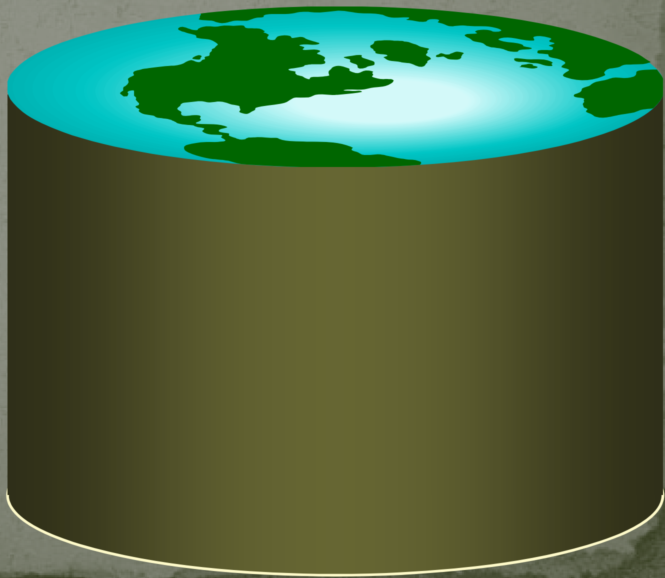
CS 505: Intermediate Topics to Database Systems

Instructor: Jinze Liu

Fall 2008



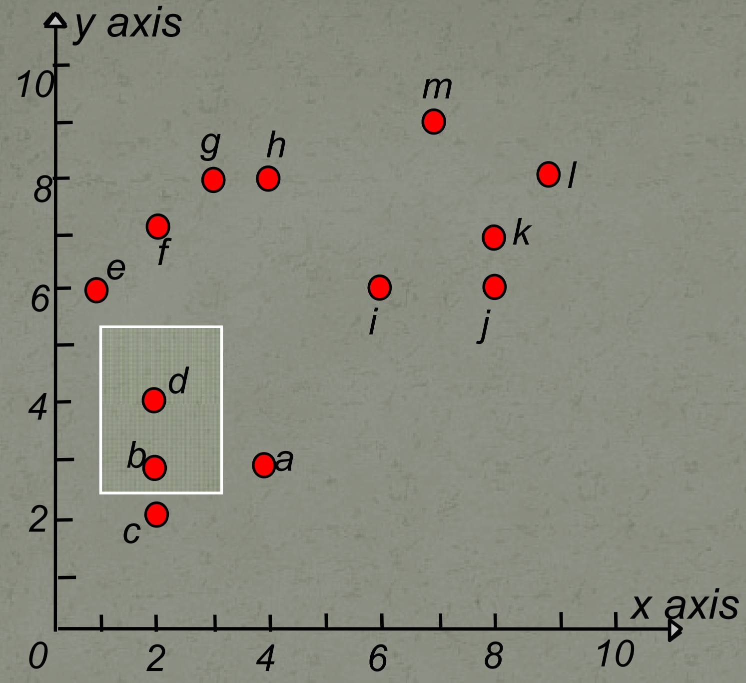
Beyond ISAM, B-, and B^+-trees

- Other tree-based indexes: R-trees and variants, GiST, etc.
- Hashing-based indexes: extensible hashing, linear hashing, etc.
- Text indexes: inverted-list index, suffix arrays, etc.
- Other tricks: bitmap index, bit-sliced index, etc.
  - How about indexing subgraph search?

R-Tree

- The R-tree
  - Range Query
  - Aggregation Query
- NN Query
- RNN Query
- Closest Pair Query
- Close Pair Query
- Skyline Query

