Вопросы по графам:

1. Основные понятия теории графов: граф, степень вершины, висячая вершина, изолированная вершина, смежные вершины, полный граф, псевдограф, мультиграф, двудольный граф:

Граф: Граф представляет собой непустое конечное множество вершин V и ребер E, оба конца которых принадлежат множеству V. Обозначать граф будем G (V, E)

Степень вершины: количество рёбер, инцидентных данной вершине.

Висячая вершина: вершина степени **1** (имеет только одно ребро).

Изолированная вершина: вершина степени **0** (не соединена ни с какими другими вершинами).

Смежные вершины: две вершины называются **смежными**, если они соединены ребром.

Полный граф: граф, в котором **каждая пара вершин соединена ребром**.

- Обозначение: Kn, где n — число вершин.
- Число рёбер: n(n−1)2.

Псевдограф — граф, в котором:

- допускаются **петли** (рёбра, соединяющие вершину саму с собой),
- могут быть **кратные рёбра** (несколько рёбер между одной парой вершин).

Мультиграф — граф, в котором:

- допускаются **кратные рёбра**,
- **нет петель**.

Двудольный граф: граф, вершины которого можно разбить на **два непересекающихся подмножества** U и V так, что:

- каждое ребро соединяет вершину из U с вершиной из V,
- **нет рёбер внутри U или внутри V**.

Обозначение: G=(U,V,E)

2. Способы задания неорентированных графов. Что такое матрица смежности и матрица инцидентности? Как по матрице смежности определить вид графа? Как по матрицам смежности и инцидентности определить степень вершины?

Способы задания неориентированных графов

Неориентированные графы можно задавать:

1. **Геометрически** (рисунком с точками и линиями).
2. **Матрицей смежности**.
3. **Матрицей инцидентности**.

---

1. Матрица смежности

Определение:
Матрица смежности представляет собой квадратную таблицу

размерами n × n, где n – число вершин графа. Строкам и колонкам матрицы ставятся в соответствие вершины, а на пересечениях строк и колонок записываются числа, показывающие, сколько ребер соединяют соответствующие вершины графа

Как определить вид графа по матрице смежности:

- **Простой граф**:
  - Все элементы 0 или 1.
  - На главной диагонали только 0 (нет петель).
- **Мультиграф**:
  - Есть элементы >1 (кратные рёбра).
  - На главной диагонали 0 (нет петель).
- **Псевдограф**:
  - На главной диагонали есть ≥1 (есть петли).

Как найти степень вершины:
Степень вершины i = **сумма элементов строки i** + **элемент на главное диагонали**

---

2. Матрица инцидентности

Определение:
Прямоугольная матрица n×m, где n — число вершин, m — число рёбер. Элемент bij:

- 1, если вершина i инцидентна ребру j.
- 2, если ребро j — **петля** для вершины i.
- 0, если вершина и ребро не связаны.

Как найти степень вершины:
Степень вершины i = **сумма элементов строки i**.

3. Операции над графами: дополнение, пересечение, объединение, сумма по модулю два. Что такое изоморфные графы.

Дополнение графа G — это граф с тем же множеством вершин, но рёбрами, которые отсутствуют в исходном графе L. Если в L есть ребро между вершинами u*u* и v*v*, то в G его нет, и наоборот.

Пересечение графов G1∩G2 — это граф, содержащий только те рёбра, которые присутствуют и в G1, и в G2. Вершины графа G1∩G2 — это пересечение множеств вершин G1 и G2.

Объединение 3графов G1∪G2 — это граф, содержащий все рёбра и вершины из G1 и G2. Если вершина или ребро присутствует хотя бы в одном из графов, оно будет в объединении.

Сумма по модулю два G1⊕G2 — это граф, содержащий рёбра, которые присутствуют либо в G1, либо в G2, но не в обоих одновременно. Вершины графа G1⊕G2 — это объединение множеств вершин G1 и G2.

Изоморфные графы — это графы, которые имеют одинаковую структуру, но могут быть по-разному изображены или обозначены. Графы G1 и G2 называются изоморфными, если существует **биекция** (взаимно однозначное соответствие) между их вершинами, сохраняющая смежность. То есть, если вершины u*u* и v*v* смежны в G1, то их образы f(u)*f*(*u*) и f(v)*f*(*v*) смежны в G2, и наоборот.

4. Основные понятия для связных графов: маршрут, цепь, простая цепь, замкнутая и разомкнутая цепь, цикл, связный и несвязный граф, степень связности, матрица расстояний, эксцентриситет вершин, радиус и диаметр графа.

