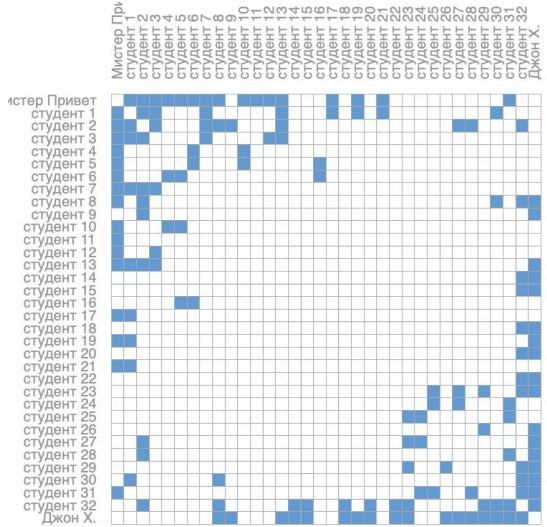
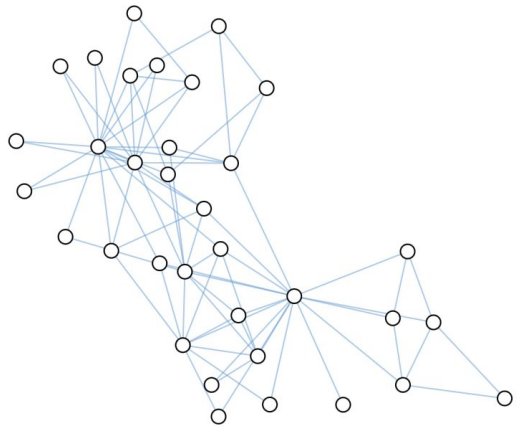
# Graph Neural Network

Elnikov Vlad

# Пример “Социальные сети как граф”

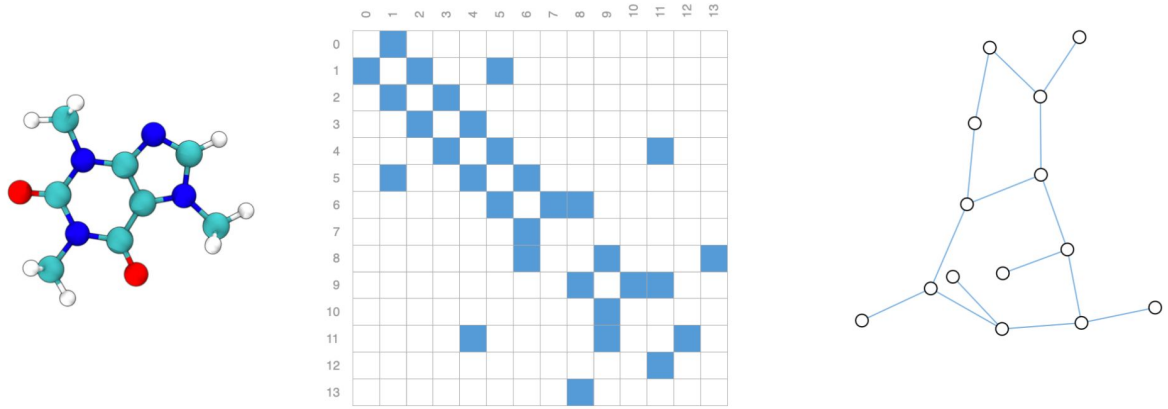
Социальные сети являются инструментами для изучения закономерностей коллективного поведения людей, учреждений и организаций. Мы можем построить граф, представляющий группы людей, моделируя людей как узлы, а их отношения как рёбра.



# Пример “Молекулы как граф”

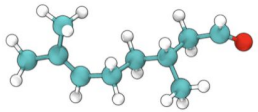
Молекулы являются строительными блоками материи и построены из атомов и электронов в 3D-пространстве.Очень удобно и распространено описать этот 3D-объект как граф, где узлы - это атомы, а рёбра - ковалентные связи.

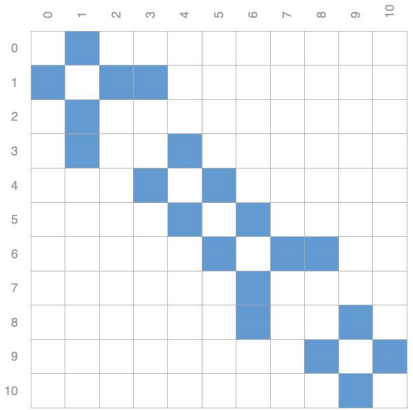


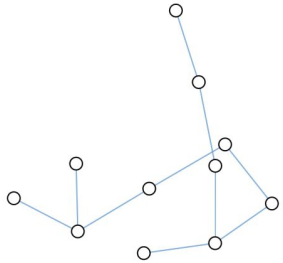

(Слева) 3D-представление молекулы кофеина (Центр) Матрица смежности связей в молекуле (справа) Графическое представление молекулы.

# Виды задач на графы:

Существует три общих типа задач прогнозирования на графиках: уровень графа, уровень узла и уровень ребра.
В задаче на уровне графа мы предсказываем одно свойство для всего графа. Для задачи на уровне узла мы предсказываем некоторое свойство для каждого узла графа. Для задачи уровня ребер мы хотим предсказать свойство или наличие рёбер в графе.

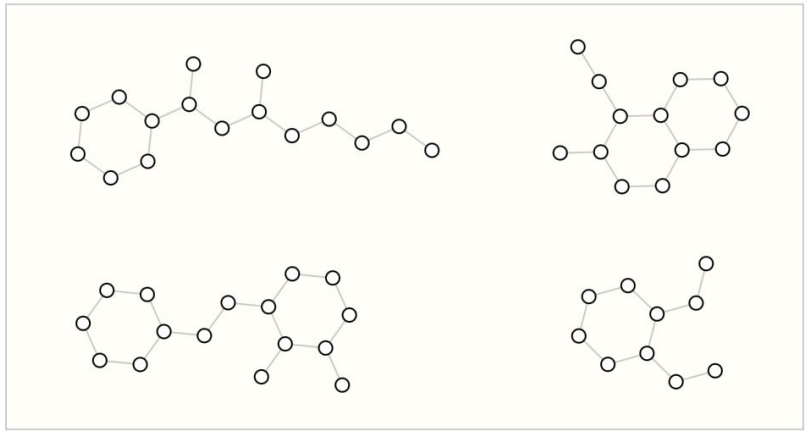




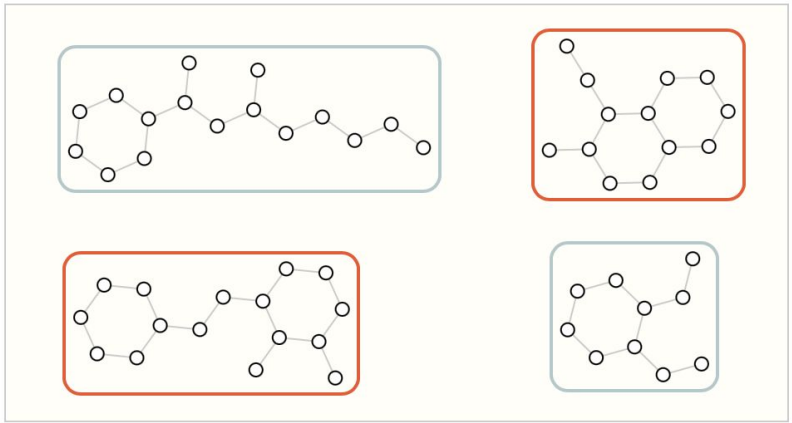



# Задачи(уровень графа)

В задаче на уровне графа наша цель - предсказать свойство всего графа. Например, для молекулы, представленной в виде графика, мы можем захотеть предсказать ее вид.