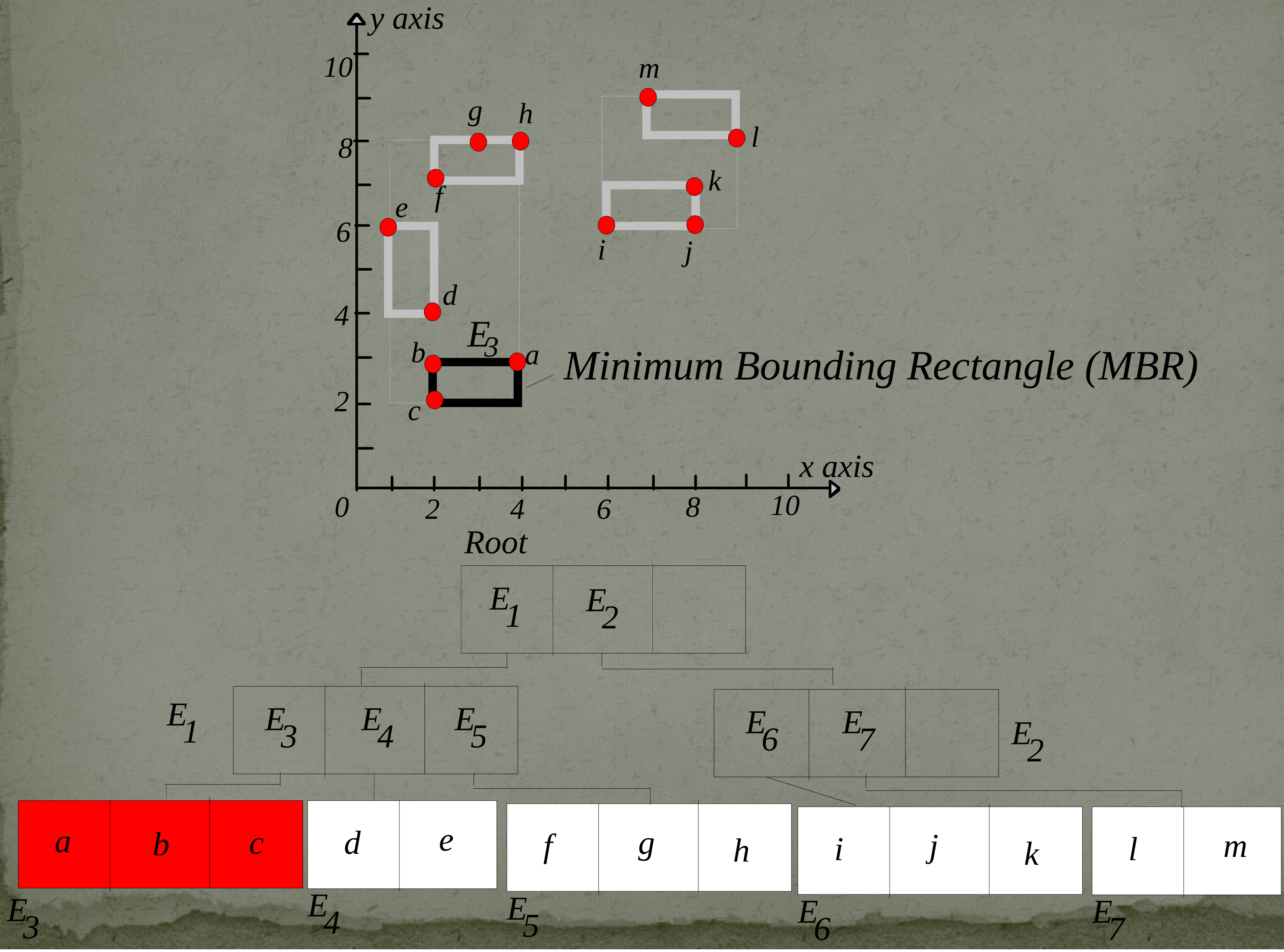
R-Tree Motivation



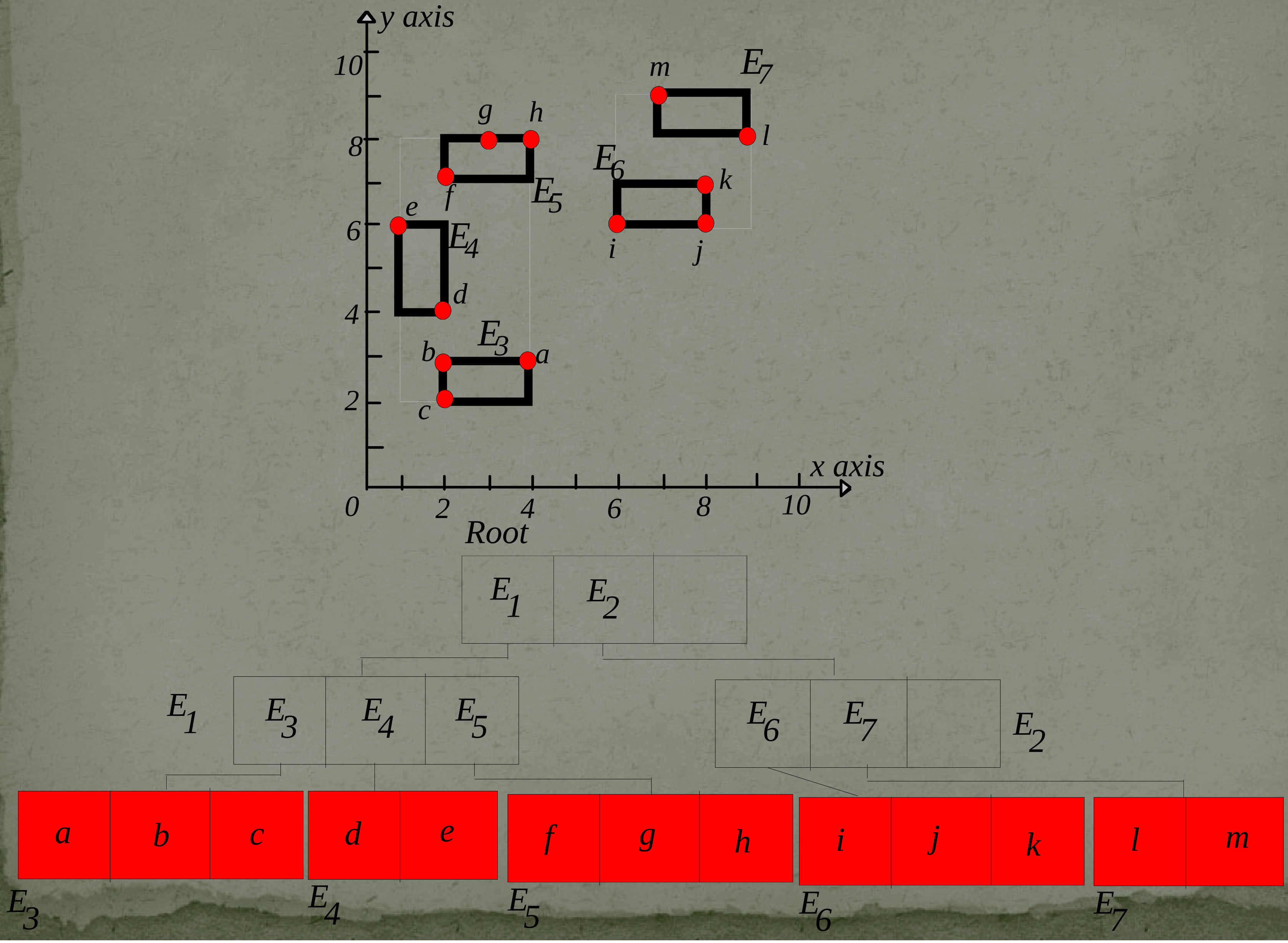
***Range query**: find the objects in a given range.*
E.g. find all hotels in Boston.

No index: scan through all objects. NOT EFFICIENT!

