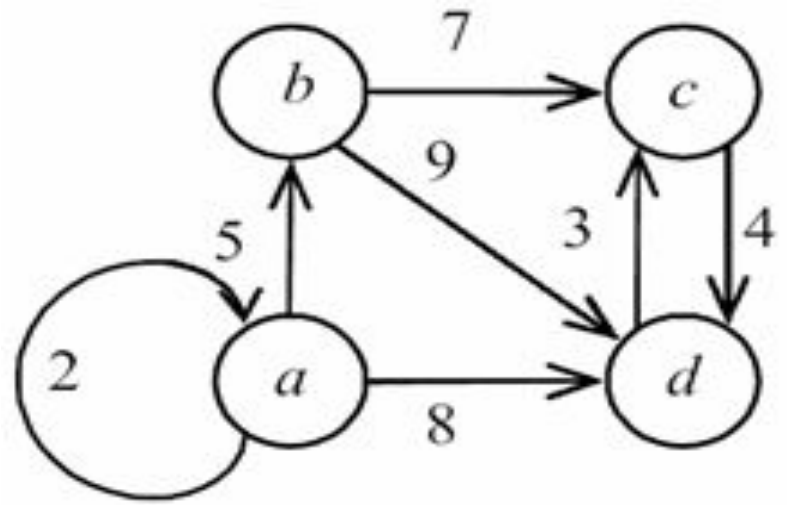
# Способы представления графов. Список рёбер




| | | | | | | |
|---|---|---|---|---|---|---|
| a | a | a | b | b | c | d |
| a | b | d | c | d | d | c |
| 2 | 5 | 8 | 7 | 9 | 4 | 3 |

***2. Матрица смежности*** – это *двумерный массив* размерности n x n, значения элементов которого характеризуются *смежностью вершин графа*. При этом значению элемента матрицы присваивается количество ребер, которые соединяют соответствующие вершины.

# Способы представления графов. Матрица смежности



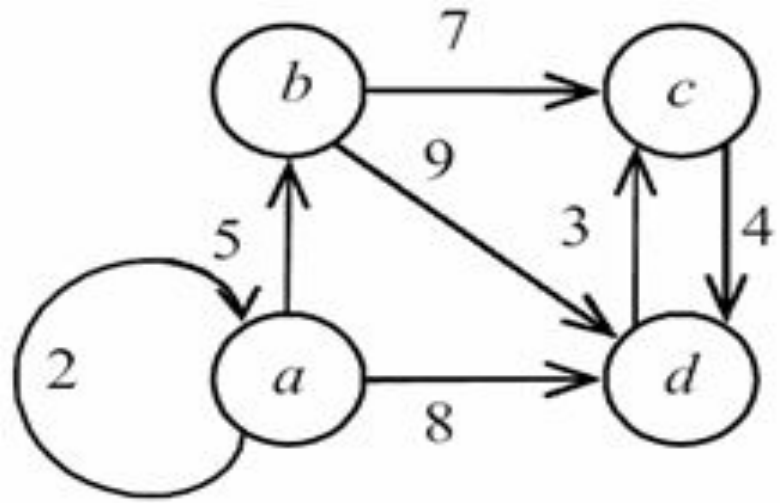

| | a | b | c | d |
|---|---|---|---|---|
| a | 2 | 5 | 0 | 8 |
| b | 0 | 0 | 7 | 9 |
| c | 0 | 0 | 0 | 4 |
| d | 0 | 0 | 3 | 0 |

## Способы представления графов. Матрица инцидентности

***3. Матрица инцидентности*** – это *двумерный массив* размерности

NxM, в котором указываются связи между *инцидентными* элементами графа (*ребро* и вершина). Столбцы матрицы соответствуют *ребрам,* строки – вершинам. Ненулевое значение в ячейке матрицы указывает связь между вершиной и *ребром*. Данный способ является самым емким для хранения и облегчает нахождение циклов в графе.

