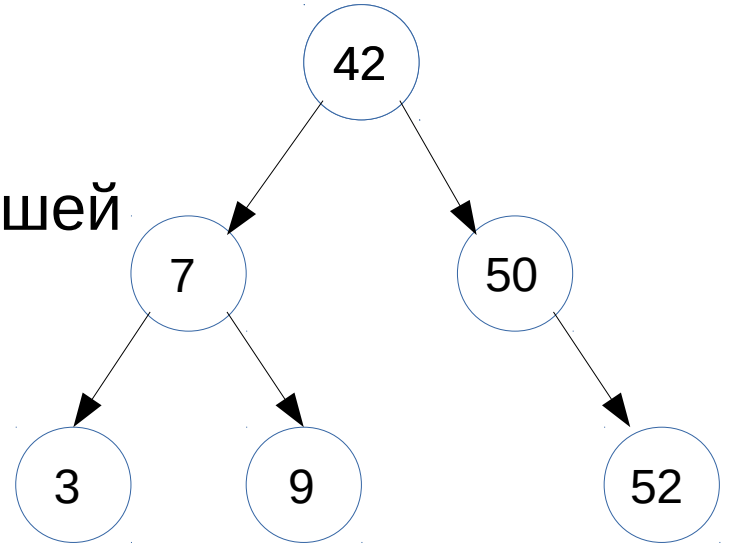
Продвинутые методы приближенного поиска ближайших соседей: k-d tree и (H)NSW

Чердаков Михаил
ВШЭ, Факультет компьютерных наук
27 сентября 2019 г.

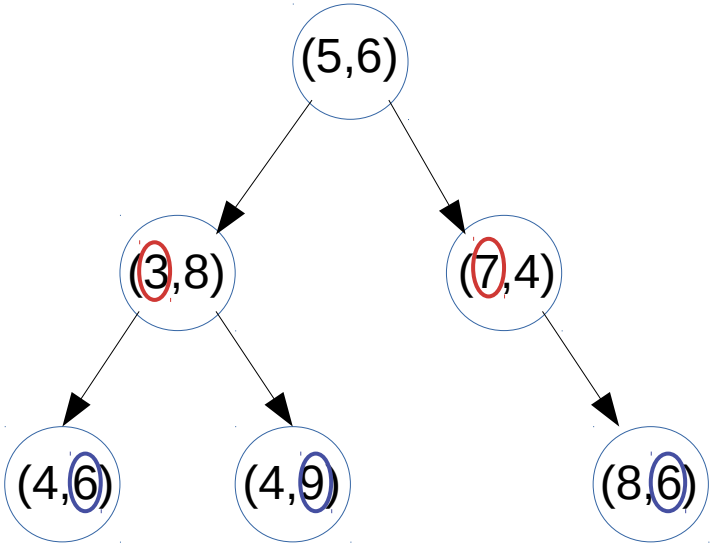
k-d tree (k-dimentional tree)

- Данные размерности k – вектор вида (x_1 x_2 ... x_k), состоящий из признаков объекта
- Цель – разбить данные на области (k-мерные параллелепипеды)
- Поиск нужной области для новой точки должен осуществляться за O(log(N))

- 1-d tree
  - Обычный двоичный поиск
  - Сложность поиска одной ближайшей вершины - O(logN)

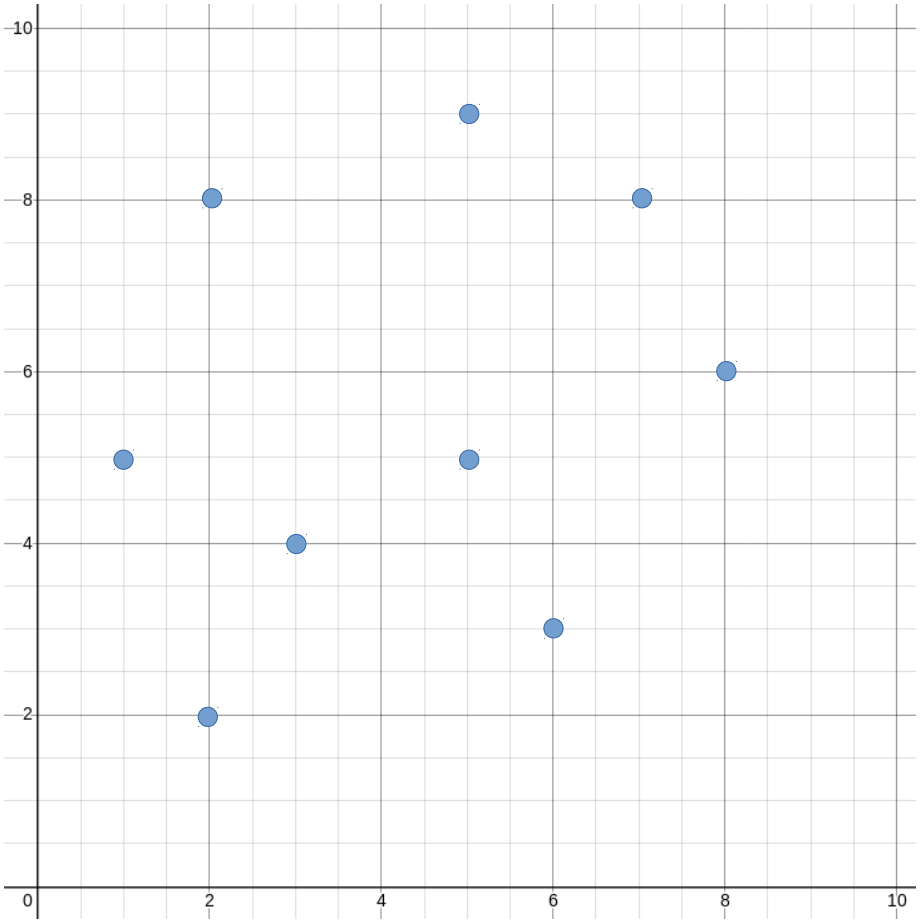


- Двоичное дерево поиска
- На каждом шаге сравниваем по одной координате
- Координаты выбираем по порядку

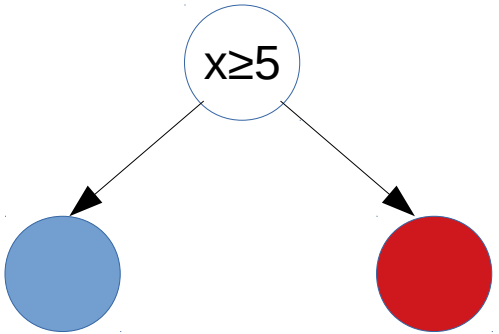


Построение k-d tree

- Для наглядности, k=2
- На каждом шаге будем находить медиану и делить данные на две части
- Начнем с координаты x
- Будем делить до тех пор, пока в каждом прямоугольнике останется не более двух точек