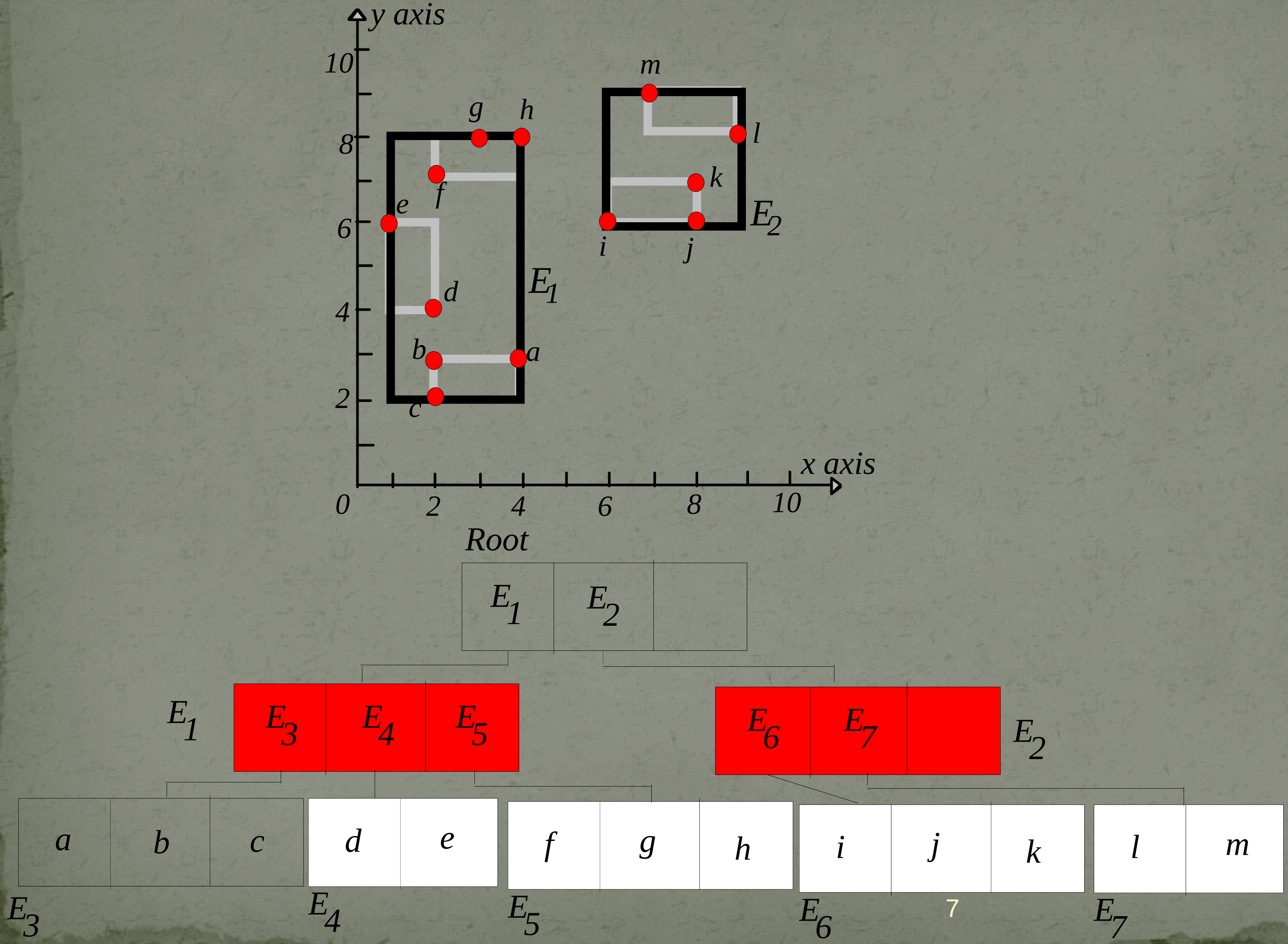
R-Tree: Clustering by Proximity



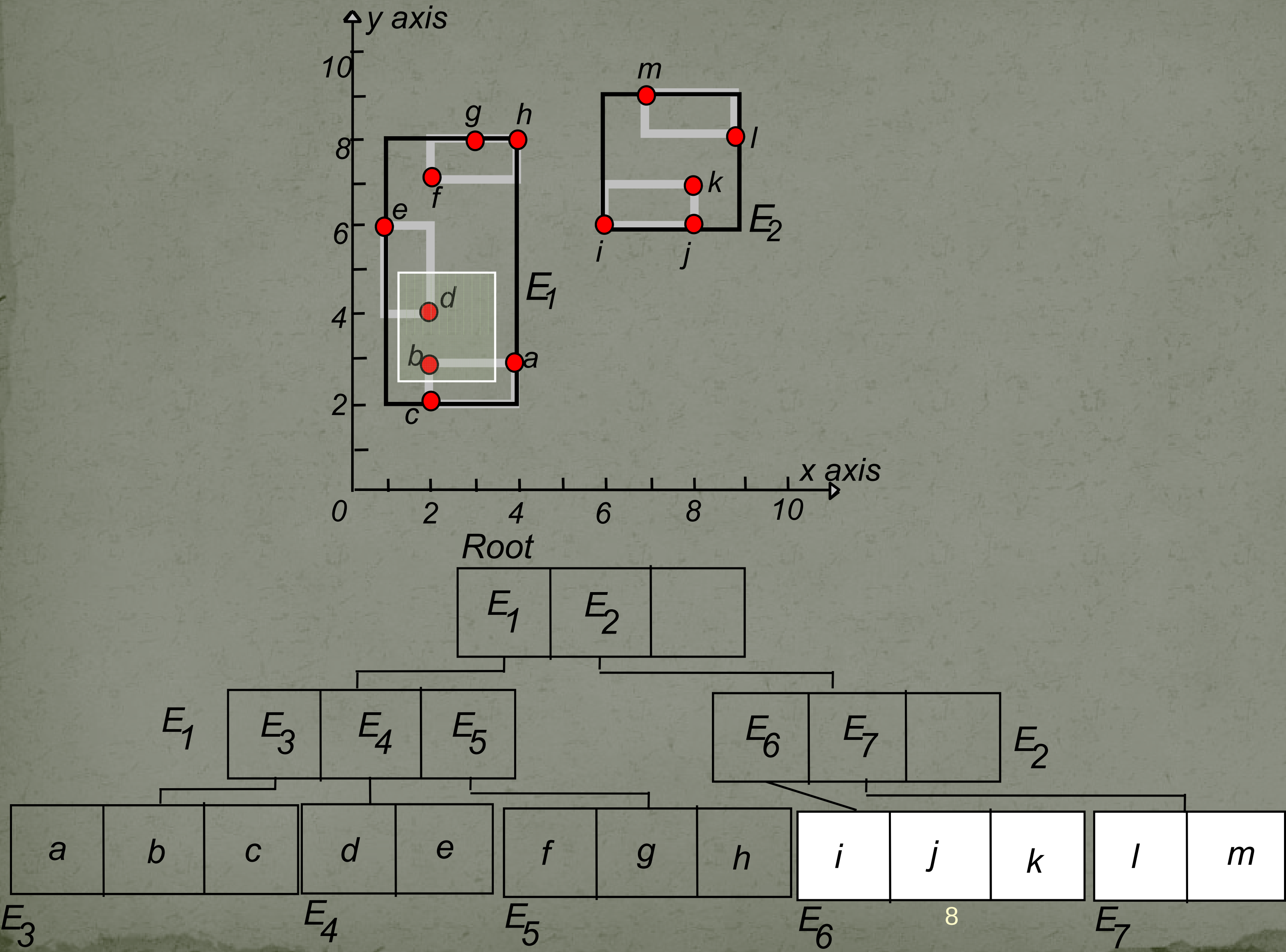
R-Tree


y axis
x axis
0
2
4
6
8
10
a
b
c
d
e
f
g
h
i
j
k
l
m
E3
E4
E5
E6
E7
Root
E1
E2
E1
E3
E4
E5
E6
E7
E2
a
b
c
d
e
f
g
h
i
j
k
l
m
E3
E4
E5
E6
E7

