Графы и графовые вычисления

МОДУЛЬ 10

ШКОЛА БОЛЬШИХ ДАННЫХ

Знакомьтесь: граф

- (математика) совокупность множества вершин и множества ребер
- **конечный** и **бесконечный**
- **ориентированный** (дуги) и **неориентированный** (ребра)
- **связный** и **несвязный**
- смежные вершины, **маршрут** (путь)
- **дерево** - частный случай (связный без нетривиальных циклов)

Примеры использования

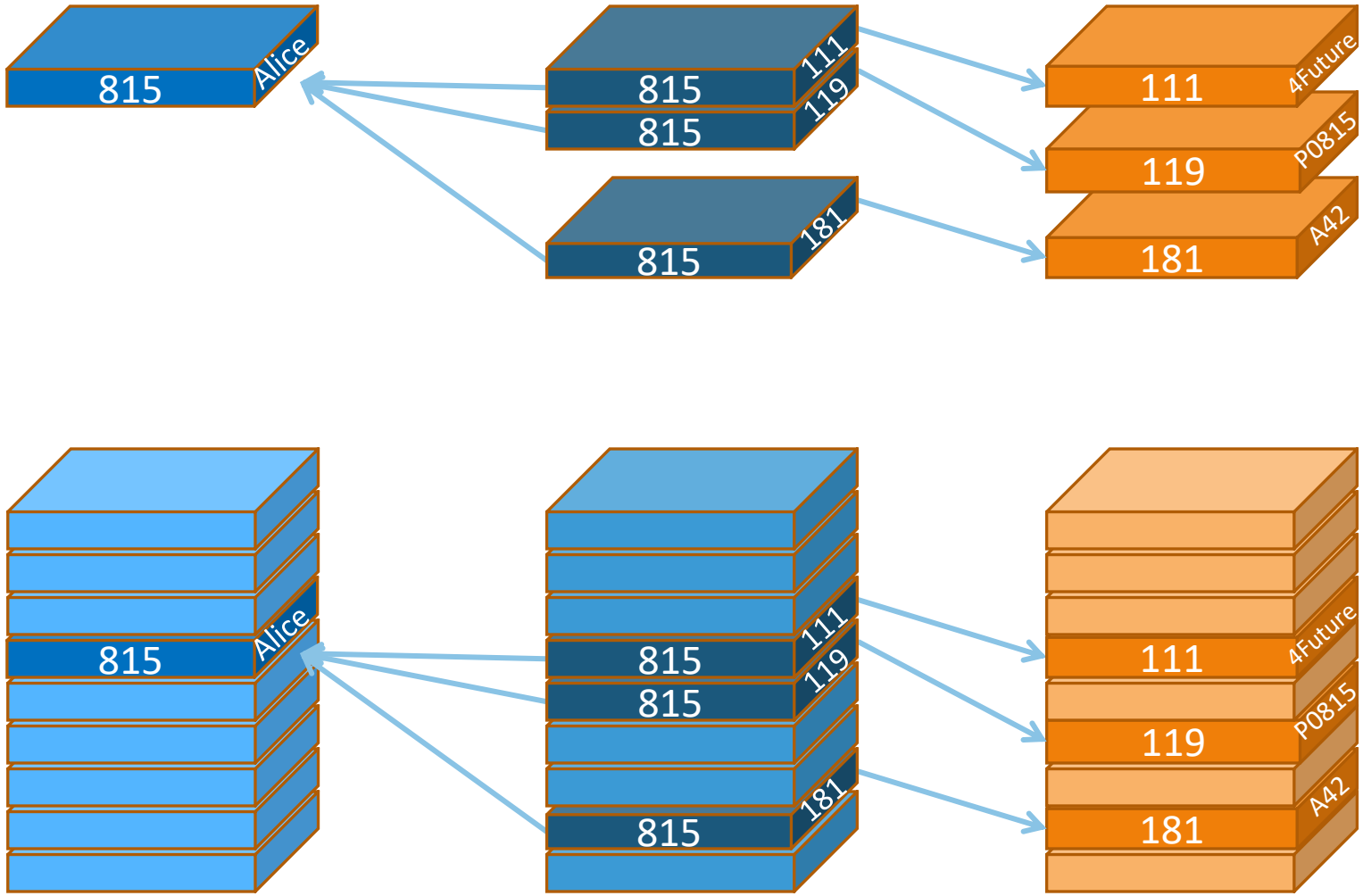
- предсказание поведения и назначение действия для динамически меняющихся групп
- 3 паттерна
  - ✓ пути распространения ("как оно распространяется")
  - ✓ поток и влияние ("каковы объемы, стоимость и точки управления")
  - ✓ взаимодействие и устойчивость ("взаимодействие объектов и потенциал изменения")
- в жизни
  - ✓ рекомендательные системы (на основе классификации)
- борьба с мошенничеством (разберем пример)
- обеспечение надежности (самые важные точки отказа)
- lineage, права доступа

Графовая модель данных

(на примере neo4j)

