CSEG601 & CSE5601
Spatial Data Management & Application :

RNN Query Processing method using R-tree-

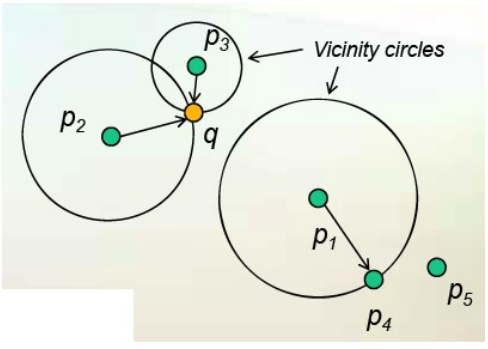
Sungwon Jung

Big Data Processing & Database Lab
Dept. of Computer Science and Engineering
Sogang University
Seoul, Korea
Tel: +82-2-705-8930
Email : jungsung@sogang.ac.kr

RNN Definition

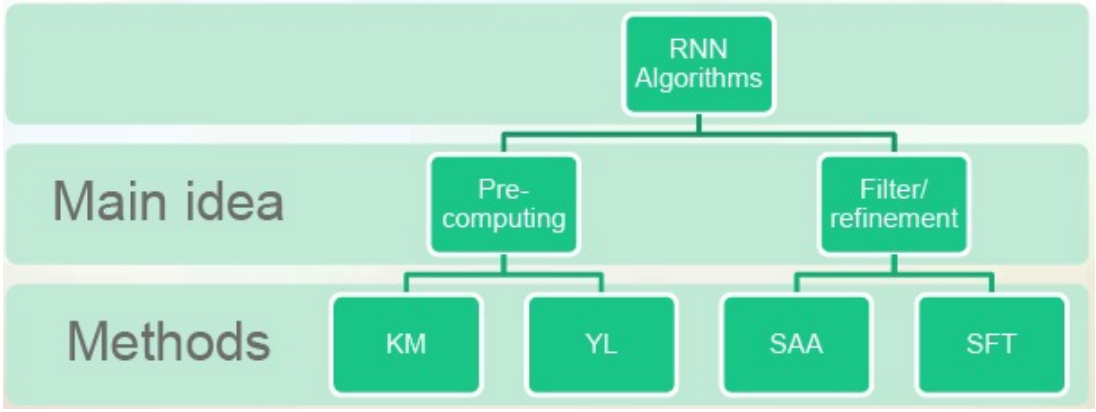
- A data point p is the reverse nearest neighbor of query point q, if there is no point p' such that *dist(p', p)< dist(q, p),* i.e. q is the NN of p.

$\text{NN}(p_2) = \text{NN}(p_3) = \text{q}$
$\text{RNN(q)} = \{p_2, p_3\}$



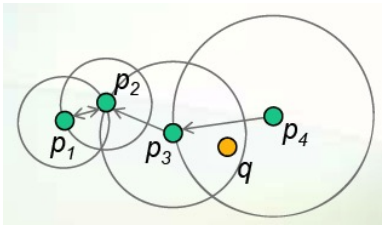
- In our example, p_2, p_3 are the houses for which q is the nearest restaurant
- Is RNN a symmetric relation?

Related Works

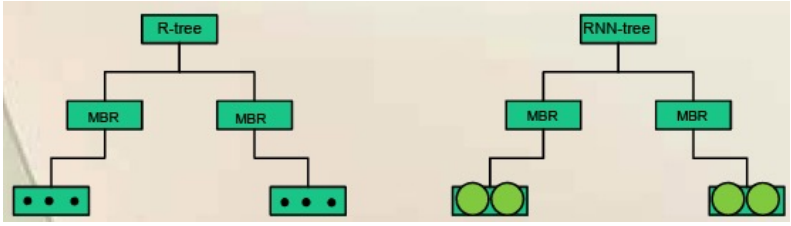


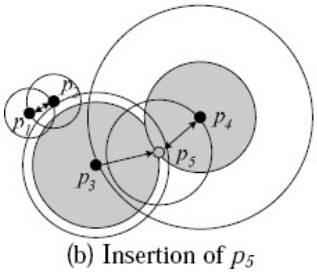
Original RNN method [KM00]

- For all *p:*
  1. Pre-compute NN(*p*)
  2. Represent *p* as a vicinity circle
  3. Index the MBR of all circles by an R-tree(Named RNN-tree)
  4. RNN(*q*)= all circles that contain *q*