R-Tree

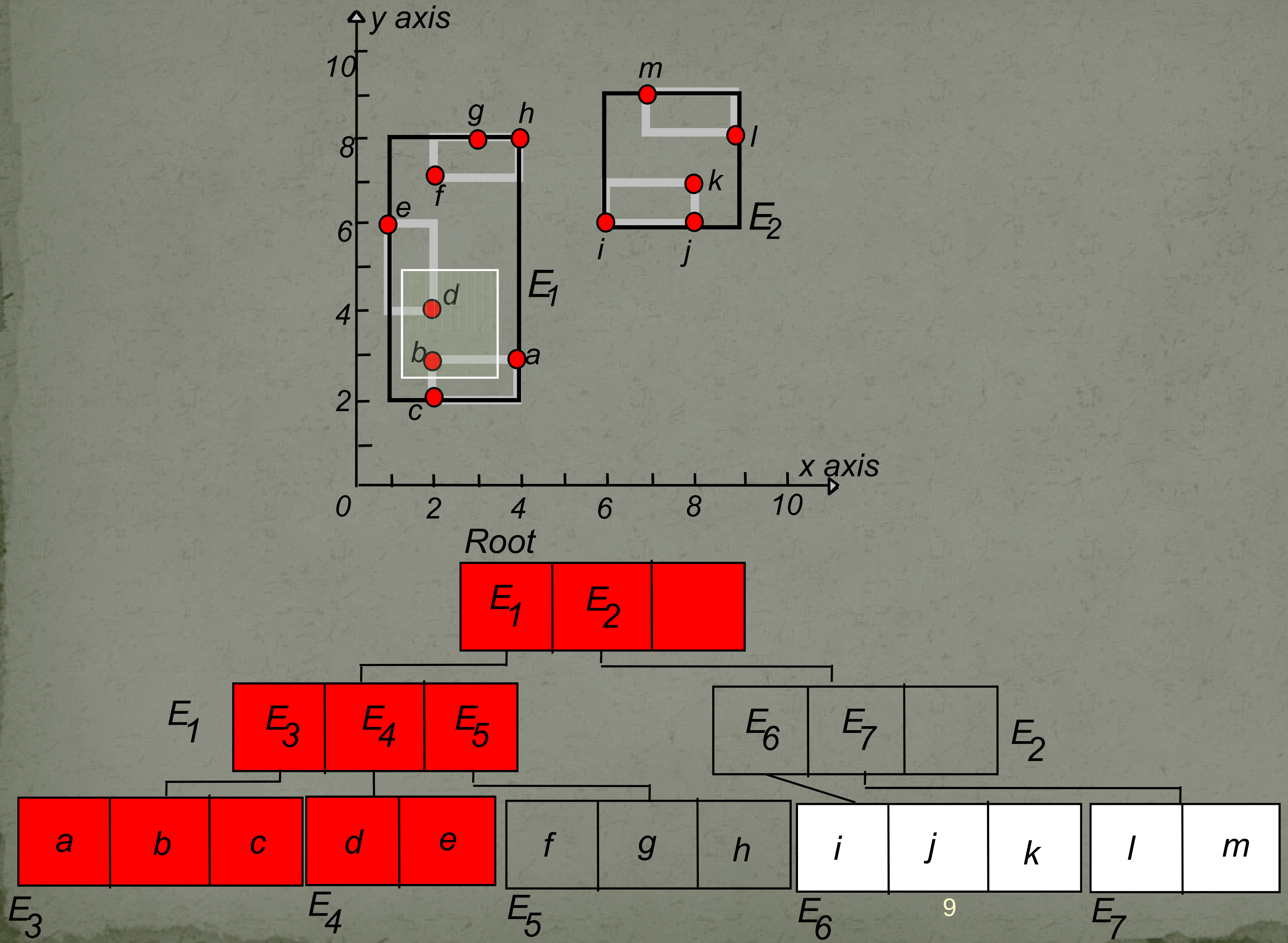

y axis
x axis
0
2
4
6
8
10
a
b
c
d
e
f
g
h
i
j
k
l
m
E_1
E_2
Root
E_1 E_2
E_1: E_3 E_4 E_5
E_2: E_6 E_7
E_3: a b c
E_4: d e
E_5: f g h
E_6: i j k
E_7: l m

Range Query

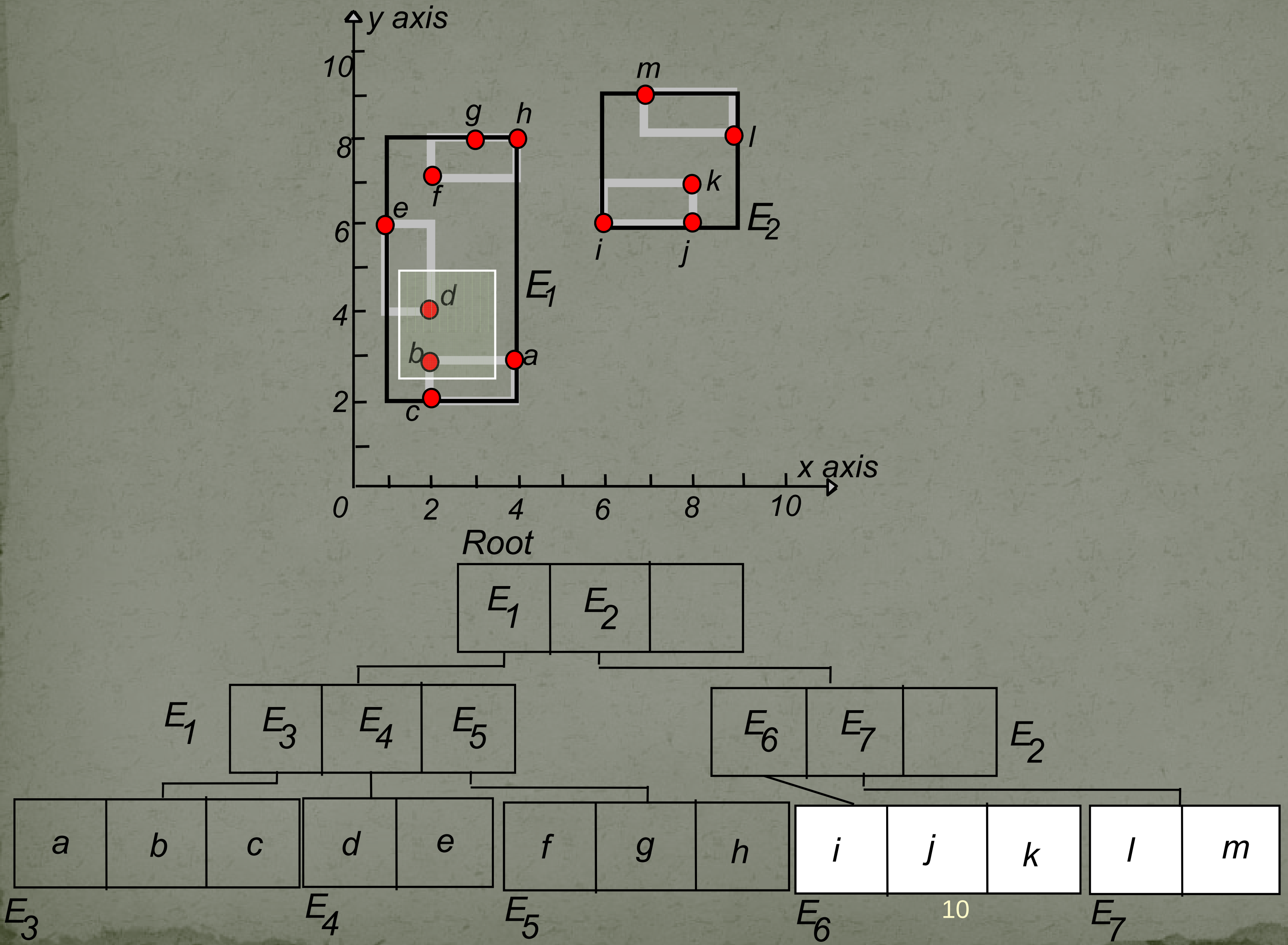


y axis
x axis
0
2
4
6
8
10
m
g
h
l
f
k
e
E2
i
j
d
E1
b
a
c
Root
E1
E2
E1
E3
E4
E5
E6
E7
E2
a
b
c
d
e
f
g
h
i
j
k
l
m
E3
E4
E5
E6
E7