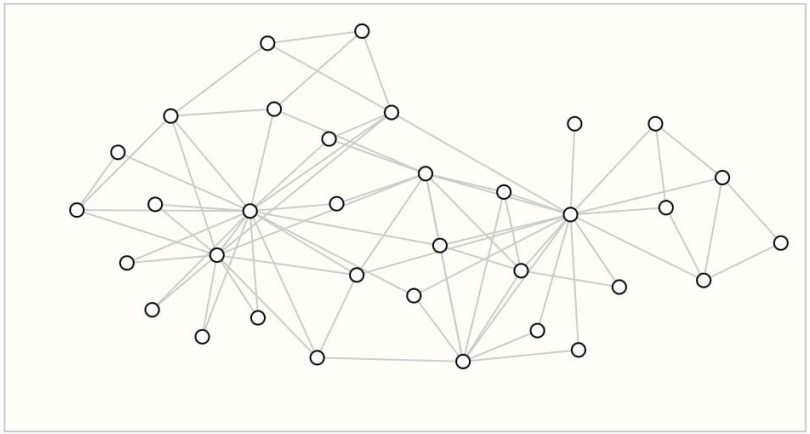
**Вход:** графики



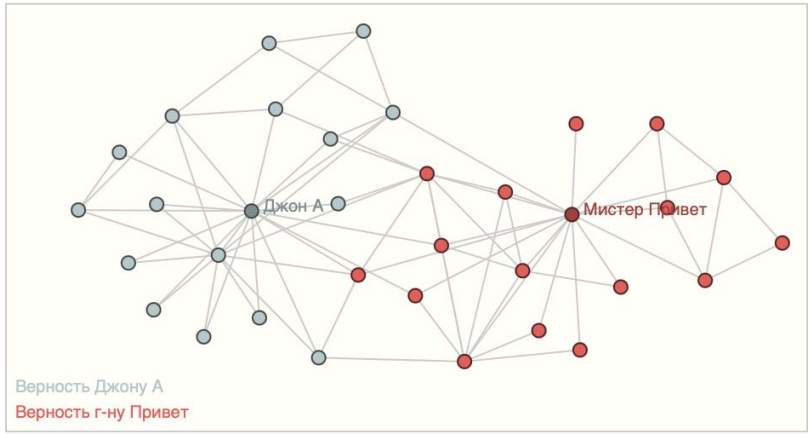
**Вывод:** метки для каждого графа (например, "содержит ли граф два кольца?")

# Задачи(уровень вершины)

Данный для примера набор данных представляет собой единый граф социальной сети, состоящий из людей, которые присягнули на верность одному из двух клубов каратэ после политического раскола.



