# 2.4.2 k-nächste Nachbarn Anfragen

## *2.4.2 k-nächste Nachbarn* (*k*-NN) *Anfragen*

- Allgemeines
  - Eigenschaften
    - Benutzer gibt Anfrageobjekt *q* und Anzahl *k* vor
    - Ergebnis enthält die *k* nächsten Nachbarn von *q*
    - Mehrdeutigkeiten müssen wiederum sinnvoll behandelt werden
  - Formal
    - Deterministisch

      kleinste Menge $NN(q,k) \subseteq DB$ mit mindestens *k* Objekten, sodass

      $$\forall o \in NN(q,k), \forall o' \in DB - NN(q,k) : dist(q,o) < dist(q,o')$$

    - Nicht-deterministisch

      Menge $NN(q,k) \subseteq DB$ mit exakt *k* Objekten, sodass

      $$\forall o \in NN(q,k), \forall o' \in DB - NN(q,k) : dist(q,o) \leq dist(q,o')$$

    - Klar: $NN(q,1) \equiv NN(q)$

# 2.4.2 k-nächste Nachbarn Anfragen

- Basisalgorithmus (sequential scan): nichtdeterministisch

```
NN-SeqScan(DB, q, k)
    result = LIST OF (dist:REAL, p:OBJECT) ORDERED BY dist DESCENDING;
  result = [];
  FOR i=1 TO k DO
    result.insert(dist(q, getObject(i), getObject(i));
  FOR i=k+1 TO n DO
    IF dist(q, DB.getObject(i)) ≤ result.getFirst().dist THEN
      result.deleteFirst();
      result.insert(dist(q, getObject(i), getObject(i));
  RETURN result;
```

  - Bemerkung:
    - Liste *result* wird als Heap anstelle einer sortierten Liste implementiert
      - Grund: Die Liste *result* braucht nicht vollständig sortiert sein – es reicht sicherzustellen, dass das erste Element in *result* immer den größten Distanzwert hat. Ermöglicht Einfügen in O(log(k)) .

# 2.4.2 k-nächste Nachbarn Anfragen

- Algorithmen mit Index
  - Grundsätzlich lassen sich alle Algorithmen zur NN-Suche auf *k*-NN-Suche erweitern, egal ob Index-basiert oder mehrstufig.
    - **Pruningdistanz** ist entsprechend immer die Distanz zum aktuell gefundenen *k*-NN (erstes Element in result-Liste)
    - Approximationsdistanzen können nur eingeschränkt benutzt werden
      - MINDIST: keine Einschränkung, da MINDIST nur zur Distanzabschätzung von Fehltreffern (true drops) verwendet wird.
      - MAXDIST: Einschränkung hängt vom Gebrauch ab. Bei Verwendung von MAXDIST zur Bestimmung von Treffern muß die Anzahl der Objekte deren Distanz durch MAXDIST approximiert wird mit dem k-Parameter abgeglichen werden.
      - MINMAXDIST: Beim Algorithmus nach [RKV] kann MINMAXDIST zu einer Seite nur wie Distanz zu einem Punkt gewertet werden und nicht als Gesamt-Pruningdistanz => MINMAXDIST lohnt sich i.A. nicht für *k*-NN Anfragen

# 2.4.2 k-nächste Nachbarn Anfragen

- Algorithmen mit Multi-Step Architektur
  - Alle drei Alternativen zur 1NN-Anfrage sind leicht erweiterbar
    - **Auswertung mit Bereichsanfrage**
      - *k*-NN Anfrage statt NN Anfrage im Filter und Refinement anpassen

        (siehe Übung)
    - **Unmittelbare Verfeinerung**
      - erweitere *k*-NN-Algorithmus statt NN-Algorithmus um entsprechende Aufrufe
    - **Auswertung nach Priorität**
      - benutze *k*-NN-Distanz als Abbruchkriterium (siehe Übung)

        Bemerkung: Hier gilt die Verfeinerungsoptimalität nur unter der Annahme der Beschränkung auf einen LB-Filter (Grund siehe später)

# 2.4.2 k-nächste Nachbarn Anfragen

- Verfeinerungsoptimale *k*-NN Anfrage
  - Ziel:
    - Kandidatenmenge im Filterschritt durch Berücksichtigung weiterer Filterkriterien oder zusätzlicher Information weiter reduzieren
  - Motivation:
    - Exakte Distanzberechnung sehr teuer im Vergleich zur Filterdistanzberechnung

      $\Rightarrow$ Oft lohnt es sich den Filter durch zusätzliche Filterinformationen zu verbessern
    - In vielen Applikationen können Ähnlichkeitsdistanzen sowohl nach unten als auch nach oben hin effizient abgeschätzt werden
    - Idee:
      - zusätzlich zur unteren Distanzabschätzung (LB) wird auch obere Distanzabschätzung (UB) im Filterschritt miteinbezogen

        $\Rightarrow$ bessere Filterung von Kandidaten mit unterer und oberer Distanzabschätzung

        $\Rightarrow$ weniger Objekte müssen verfeinert werden.
    - Ziel: Optimale Auswertung mit unterer und oberer Distanzabschätzung

# 2.4.2 k-nächste Nachbarn Anfragen

- Aber …
  Mehrstufige NN-Anfragebearbeitung mit Auswertung nach Priorität ist „ optimal bzgl. der Anzahl der Verfeinerungen“ (s. Folie 99)

  Wiederspruch?