Range Query

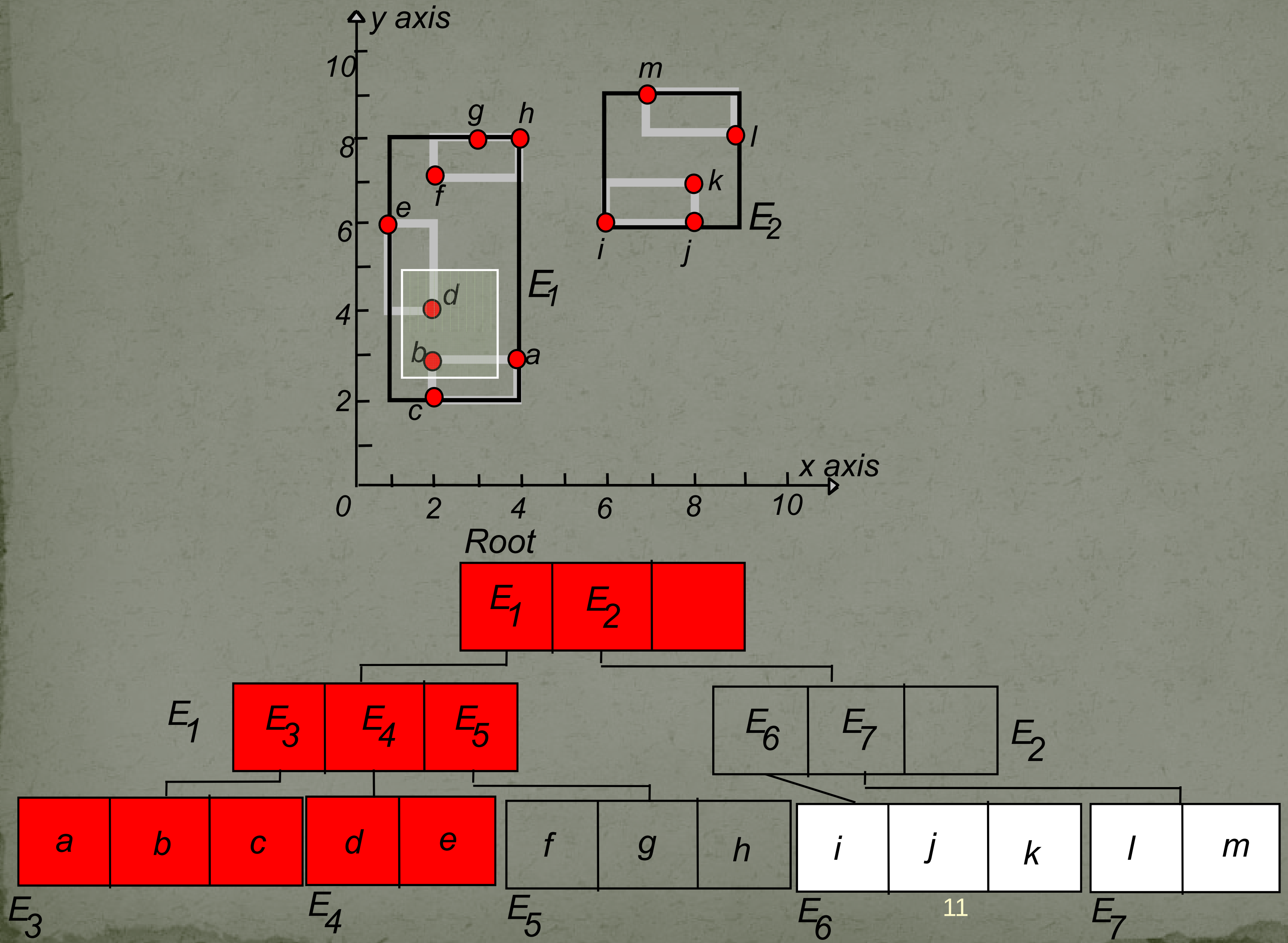

y axis
x axis
0
2
4
6
8
10
a
b
c
d
e
f
g
h
i
j
k
l
m
E_1
E_2
Root
E_1 E_2
E_3 E_4 E_5
E_6 E_7
E_3
E_4
E_5
E_6
E_7

Range Query



y axis
x axis
0
2
4
6
8
10
a
b
c
d
e
f
g
h
i
j
k
l
m
E_1
E_2
Root
E_1 E_2
E_1 E_3 E_4 E_5
E_6 E_7 E_2
a b c
d e
f g h
i j k
l m
E_3
E_4
E_5
E_6
E_7

Range Query



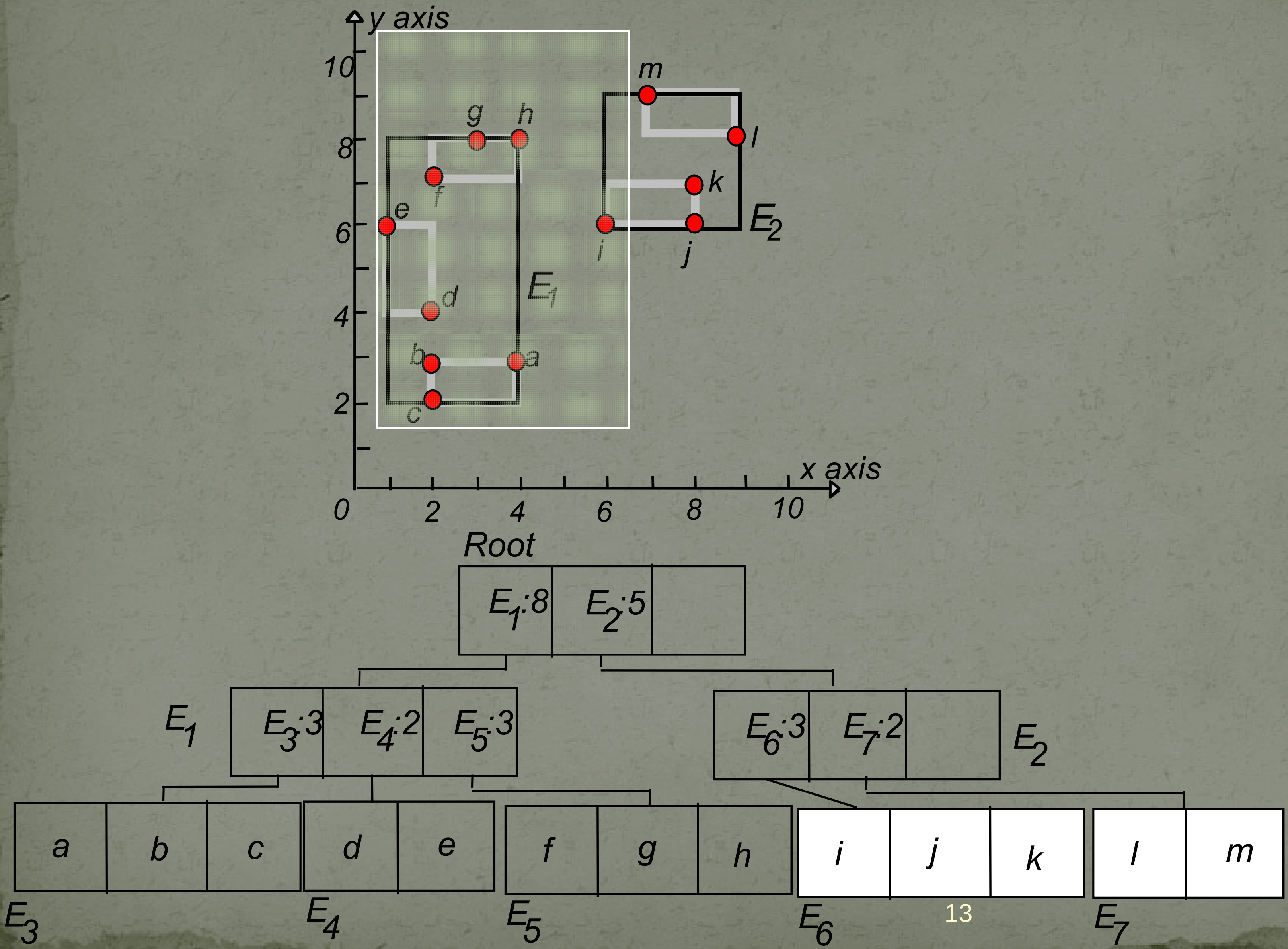
y axis
x axis
0
2
4
6
8
10
m
g
h
l
f
k
e
E2
i
j
d
E1
b
a
c
Root
E1
E2
E3
E4
E5
E6
E7
a
b
c
d
e
f
g
h
i
j
k
l
m
E3
E4
E5
E6
E7

Aggregation Query

- *Given a range, find some aggregate value of objects in this range.*
- COUNT, SUM, AVG, MIN, MAX
- E.g. find the total number of hotels in Massachusetts.