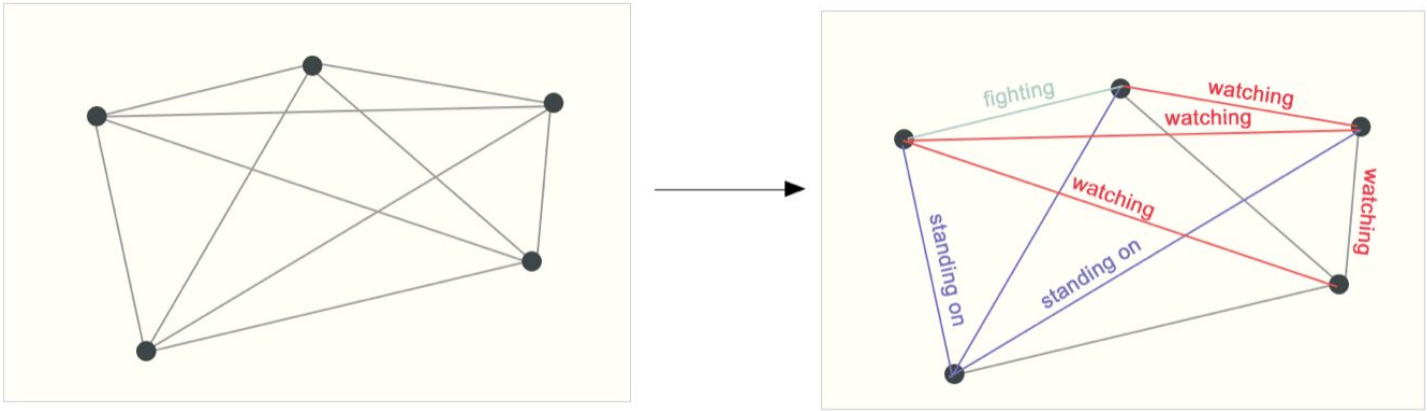
**Вход:** график с незамеченными узлами




**Вывод:** метки графических узлов

# Задачи(уровень ребра)

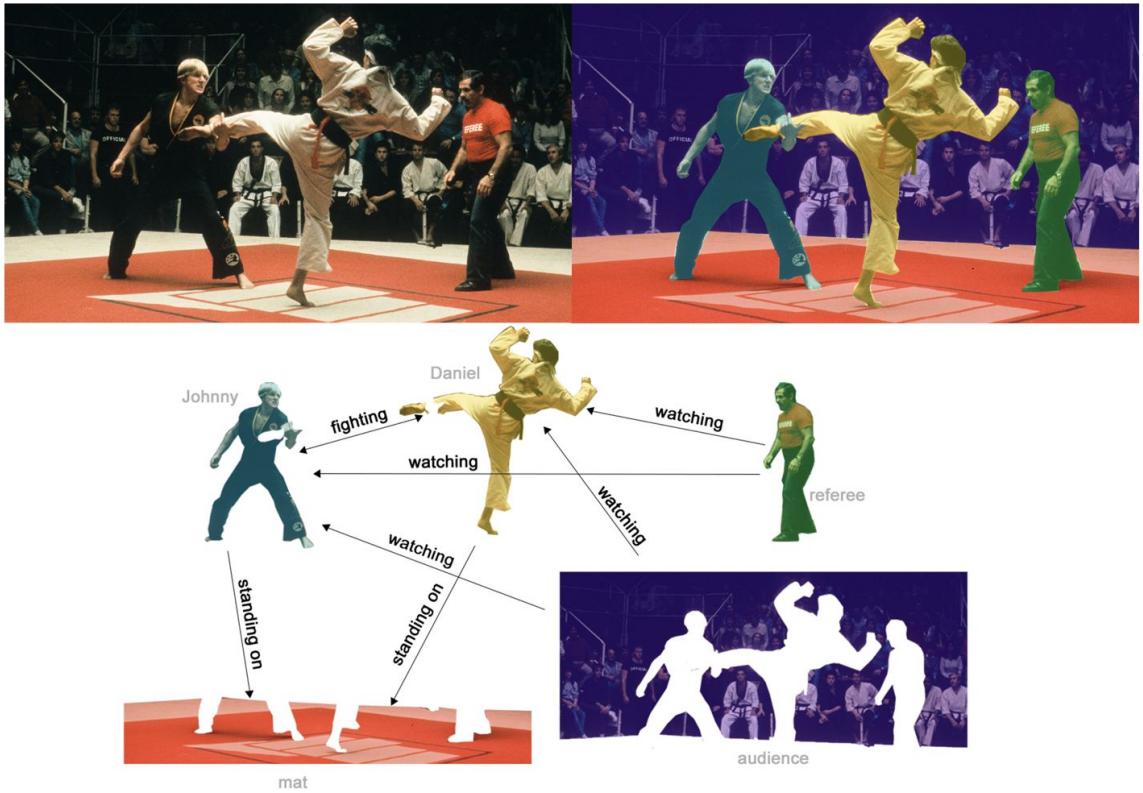
Одним из примеров вывода на уровень ребра является понимание сцены изображения. Помимо идентификации объектов на изображении, модели глубокого обучения могут быть использованы для прогнозирования взаимосвязи между ними.




**Input:** fully connected graph, unlabeled edges

**Output:** labels for edges

# Задачи(уровень ребра)



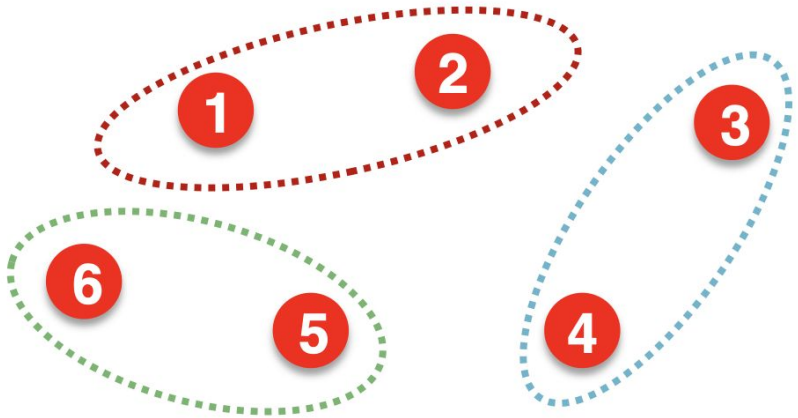
# Разделение графа

Обычное разделение на обучающую и тестовую выборку:

Training

Validation

Test



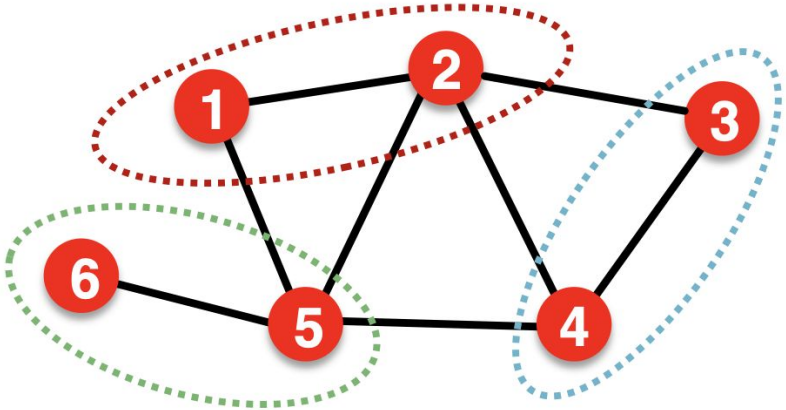
# Разделение графа

Проблемой разделения графа на обучающую и тестовую выборку заключается в том, что вершины больше не независимы.

Training

Validation

Test



# Разделение графа (Transductive setting)

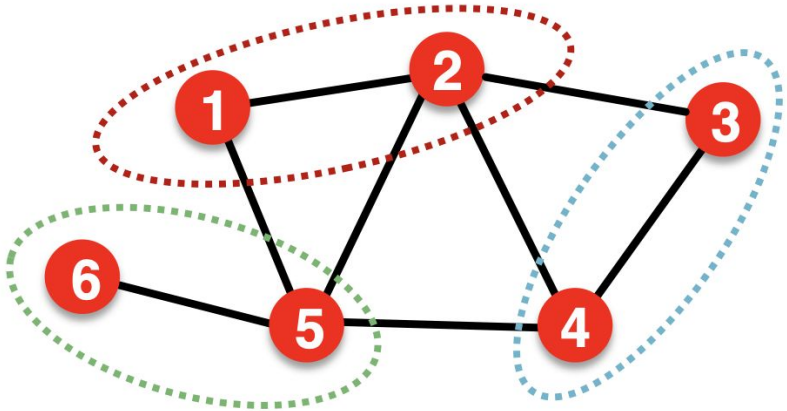
Первый способ: разделить только вершины графа

Мы вычисляем embedding, используя весь граф, и тренируемся, используя метки узлов 1 и 2; тестируемся, используя 5 и 6.

Training

Validation

Test



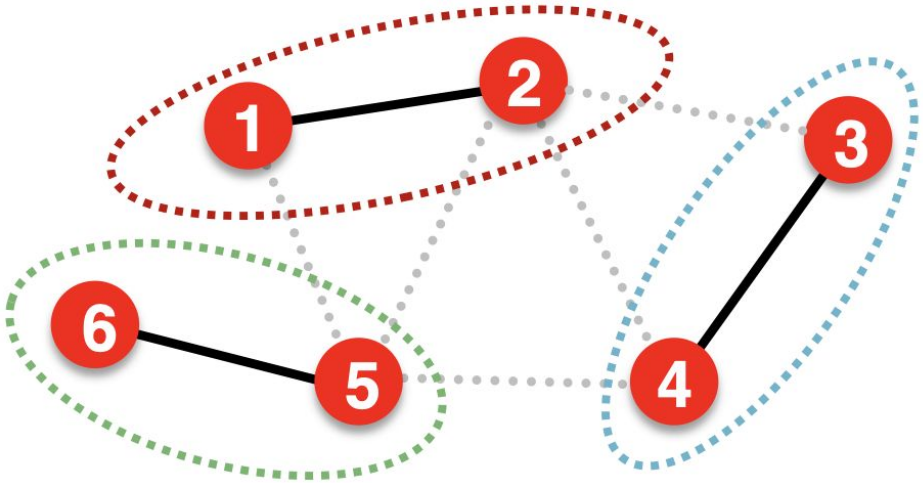
# Разделение графа (Inductive setting)

Второй способ: убрать ребра между различными выборками. Embedding вычисляется на каждой выборке отдельно.