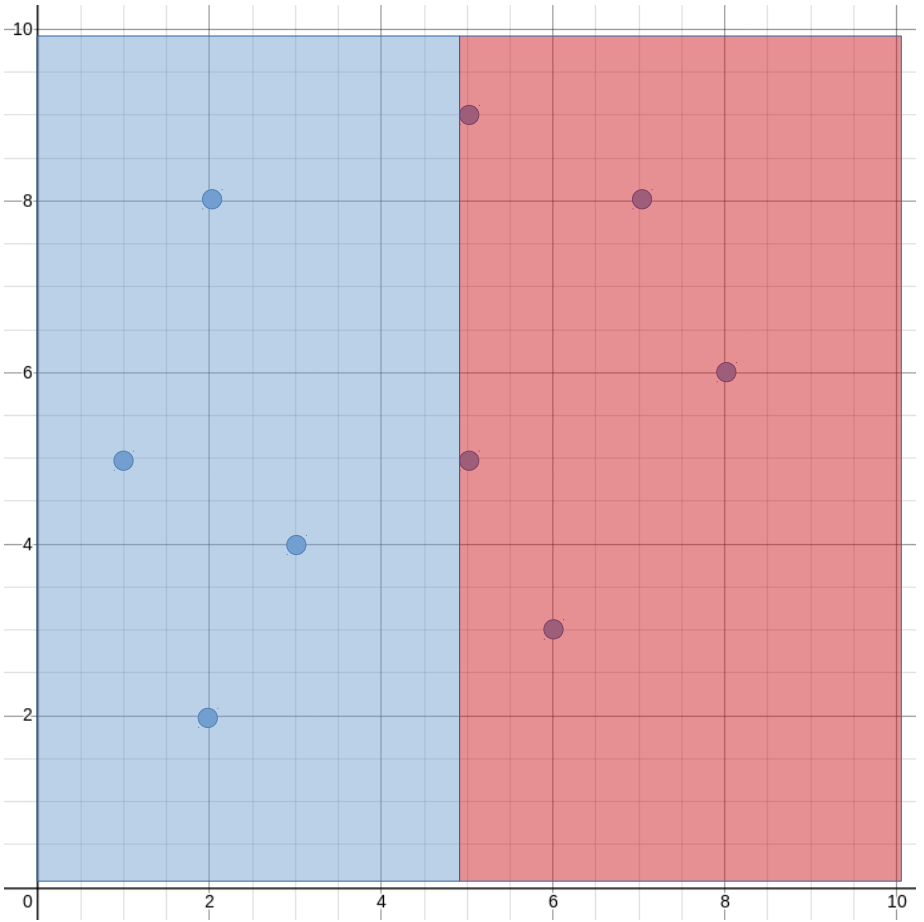


Построение k-d tree

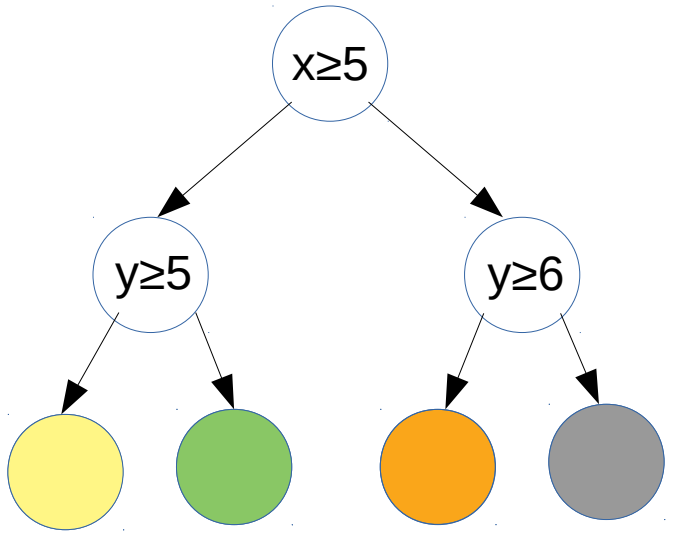


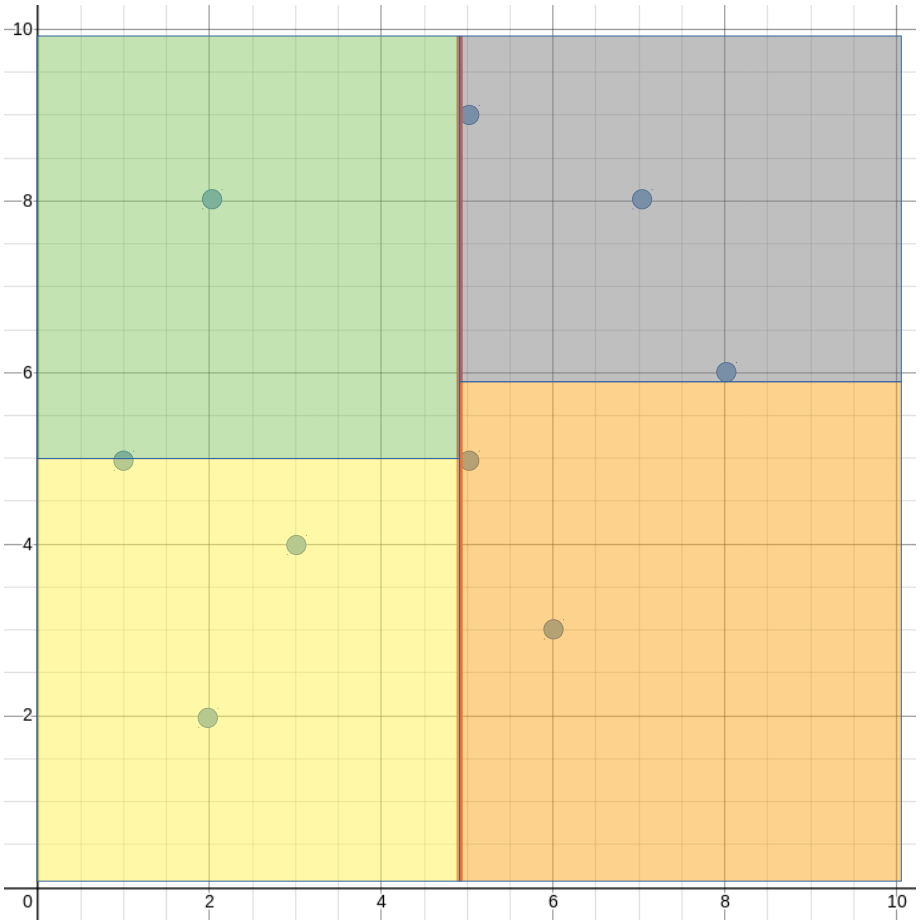
(2, 2), (3, 4)
(1, 5), (2, 8)

