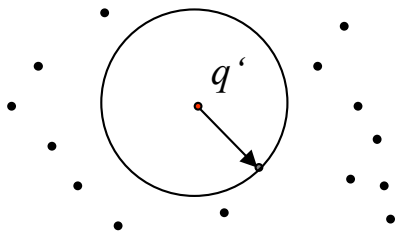
2.4 Nächste Nachbarn Anfragen

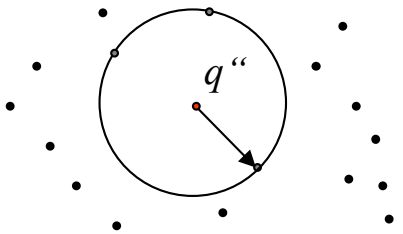
2.4 Nächste Nachbarn Anfragen

2.4.1 Nächste Nachbarn Anfrage

- Allgemeines
  - Eigenschaften
    - Benutzer gibt Anfrageobjekt *q* vor
    - Ergebnis enthält das Objekt, das die geringste Distanz zu *q* hat
    - Mehrdeutigkeiten müssen sinnvoll behandelt werden (mehrere nächste Nachbarn, oder nichtdeterministisch ein Objekt)
  - Formal $NN(q) = \{o \in DB \mid \forall x \in DB : dist(q,o) \leq dist(q,x)\}$



Eindeutiges Ergebnis



Mehrdeutiges Ergebnis

2.4.1 Nächste Nachbarn Anfragen

- Basisalgorithmus (sequential scan): nichtdeterministisch

```
NN-SeqScan(DB, q)
    result = ∅;
    stopdist = +∞;
    FOR i=1 TO n DO
        IF dist(q, DB.getObject(i)) ≤ stopdist THEN
            result := getObject(i);
            stopdist = dist(q, DB.getObject(i));
    RETURN result;
```

- Algorithmus mit Index: Einfache Tiefensuche
  - Unterschied zur Range-Query
    - Nächste Nachbar kann beliebig weit vom Anfragepunkt weg liegen
    - Gestalt der Query zunächst unbekannt
    - Es kann zunächst nicht anhand der Seitenregion entschieden werden, ob eine Seite gebraucht wird
    - Ob eine Seite gebraucht wird, hängt auch von dem Inhalt der anderen Seiten ab

2.4.1 Nächste Nachbarn Anfragen

- Kennt man NN-Distanz, würde Range Query ausreichen
- Kennt man ein beliebiges Objekt, kann man dessen Abstand als obere Schranke für die NN-Distanz nutzen
- Kennt man mehrere Objekte, kann man den geringsten Abstand als obere Schranke für die NN-Distanz nutzen

- Umformulierung des RQ-Algorithmus: Einfache Tiefensuche
  - Verwende als ε die kleinste Distanz zu den bisher gefunden Nachbarn

```
Globale Variable: stopdist = +∞;
NN-Index-Simple-TS(pa, q)          // pa = Diskadress z.B. der Wurzel des Indexes
        result = ∅;
    p := pa.loadPage();
    IF p.isDataPage() THEN
        FOR i=0 TO p.size() DO
            IF dist(q, p.getObject(i)) ≤ stopdist THEN
                result := getObject(i);
                stopdist = dist(q, p.getObject(i));
        ELSE                                // p ist Directoryseite
        FOR i=0 TO p.size() DO
            IF MINDIST(q, p.getRegion(i)) ≤ stopdist THEN
                result := NN-Index-Simple-TS(p.childPage(i), q)
        RETURN result;
```

2.4.1 Nächste Nachbarn Anfragen

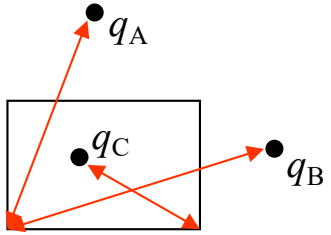
- Nachteil des einfachen Tiefensuch-Algorithmus
  - Initialisierung: stopdist = +∞
  - Dadurch: Start mit beliebigem Pfad
  - Folge: die ersten gefundenen Objekte sind meist sehr weit vom Anfrageobjekt entfernt => stopdist ist wenig selektiv
  - Verbesserung: beginne Pfad, der möglichst nah zum Anfrageobjekt liegt

2.4.1 Nächste Nachbarn Anfragen

- Algorithmus mit Index: Tiefensuche nach [RKV 95]

  [Roussopoulos, Kelley, Vincent. Proc. ACM Int. Conf. Management of Data (SIGMOD), 1995]

  - Vermeidet langsame Einschränkung des Suchraums durch
    - Verwendung der Seitenregionen zur Abschätzung der NN-Distanz
    - Priorisierung der Tiefensuche nach Distanz der Seitenregion zur Query
  - Neben MINDIST weitere Abschätzungen der NN-Distanz durch:
    - MAXDIST
      - Maximale Distanz zwischen Query und allen Punkten der Seitenregion
      - NN-Distanz kann nicht schlechter als MAXDIST werden