Training

Validation

Test



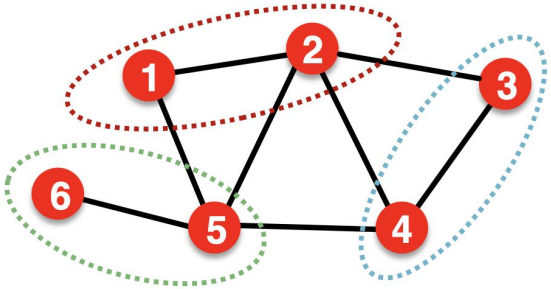
# Разделение графа (Минусы)

Первый имеет информацию о тестовой выборке во время обучения. Не подходит для задач о предсказании графа.

Training

Validation

Test

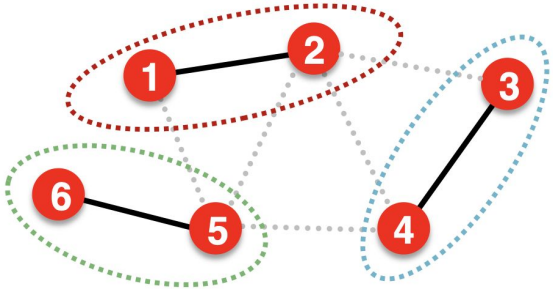



Второй теряет ребра, что может быть критично для маленьких графов

Training

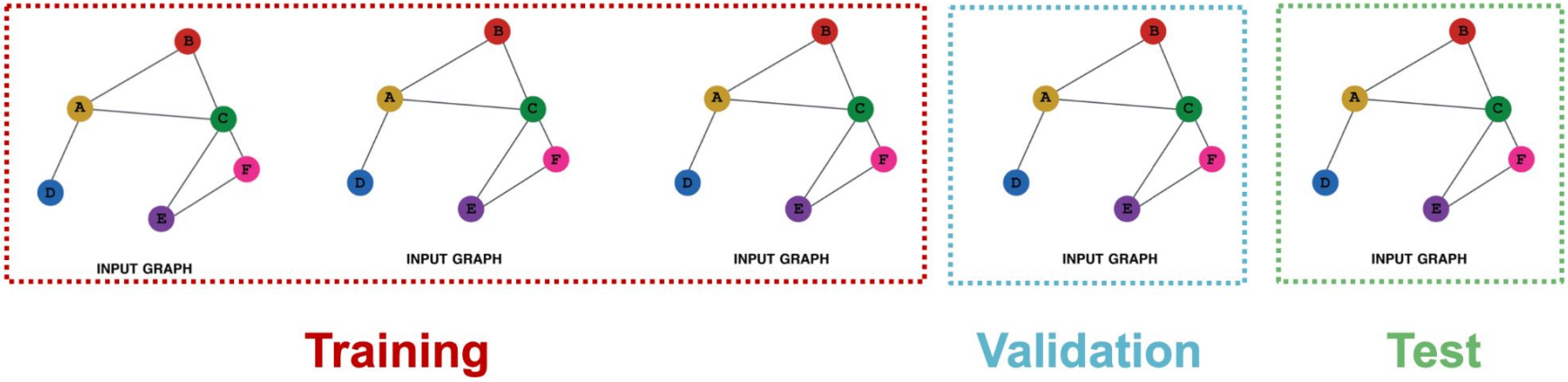
Validation

Test



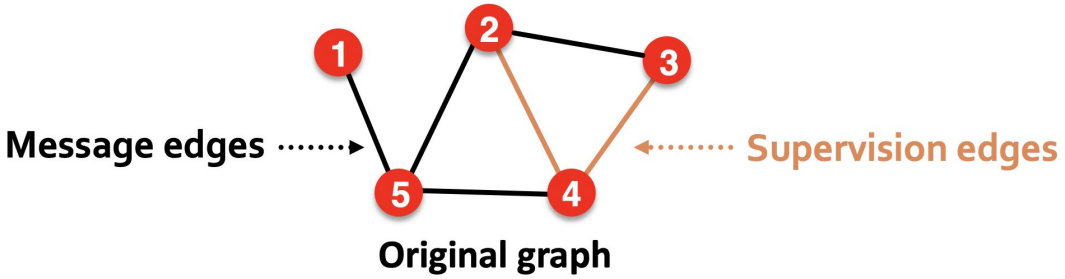
# Разделение графа

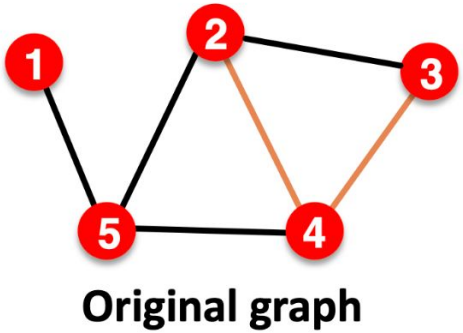
Разные компоненты связности можно разделять так. Также такое распределение получается при использовании второго способа разделения графа.

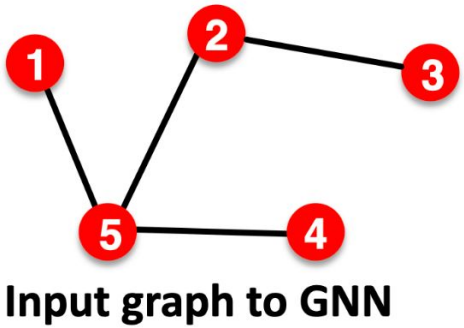


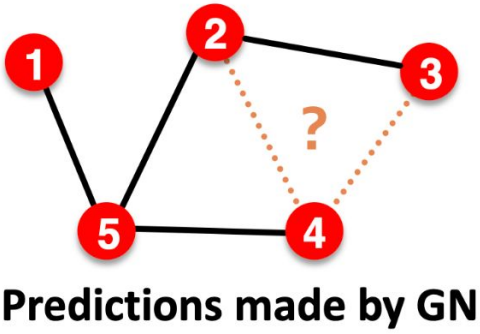
# Разделение графа

Задачи о предугадывании ребер.