  Können wir denn das dann durch weitere Filter verbessern?

- Grundsätzlich gilt:
  - Unterscheidung der Optimalität:

    - Optimal bzgl I/O Kosten (Seitenzugriffe im Index) = Kosten im Filterschritt

    - Refinement-Optimal (R-Optimal) bzgl CPU Kosten für die Berechnung der exakten Distanz = Kosten im Verfeinerungsschritt der Mehrstufigen Anfragebearbeitung

    - Optimalität eines mehrstufigen Anfragealgorithmus hängt von der Annahme der im Filterschritt zur Verfügung stehenden Distanzinformation ab, d.h. mehr Information im Filterschritt

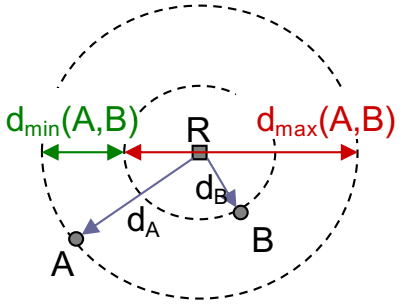
      $\Rightarrow$ höhere Selektivität des Filters

      $\Rightarrow$ weniger Seitenzugriffe, weniger Verfeinerungen

    Kein Wiederspruch, wenn bei der „NN-MultiStep mit Auswertung nach Priorität“ **nur** die lower-bounding (LB-) Filterdistanz verwendet wird.

# 2.4.2 k-nächste Nachbarn Anfragen

- Beispiele für die Ermittlung von unterer bzw. oberer Distanzabschätzung:
  - Abschätzung basierend auf Referenzpunkten (z.B. M-tree)



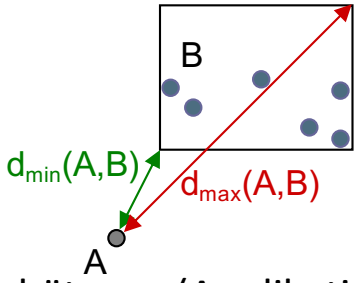

$d_{min}(A,B) = |d_A - d_B|$

$d_{max}(A,B) = d_A + d_B$

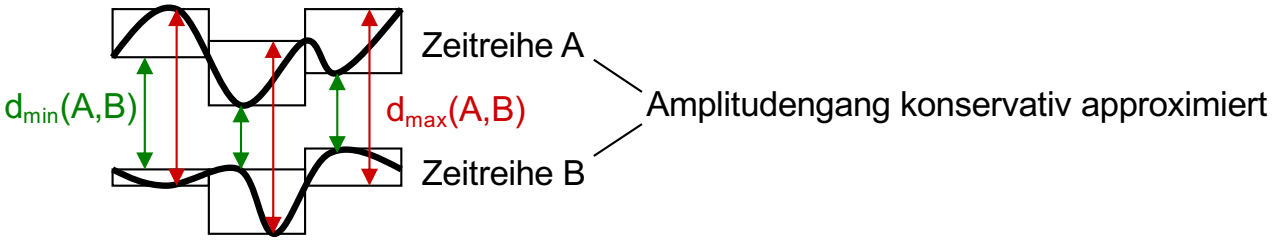
$d_{min}(A,B) \leq d(A,B) \leq d_{max}(A,B)$

LB exakt UB

  - Abschätzung basierend auf Regionen (z.B. R-tree)



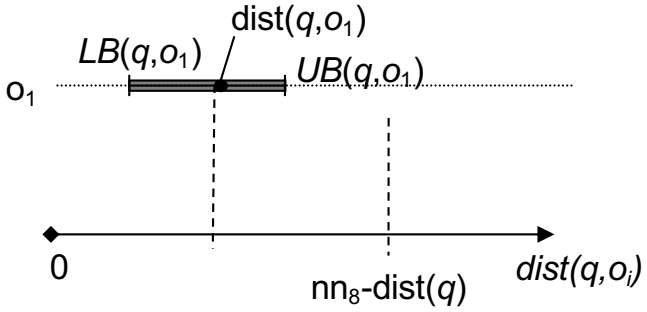

  - Individuelle Abschätzung (Applikations/Daten abhängig)