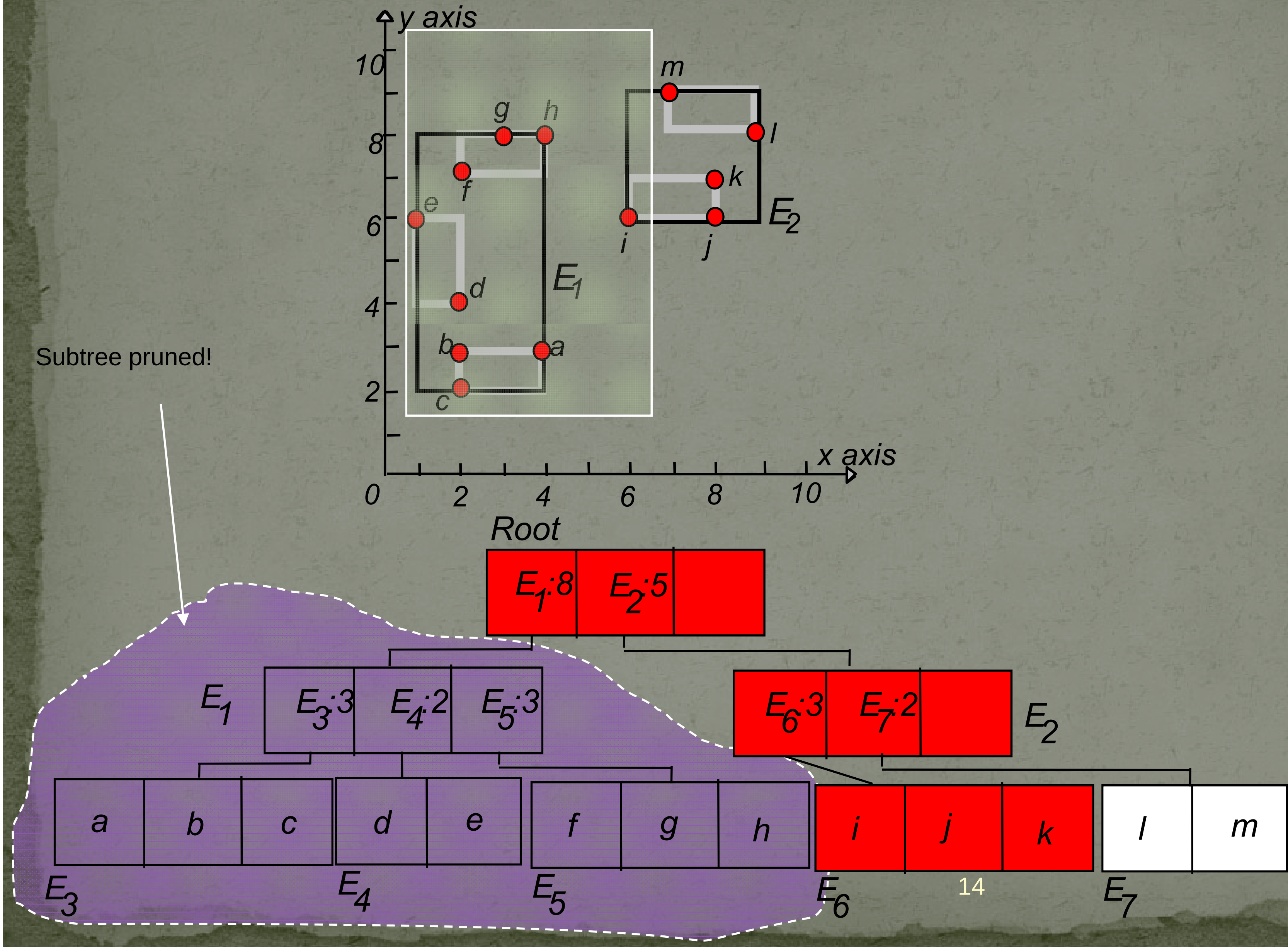
- Straightforward approach: reduce to a range query.
- Better approach: along with each index entry, store aggregate of the sub-tree.

Aggregation Query



Aggregation Query

Subtree pruned!



Content

- The R-tree
  - Range Query
  - Aggregation Query
- **NN Query**
- RNN Query
- Closest Pair Query
- Close Pair Query
- Skyline Query

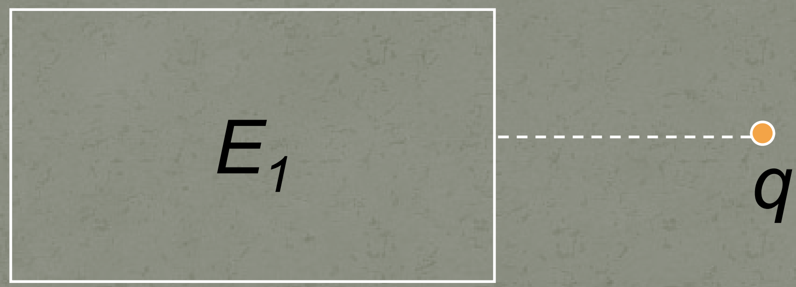
Nearest Neighbor (NN) Query

- Given a query location *q*, find the nearest object.

- E.g.: given a hotel, find its nearest bar.

A Useful Metric: MINDIST

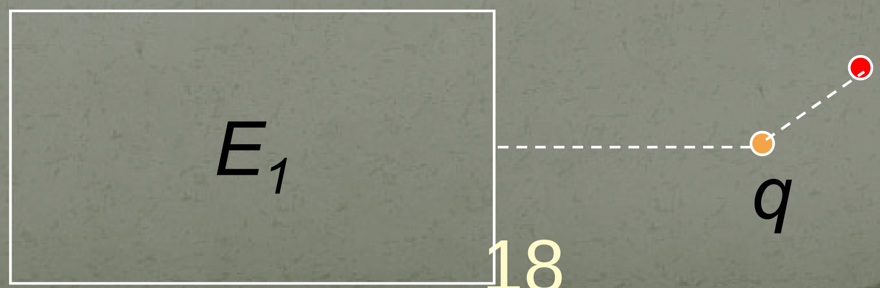
- Minimum distance between *q* and an MBR.



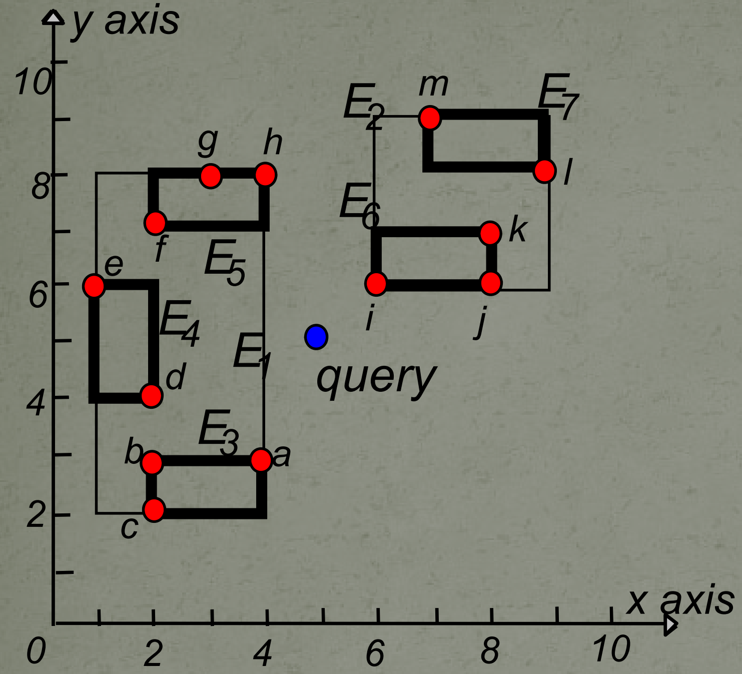
- It is an lower bound of $d(o, q)$ for every object *o* in E_1.
- $\text{MINDIST}(o, q) = d(o, q)$.