- Needs two trees: RNN-tree & R-tree

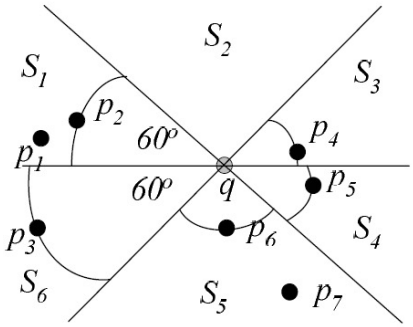




(b) Insertion of p_5

SAA

- Elimination of the need for pre-computing all NNs in filter/ refinement methods
- SAA:
  - Divide the space around query into six equal regions
  - Find NN(q) in all regions (candidate keys)
  - Either (i) or (ii) holds for each candidate key p
    - (i) p is in RNN(q)
    - (ii) No RNN(q) in S_i
    - RNN(q)= $\{p_6\}$
- Any Drawbacks?
  - The number of regions increases exponentially with the dimensionality

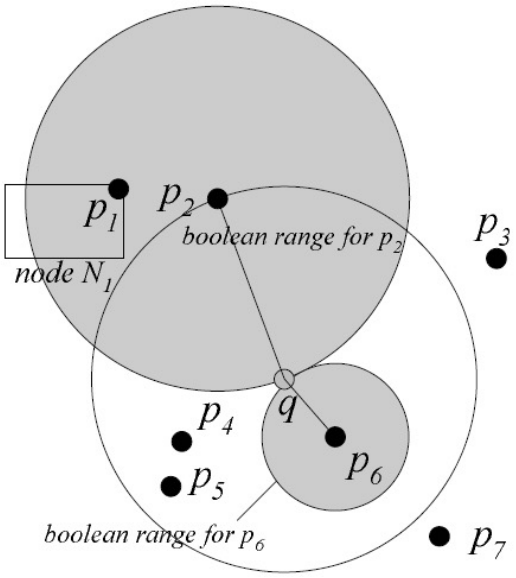


SFT

1. Find the kNNs of the query q (k candidates)
2. Eliminate the points that are closer to other candidates than q.
3. Apply *Boolean range queries* to determine the actual RNNs

- A Boolean range query terminates immediately when
  1. the first data point is found
  2. The entire side of a node MBR lies within a circle

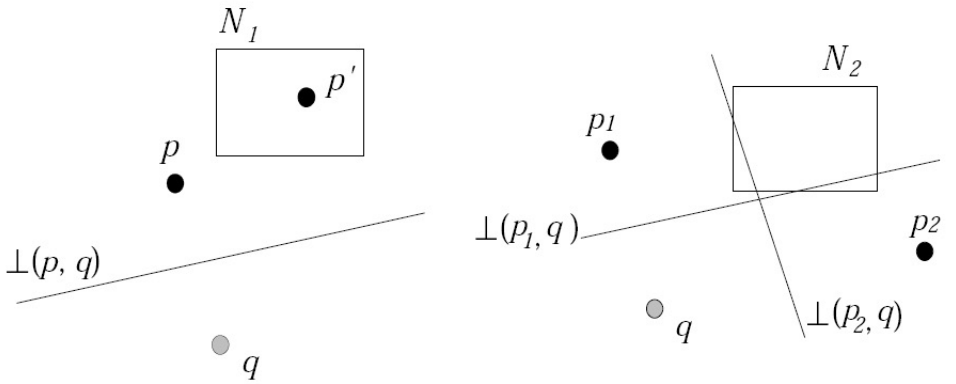
  e.g.,) minmaxdist(N_1,p_2) ≤ dist(p_2, q)
- Drawbacks?
  - False misses
  - Choosing a proper k



Step 1: Given k=4, Find 4NN Candidates Set = {p_6, p_5, p_4, p_2}
Step 2: Discard p_4 and p_6 since they are closer to each other than q
Step 3: Boolean Range Queries (p_2, dist(p_2,q)) and (p_6, dist(p_6,q))
Step 4: Discard p_2 since minmaxdist(N_1,p_2) <= dist(p_2,q)
False miss: p_3 due to k =4.