- **сущность** = вершина (**“node”**)
  - ✓ может иметь тип (**“label”**)
- **связь** между сущностями = ребро (**“relationship”**)
  - ✓ должна иметь тип (**“relationship type”**)
- **свойства** (атрибуты) = **“properties”**

Графовая модель vs реляционная



Графы и Resource Description Framework (RDF)

небо - синее (субъект = **небо**, предикат = **имеет цвет**, объект = **синий**)

- попытка создать метаданные интернета
- тройка (**“triplet”**)
  - ✓ субъект
  - ✓ предикат
  - ✓ объект
- специализированные хранилища (**“tripplestores”**)
- специализированный язык запросов (**“SPARQL”**)
- семантический механизм **рассуждений**

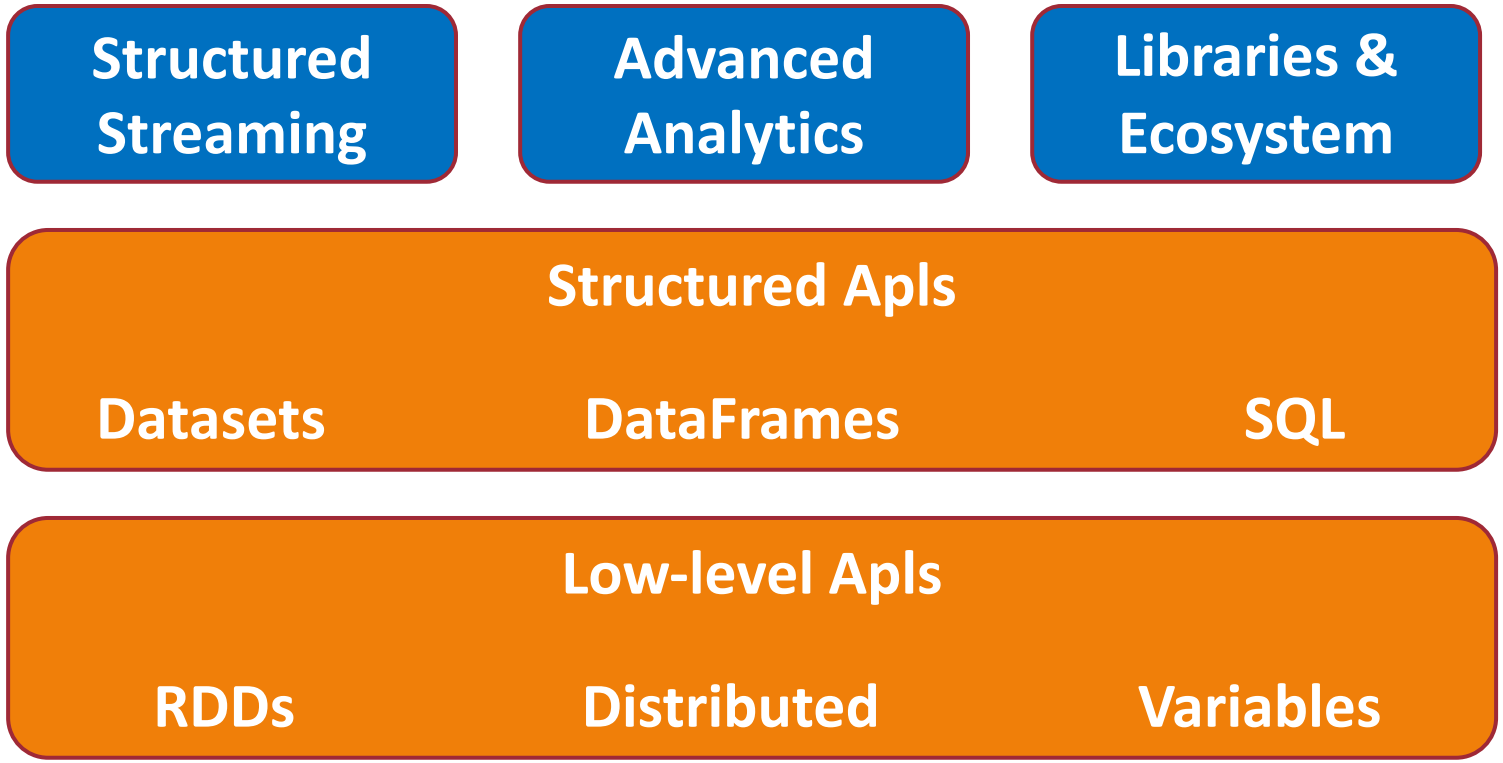
Что есть для работы с графами

- **Amazon Neptune** (https://aws.amazon.com/neptune)
- **Neo4j** (http://neo4j.com)
- **Spark**
- **TigerGraph** (https://www.tigergraph.com/ - The World's Fastest and Most Scalable Graph Platform)
- много вариантов поддержки **RDF**

Neo4j

- **графовая СУБД**
  - ✓ наиболее популярная в классе (по DB-Engines)
- поддержка **ACID**
- собственная система хранения и обработки
- **Cypher** - язык запросов
- **community edition**
  - ✓ просто устанавливается
  - ✓ вместе с документацией
  - ✓ красивый UI
- **Spark 3.0**