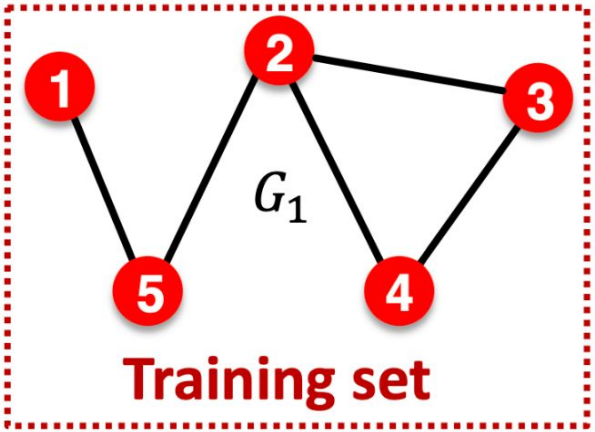



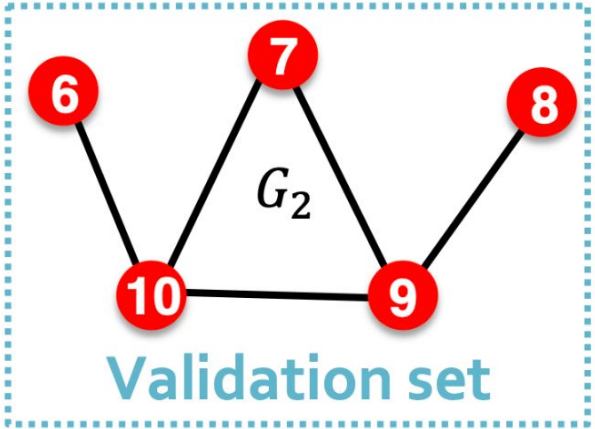



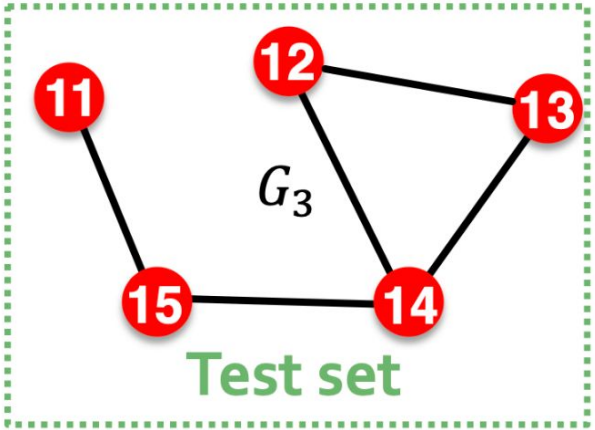



# Разделение графа

Первый способ. Попробовать угадывать разные ребра.

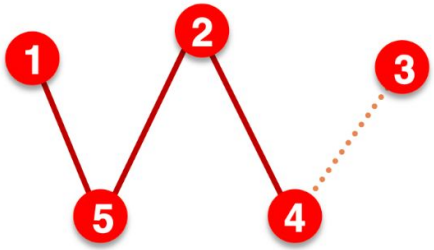


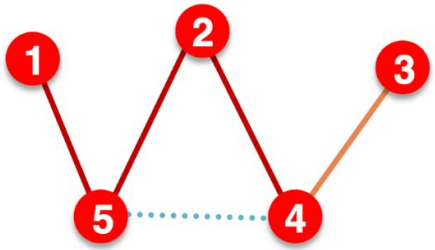



# Разделение графа

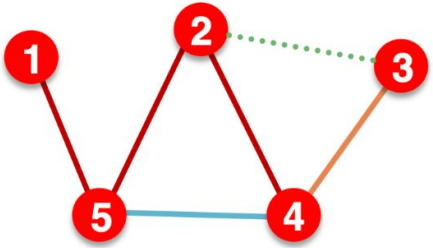
Второй способ. Постепенно добавлять ребра.




**(1) At training time:**
Use **training message edges** to predict **training supervision edges**



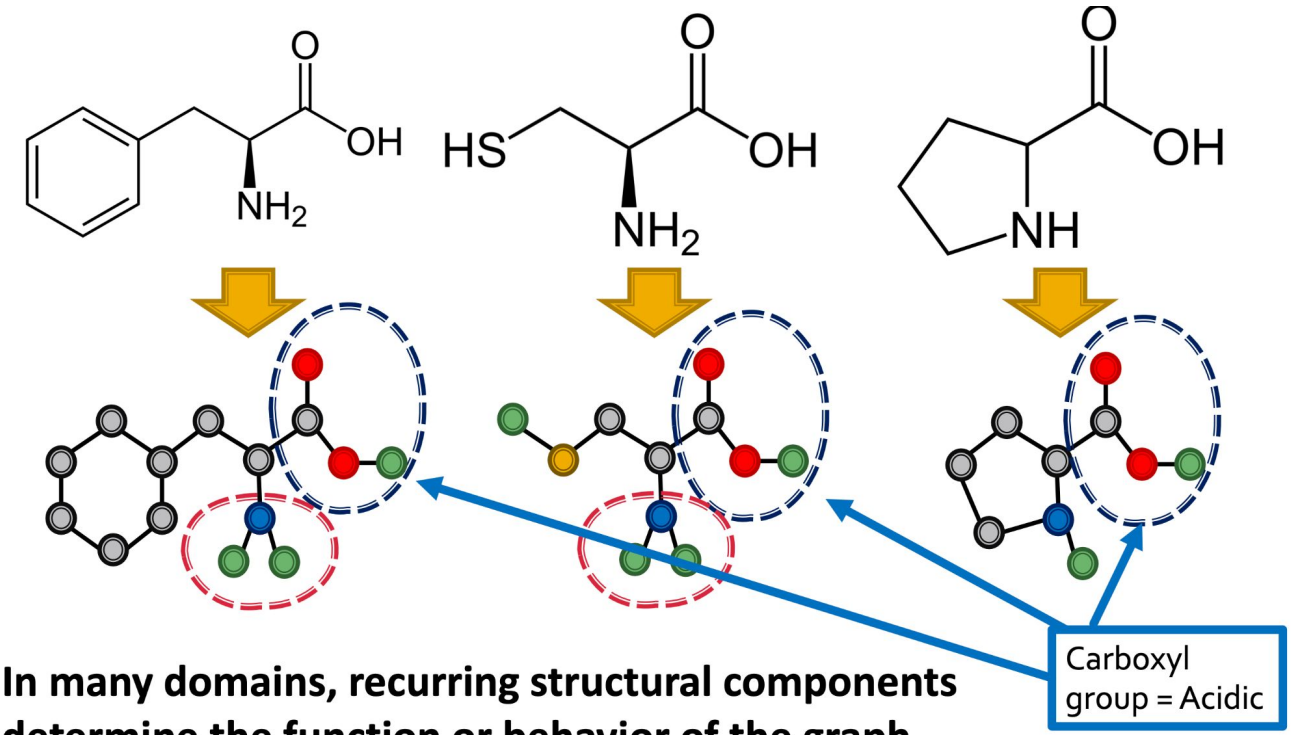

**(2) At validation time:**
Use **training message edges & training supervision edges** to predict **validation edges**




**(3) At test time:**
Use **training message edges & training supervision edges & validation edges** to predict **test edges**

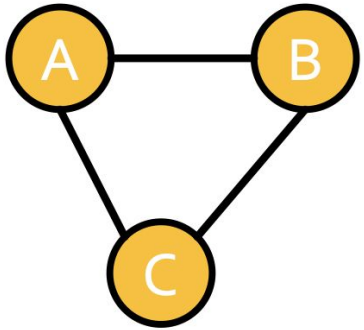
# Нахождение подграфов

Нахождение подграфов может помочь нам в обучении.