Half-plane prunning

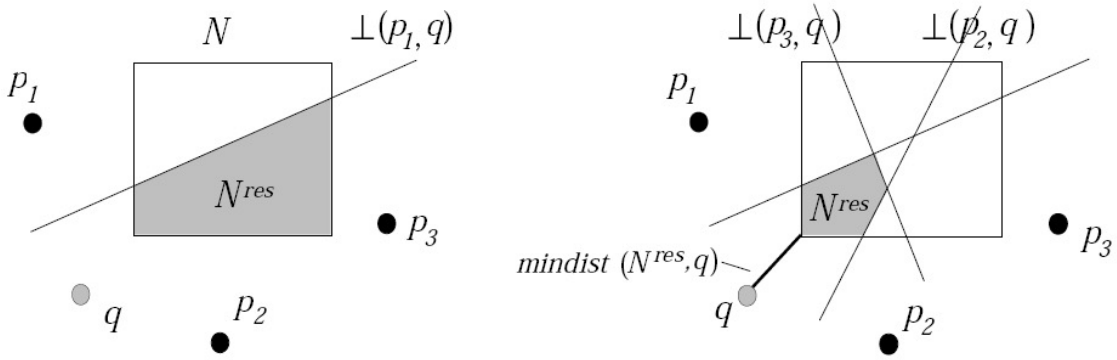
- Can p' be closer to q than p ?



(a) Pruning with one point (b) Pruning with two points

- If $p_1, p_2, \ldots, p_n$ n data points, then any node whose MBR falls inside $\cup_{i=1,\ldots,n} PL_{p_i(p_i,q)}$ cannot contain any RNN result.

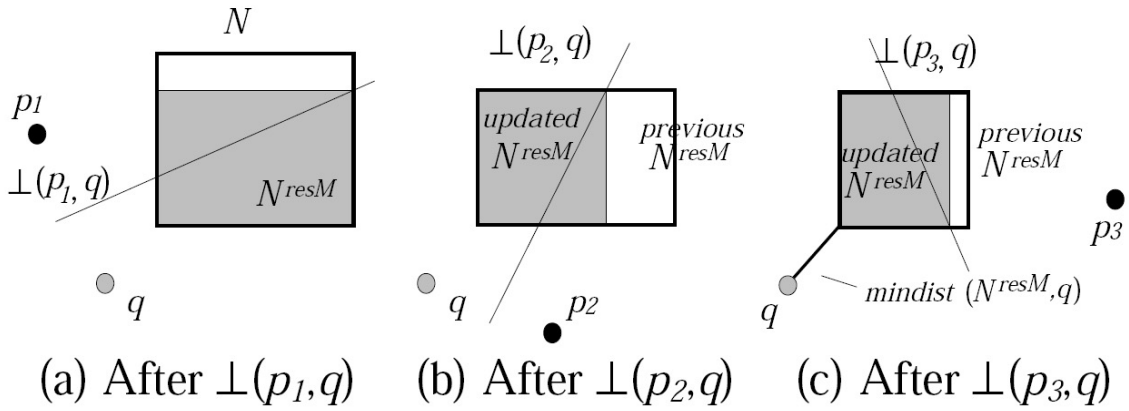
Computing the residual region



(a) After processing $\perp(q,p_1)$ (b) The final polygon

- O(n^2) processing time in terms of bisector trimming for computing N^{res}
- Computation of intersections does not scale with dimensionality