Графы и Spark



Graph API в spark: текущий статус

- **GraphX**: первая реализация (“RDD”)
- **GraphFrame**: вариант реализации для “dataframe” (модуль)
- **Spark 3.0 Graph**
  - ✓ добавлена поддержка **“cypher”**
  - ✓ **“Property Graphs”**

Что есть граф в Spark

- (математика) матрица смежности
- как представлен в Spark
  - ✓ **“vertex”** + **“edge”** RDD (GraphX)
  - ✓ **“vertex”** + **“edge”** Dataframe (GraphFrame)
- не графовая база

Основные абстракции

“GraphFrame”

- датафрейм с узлами (**“id”** плюс любой набор колонок)
- датафрейм с ребрами (**“src”**, **“dst”** плюс любой набор колонок)
- **атрибуты**
  - ✓ **“edges”**, **“vertices”**: ребра и узлы (датафрейм)
  - ✓ **“triplets”**: триплеты (датарфрейм)
  - ✓ **“degrees”**, **“inDegrees”**, **“outDegrees”**: количество "соседей" (входящих, исходащих) (датафрейм)

Основные методы GraphFrame

- информационные методы (**triangleCount()**)
- сервисные методы (**dropIsolatedVertices()**, **filterEdges()**, **filterVertices()**, **find()**)
- передача сообщений (**aggregateMessages()**)
- алгоритмы (**bfs()**, **connectedComponents()**, **stronglyConnectedComponents()**, **labelPropagation()**, **pageRank()**, **parallelPersonalizedPageRank()**, **shortestPaths()**, **svdPlusPlus()**)
- **pregel** - реализация парадигмы (см. ниже)

Pregel

Pregel

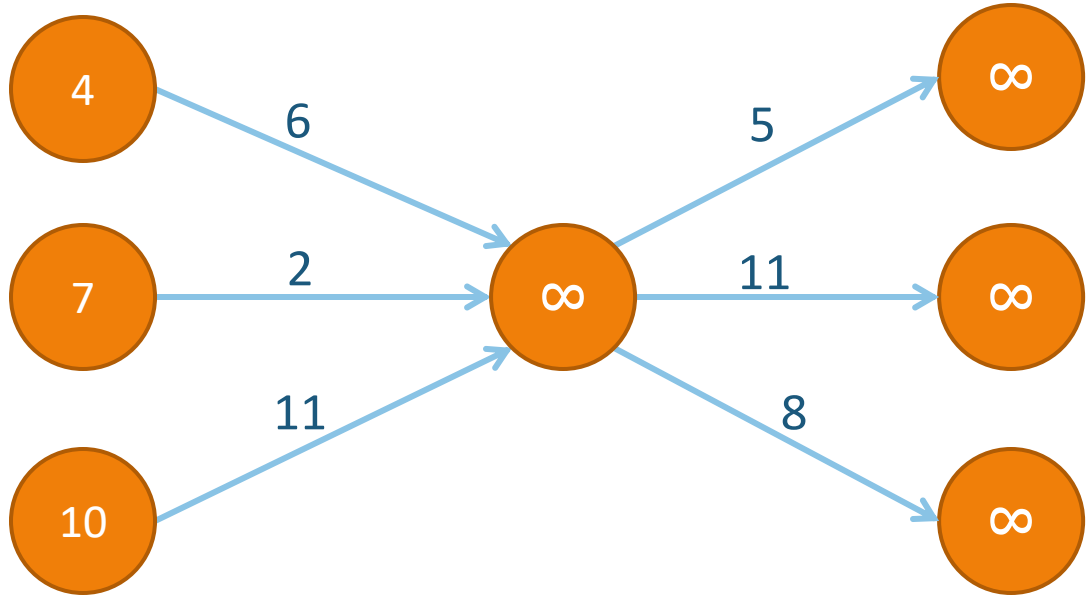
Pregel (a portmanteu of the words Parallel, Graph, and Google)

- **data flow** парадигма для решения графовых задач
- последовательность **итераций**
  - ✓ каждый узел может получить сообщение
  - ✓ обработать его (изменив состояние)
  - ✓ послать сообщение
- последовательность заканчивается, когда не послано ни одного сообщения
- результат: **состояние узлов графа**
- так можно реализовать `page rank`, `shortest path`, ...