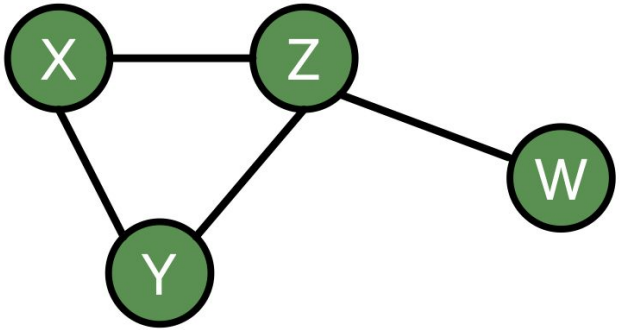


# Нахождение подграфов

Мы хотим узнать для графа G2, есть ли у него подграф, изоморфный графу G1



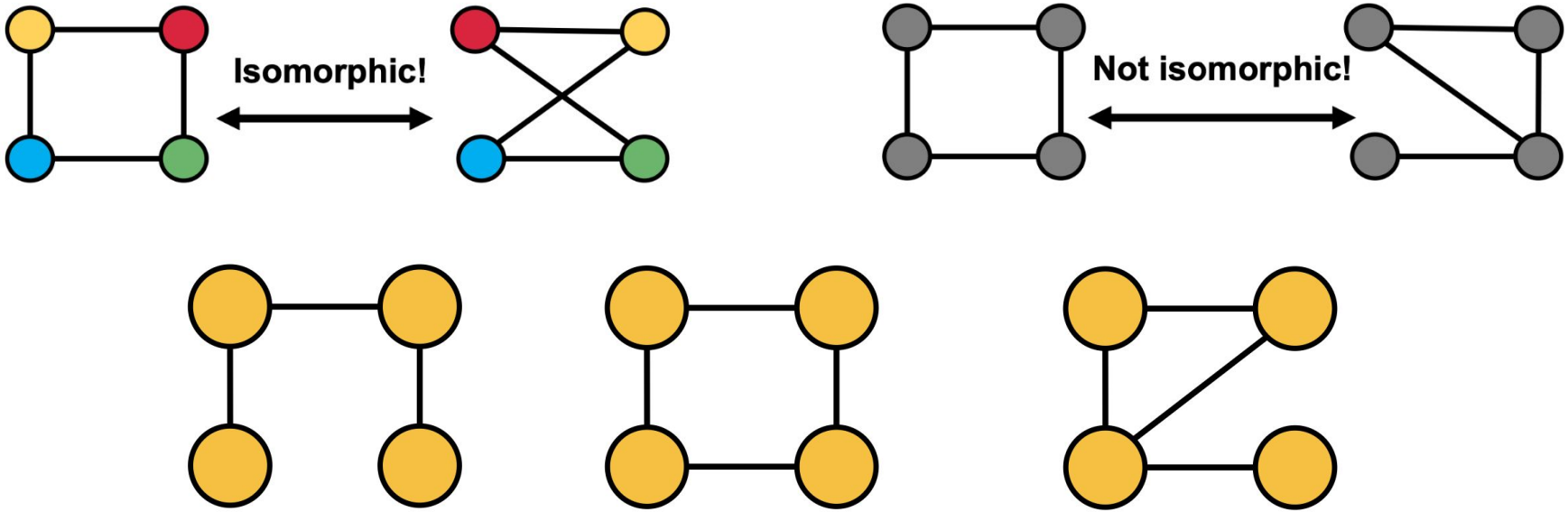

$G_1$



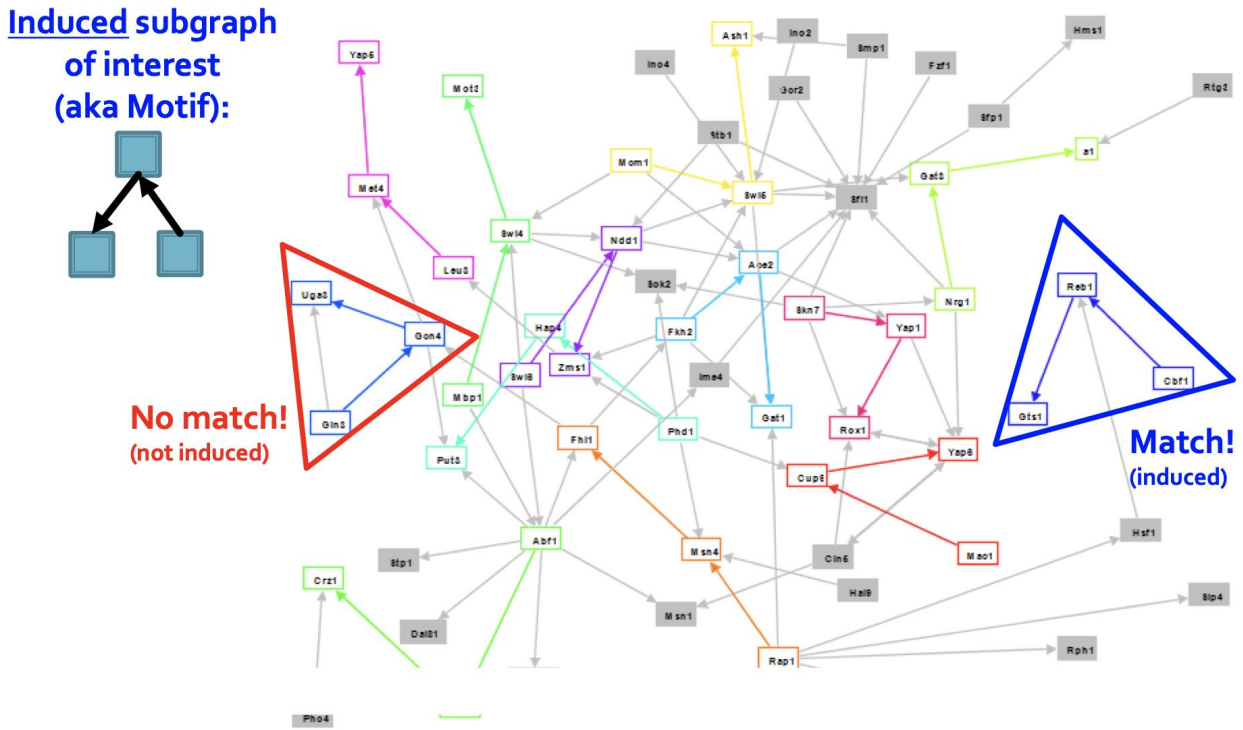

$G_2$

# Проблема нахождение подграфов

Задача изоморфизма двух графов является NP-полной.