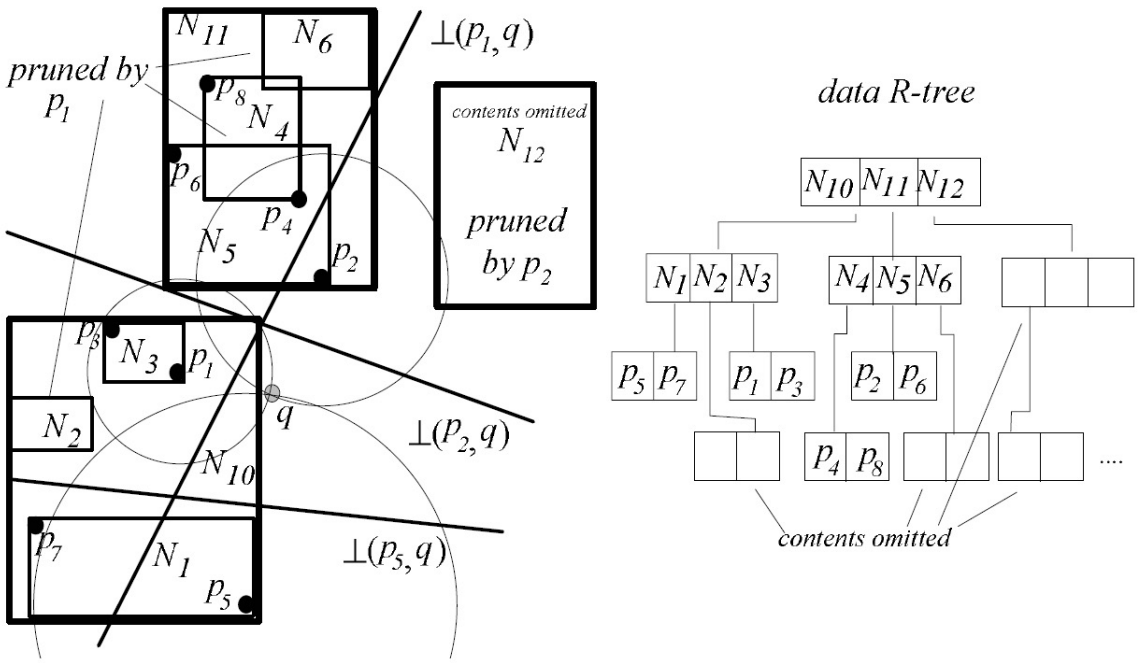
Approximating the residual MBR



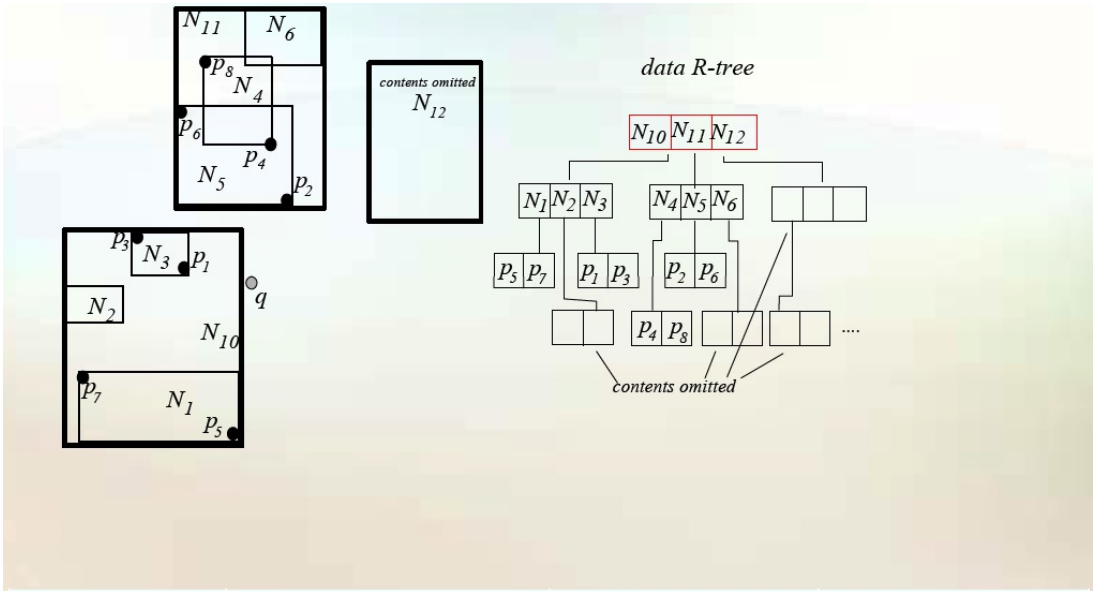
(a) After $\perp(p_1,q)$ (b) After $\perp(p_2,q)$ (c) After $\perp(p_3,q)$

- An MBR can be pruned if its residual region is empty
- The approximation is a superset of the real residual region
- We can prune an MBR if its approximate residual is empty
- Good news:
  - *O(n)* processing time for computing N^{resM}
  - No more hyper-polyhedrons to make the intersection computation complex

TPL algorithm for Single RNN – example(1)