Pregel: поиск кратчайшего пути

Состояние перед итерацией N

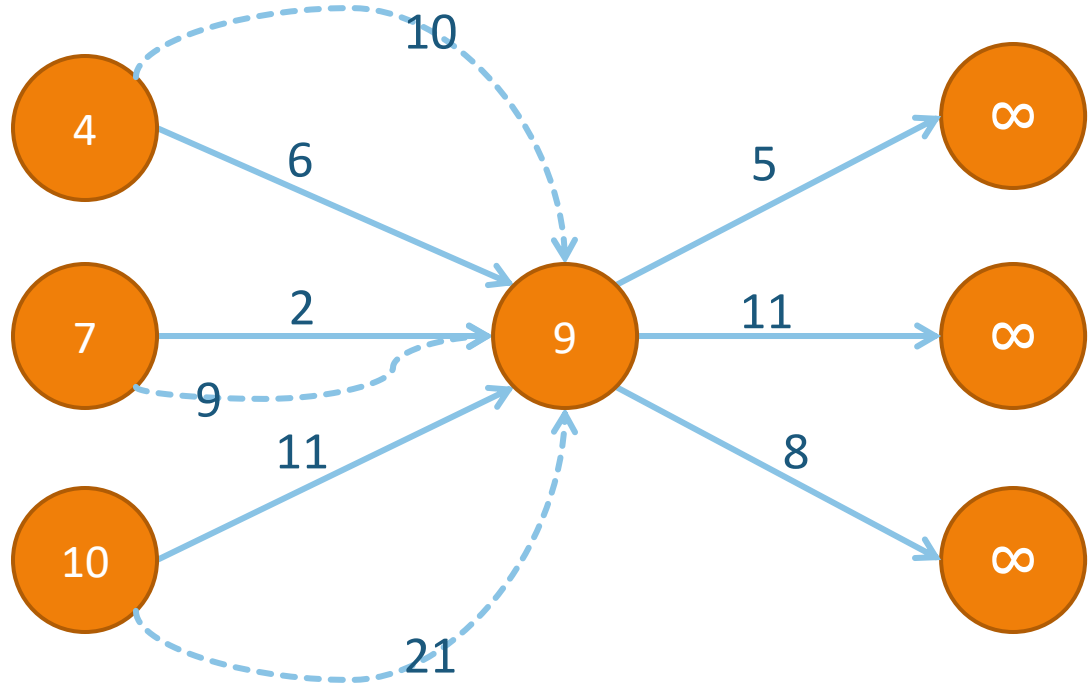
- ✓ в узле записан кратчайший путь к нему
- ✓ на ребре - расстояние до соседнего узла



Pregel: поиск кратчайшего пути

Итерация N

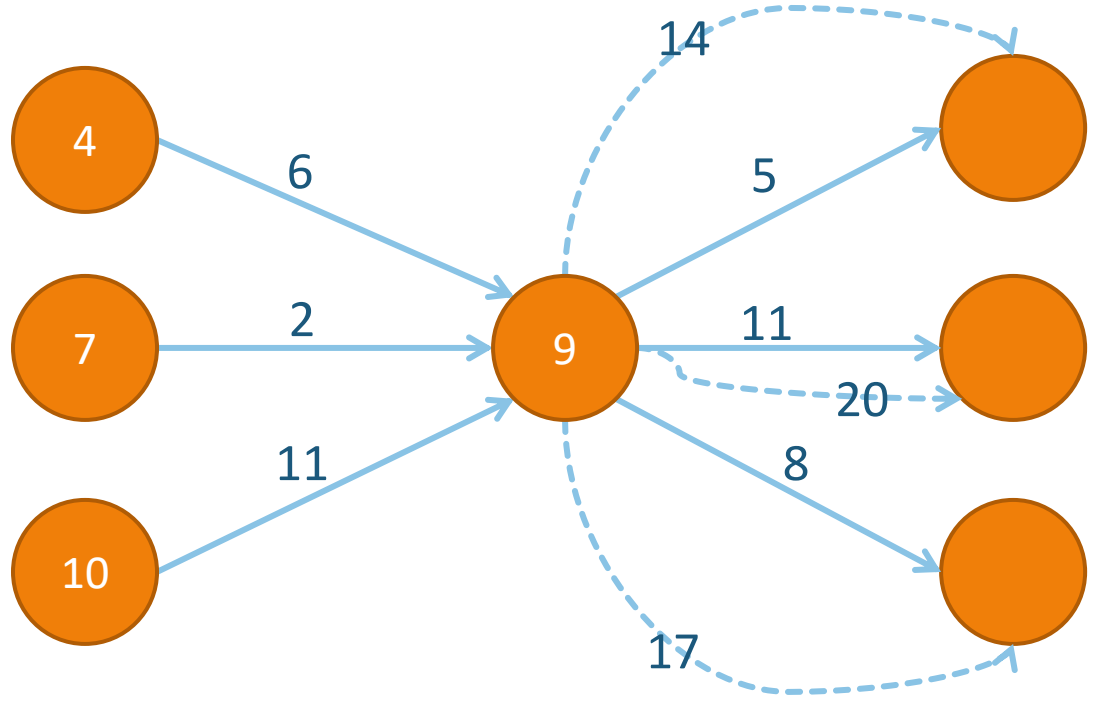
- ✓ три узла отправляют соседу (центр) рассчитанные значения кратчайшего пути к нему