NN Basic Algorithm

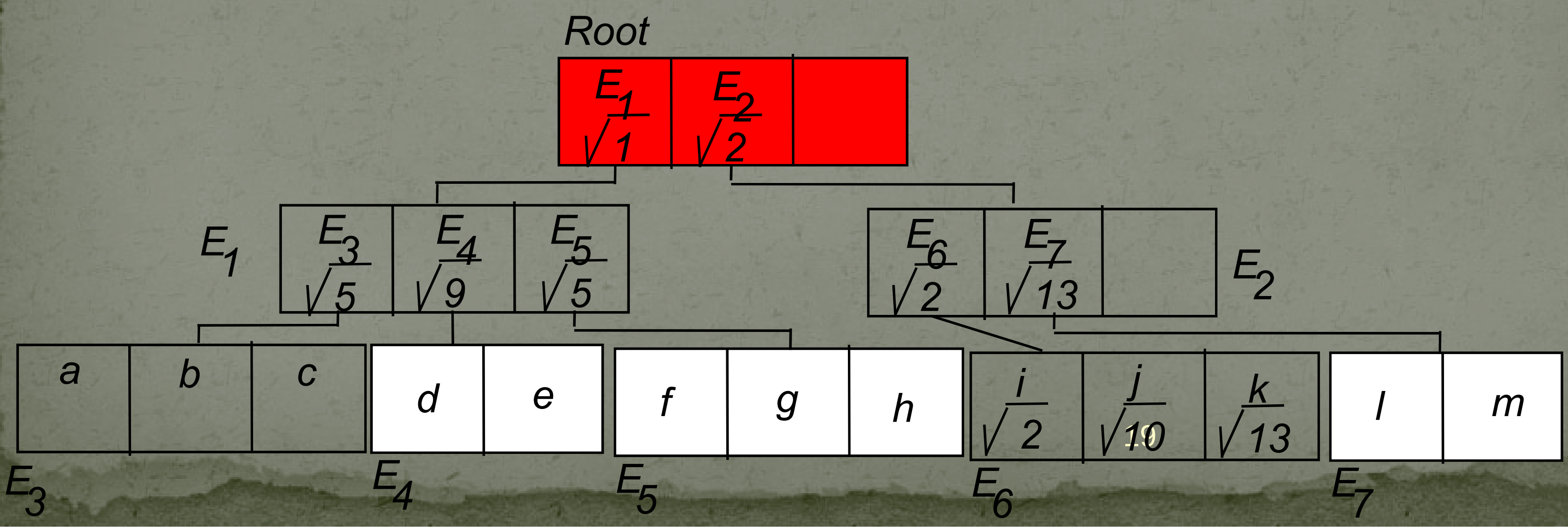
- Keep a heap *H* of index entries and objects, ordered by MINDIST.
- Initially, *H* contains the root.
- While $H \neq \phi$
  - Extract the element with minimum MINDIST
  - If it is an index entry, insert its children into *H*.
  - If it is an object, return it as NN.
- End while



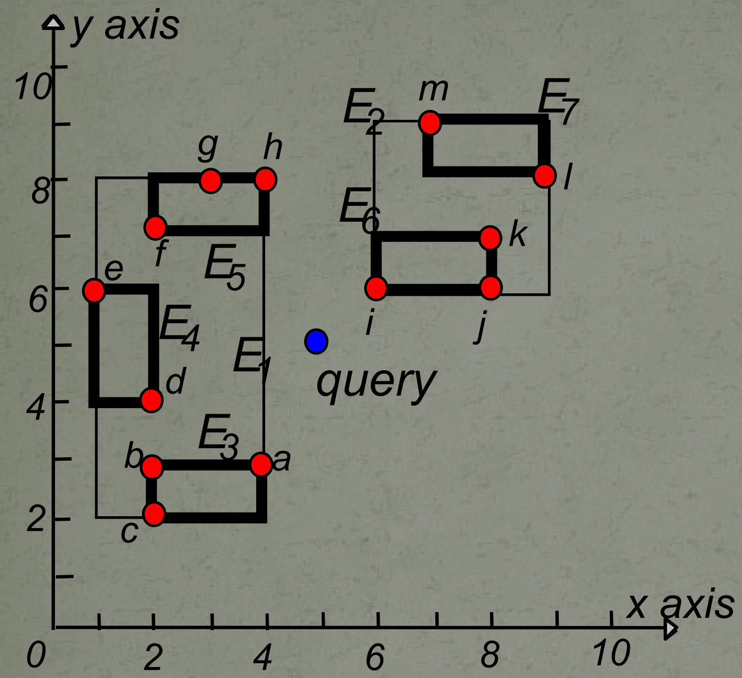
NN Query Example



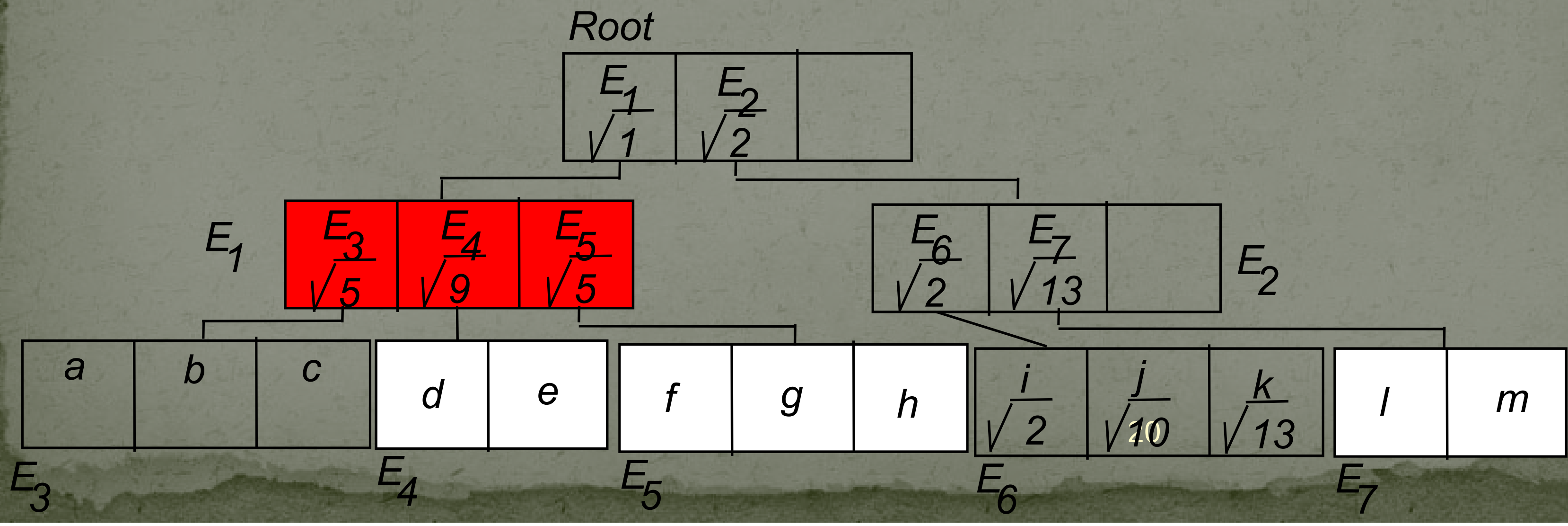
Action	Heap						
Visit Root	$E_1 \sqrt{1}$	$E_2 \sqrt{2}$					