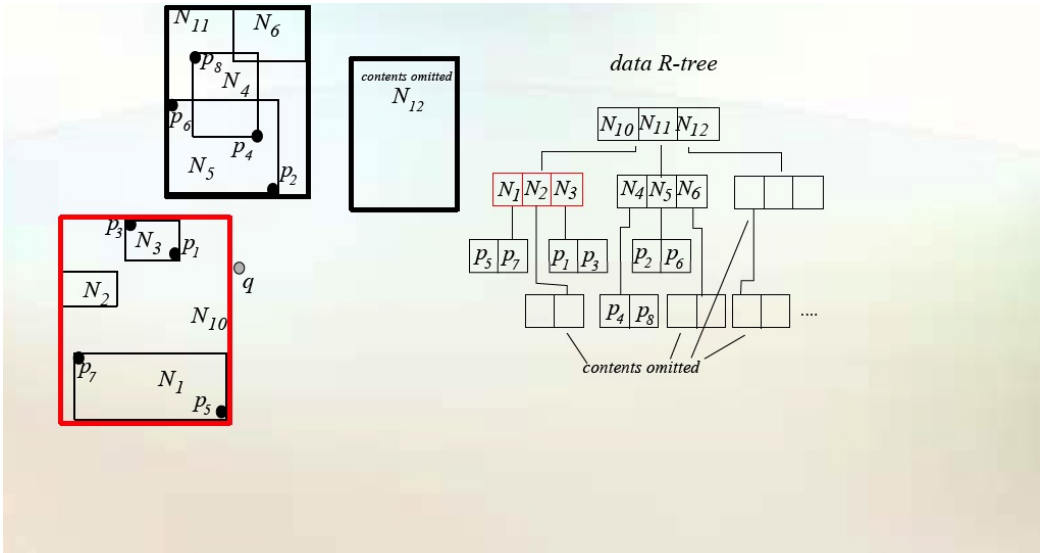


Flitering step - 1



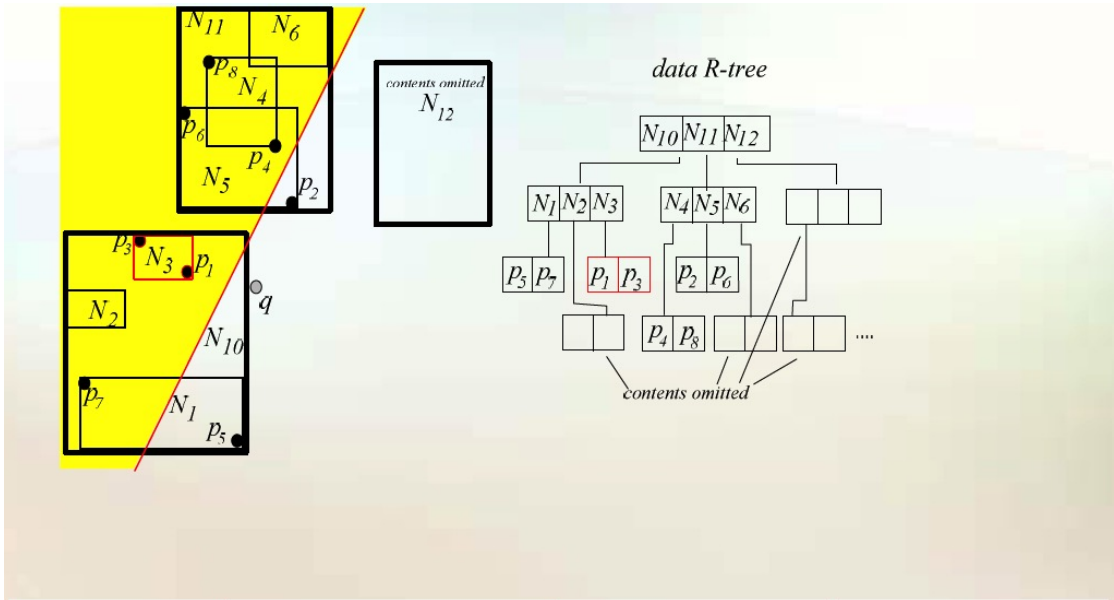
Action	Heap	S_{cnd}	S_{rfn}
Visit root	$\{N_{10}, N_{11}, N_{12}\}$	{}	{}

Flitering step - 2



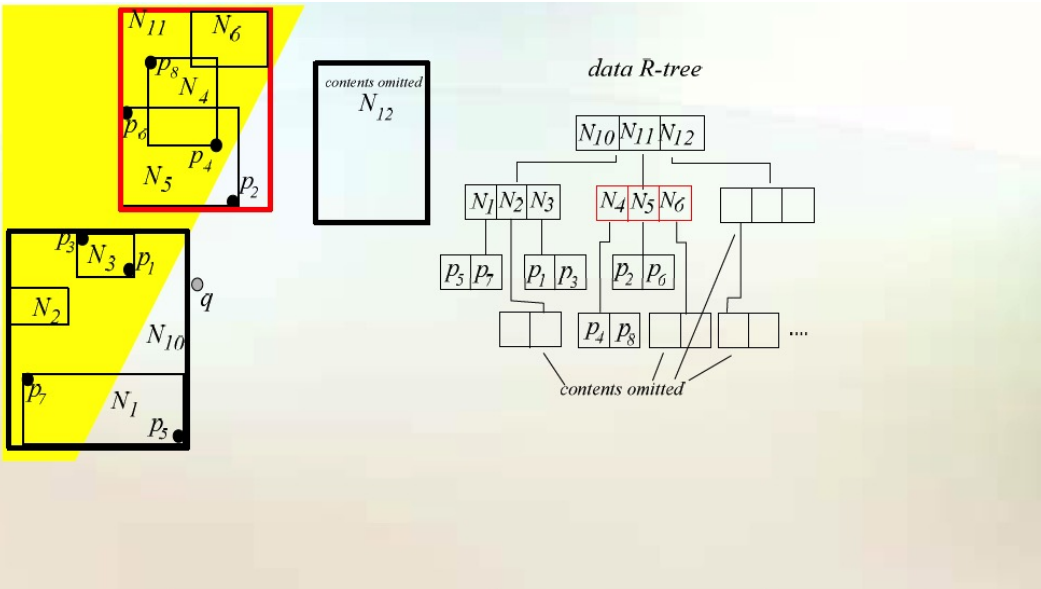
Action	Heap	S_{cnd}	S_{rfn}
Visit N_{10}	$\{N_3, N_{11}, N_2, N_1, N_{12}\}$	{}	{}