Pregel: поиск кратчайшего пути

Итерация N+1

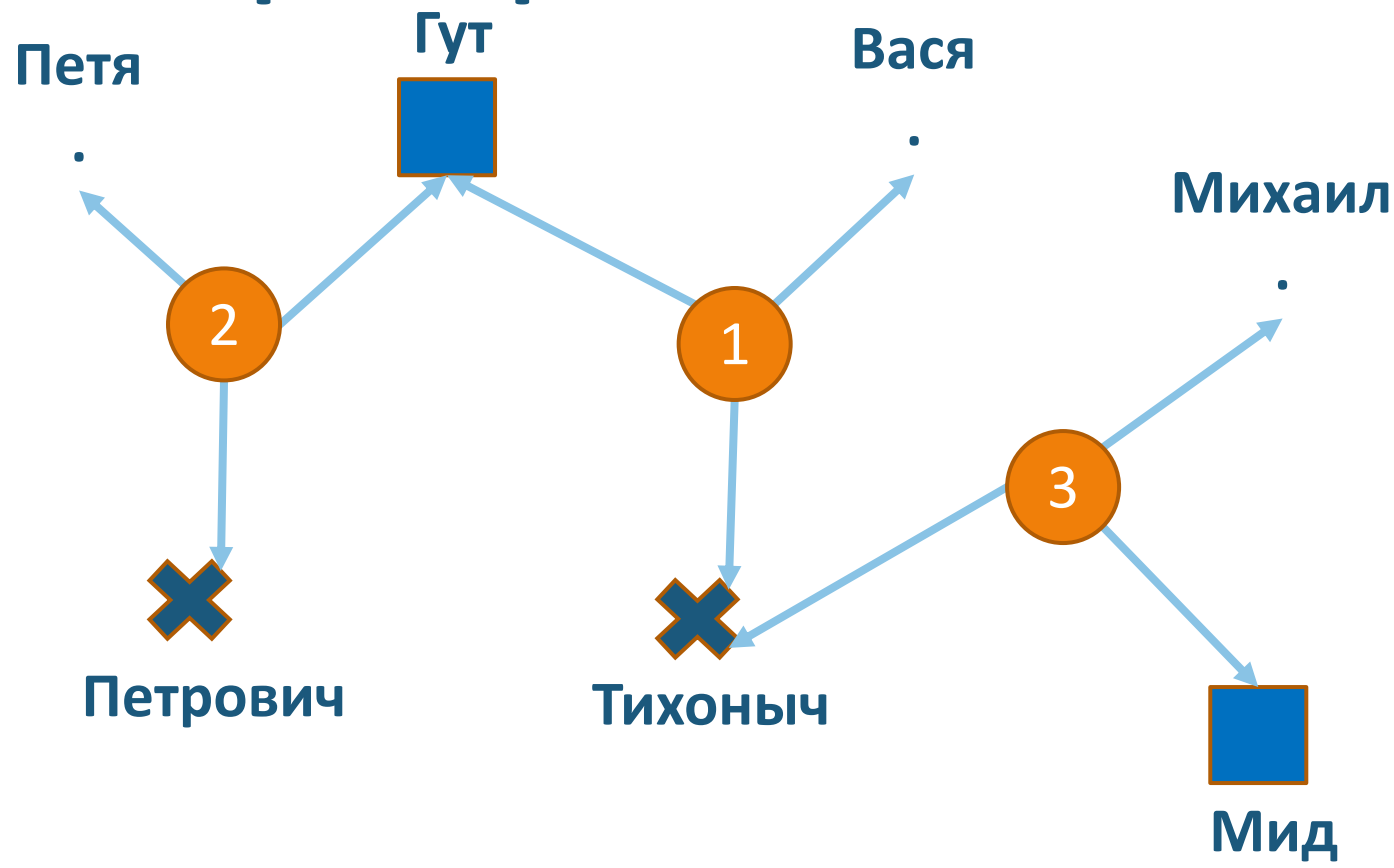
- ✓ центральный узел сохраняет минимальное из полученных значений
- ✓ центральный узел отправляет своим соседям рассчитанные расстояния до них