Маршрут – последовательность вершин и ребер, где каждое ребро соединяет предыдущую и следующую вершины.

Цепь – маршрут без повторяющихся ребер.

Простая цепь – маршрут без повторяющихся вершин и ребер.

Замкнутая цепь – начинается и заканчивается в одной вершине.

Разомкнутая цепь – начинается и заканчивается в разных вершинах.

Цикл – замкнутая простая цепь (все вершины и ребра уникальны, кроме начально и конечной вершины)

Связный граф – между любыми двумя вершинами существует путь.

Несвязный граф – имеет хотя бы две вершины, между которыми нет пути.

Степень связности - Число компонент, из которых состоит граф

Матрица расстояний (или **матрица кратчайших расстояний**) — это квадратная матрица, которая показывает длины кратчайших путей между всеми парами вершин в графе.

Эксцентриситет вершины – максимальное расстояние от данной вершины до любой другой вершины графа.

Формула: $e(v) = \max_{u \in V} d(v, u)$, где $d(v, u)$ — расстояние между вершинами v и u.

Радиус графа – минимальный эксцентриситет среди всех вершин графа.

Формула: $\mathrm{radius}(G) = \min_{v \in V} e(v)$.

Диаметр графа – максимальный эксцентриситет среди всех вершин графа (наибольшее расстояние между любыми двумя вершинами).

Формула: $\mathrm{diameter}(G) = \max_{v \in V} e(v)$.

5. Что такое обход графа. Рекурсивный и итеративный алгоритмы обхода графа в глубину. Алгоритм обхода графа в ширину.

Обход графа — это процесс посещения всех вершин графа в определенном порядке. Используется для поиска путей, проверки связности, обнаружения циклов и решения других задач.

Обход в глубину идет вглубь графа, пока не упрется в тупик, затем возвращается и исследует другие ветви. Используется для поиска циклов, компонентов связности. **Рекурсивный** алгоритм помечаем вершину как посещенную, рекурсивно вызываем функцию для всех не посещенных соседей. **Итеративный (стек)** используем стек для хранения вершин, берем вершину из стека, обрабатываем и добавляем ее соседей.

Обход в ширину посещаем все соседние вершины перед углублением дальше. Используется для поиска кратчайшего пути в невзвешенном графе. Алгоритм: используем очередь. Посещаем вершину, добавляем всех ее соседей в очередь.

6. Понятия эйлерова графа и цикла. Теорема Эйлера. Алгоритм нахождения эйлерова цикла. Понятие уникурсальной линии.

Эйлеров граф - называется эйлеровым, если в нём существует **эйлеров цикл** – замкнутый путь, проходящий через каждое ребро ровно один раз, начинается и заканчивается в одной вершине.

Теорема Эйлера

Эйлеров цикл существует, если:

Граф **связный**.

Все вершины имеют чётную степень.

Эйлеров путь (но не цикл) существует, если:

Граф связный.

Ровно две вершины имеют нечётную степень (это начало и конец пути).

Алгоритм нахождения эйлерова цикла

I. Проверка существования эйлерова цикла

1. Граф **связный**.
2. Все вершины имеют **чётную степень**.

II. Поиск цикла

1. **Инициализация:**
   - Стек `st` (для обхода вершин).
   - Вектор `eulerCycle` (результат).
   - Начинаем с любой вершины `v`, кладём её в стек.
2. **Построение цикла:**
   - Пока стек не пуст:
     - Берём вершину `v` (не удаляя из стека).
     - Если у `v` есть смежная вершина `i`:
       - Кладём `i` в стек.
       - Удаляем ребро `v-i`.
     - Иначе:
       - Переносим `v` в `eulerCycle`.
       - Удаляем `v` из стека.
3. **Завершение:**
   - Проверяем, что цикл замкнутый.
4. **Вывод:**
   - Выводим `eulerCycle` в порядке обхода.

Уникурсальная линия – это фигура (граф), которую можно нарисовать, **не отрывая карандаша от бумаги и не проходя дважды по одному ребру**.

7. Понятия гамильтонова графа и цикла. Признаки определения гамильтонова цикла. Алгоритм определения гамильтонова цикла. Понятие полугамильтонова графа.

Гамильтонов цикл — цикл в графе, проходящий через **каждую вершину ровно один раз** (кроме начальной/конечной, которая повторяется).

Гамильтонов граф — граф, содержащий гамильтонов цикл.
Достаточные условия (но не необходимые):

1. **Теорема Дирака (1952)**:
   Если в графе с $n \geq 3$ вершинами степень каждой вершины $deg(v) \geq n/2$, то граф гамильтонов.
2. **Теорема Оре (1960)**:
   Если для любых двух несмежных вершин u и v выполняется $deg(u)+deg(v) \geq n$, то граф гамильтонов.