(5, 5), (5, 9)
(6, 3), (7, 8),
(8, 6)

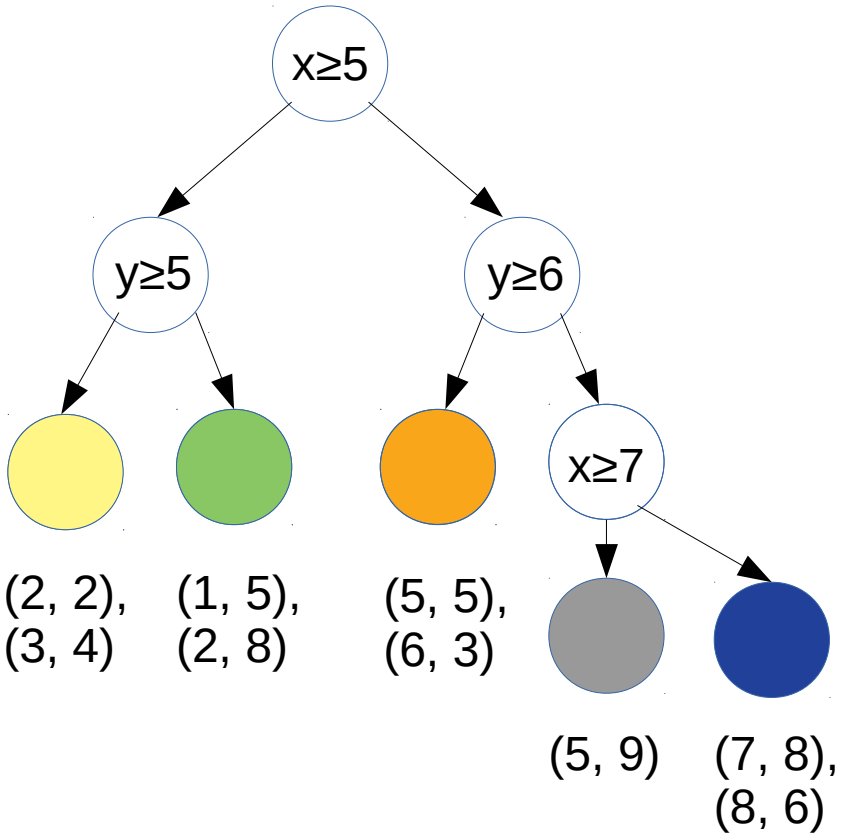


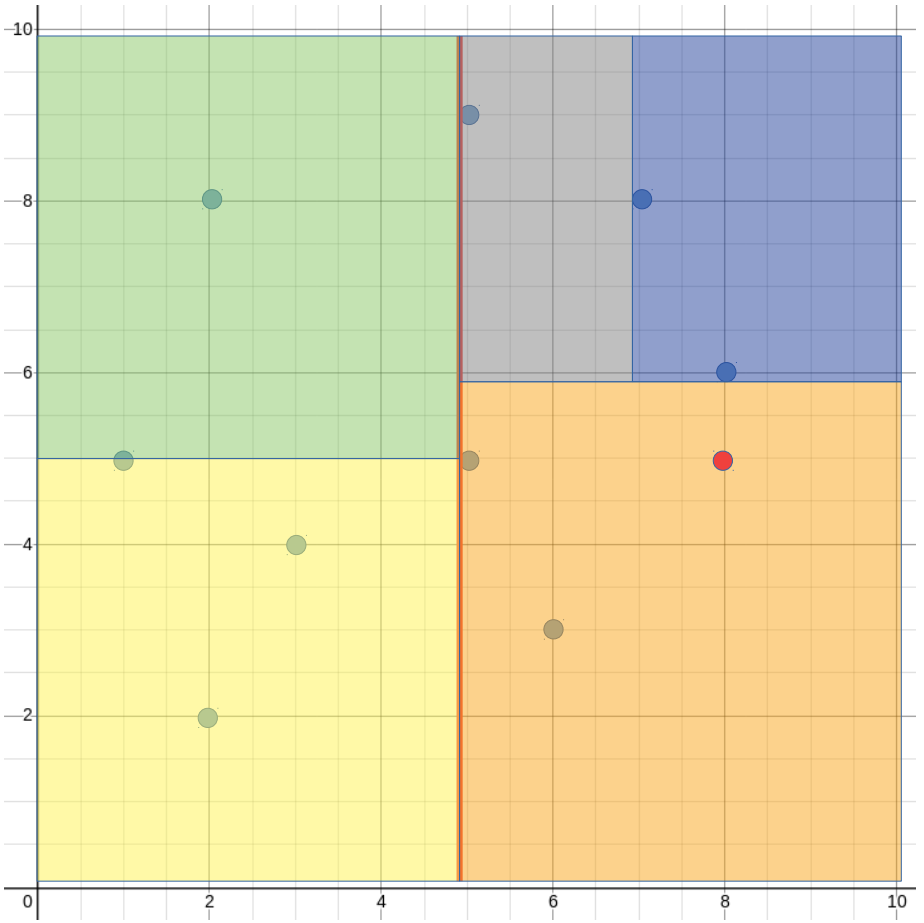
Построение k-d tree



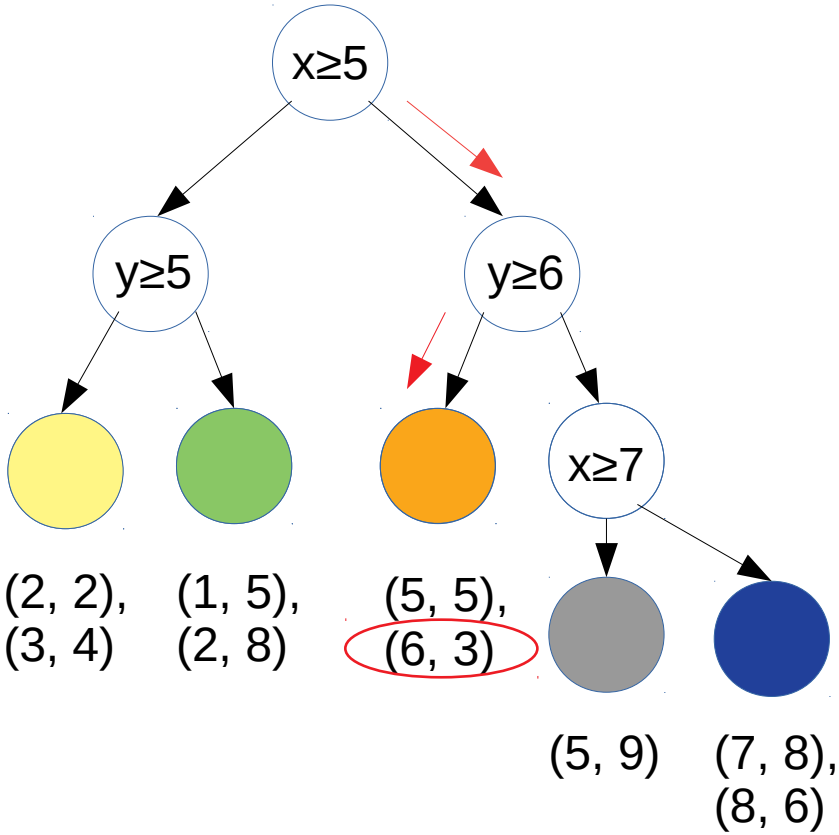


Построение k-d tree





Как найти ближайшего соседа?

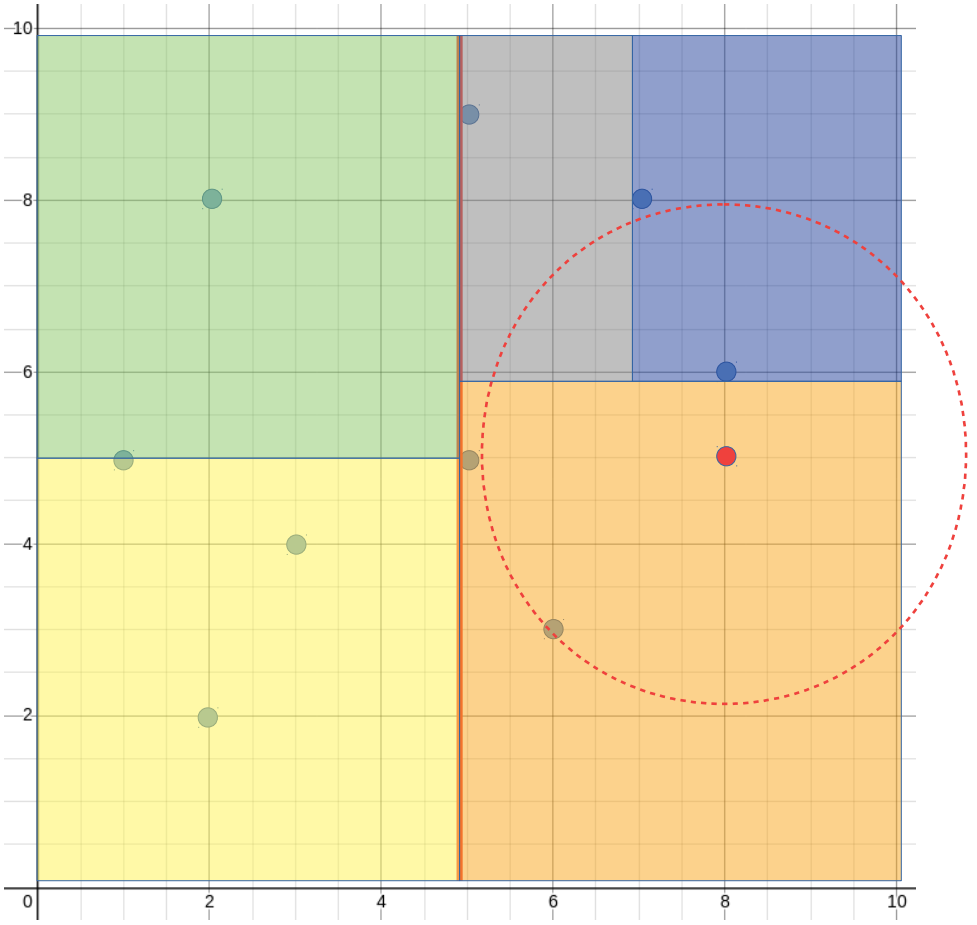


- Рассмотрим, например, точку (8, 5)
- Доходим до листа, который удовлетворяет всем условиям в узлах дерева
- Ищем расстояния до всех точек в этой области и выбираем наименьшее.

- Заметим, что найденная точка – не самая ближайшая

- Как найти точно, не сильно ухудшая время работы?

  - Проведем сферу с радиусом, равным расстоянию до найденной точки
  - Начнем движение с корня дерева.
  - На каждом шаге проверяем, пересекает ли сфера левую и правую области
  - Если пересекает, идем в этот узел.
  - Когда дойдем до листа, находим ближайшую точку, проводим сферу и повторяем.



Как найти ближайшие k точек?

- Если k не изменяется во время работы модели – можно оставлять в листах не менее k точек.
- Иначе, в обратном порядке обходим все вершины

Как еще можно делить пространство?

- Проводить “разрез” не через медиану, а через середину стороны, по которой делим
  - Быстрее, но менее точно
- SAH
  - Медленне, но точнее