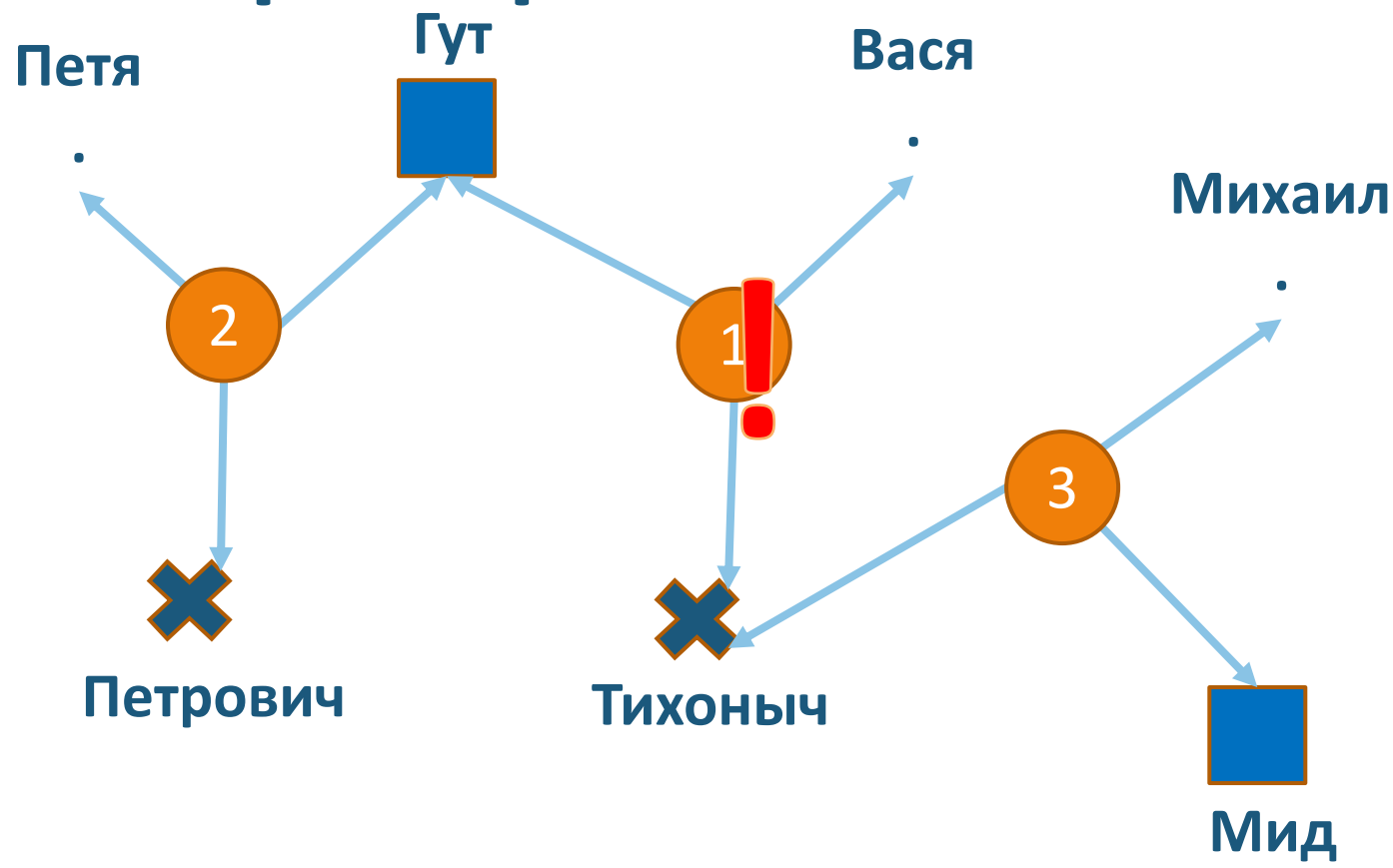
Пример: страховое мошенничество

- **незаконное получение** страхового возмещения (**по КАСКО**)
- **недостоверная информация** о страховом случае
- **сговор** с аварийным комиссаром
- сговор с агентом (реже)