Алгоритм поиска гамильтоновых циклов

1. **Инициализация:**
   - Начинаем с вершины v, добавляем её в множество S (текущая цепь).
2. **Построение цепи:**
   - На каждом шаге добавляем к S первую возможную вершину из матрицы смежности M, которая ещё не в S.
3. **Откат (backtracking):**
   - Если для текущей вершины нет допустимых вершин для добавления, возвращаемся к предыдущей вершине и пробуем другой вариант.
4. **Проверка на гамильтонов цикл:**
   - Если длина S=n и есть ребро между последней и начальной вершиной — найден гамильтонов цикл.
   - Если длина S=n, но нет замыкающего ребра — откатываемся.
5. **Завершение:**
   - Алгоритм заканчивается, когда все возможные вершины перебраны, начиная с v.
   - Все найденные циклы — все гамильтоновы циклы графа.

8. Понятия плоского графа, грани графа, планарного графа. Теорема Эйлера о плоских графов. Следствие из теоремы.

Граф называется **плоским**, если он может быть нарисован на плоскости **без пересечения рёбер** (кроме общих вершин).
Граф называется **планарным**, если он **изоморфен плоскому графу** (т.е. его можно "разложить" на плоскости без пересечений).
В плоском графе **грань** – это область, ограниченная рёбрами (включая внешнюю бесконечную грань).

Теорема:
Пусть n – число вершин связного плоского графа G
r – число его ребер
q – число граней. Тогда
$n + q = r + 2$. (1)
Эту теорему Л. Эйлер доказал в 1752 г.
Следствие:
Формула Эйлера распространяется и на многокомпонентные графы:
$n + q = r + k + 1$, (2)
где k – число компонент несвязного графа.

9. Гомеоморфные графы. Критерий Понтрягина-Куратовского.

Два графа называются **гомеоморфными**, если один можно получить из другого с помощью последовательности следующих операций:

- **Подразделение ребра** — добавление новой вершины на ребро, разбивая его на два.
- **Обратная операция (сглаживание вершины степени 2)** — если вершина имеет ровно двух соседей, её можно удалить, соединив соседей ребром.

Критерий Понтрягина-Куратовского
Этот критерий даёт необходимое и достаточное условие планарности графа, используя понятие гомеоморфности.
Формулировка:
Граф **планарен** тогда и только тогда, когда он не содержит подграфов, гомеоморфных полному графу K5 или полному двудольному графу K3,3.
Пояснение:

- G5 — полный граф с 5 вершинами, где каждая вершина соединена с каждой.
- G3,3 — полный двудольный граф, где каждая из 3 вершин одной доли соединена с каждой из 3 вершин другой доли.

Применение:
Чтобы доказать, что граф непланарен, достаточно найти подграф, гомеоморфный K5 или K3,3.

10. Понятия дерева, леса. 4 теоремы.

Несвязный граф, не содержащий циклов, называется лесом. Связный граф, не содержащий циклов, называется деревом.
Теорема 1
Всякое дерево содержит $n - 1$ ребер, где n – число вершин.
Теорема 2
Всякий лес содержит $n-k$ ребер, где k – число компонент связности.
Теорема 3
Любые две вершины дерева соединены точно одной простой цепью.
Теорема 4

Если в дереве любые две вершины соединить ребром, то в графе появится один цикл.

11. Остовные деревья. Понятие цикломатического числа. Область применения. Постановка задачи о нахождении минимального остовного дерева. Алгоритм Крускала. Алгоритм Примы.

Остовным деревом (spanning tree) связного неориентированного графа G на n вершинах будем называть подмножество его ребер размера n − 1, не содержащее циклов. Понятно, что тогда по этим ребрам можно добраться из любой вершины до любой, и эти ребра образуют дерево.

Цикломатическое число: это минимальное количество рёбер, которые нужно удалить из графа GG, чтобы он стал **ациклическим** (т.е. превратился в лес). число ребер - число вершин + число компонентов связности.

Постановка задачи о МОД

Дано:

- **Связный неориентированный взвешенный граф** G=(V,E), где:
    - V — множество вершин,
    - E — множество рёбер,
    - каждому ребру присвоен вес

Требуется:
Найти **остовное дерево** T=(V,E′), где E′⊆E, такое что:

1. связный ациклический граф
2. Суммарный вес рёбер T **минимален**

Алгоритм Краскала:

1. **Инициализация**:
    - Сортируем все рёбра графа по весу (по возрастанию).
    - Начинаем с пустого множества рёбер A0, где граф состоит из изолированных вершин.
2. **Построение минимального остова**:
    - На каждом шаге выбираем ребро с наименьшим весом.
    - Если оно соединяет две разные компоненты связности — добавляем его в остов.
    - Объединяем компоненты (через систему непересекающихся множеств — DSU).
    - Повторяем, пока не останется **одна компонента связности** (получим дерево).