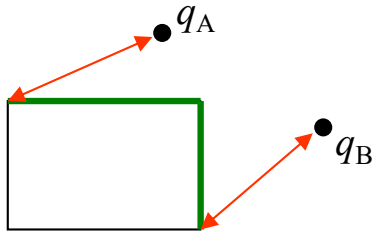


$$\mathrm{MAXDIST}(region, q) = \sqrt{\sum_{0<i\leq d} \max\{(q_i - region.UB_i)^2, (q_i - region.LB_i)^2\}}$$

2.4.1 Nächste Nachbarn Anfragen

- MINMAXDIST
  - MBRs als Seitenregionen: maximale NN-Distanz noch besser abzuschätzen
  - Auf jeder Kante des MBR muss ein Punkt liegen (sonst ist MBR nicht minimal)
  - Intuition: „nächstliegende Kante, weitester Punkt“

$$MINMAXDIST(region, q) = \sqrt{\min_{1 \le k \le d} |q_k - rm_k|^2 + \sum_{\substack{1 \le i \le d \\ i \neq k}} |q_i - rM_i|^2}$$



$$rm_i = \begin{cases} region.LB_i & \text{if } q_i \le \dfrac{region.LB_i + region.UB_i}{2} \\ region.UB_i & \text{else} \end{cases}$$

$$rM_i = \begin{cases} region.LB_i & \text{if } q_i \ge \dfrac{region.LB_i + region.UB_i}{2} \\ region.UB_i & \text{else} \end{cases}$$

2.4.1 Nächste Nachbarn Anfragen

- MINMAXDIST (cont.)
  - Für andere Geometrien (nicht MBRs) sind MINDIST und MAXDIST analog definierbar; MINMAXDIST allerdings nicht
  - Abschätzung von stopdist durch Minimum aus stopdist und MINMAXDIST (bzw. MAXDIST) aller bisher bekannten Seitenregionen (pruningdist)
  - Vor dem rekursiven Abstieg: sortieren der Kindseiten nach MINDIST (experimentell als bestes Prioritätsmaß ermittelt)

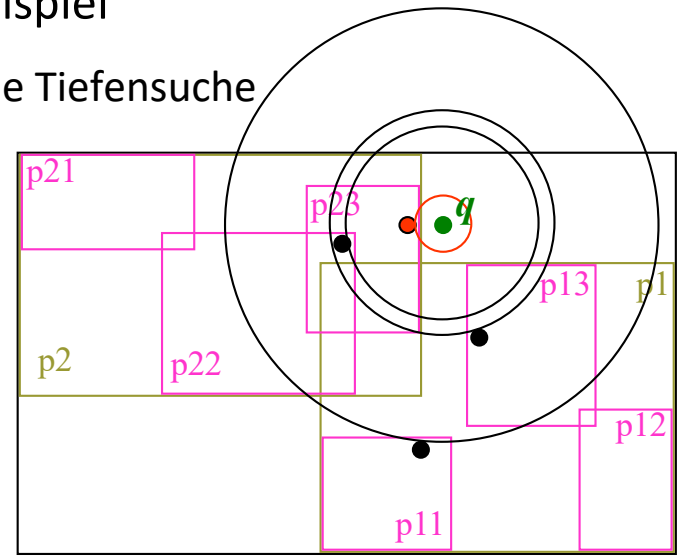
2.4.1 Nächste Nachbarn Anfragen

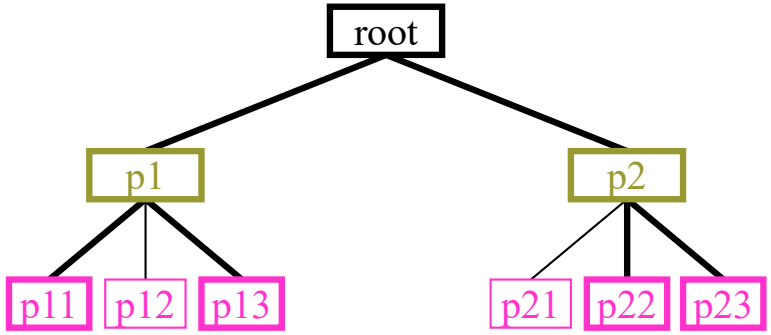
- Algorithmus: Globale Variablen: stopdist = +∞; pruningdist = +∞;

```
NN-Index-RKV-TS(pa, q)          // pa = Diskadress z.B. der Wurzel des Indexes
      result = ∅;
   p := pa.loadPage();
   IF p.isDataPage() THEN
      FOR i=0 TO p.size() DO
         IF dist(q, p.getObject(i)) ≤ stopdist THEN
            result := getObject(i);
            stopdist = dist(q, p.getObject(i));
         IF stopdist < pruningdist THEN
            pruningdist = stopdist;
      ELSE                          // p ist Directoryseite
      FOR i=0 TO p.size() DO
         IF MINMAXDIST(q, p.getRegion(i)) < pruningdist THEN
            pruningdist = MINMAXDIST(q, p.getRegion(i));
      quicksort(p.getObjectArray(), MINDIST);
      FOR i=0 TO p.size() DO
         IF MINDIST(q, p.getRegion(i)) ≤ pruningdist THEN
            result := NN-Index-RKV-TS(p.childPage(i), q);
      RETURN result;
```

