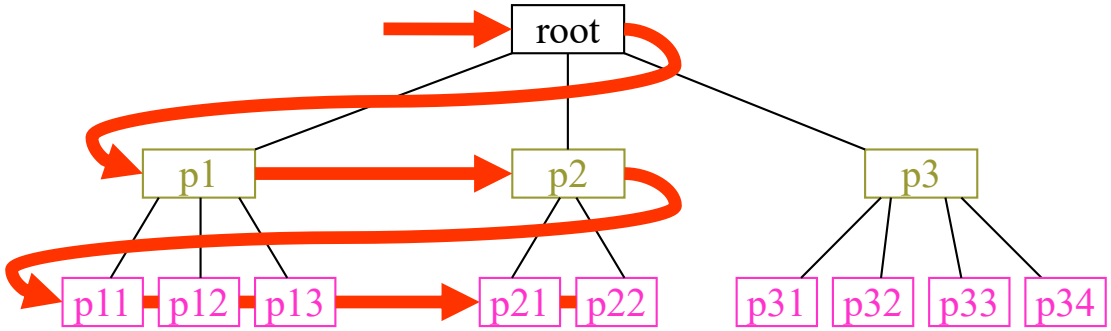
2.4.1 Nächste Nachbarn Anfragen

- Algorithmus mit Index: Breitensuche



  - Abgesehen von MINMAXDIST-Abschätzung stehen Punktdistanzen erst auf Blatt-Ebene zur Verfügung
    - Viele Zugriffe auf Directoryseiten
    - Die erste Punktdistanz hätte viele dieser Zugriffe schon verhindern können
  - Speicherintensiv
    - Worst-case: gesamte letzte Directory-Ebene muss im RAM gehalten werden

2.4.1 Nächste Nachbarn Anfragen

- Algorithmus mit Index: Prioritätssuche nach [HS 95]

  [Hjaltason, Samet. Proc. Int. Symp. on Large Spatial Databases (SSD), 1995]

  - Statt rekursivem Durchlauf: Liste der aktiven Seiten (active page list APL)
    - Seite p ist aktiv genau dann wenn folgende Bedingungen erfüllt sind:
      - p wurde noch nicht geladen
      - Elternseite von p wurde bereits geladen
      - MINDIST(q, p.getRegion()) ≤ pruningdist
    - APL wird mit Wurzel des Indexes initialisiert
    - Seiten in APL nach MINDIST zum Anfrageobjekt aufsteigend sortiert
    - Algorithmus entnimmt immer die erste Seite aus APL (mit kleinster MINDIST)
    - Entnommene Seite wird geladen und verarbeitet: („verfeinert“)
      - Datenseiten werden wie bisher verarbeitet
      - Directoryseiten: Kindseiten mit MINDIST ≤ pruningdist in APL einfügen
    - Ändert sich pruningdist werden Seiten mit MINDIST > pruningdist alternativ:
      - aus APL entfernt
      - als gelöscht markiert
      - ohne explizite Markierung später ignoriert

2.4.1 Nächste Nachbarn Anfragen

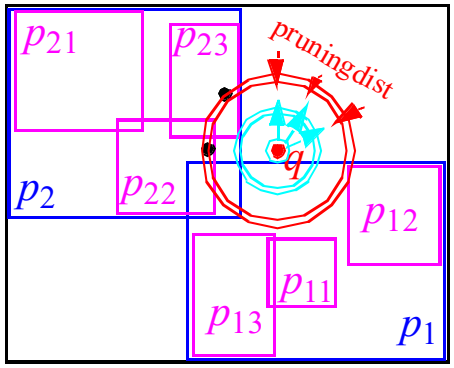
- Algorithmus:

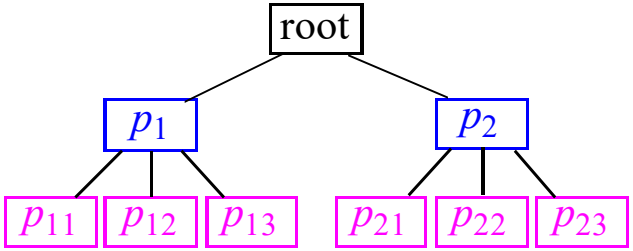
Globale Variablen: stopdist = +∞; pruningdist = +∞;

```
NN-Index-HS(pa, q)              // pa = Diskadress z.B. der Wurzel des Indexes
      result = ∅;
   apl = LIST OF (dist:Real, da:DiskAdress) ORDERED BY dist ASCENDING
   apl = [(0.0, pa)]
   WHILE NOT apl.isEmpty() AND apl.first().dist ≤ pruningdist DO
      p := apl.getFirst().da.loadPage();
      apl.deleteFirst();
      IF p.isDataPage() THEN

         (* wie bisher *)

      ELSE                              // p ist Directoryseite
         FOR i=0 TO p.size() DO
            IF MINDIST(q, p.getRegion(i)) ≤ pruningdist THEN
               apl.insert(MINDIST(q, p.getRegion(i)), p.childPage(i));
      RETURN result;
```