Эвристика SAH

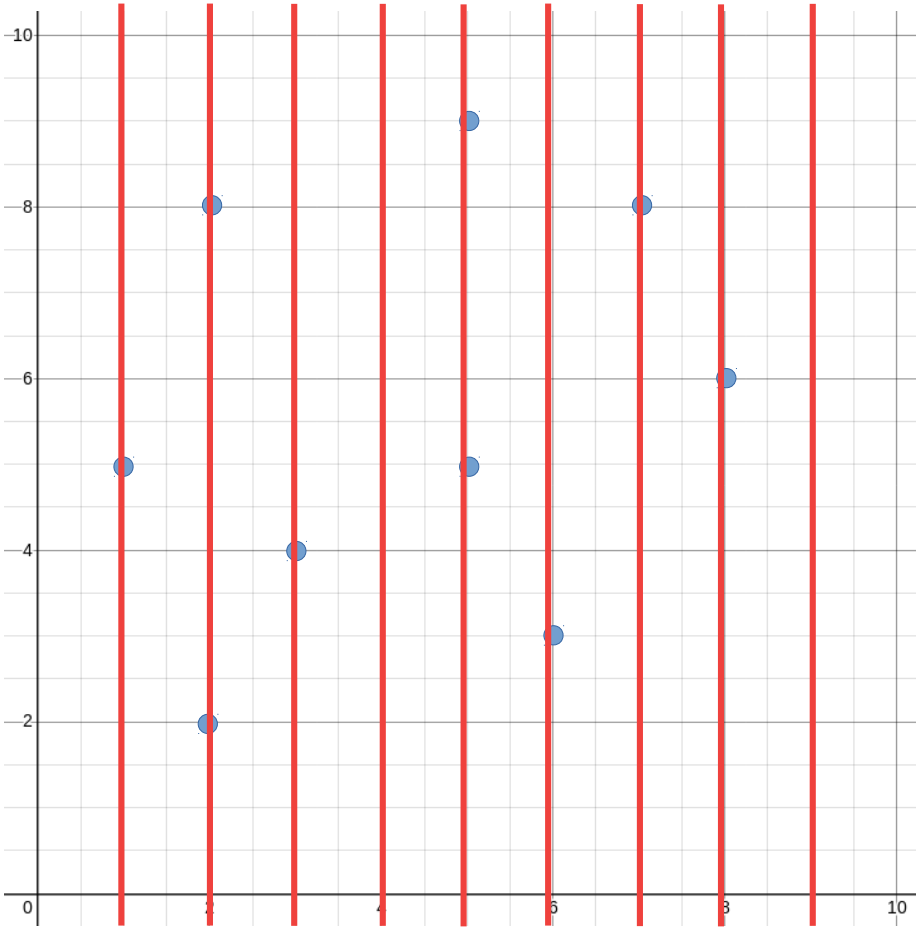
- Делим пространство на n равных частей
- Для каждого сечения считаем функцию SAH и выбираем argmin(f(x)):

$f(x) = C + SA_L(x) * N_L(x) + SA_R(x) * N_R(x)$

x – точка деления

$SA_{L,\,R}$ – площадь поверхности слева и справа от плосткости сечения

$N_{L,\,R}$ – количество точек слева и справа от плосткости сечения



Сложность и затраты по памяти

- Затраты по памяти – O(N)
- Сложность – O(logN) или O(N) в зависимости от реализации

Когда применять?

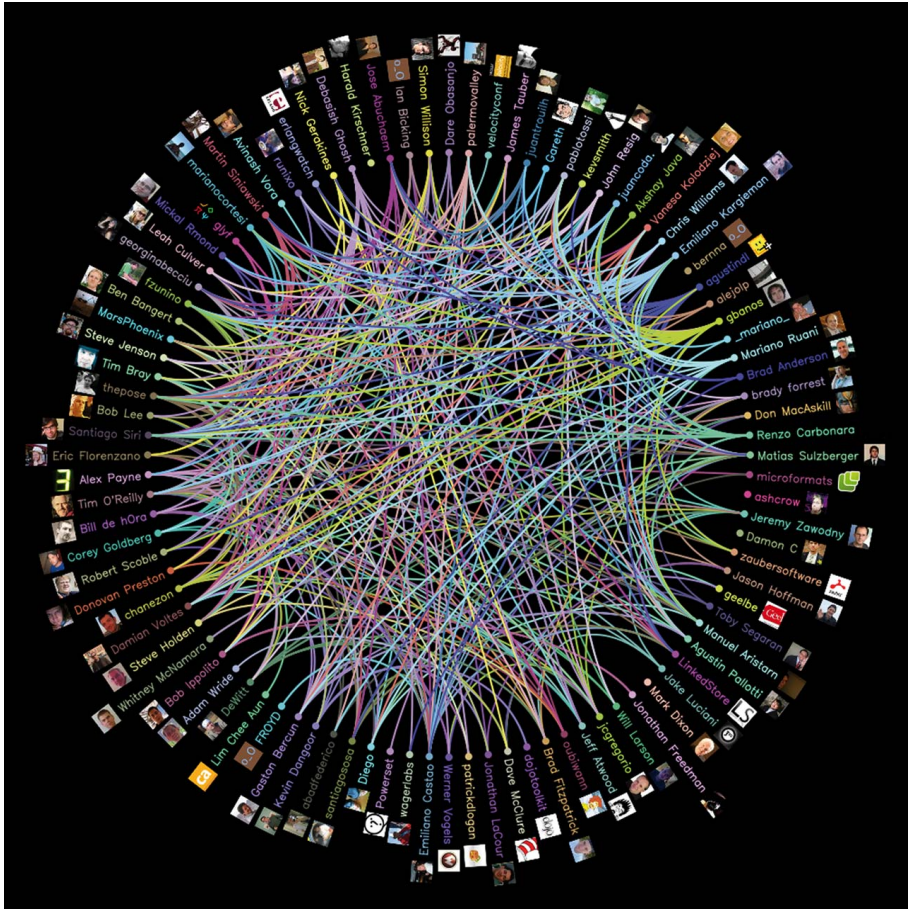
- Если количество измерений не слишком велико, иначе точность будет слишком низкой
- Если точность важна не в такой степени, как скорость работы единичного запроса

Граф “маленький мир” (мир тесен)

- Если взять две вешины такого графа, то с большой вероятностью они не являются смежными
- Из любой вершины другие достижимы за небольшое количество переходов (порядка log N)
- Другое определение: граф, который обеспечивает логарифмическое математическое ожидание между любыми двумя вершинами при достаточно маленькой средней степени вершин

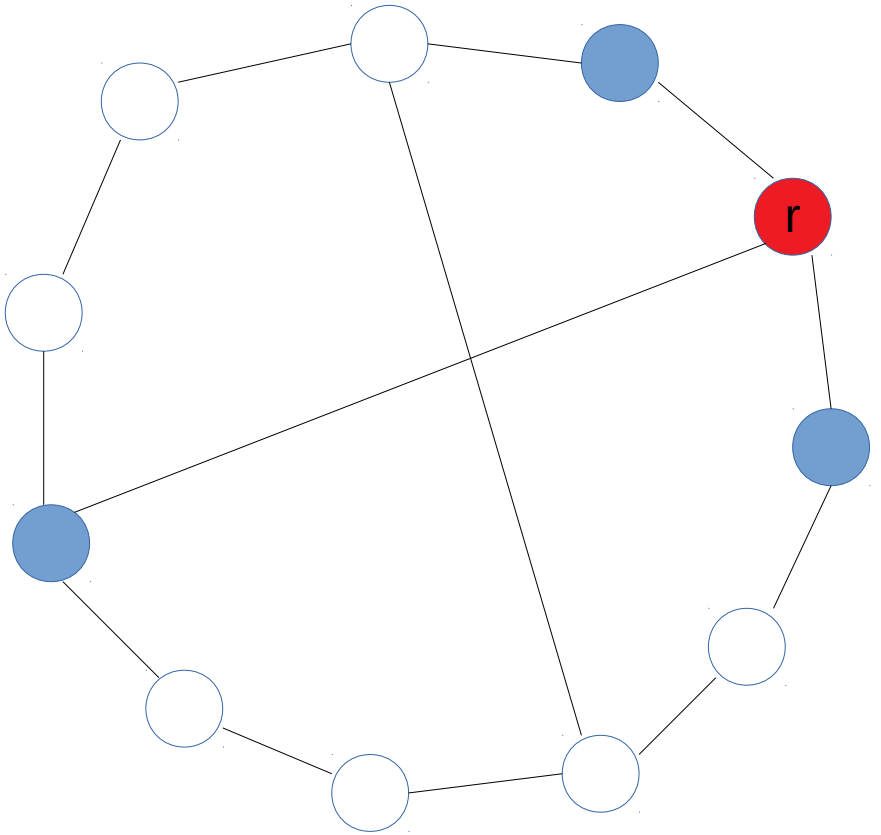
Примеры графов типа “маленький мир”