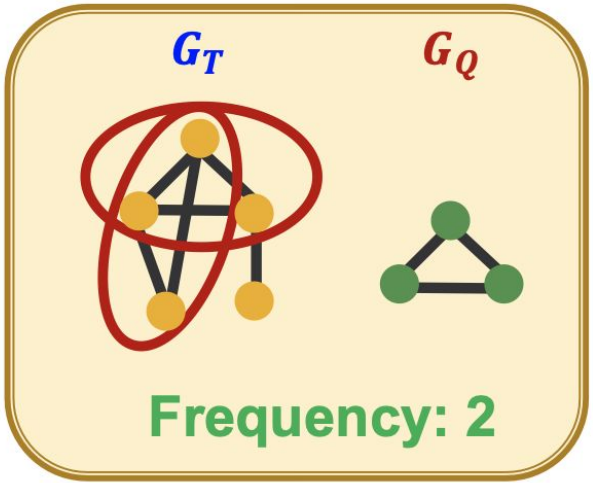
# “Мотивы”(Motifs)

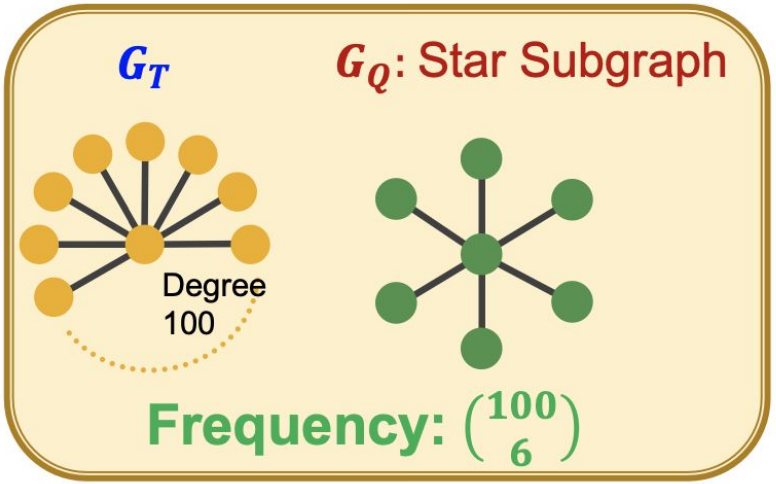


# “Мотивы”(Motifs)

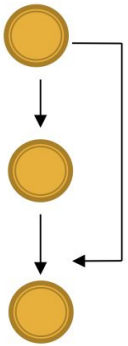
Помогают понять как устроен граф, по частоте вхождения.

Помогают нам делать прогнозы, основанные на присутствии или отсутствие присутствия в наборе данных.

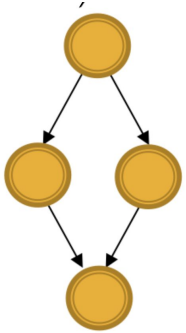


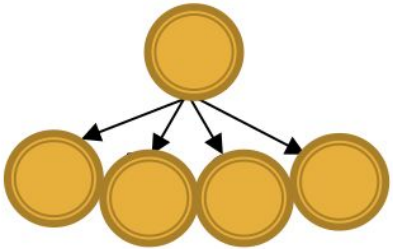
# “Мотивы”(Motifs)-примеры



Петли обратной связи: обнаруживаются в сетях нейронов, где они нейтрализуют “биологический шум”.



Параллельные петли: встречаются в пищевых сетях



Модули с одним входом: встречаются в сетях контроля генов

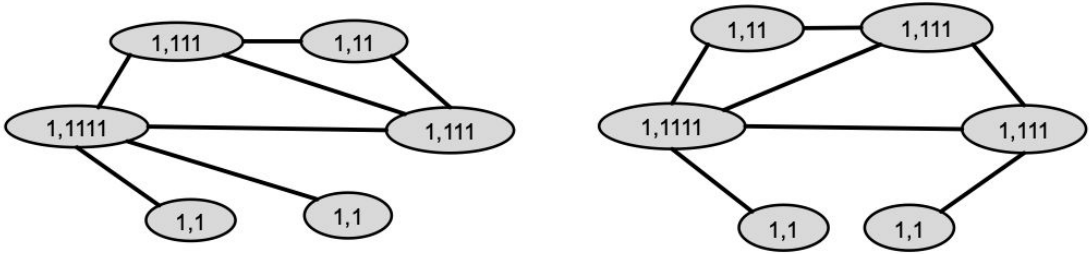
# Раскрашивание графа

$$c^{(k+1)}(v) = \text{HASH}\left(c^{(k)}(v), \left\{c^{(k)}(u)\right\}_{u \in N(v)}\right)$$

c(v) - цвет, v - вершина,N(v) -соседи

- Assign initial colors

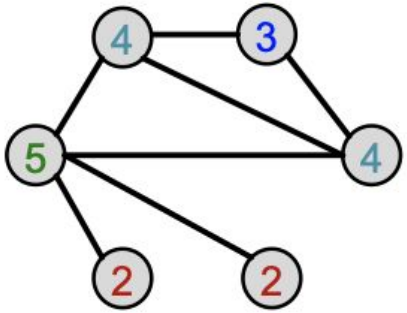

- Aggregate neighboring colors

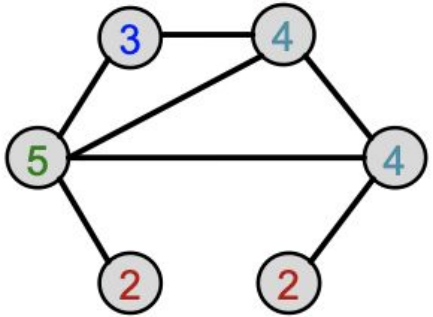


# Раскрашивание графа

С помощью хэш-функции основываясь на положении вершины в графе (ее ребрах) мы раскрашиваем наш граф. Так мы получаем сравнение неизоморфных графов.