2.4.1 Nächste Nachbarn Anfragen

- Beispiel





APL:

1. root
2. p_1 p_2
3. p_2 p_{12} p_{13} p_{11}
4. p_{23} p_{22} p_{12} p_{13} p_{11} p_{21}
5. p_{22} p_{12}

- Eigenschaften
  - Allgemein
    - Seiten werden nach aufsteigendem Abstand geordnet zugegriffen (blaue Kreise)
    - pruningdist wird kleiner, sobald nähergelegenes Objekt gefunden (rote Kreise)
    - Anfragebearbeitung stoppt, wenn beide Kreise sich treffen

2.4.1 Nächste Nachbarn Anfragen

- Speicherbedarf
  - Wie bei Breitensuche kann gesamter unterste Directorylevel in APL stehen
  - Dieser Fall is allerdings unwahrscheinlicher als bei Breitensuche
  - Speicherkomplexität O(n) (Tiefensuche O(log n)

2.4.1 Nächste Nachbarn Anfragen

- Optimalität des Verfahrens

  [Berchtold, Böhm, Keim, Kriegel. ACM Smp. Principles of database Systems (PODS), 1997]

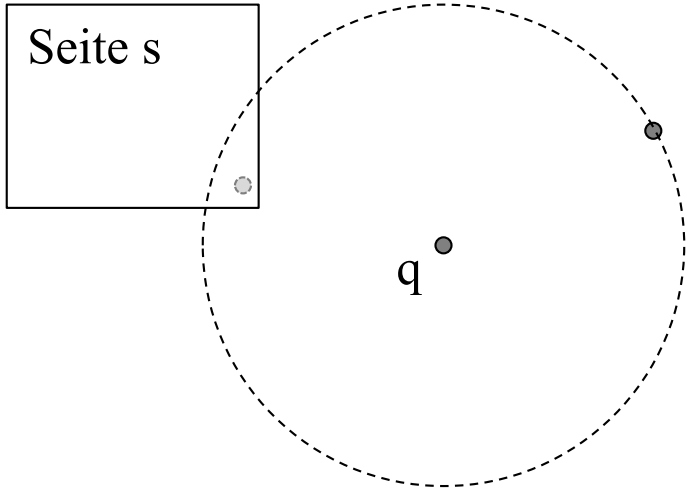
  - Prioritätssuche nach [HS 95] ist optimal bzgl. der Anzahl der Seitenzugriffe
  - Beweis (Überblick):
    - Lemma 1: jeder korrekte Algorithmus muss mind. die Seiten laden, die von der NN-Kugel um q berührt werden
    - Lemma 2: das Verfahren greift auf Seiten in aufsteigendem Abstand von q zu
    - Lemma 3: keine Seite s wird zugegriffen, mit MINDIST(q, s) > NN-Distanz(q)

2.4.1 Nächste Nachbarn Anfragen

- **Lemma 1:** Ein korrekter NN-Algorithmus muss mind. die Seiten *s* laden, die MINDIST(q, s) ≤ NN-Distanz(q) erfüllen.

  Beweis: Angenommen eine Seite *s* mit MINDIST(q, s) ≤ NN-Distanz(q) wird nicht geladen. Dann kann diese Seite Punkte enthalten (als Datenseite; Directoryseiten können im entspr. Teilbaum Punkte speichern), die näher am Anfragepunkt liegen als der nächste Nachbar. Der nächste Nachbar ist also nicht als solcher validiert, da über Punkte in einem Teilbaum keine Infos bekannt sind, außer dass sie in der entsprechenden Region liegen.