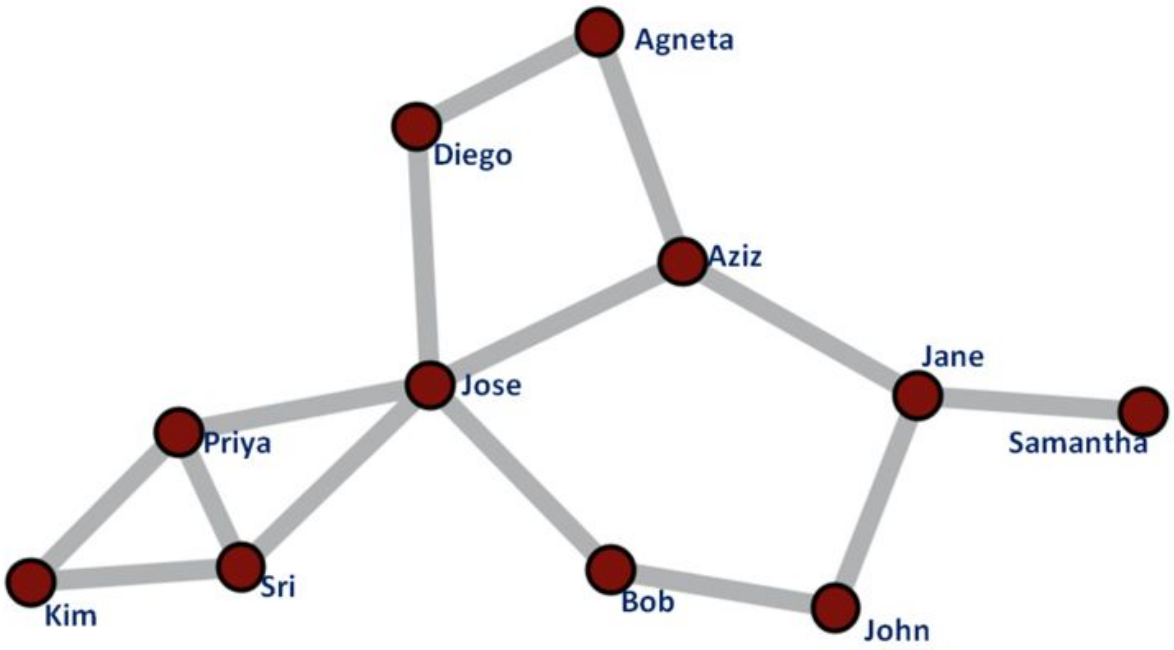
# Способы представления графов. Матрица инцидентности




| | $a$ | $b$ | $c$ | $d$ |
|---|---|---|---|---|
| $(a, a)$ | 2 | 0 | 0 | 0 |
| $(a, b)$ | 0 | 5 | 0 | 0 |
| $(a, d)$ | 0 | 0 | 0 | 8 |
| $(b, c)$ | 0 | 0 | 7 | 0 |
| $(b, d)$ | 0 | 0 | 0 | 9 |
| $(c, d)$ | 0 | 0 | 0 | 4 |
| $(d, c)$ | 0 | 0 | 3 | 0 |

- У большинства людей друзей меньше, чем в среднем у их друзей.
- Несмотря на видимую парадоксальность утверждения, оно вполне математически логично и выводится из базовых принципов теории графов. В 2012 году это утверждение подтвердили исследователями Корнеллского университета, которые проанализировали721 миллион пользователей Facebook. Читать полностью: https://42.tut.by/576746
- https://www.economist.com/blogs/economist-explains/2013/04/economist-explains-why-friends-more-popular-paradox

**Парадокс дружбы**




Простой социальный граф с наблюдаемым парадоксом дружбы: почти у всех (кроме Jose и Jane) друзей меньше, чем в среднем у их друзей. Изображение: wikipedia.org

# Обходы в графе

**Обходом графов (поиском на графах)** - это процесс *систематического* просмотра всех ребер или *вершин графа* с целью поиска ребер или вершин, удовлетворяющих некоторому условию.

К стандартным и наиболее распространенным методам относятся:

- ***поиск в глубину*** (Depth First Search, *DFS*);
- ***поиск в ширину*** (Breadth First Search, *BFS*).

*Эти методы чаще всего рассматриваются на *ориентированных графах*, но они применимы и для неориентированных, *ребра* которых считаются *двунаправленными*