Flitering step - 3



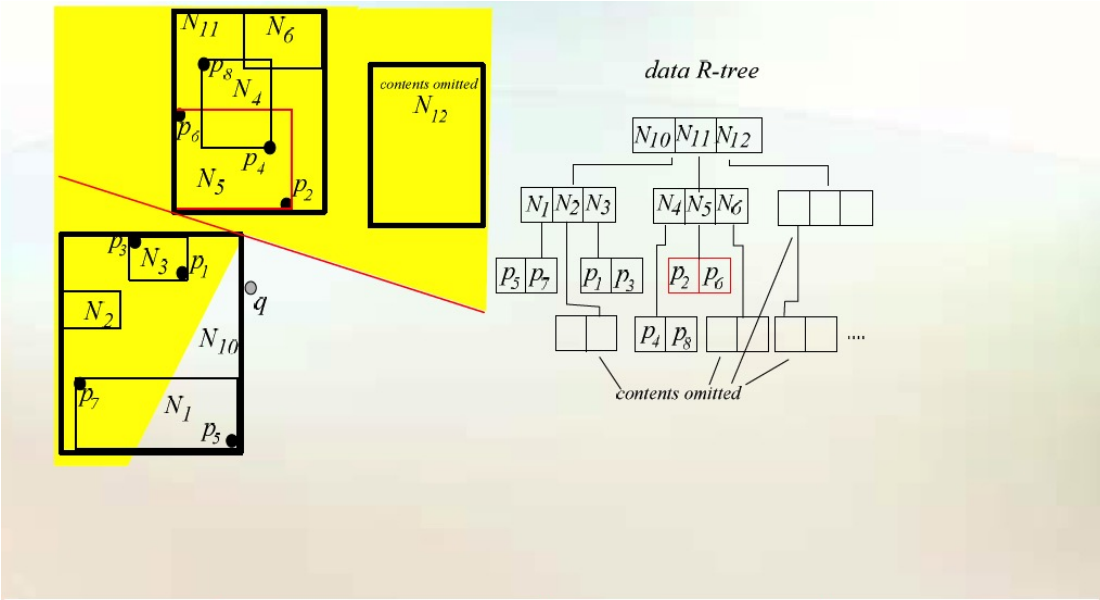
Action	Heap	S_{cnd}	S_{rfn}
Visit N_3	$\{N_{11}, N_2, N_1, N_{12}\}$	$\{p_1\}$	$\{p_3\}$

Flitering step - 4



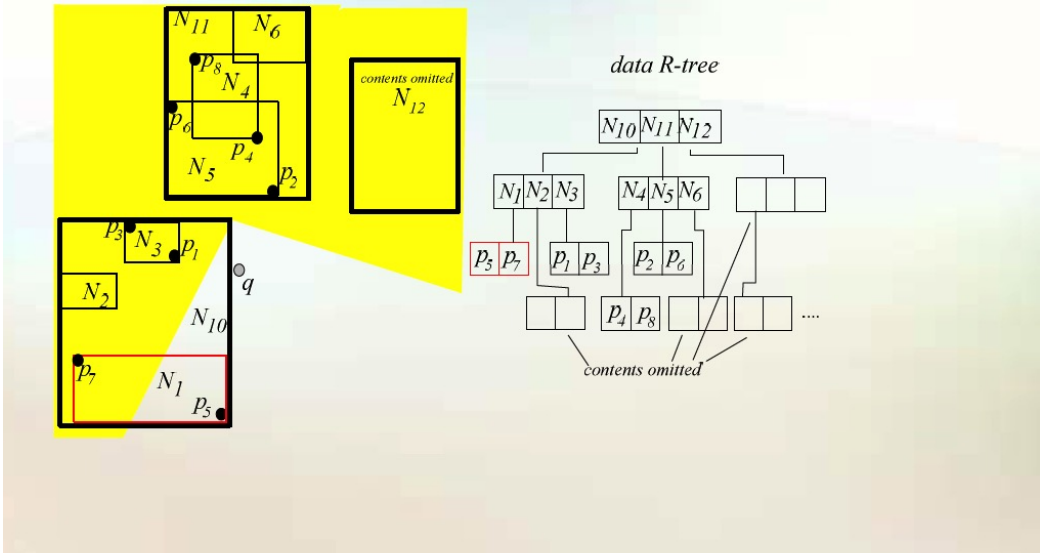
Action	Heap	S_{cnd}	S_{rfn}
Visit N_{11}	$\{N_5, N_2, N_1, N_{12}\}$	$\{p_1\}$	$\{p_3, N_4, N_6\}$