- **Injectively** HASH the aggregated colors



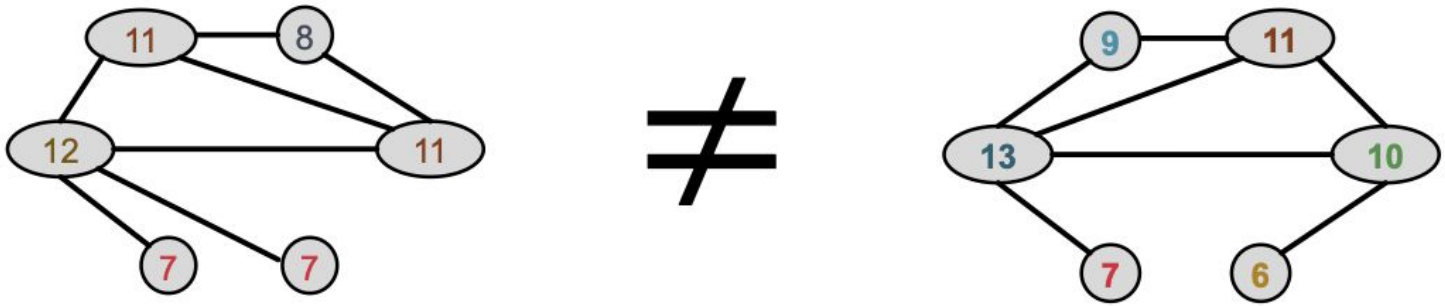




HASH table: **Injective!**

| | | |
|---|---|---|
| 1,1 | --> | 2 |
| 1,11 | --> | 3 |
| 1,111 | --> | 4 |
| 1,1111 | --> | 5 |

# Раскрашивание графа

Процесс продолжается до тех пор, пока не будет достигнута стабильная окраска.
Два графа можно назвать “изоморфными”, если они имеют одинаковый набор цветов.



# Функции ошибок

Функцию ошибок нужно брать отталкиваясь от задачи. Если задача хочет раскрасить вершины в разные цвета, то подойдут ошибки типичные для задач классификации - SVM, LogLoss. Если задача требует расставить веса в ребрах, то можно использовать MSE. Точно не указывается какие функции ошибок лучше для работы с графами, так как все зависит от задачи.

Далее разберем наиболее популярные функции ошибок для задач графов.

# Функции ошибок (наиболее популярные)

Данная функция ошибок нужна, для задач угадывания узлов.

$$\mathcal{L}(y_v, \hat{y_v}) = -\sum_c y_{vc} \log \hat{y_{vc}}.$$

$$\mathcal{L}_G = \frac{\sum_{v \in \mathrm{Lab}(G)} \mathcal{L}(y_v, \hat{y_v})}{|\mathrm{Lab}(G)|}$$

$\hat{y_{vc}}$ is the predicted probability that node $v$ is in class $c$.

# Функции ошибок (наиболее популярные)

Данная функция потерь чаще всего используется для задач построения ребер в графе. (Можно заметить большое сходство с LogLoss).

$$\mathcal{L}(y_v, y_u, e_{vu}) = -e_{vu} \log(p_{vu}) - (1 - e_{vu}) \log(1 - p_{vu})$$
$$p_{vu} = \sigma(y_v^T y_u)$$

where $\sigma$ is the sigmoid function, and $e_{vu} = 1$ iff there is an edge between nodes $v$ and $u$, being $0$ otherwise.

# Генерация графов

Многие графы даны нам изначально, но что делать если нам нужно создать новый?

Image credit: Medium

**Social Networks**

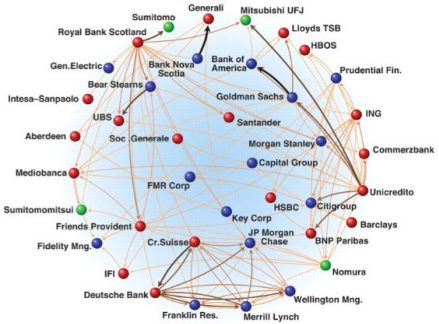



Image credit: Science

**Economic Networks**

Image credit: Lumen Learning

**Communication Networks**

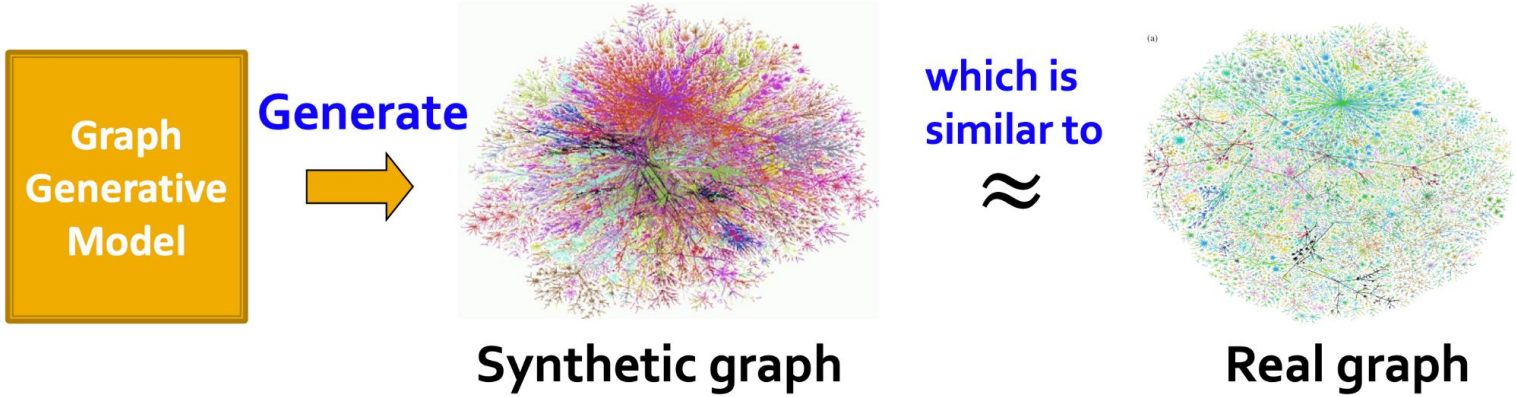
# Генерация графов

Что нам нужно, для генерации графов:

- Понимание - мы можем понять, как выглядит граф
- Прогнозы - мы можем предсказать, как будет развиваться граф в дальнейшем
- Моделирование - мы можем использовать тот же процесс для создания новых экземпляров графа
- Обнаружение аномалий - мы можем решить, является ли граф нормальным / ненормальным

# Генерация графов

Шаг 1: Свойства графиков реального мира
Шаг 2: Традиционные модели генерации графов
Шаг 3: Генеративные модели глубокого графа



# Итог

Чтобы решить задачу графов надо:

- Какой тип задачи перед нами стоит (уровень графа, вершины, ребра).
- Каким способом предпочтительнее разделить выборку.
- Выделить мотивы которые нам интересны и часто встречаются.
- Попробовать найти “изоморфные” по цвету вершины.
- Подобрать функцию ошибок.
- Обучить нашу модель.

# Материалы

https://web.stanford.edu/class/cs224w/

https://distill.pub/2021/gnn-intro/

https://distill.pub/2021/understanding-gnns/#learning