- Социальные сети
- p2p сети
- Нейронные сети мозга



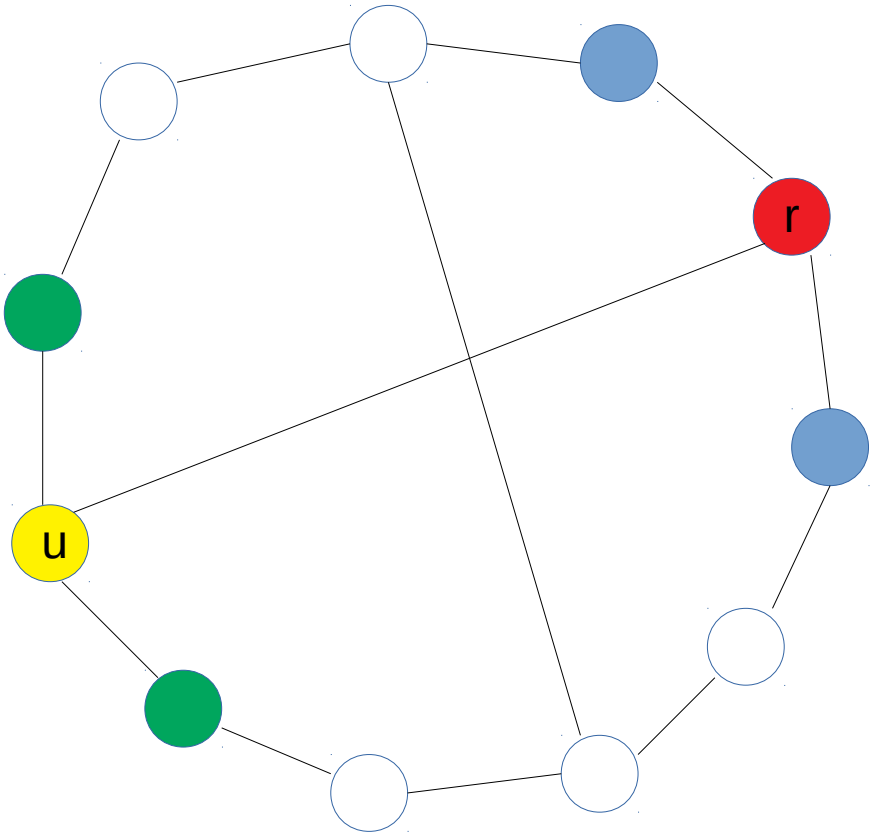
Поиск k ближайших соседей (NSW)

- Пусть q – искомая точка
- Выбираем случайную точку r
- Добавляем всех соседей r в TreeSet (candidates)
- Добавляем всех соседей в TreeSet (result)
- Если соседей больше, чем k - обновляем k лучших.



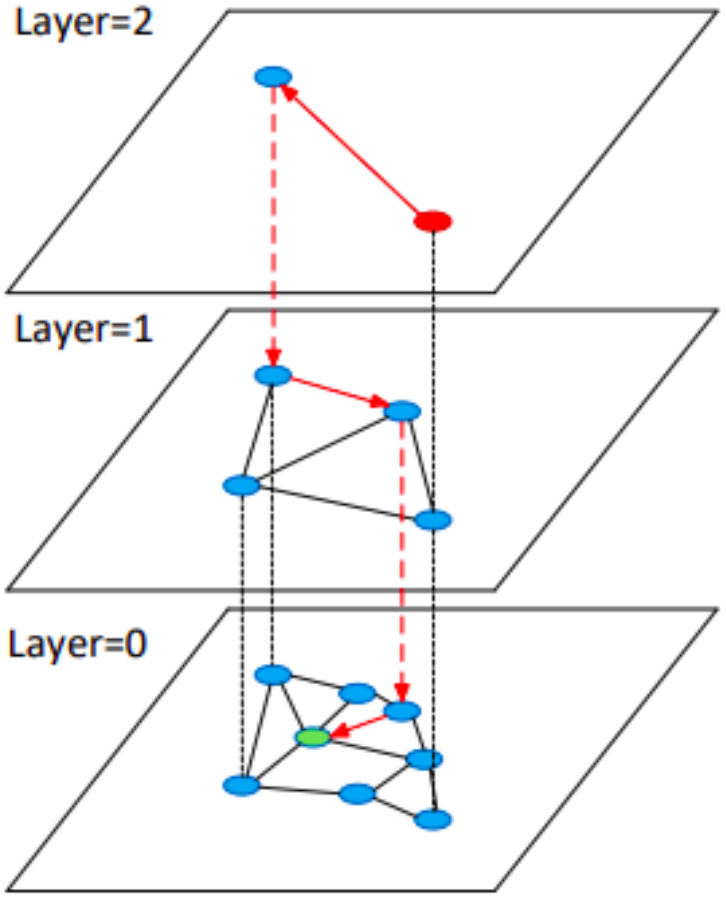
Поиск k ближайших соседей (NSW)

- Из всех вершин, находящихся в candidates, выбираем наиболее близкую к q (назовем ее u), удаляем ее из candidates.
- Если эта вершина дальше, чем k-ая вершина в result, останавливаемся
- Добавляем всех соседей u в candidates и обновляем result
- Повторяем m раз все предыдущие шаги (с новыми случайными вершинами в начале)