Flitering step - 5



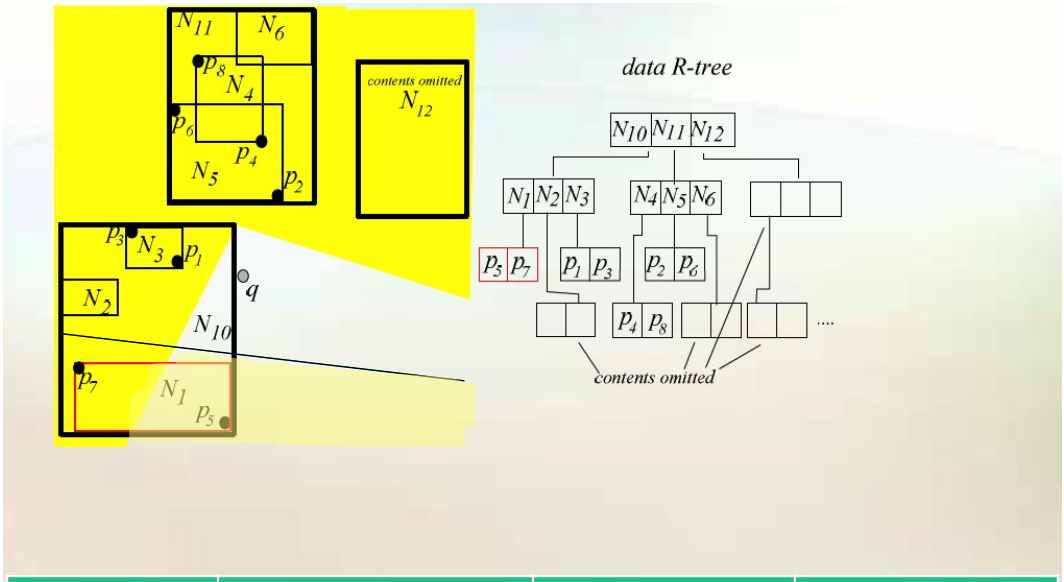
Action	Heap	S_{cnd}	S_{rfn}
Visit N_5	$\{N_2, N_1, N_{12}\}$	$\{p_1, p_2\}$	$\{p_3, N_4, N_6, p_6\}$

Flitering step - 6



Action	Heap	S_{cnd}	S_{rfn}
Visit N_1	$\{N_{12}\}$	$\{p_1, p_2, p_5\}$	$\{p_3, N_4, N_6, p_6, N_2, p_7\}$