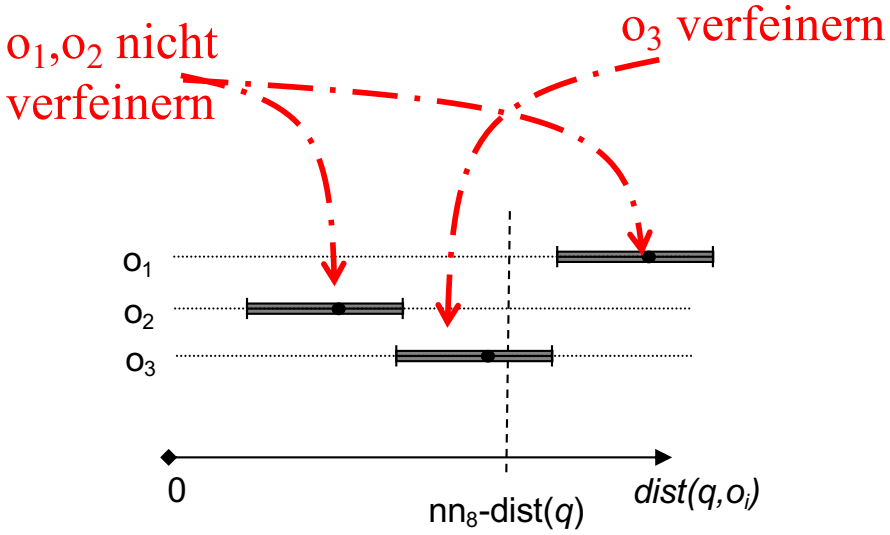


# 2.4.2 k-nächste Nachbarn Anfragen

- Im folgenden definieren wir:
  - *$nn_k$-dist(q):=* k-nächsten-Nachbar-Distanz von Anfrageobjekt *q.*
  - *LB(q,o) (OB(q,o)) := untere (obere) Distanzabschätzung zwischen q und o.*
  - Grundsätzlich gilt !!!!
    Alle Objekte *o* deren Filterdistanzen die Eigenschaft $LB(q,o) \leq nn_k\text{-dist}(q) \leq UB(q,o)$ erfüllen, **müssen** verfeinert werden.

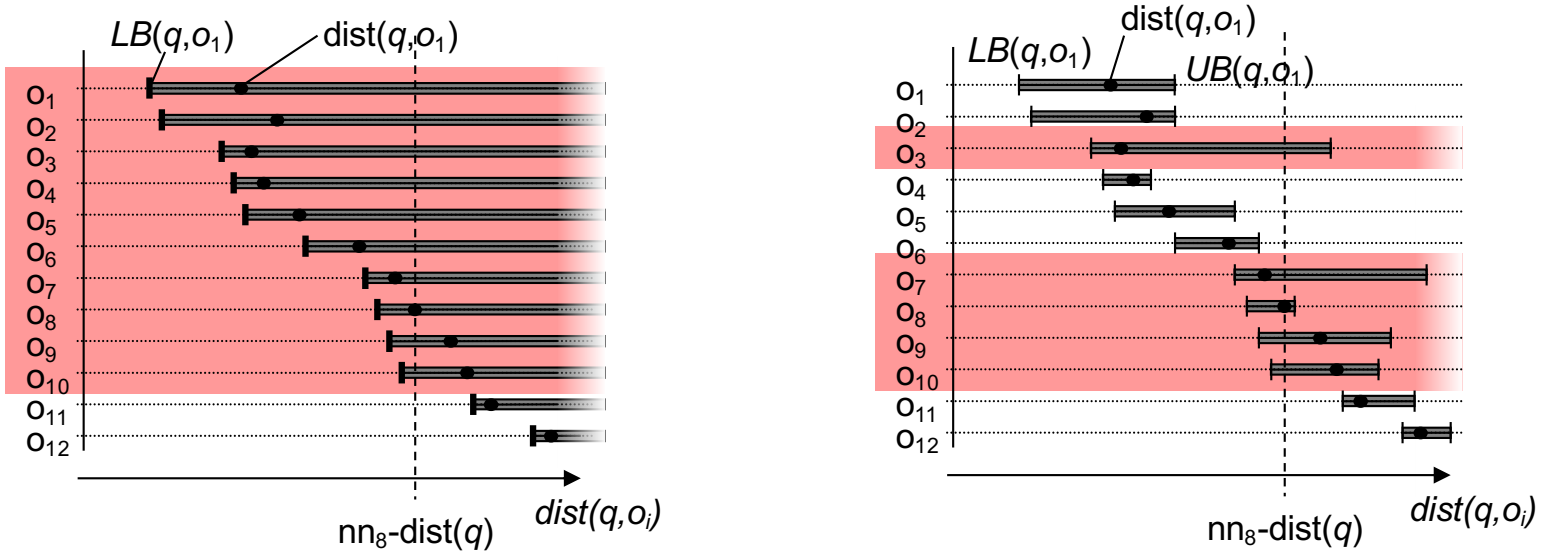
Warum gilt dies?



# 2.4.2 k-nächste Nachbarn Anfragen

- LB-basierte *k*-NN-Suche vs. (LB+UB)-basierte *k*-NN-Suche




- Beispiel: k-NN Anfrage mit k = 8
  - Bei der LB-basierten *k*-NN-Suche müssen mehr Objekte verfeinert werden als bei der (LB+UB)-basierten *k*-NN-Suche
    - LB-basierte *k*-NN-Suche muss 10 Objekte verfeinern, während die (LB+UB)-basierte *k*-NN-Suche nur 5 Objekte verfeinern muss
    - Gilt nur bei der Annahme: Die Berechnung der exakten Distanz für die Resultatmenge ist nicht erforderlich !!!