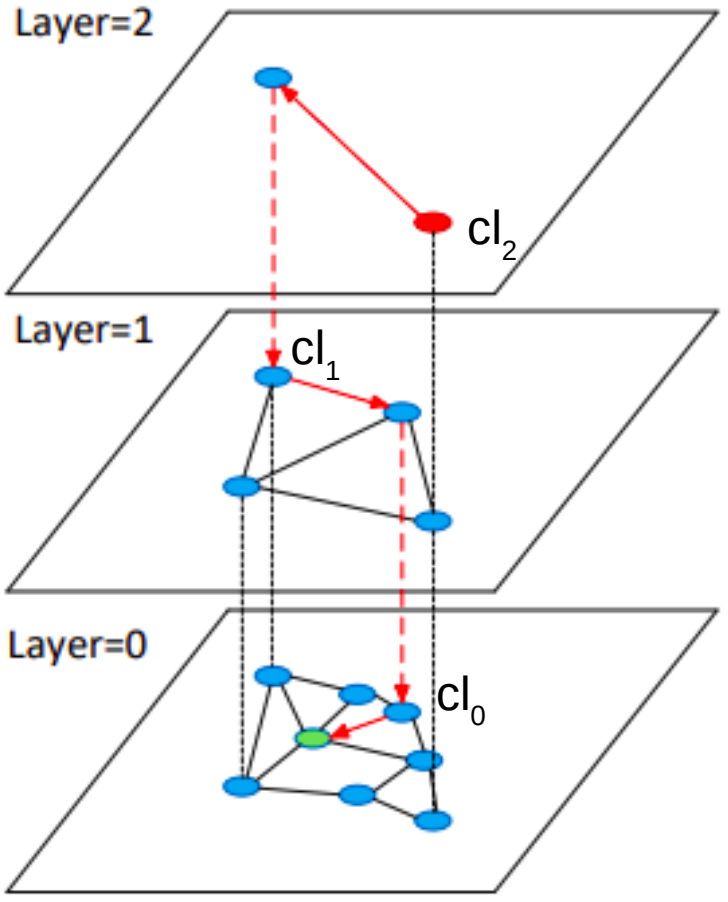
HNSW (Hierarchical Navigable Small World)

- Идейно схож с NSW, но теперь мы имеем дело с иерархией графов
- На нулевом слое предоставлены все объекты, а по мере увеличения номера слоя – все меньшая и меньшая их подвыборка
- При этом все объекты, которые есть на слое n + 1, есть и на слое n
- Степень вершины на каждом уровне ограничена константой



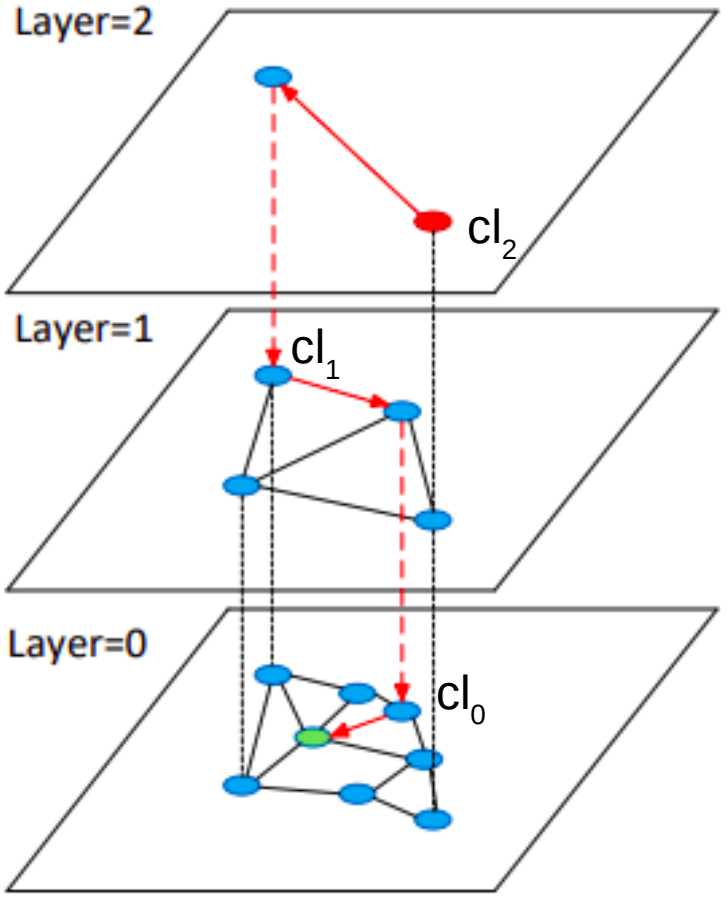
Поиск k ближайших точек в HNSW

- Идем с верхнего уровня к нижнему
- На верхнем уровне выбираем случайную вершину cl_{maxl}
- Ищем из соседей cl_{maxl} ближайшую к q.
- Объявляем эту вершину как cl_{maxl-1}
- Повторяем, пока не дойдем до cl_0



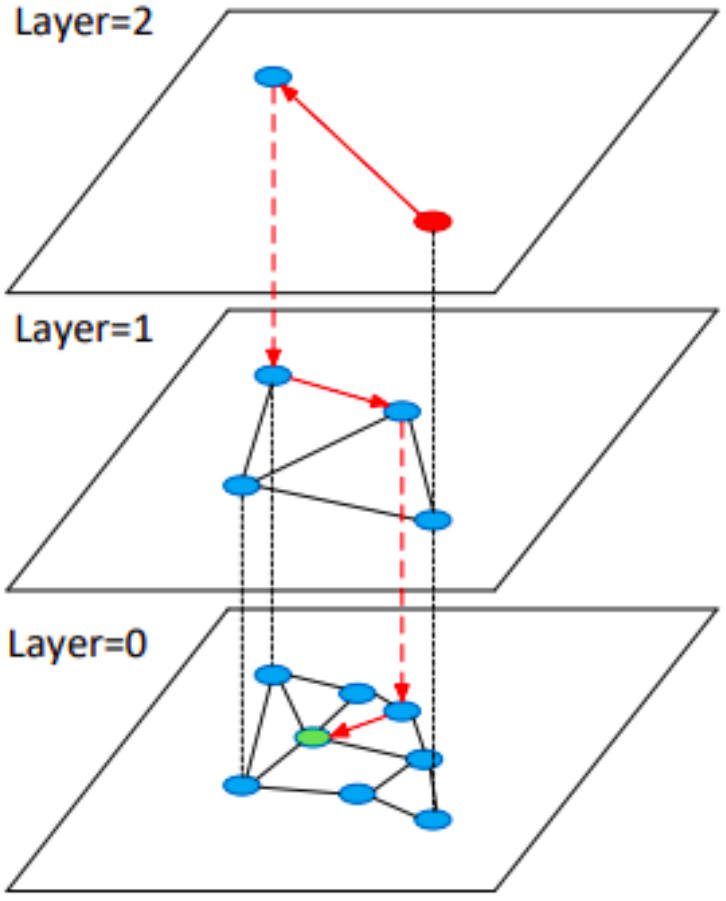
Поиск k ближайших точек в HNSW

- На самом нижнем слое начинаем поиск из cl_0
- Ищем k ближайших соседей как в алгоритме NSW
- В сравнении с NSW, вероятность попасть в локальный минимум меньше