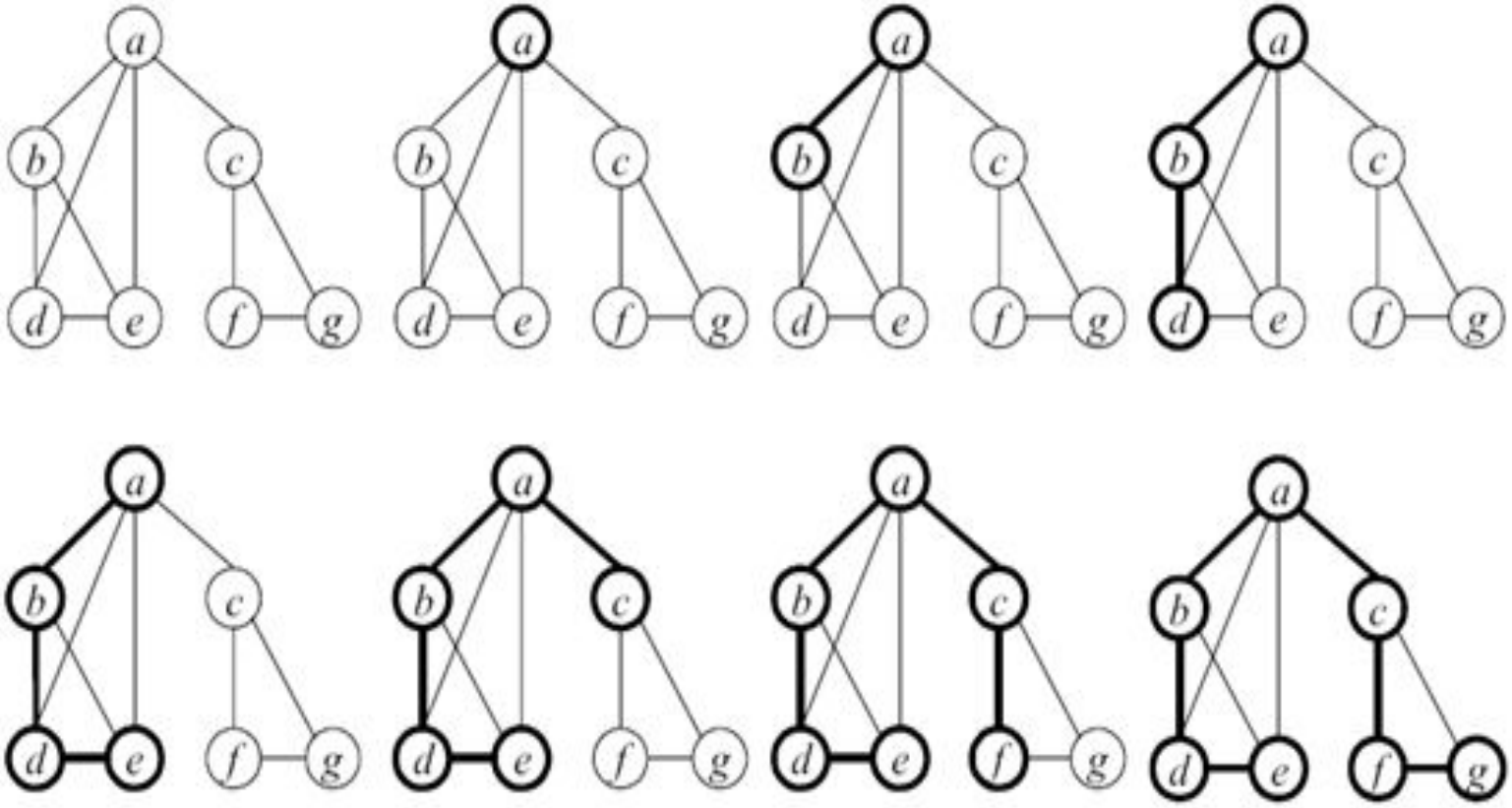
# *Алгоритм поиска в глубину*

**Шаг 1.** Всем вершинам графа присваивается *значение* не посещенная. Выбирается первая *вершина* и помечается как посещенная.

**Шаг 2.** Для последней помеченной как посещенная вершины выбирается смежная *вершина*, являющаяся первой помеченной как не посещенная, и ей присваивается *значение* посещенная. Если таких вершин нет, то берется предыдущая помеченная *вершина*.

# *Алгоритм поиска в глубину*

**Шаг 3.** Повторить шаг 2 до тех пор, пока все вершины не будут помечены как посещенные.



# *Алгоритм поиска в ширину*

**Шаг 1.** Всем вершинам графа присваивается *значение* не посещенная. Выбирается первая *вершина* и помечается как посещенная (и заносится в *очередь*).

**Шаг 2**. Посещается первая *вершина* из очереди (если она не помечена как посещенная). Все ее соседние вершины заносятся в *очередь*. После этого она удаляется из очереди.

**Шаг 3.** Повторяется шаг 2 до тех пор, пока *очередь* не пуста

# Поиск кратчайших путей в графе

Нахождение *кратчайшего пути* на сегодняшний день является жизненно необходимой задачей и используется практически везде, начиная от нахождения оптимального маршрута между двумя объектами на местности (например, *кратчайший путь* от дома до университета), в системах автопилота, для нахождения оптимального маршрута при перевозках, *коммутации* информационного пакета в сетях и т.п.

# *Алгоритм Дейкстры*

**Шаг 1.** Всем вершинам, за исключением первой, присваивается *вес* равный бесконечности, а первой вершине – 0.

**Шаг 2.** Все вершины не выделены.

**Шаг 3.** Первая *вершина* объявляется текущей.

**Шаг 4.** *Вес* всех невыделенных вершин пересчитывается по формуле: *вес* невыделенной вершины есть минимальное число из старого веса данной вершины, суммы веса текущей вершины и веса *ребра*, соединяющего текущую вершину с

# *Алгоритм Дейкстры*

**Шаг 5.** Среди невыделенных вершин ищется *вершина* с минимальным весом. Если таковая не найдена (то есть *вес* всех вершин равен бесконечности), то *маршрут* не существует. Следовательно, *выход*. Иначе, текущей становится найденная *вершина*. Она же выделяется.

**Шаг 6**. Если текущей вершиной оказывается конечная, то *путь* найден, и его *вес* есть *вес* конечной вершины.

**Шаг 7.** Переход на шаг 4.