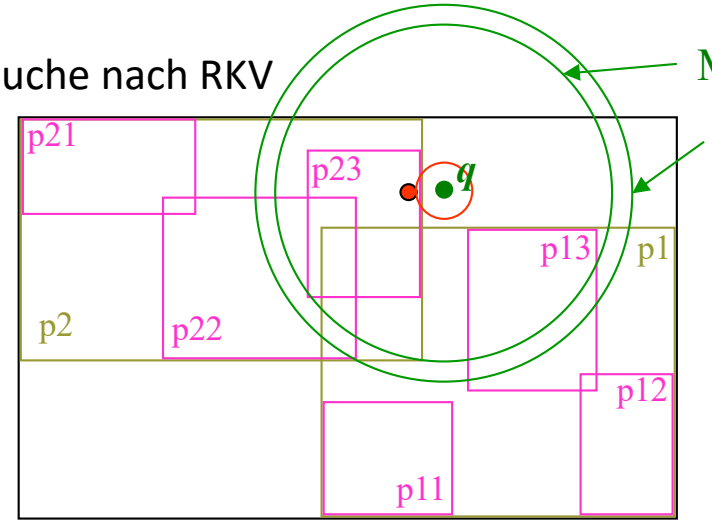
2.4.1 Nächste Nachbarn Anfragen

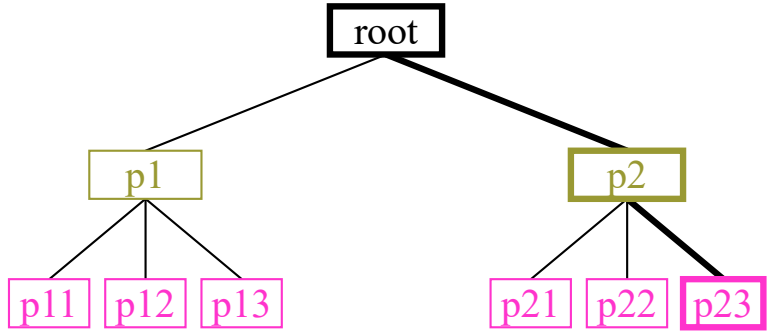
- Ablaufbeispiel
  - Einfache Tiefensuche





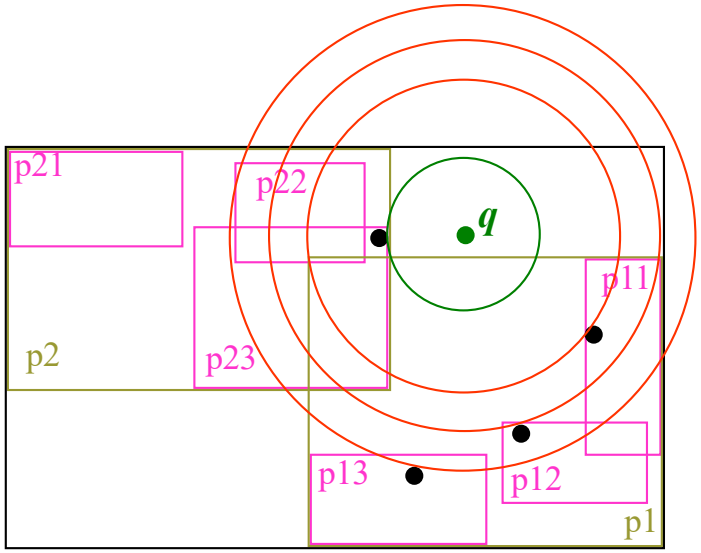
  - Tiefensuche nach RKV

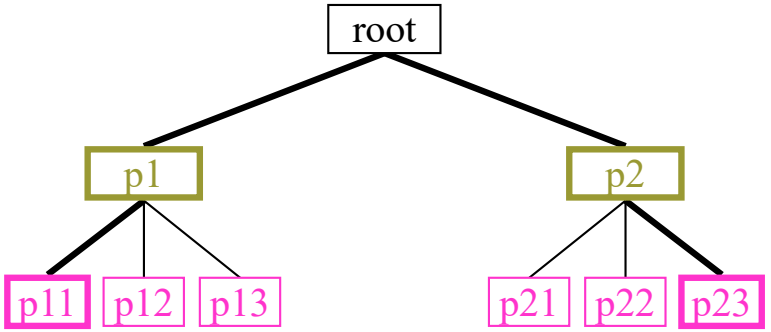




2.4.1 Nächste Nachbarn Anfragen

- Fazit:
  - Priorisierung mit MINDIST bewirkt Reduktion der Seitenzugriffe von 7 auf 3
  - MINMAXDIST verbessert Pruning-Distanz, verhindert hier aber keine Seitenzugriffe
  - Trotz Priorisierung: Tiefendurchlauf kann prinzipiell stark fehlgeleitet werden wenn z.B. eine Seite auf dem ersten Level sehr nah am Queryobjekt liegt, ihre Kindseiten aber relativ weit weg





  - Bei Start mit p_2 hätte keine der Kindseiten von p_1 geladen werden müssen