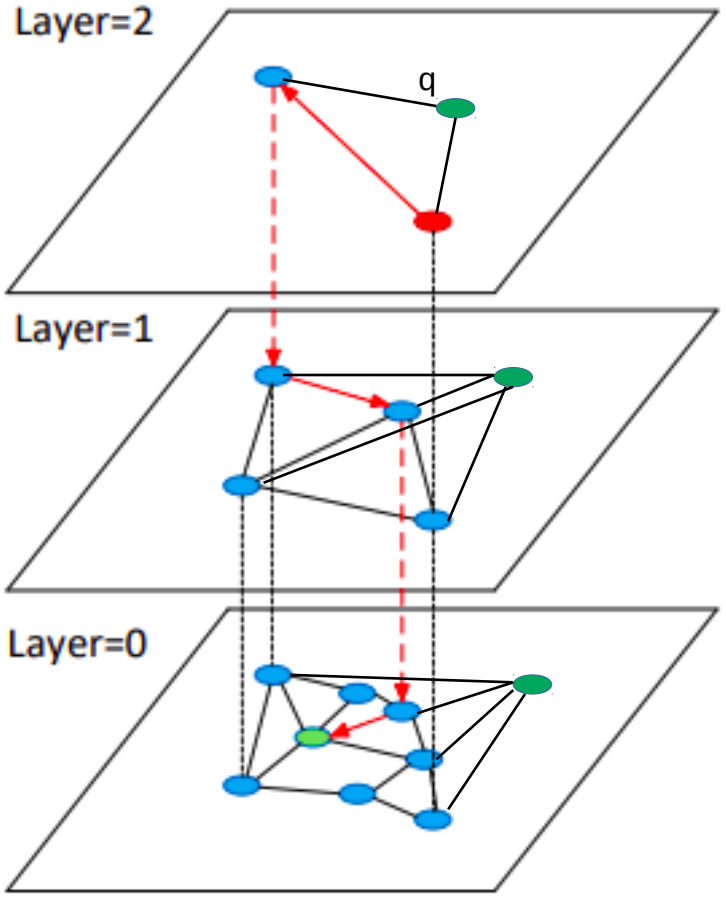
Добавление новой вершины в HNSW

- Пусть нам нужно добавить вершину q
- Случайно выбираем число mL – максимальный уровень, на котором будет представлена q
- Распределение экспоненциально убывающее, например:
  mL = -ln(rand(eps, 1))



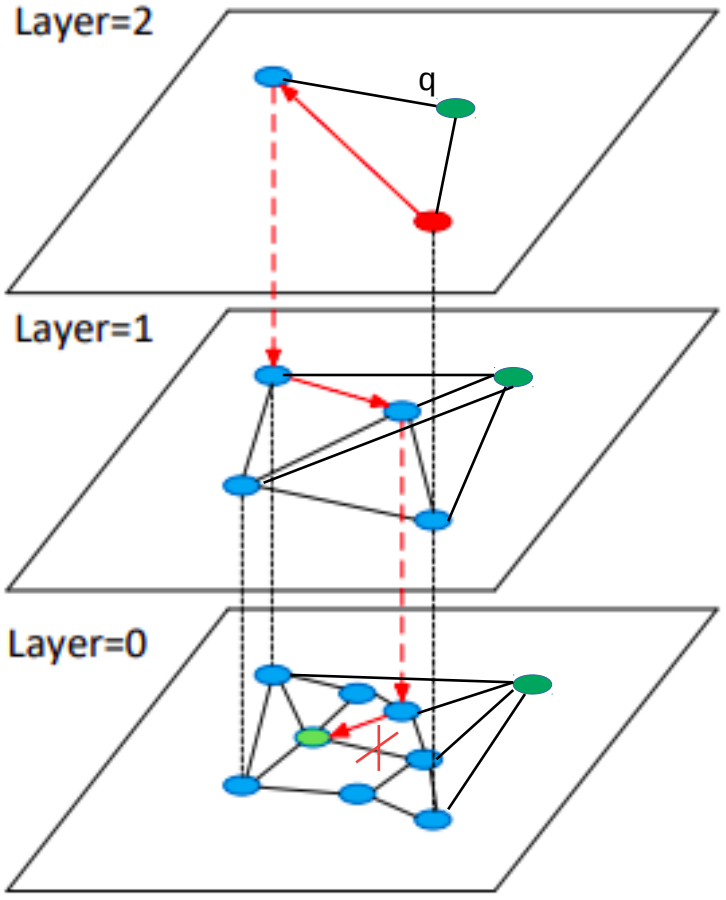
Добавление новой вершины в HNSW

- Если mL > |HNSW| - 1, то на уровнях с |HNSW| до mL добавляем вершину q без связей
- Пусть m – константа, ограничивающая степень любой вершины
- На каждом слое ищем m ближайших с q соседей



Добавление новой вершины в HNSW

- Удаляем лишние связи так, чтобы у каждой вершины степнь была не более m
- Если d – степень какой-либо вершины, соседней с q, то нужно найти d – m самых дальних от нее вершин и удалить эти связи



Преимущества HNSW

- State-of-the-art результаты
- Простой для понимания алгоритм (по сравнению с другими алгоритмами, дающими такие же хорошие результаты)
- Эффективная и удобная в использовании реализация в библиотеке nmslib (C++) с интерфейсом для Python

Контрольные вопросы по теме

- Опишите метод, которым нужно делить пространство в алгоритме k-d tree так, чтобы конечное дерево было уравновешенным
- Какую длину пути (асимптотически) между двумя любыми вершинами гарантирует граф small world (маленький мир)?
- Как происходит поиск ближайших соседей в модели NSW?

Источники

- С формальным описанием:
  - Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs, Yu. A. Malkov, D. A. Yashunin (ссылка)
  - An introductory tutorial on kd-trees, Andrew W. Moore (ссылка)
- Почитать, чтобы было понятнее:
  - Статья на хабре про HNSW и другие крутые методы (ссылка)
  - Статья на хабре про kd-tree - рассматривается немного более общая задача, но тоже интересно (ссылка)