Flitering step - 7

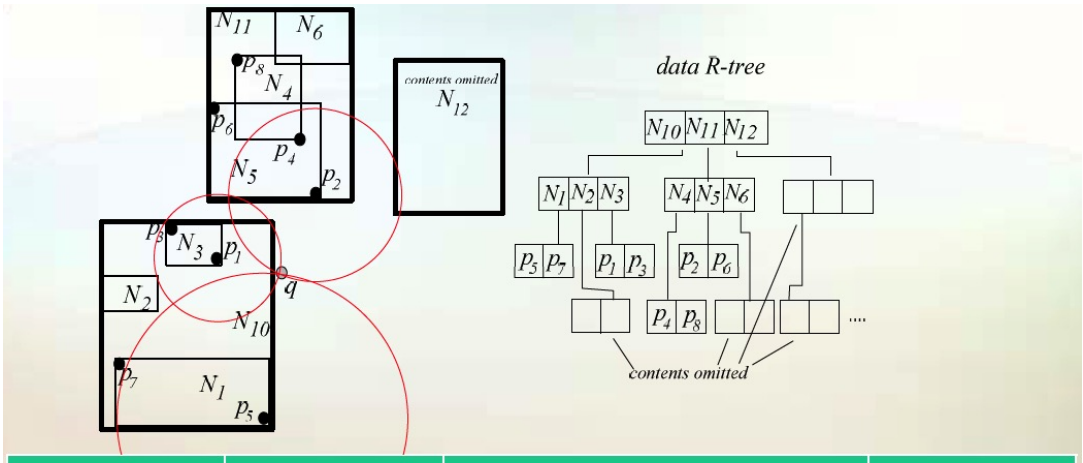


Action	Heap	S_{cnd}	S_{rfn}
	{}	$\{p_1, p_2, p_5\}$	$\{p_3, N_4, N_6, p_6, N_2, p_7, N_{12}\}$

Refinement Heuristics

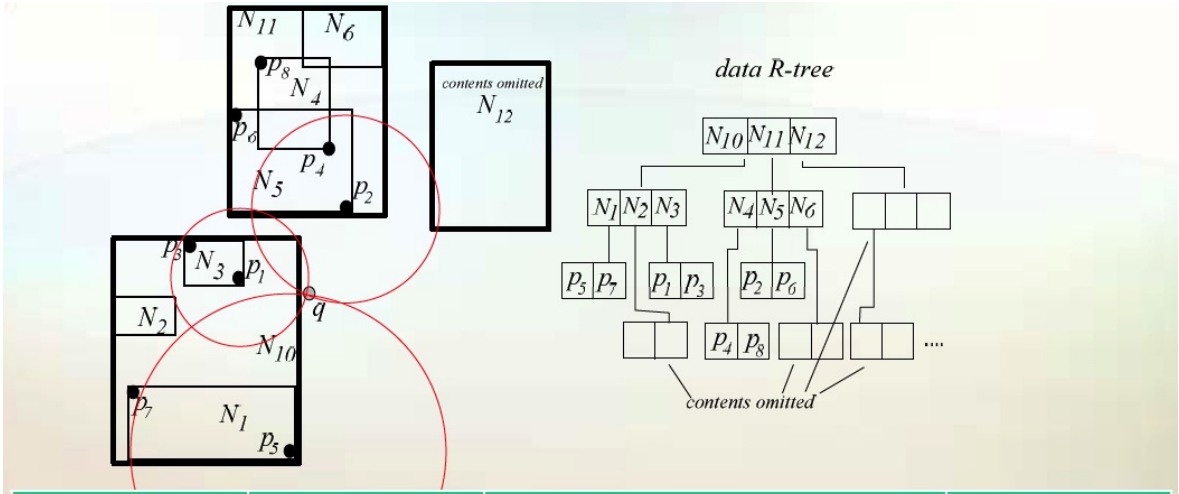
- Let P_{rfn} be the set of points and N_{rfn} be the set of nodes in S_{rfn}
- A point p from S_{cnd} can be discarded as a false hit if there is a point $p' \in P_{rfn}$ such that either of the following hold:
  - (i) $dist(p,p') < dist(p,q)$
  - (ii) There is a node MBR $N \in N_{rfn}$ such that $minmaxdist(p,N) < dist(p,q)$
- A candidate point can be eliminated if it is closer to another candidate point than to the query
- A point p from S_{cnd} can be reported as an actual result if the following two conditions hold:
  - (i) There is no point $p' \in P_{rfn}$ such that $dist(p,p') < dist(p,q)$
  - (ii) For every node $N \in N_{rfn}$: $mindist(p,N) \geq dist(p,q)$
- If none of the above works, visit all node MBRs $N \in N_{rfn}$ where $mindist(p,N) < dist(p,q)$ and use the mentioned heuristics considering the newly visited entries

Refinement step - 1



Action	Scnd	Srfn	Actual results
	$\{p_1, p_2, p_5\}$	$\{p_3, N_4, N_6, p_6, N_2, p_7, N_{12}\}$	{}
Invalidate p_1	$\{p_2, p_5\}$	$\{N_4, N_6, N_2, N_{12}\}$	{}
Validate p_5	$\{p_2\}$	$\{N_4, N_6, N_2, N_{12}\}$	{}
Remove N_6, N_2	$\{p_2\}$	$\{N_4, N_{12}\}$	$\{p_5\}$

Refinement step - 2



Action	Scnd	Srfn	Actual results
	$\{p_2\}$	$\{N_4, N_{12}\}$	$\{p_5\}$
Access N_4	$\{p_2\}$	$\{p_4, p_8, N_{12}\}$	$\{p_5\}$
Invalidate p_2	{}	$\{N_{12}\}$	$\{p_5\}$