3. **Результат**:
   Множество выбранных рёбер образует **минимальное остовное дерево** (МОД).

Алгоритм Прима:

1. **Начинаем с произвольной вершины** (например, вершины `A`).
2. **На каждом шаге:**
   - Выбираем **минимальное ребро**, соединяющее уже включённые в дерево вершины с оставшимися.
   - Добавляем это ребро и новую вершину в дерево.
3. **Повторяем**, пока все вершины не будут включены в дерево.

12. Ориентированные графы. Понятие дуги, орграфа, основания орграфа, смешенного графа. Степень вершин орграфа.

Ориентированный граф— это граф, в котором рёбра имеют направление.

Дуга — направленное ребро

Орграф – граф состоящий из дуг

Основание орграфа — неориентированный граф, полученный из орграфа удалением направлений дуг

Смешанный граф — граф, содержащий как ориентированные, так и неориентированные рёбра.

Степени вершин в орграфе

В орграфе различают две степени для каждой вершины:

Входа — количество дуг, **выходящих** из вершины

Выхода — количество дуг, **входящих** в вершину

13. Матрица смежности. Матрица инцидентности, матрица смежности дуг орграфа.

1. **Матрица смежности** ($n \times n$):
   - Для **неориентированного графа**: симметричная, $A[i][j] = 1$, если есть ребро $\{i, j\}$.
   - Для **орграфа**: $A[i][j] = 1$, если есть дуга ($i \to j$).
2. **Матрица инцидентности** ($n \times m$, где m — число рёбер/дуг):
   - Для **неориентированного графа**: $B[v][e] = 1$, если вершина v принадлежит ребру e.
   - Для **орграфа**: $B[v][e] = -1$ (начало дуги), 1 (конец дуги), 0 (иначе).
3. **Матрица смежности дуг** ($m \times m$, только для орграфа):
   - $C[a][b] = 1$, если конец дуги a совпадает с началом дуги b.

14. Маршруты, цепи и циклы в орграфах. Достижимая вершина. Сильно связный, слабо связный, несвязный орграф.

1. Основные понятия

- **Маршрут** – последовательность дуг, где конец каждой предыдущей дуги совпадает с началом следующей.
- **Цепь** – маршрут **без повторения дуг**.
- **Простая цепь** – цепь **без повторения вершин** (кроме, возможно, начала и конца).
- **Цикл** – замкнутая цепь (начало и конец совпадают).

2. Достижимость

- Вершина v **достижима** из u, если существует путь $u \to v$.

3. Связность орграфов

1. **Сильно связный** – из любой вершины достижима любая другая (включая обратные пути).
   - Пример: цикл $1 \to 2 \to 3 \to 1$
2. **Слабо связный** – основание графа (без ориентации) связно, но не сильно связный.
   - Пример: $1 \to 2 \leftarrow 3$
3. **Несвязный** – есть хотя бы две недостижимые друг для друга вершины.

15. Эйлерова цепь. Теорема о том, что орграф содержит замкнутую эйлерову цепь.

Ориентированная замкнутая цепь называется **эйлеровой**, если она содержит все дуги графа (эйлеров цикл). Если ориентированная разомкнутая цепь содержит все дуги графа, то такая цепь называется **полуэйлеровой**.

Теорема

Орграф содержит замкнутую эйлерову цепь тогда и только тогда, когда он является слабо связным и когда каждая вершина имеет степень входа, равную степени выхода.

Следствие из теоремы

Ориентированный граф содержит разомкнутую эйлерову цепь, если одновременно выполняются следующие условия:

а) орграф является слабо связным;

б) в орграфе существует одна вершина, степень выхода которой на единицу больше степени входа;

в) в орграфе существует одна вершина, степень входа которой на единицу больше степени выхода;

г) степень входа каждой из остальных вершин равна степени выхода.

16. Взвешенный граф. Алгоритм. Постановка задачи нахождения кратчайших путей. Алгоритм Дейкстры. Алгоритм Флойда. Алгоритм Белмана-Форда.

Постановка задачи нахождения кратчайших путей

Найти **кратчайший путь** от s до всех других вершин v∈V (или до конкретной t)

. Алгоритм Дейкстры (1959)

Назначение: Нахождение кратчайших путей от фиксированной вершины до всех остальных в графе с **неотрицательными весами**.