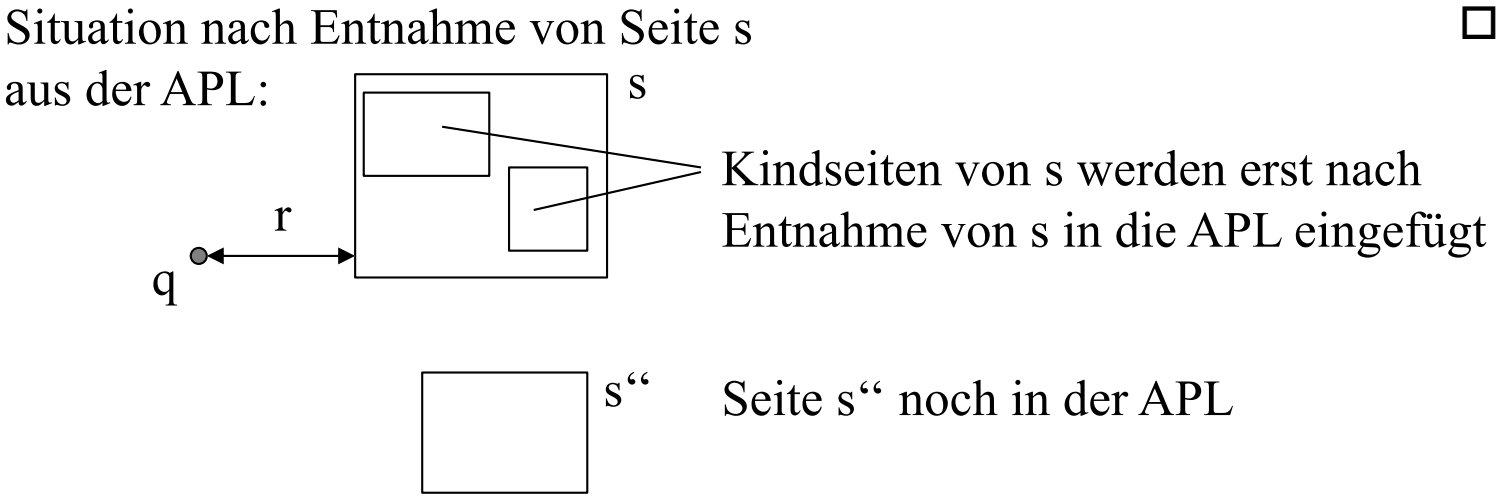
  □



2.4.1 Nächste Nachbarn Anfragen

- **Lemma 2:** Das Verfahren greift auf die Seiten des Index aufsteigend sortiert nach MINDIST zu.

  Beweis: Die Seiten werden in aufsteigender Reihenfolge aus der APL entnommen. Es muss also nur sichergestellt werden, dass nach Entnahme von Seite *s* keine Seiten *s'* mehr in APL eingefügt werden, mit MINDIST(*q*,*s'*) < *r* := MINDIST(*q*,*s*). Alle Seiten, die nach Entnahme von s in APL eingefügt werden, sind entweder Kindseiten von *s* oder Kindseiten von Seiten *s''* mit MINDIST(*q*,*s''*) ≥ *r*. Da die Region einer Kindseite in der Region der Elternseite vollständig eingeschlossen ist, ist die MINDIST einer Kindseite nie kleiner als die der Elternseite. Daher haben alle später eingefügten Seiten eine MINDIST ≥ *r*. □



2.4.1 Nächste Nachbarn Anfragen

- **Lemma 3:** Das Verfahren greift auf keine Seite *s* zu, mit MINDIST(*q*,*s*) > NN-Distanz(*q*).

  Beweis: Nach Lemma 2 können nach Zugriff auf Seite *s* nur Punkte *p* gefunden werden, mit dist(*q*, *p)* > MINDIST(*q*, *s*). Wäre vor Zugriff auf *s* ein Punkt *p* mit dist(*q*, *p)* < MINDIST(*q*, *s*) gefunden worden, dann wäre s aus der APL gelöscht worden bzw. der Algorithmus hätte vor der Bearbeitung von *s* angehalten.

  □

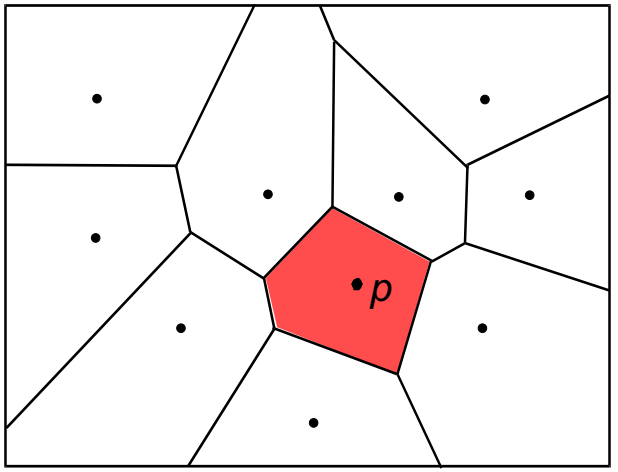
- Aus Lemma 1-3 ergibt sich, dass der Algorithmus nach [HS 95] optimal bzgl. der Anzahl der Seitenzugriffe ist.

2.4.1 Nächste Nachbarn Anfragen

- Hybrider Algorithmus: Voronoi-Diagramme

  [Berchtold, Ertl, Keim, Kriegel, Seidl. Proc. Int. Conf. Data Engineeering (ICDE), 1998]

  - Nur für Vektordaten!!!
  - Idee:
    - Berechne für jeden Punkt *p* den Teil des Datenraumes in dem *p* der nächste Nachbar ist (Voronoi-Zellen)
    - Speichere Voronoi-Zellen in DB
    - NN-Anfrage entspricht Punktanfrage
      mit *q* auf den Voronoi-Zellen
      => Punkt *p*, in dessen Voronoi-Zelle *q* liegt, ist der NN von *q*



  - Problem:
    - Voronoi-Zellen sind konvexe Polygone
    - Ab $d > 2$ sehr komplex (große Anzahl Eckpunkte)
  - Lösung: Approximation der Voronoi-Zellen (z.B. mit MBR)
    - Nur Filterschritt, da MBRs sich überlappen können, und q in mehrere dieser MBRs liegen kann
      => Verfeinerung der Kandidaten mit exakten Punktdistanzen