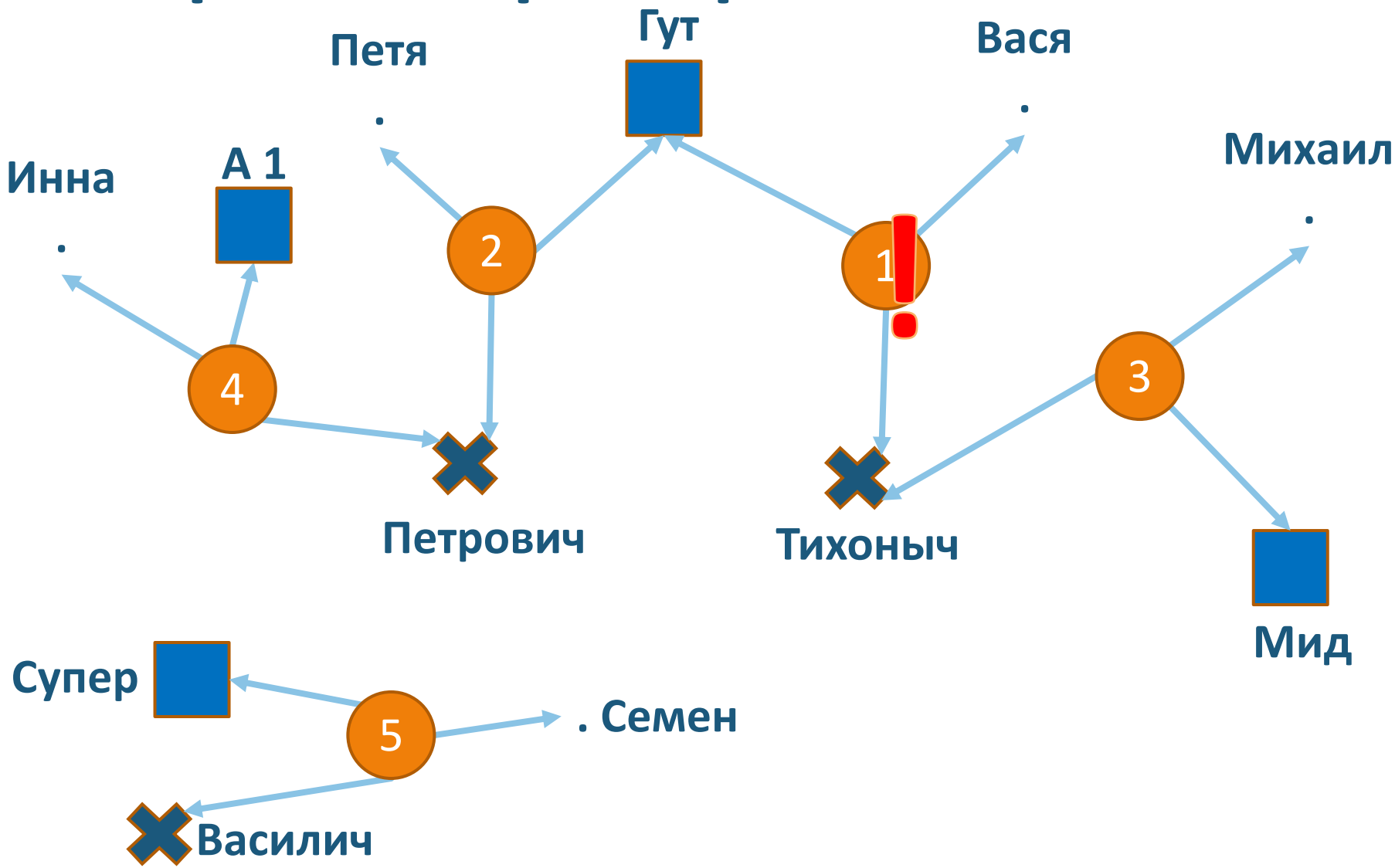
Разберем на примере



Разберем на примере

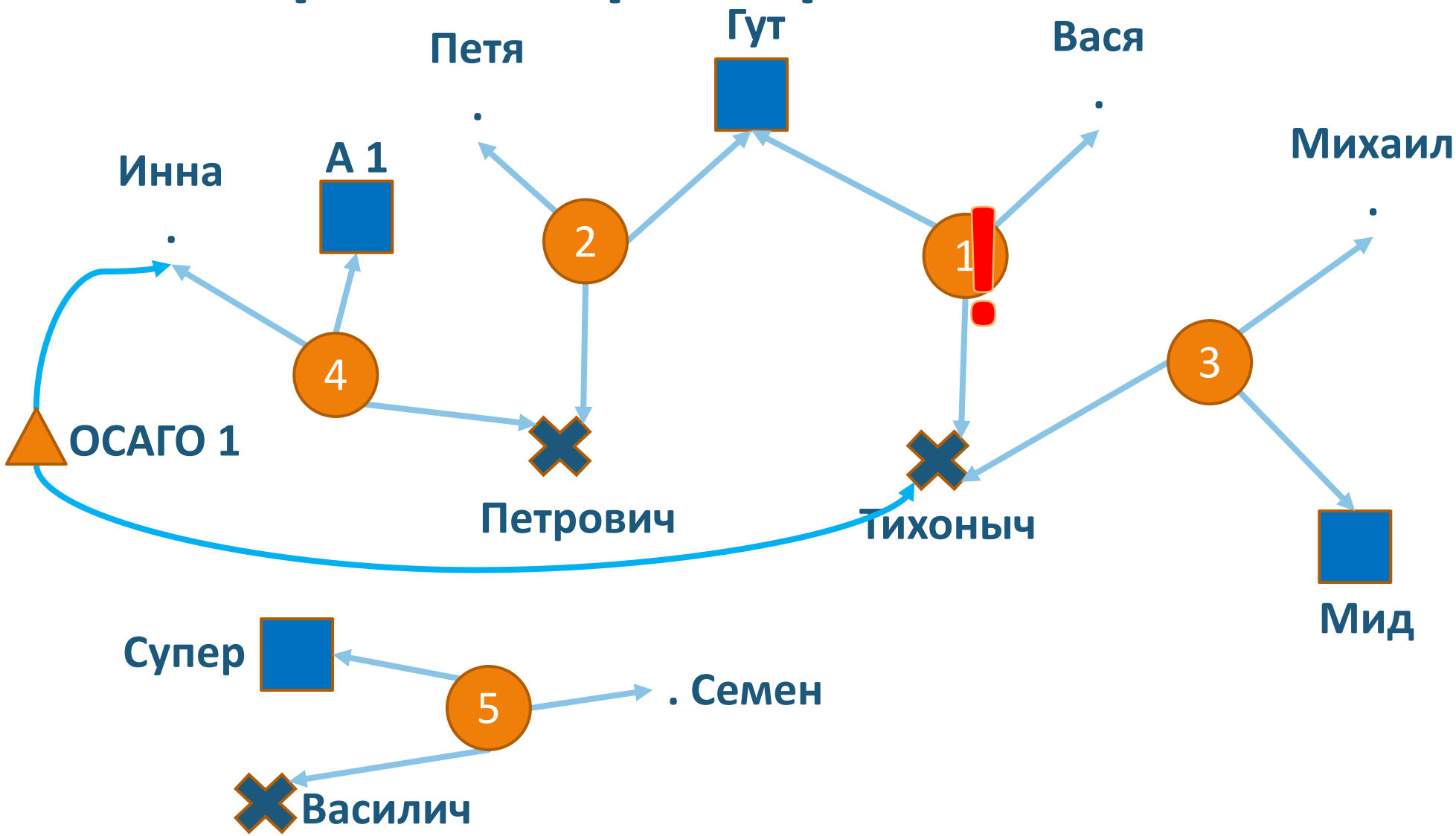


Разберем на примере



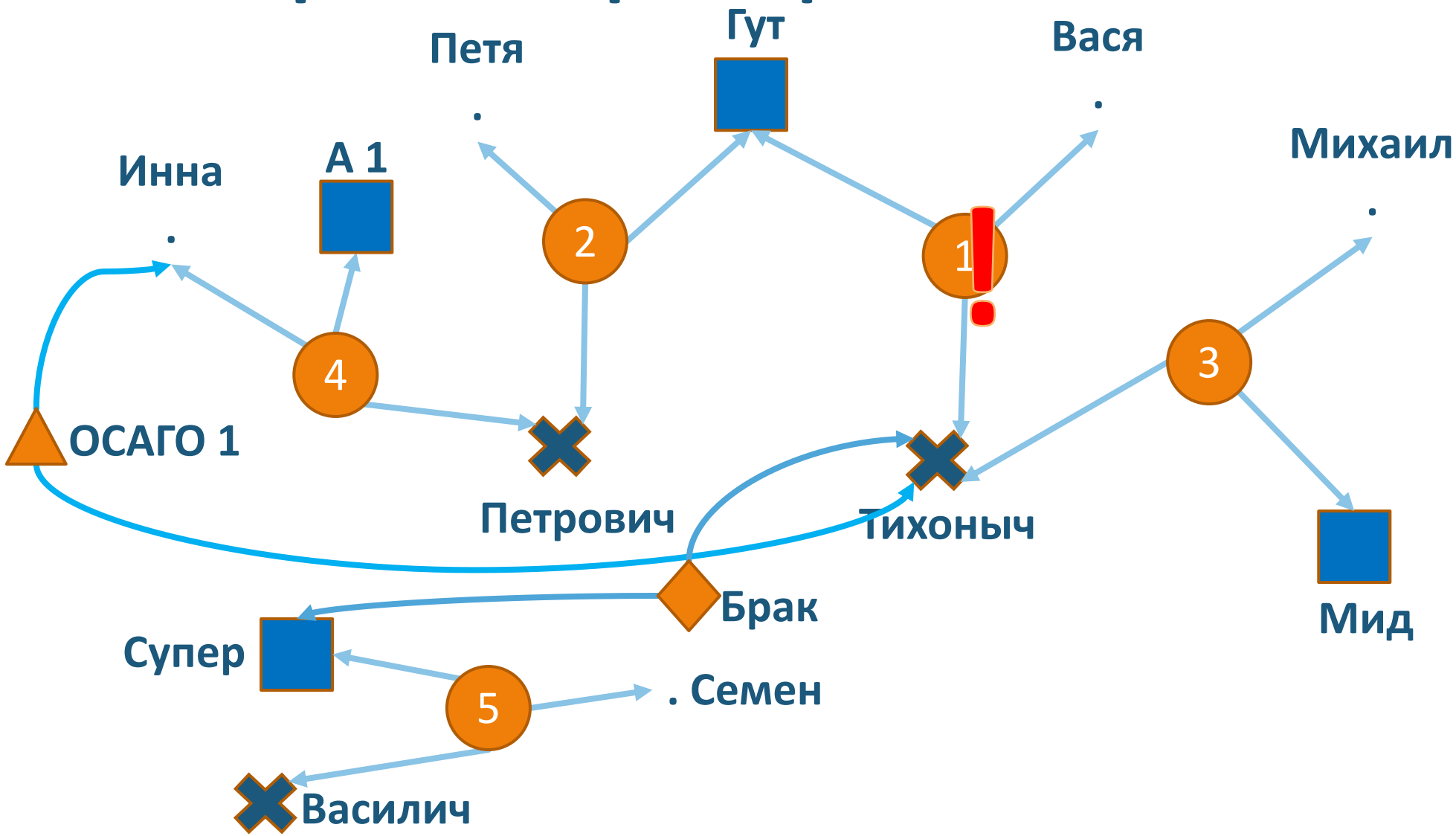
ШКОЛА БОЛЬШИХ ДАННЫХ

Разберем на примере



ШКОЛА БОЛЬШИХ ДАННЫХ

Разберем на примере



Сложности

- как построить граф (направленность, что есть ребра)?
- вклад **"связи"** в вероятность мошенничества?
- как влияет **расстояние**?
- каким **алгоритмом** считать?
- что считать алгоритмом, что проверять "глазами"?

Вопросы?