# Ответ 2

## Объясните, как многоуровневая структура HNSW-графа в Qdrant обеспечивает баланс между скоростью поиска и точностью результатов, и как выбор метрики (например, косинусное сходство) влияет на работу этого механизма в контексте текстовых эмбеддингов.

На основе предоставленного учебного материала, многоуровневая структура HNSW-графа в Qdrant и выбор метрики, такой как косинусное сходство, работают синергетически для достижения баланса между скоростью и точностью поиска. Этот механизм можно объяснить в двух частях: структура графа и математическая основа для навигации по нему.

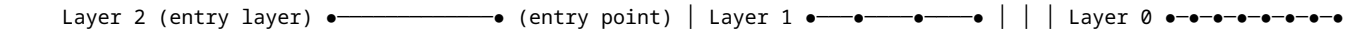
### 1. Многоуровневая структура HNSW: Баланс скорости и точности

Сердцем механизма Approximate Nearest Neighbor (ANN) поиска в Qdrant является алгоритм HNSW (Hierarchical Navigable Small World). Его эффективность обусловлена многоуровневой графовой структурой, которая обеспечивает компромисс между скоростью и точностью.

#### Концепция и структура

Как указано в материале, HNSW-граф можно представить по аналогии с картой дорог. Чтобы быстро преодолеть большое расстояние, вы используете скоростные автомагистрали (long-range connections), а для точного прибытия в пункт назначения — локальные дороги.

1. **Многоуровневая структура:** HNSW строит несколько слоев графа.
    - **Нижний уровень (Layer 0):** Содержит абсолютно все векторы в коллекции. Этот уровень обеспечивает максимальную точность.
    - **Верхние уровни (Layer 1, Layer 2, ...):** Каждый последующий слой является все более разреженным подмножеством узлов из слоя ниже. Эти слои служат "экспресс-шоссе" для быстрой навигации. Узел попадает на верхний уровень с определенной вероятностью, что обеспечивает логарифмическую зависимость количества слоев от общего числа векторов.




#### Процесс поиска и его эффективность

Процесс поиска в HNSW-графе напрямую демонстрирует, как достигается баланс:

1. **Начало поиска:** Поиск начинается с точки входа (entry point) на самом верхнем, наиболее разреженном слое.
2. **Быстрая навигация (Скорость):** На верхних уровнях используется жадный алгоритм. Система движется по графу к узлу, ближайшему к запросному вектору $\vec{q}$. Поскольку эти слои разрежены, каждый шаг позволяет "перепрыгивать" через большие области векторного пространства. Это снижает вычислительную сложность с линейной $O(N)$ до логарифмической $O(\log N)$.
3. **Пошаговое уточнение (Точность):** Когда на текущем уровне находится локальный минимум (узел, ближе которого нет ни одного из его соседей), этот узел используется как точка входа для поиска на уровне ниже.
4. **Финальный поиск:** Процесс рекурсивно повторяется до тех пор, пока не будет достигнут `Layer 0`. На этом самом плотном уровне, содержащем все векторы, выполняется наиболее точный локальный поиск для нахождения ближайших соседей.

Таким образом, **баланс достигается за счет того, что верхние уровни графа обеспечивают скорость, позволяя быстро приблизиться к нужной области пространства, а нижние уровни обеспечивают точность, выполняя детальный поиск в этой уже локализованной области.** Это позволяет находить "достаточно близкие" векторы на порядки быстрее, чем полный перебор (brute-force).

### 2. Влияние метрики: Косинусное сходство для текстовых эмбеддингов

Чтобы HNSW-граф функционировал, ему необходим способ измерения "расстояния" или "близости" между узлами-векторами. Этот способ определяется выбранной метрикой. В контексте текстовых эмбеддингов, полученных от **transformers**-моделей, наиболее распространенной и подходящей метрикой является косинусное сходство.

**Математическая основа косинусного сходства**

Как объясняется в учебном материале, косинусное сходство измеряет косинус угла между двумя векторами, что позволяет оценить их направленность независимо от их длины.

- **Шаг 1: Определение символов.** Формула выводится из определения скалярного произведения векторов $\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\|\|\mathbf{B}\| \cos(\theta)$, где:
  - $\mathbf{A}$ и $\mathbf{B}$ — это два вектора, например, $\mathbf{A} = [A_1, A_2, \ldots, A_n]$.
  - $\mathbf{A} \cdot \mathbf{B}$ — скалярное произведение, равное $\sum_{i=1}^{n} A_i B_i$.
  - $\|\mathbf{A}\|$ — норма (длина) вектора, равная $\sqrt{\sum_{i=1}^{n} A_i^2}$.
  - $\theta$ — угол между векторами.
- **Шаг 2: Выражение косинуса угла.** Из определения скалярного произведения, косинус угла можно выразить как:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

- **Шаг 3: Полная формула.** Подставляя определения, получаем полную формулу для вычисления сходства:

$$\text{similarity} = \cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

**Влияние на работу HNSW в контексте текстовых эмбеддингов**

Выбор косинусного сходства имеет ключевое практическое значение для работы с текстовыми данными:

1. **Фокус на семантике:** Для текстовых эмбеддингов важна *ориентация* вектора, которая кодирует семантическое направление (смысл), а не его длина. Материал приводит пример: фразы "AI is powerful" и "Artificial intelligence is very powerful" семантически очень близки. Их векторы будут указывать почти в одном направлении, но могут иметь разную длину. Косинусное сходство, благодаря делению на нормы векторов (нормализации), нивелирует влияние длины и фокусируется исключительно на направлении, что корректно отражает семантическую близость.

2. **Навигация по графу:** При работе HNSW-алгоритма в Qdrant именно эта метрика используется на каждом шаге поиска. Когда система находится в определенном узле графа, она вычисляет косинусное сходство между запросом $\vec{q}$ и всеми соседними узлами. Узел с наибольшим значением сходства (ближе к 1) выбирается как следующий шаг на пути к цели. Таким образом, **выбор метрики напрямую определяет, как HNSW будет "видеть" пространство и по какому пути он будет перемещаться от "скоростных шоссе" верхних уровней к "локальным дорогам" нижнего уровня.** Использование косинусного сходства гарантирует, что эта навигация будет основана на семантической близости, что критически важно для задач вроде **RAG**.