Ключевые принципы:

1. **Инициализация**:
   - Начальной вершине `s` присваивается метка 0.
   - Всем остальным вершинам присваивается метка ∞.
2. **Основной процесс**:
   - На каждом шаге выбирается вершина `u` с минимальной временной меткой.
   - Для всех соседей `v` вершины `u` выполняется релаксация:
     - Если найден более короткий путь через `u`, метка `v` обновляется.
     - $m(v)=\min(m(v),m(u)+\rho(u,v))$
3. **Завершение**:
   - Алгоритм завершается, когда все вершины получат окончательные метки.
   - Метки представляют длины кратчайших путей от `s`.

Особенности:

- Не работает с отрицательными весами.
- Использует жадную стратегию выбора вершин.
- Эффективен для разреженных графов.

2. Алгоритм Флойда-Уоршелла

Назначение: Нахождение кратчайших путей **между всеми парами вершин**.

Ключевые принципы:

1. **Инициализация**:
   - Матрица расстояний заполняется весами рёбер.
   - Диагональные элементы (пути в самих себя) устанавливаются в 0.
2. **Динамическое программирование**:
   - Для каждой промежуточной вершины `k`:
     - Для каждой пары вершин `(i, j)`:
       - Проверяется, улучшит ли путь через `k` текущее расстояние.
3. **Обнаружение циклов**:
   - После завершения проверяется наличие отрицательных циклов.

Особенности:

- Работает с любыми весами, кроме отрицательных циклов.
- Высокая вычислительная сложность.
- Эффективен для плотных графов.

3. Алгоритм Беллмана-Форда (1958/1962)

Назначение: Нахождение кратчайших путей из одной вершины в графе с **произвольными весами**.

Ключевые принципы:

Алгоритм Беллмана-Форда:

1. **Инициализация**

   - Всем вершинам, кроме стартовой, присвой расстояние = ∞
   - Стартовой вершине — расстояние = 0

2. **Релаксация рёбер**
   Повтори (V-1) раз:

   - Для каждого ребра (u → v) проверь:
     Если расстояние до u + вес ребра < текущего расстояния до v:

     - Обнови расстояние до v

     - Запомни u как предшественника v

3. **Проверка на отрицательные циклы**

   - Ещё раз пройди по всем рёбрам
   - Если найдётся ребро, где можно улучшить расстояние — есть отрицательный цикл

Особенности:

- Медленнее Дейкстры, но обрабатывает отрицательные веса.
- Способен обнаруживать отрицательные циклы.
- Используется в сетевых протоколах.

17. Постановка задача коммивояжера. Алгоритм решения задачи.

Формулировка:

Дано:

- n городов (вершин графа)
- Матрица расстояний dij между каждой парой городов i и j

Требуется:

Найти **гамильтонов цикл** минимальной длины, проходящий через каждый город **ровно один раз** и возвращающийся в исходный город.

Жадный алгоритм "Ближайший сосед"

Как работает:

1. Начинаем с любого города (обычно первого)
2. На каждом шаге выбираем ближайший непосещённый город
3. Добавляем его в маршрут
4. Повторяем, пока не посетим все города
5. Возвращаемся в начальный город

18. Постановка задачи о максимальном потоке. Алгоритм решения задачи.

Алгоритм Форда-Фалкерсона

Шаг 1: Начало

- Назначаем всем трубам начальный поток = 0
- Строим **остаточную сеть** — копию исходной сети, где пропускная способность показывает, сколько ещё можно пропустить

Шаг 2: Поиск увеличивающего пути

Ищем **любой путь** от истока к стоку в остаточной сети, где:

- Все трубы на пути имеют остаточную пропускную способность > 0

Пример:

Исток → A → B → Сток (если все рёбра A→B и B→Сток могут пропустить воду)

Шаг 3: Увеличение потока

- Находим **минимальную пропускную способность** на этом пути (это «узкое место»)
- Увеличиваем поток вдоль всего пути на эту величину
- Обновляем остаточную сеть:
  - Для каждой трубы на пути уменьшаем остаточную пропускную способность
  - Добавляем обратные трубы (чтобы «отменять» поток при необходимости)

Пример:

Если минимальная пропускная способность на пути = 5, добавляем +5 ко всему потоку.

Шаг 4: Повторение

Повторяем шаги 2–3, пока есть хотя бы один увеличивающий путь.

Шаг 5: Результат

Сумма потоков, выходящих из истока (или входящих в сток) — это **максимальный поток**.