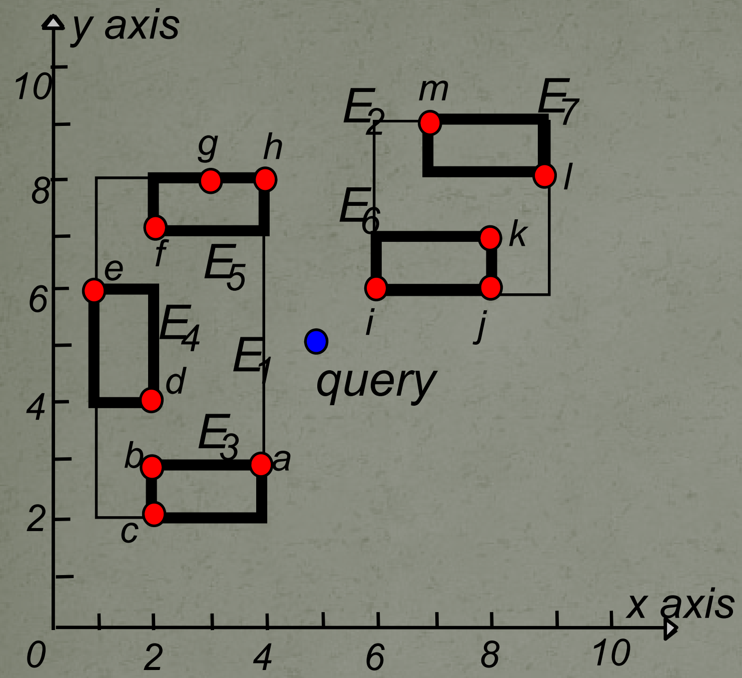
NN Query Example



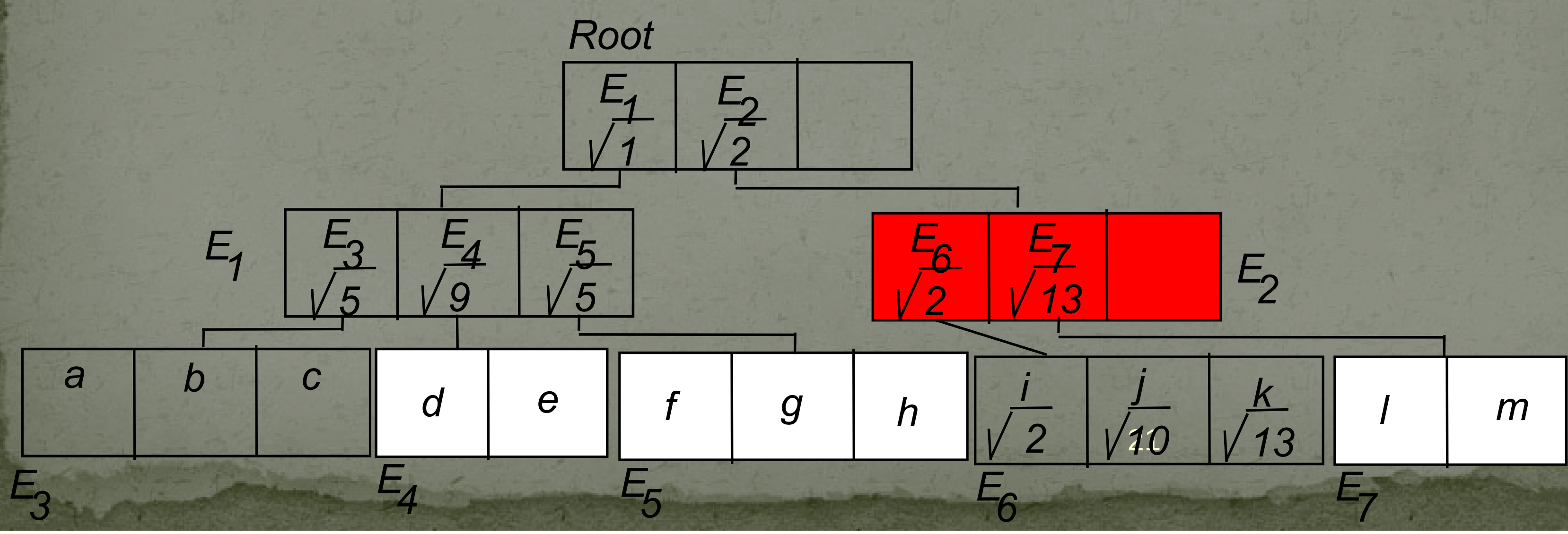
Action	Heap						
Visit Root	$E_1 \sqrt{1}$	$E_2 \sqrt{2}$					
follow E_1	$E_2 \sqrt{2}$	$E_3 \sqrt{5}$	$E_5 \sqrt{5}$	$E_4 \sqrt{9}$			



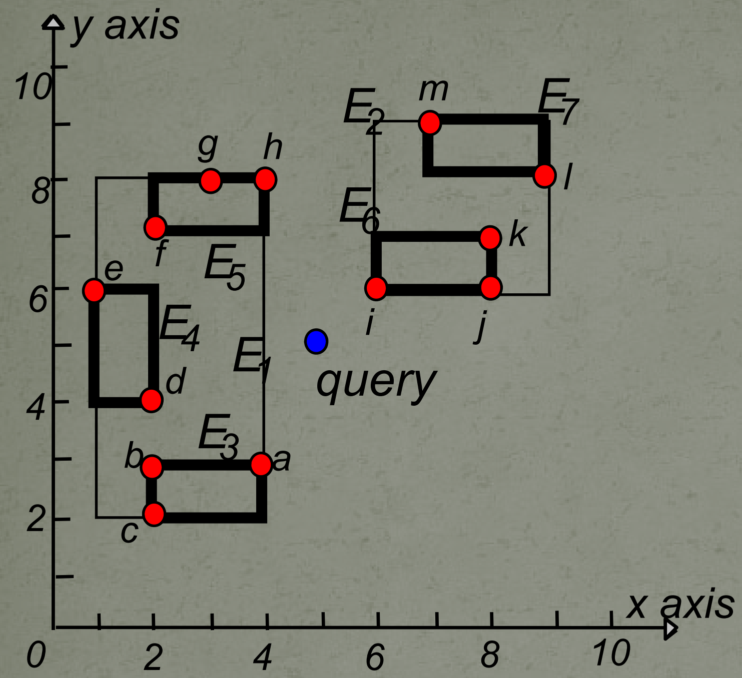
NN Query Example



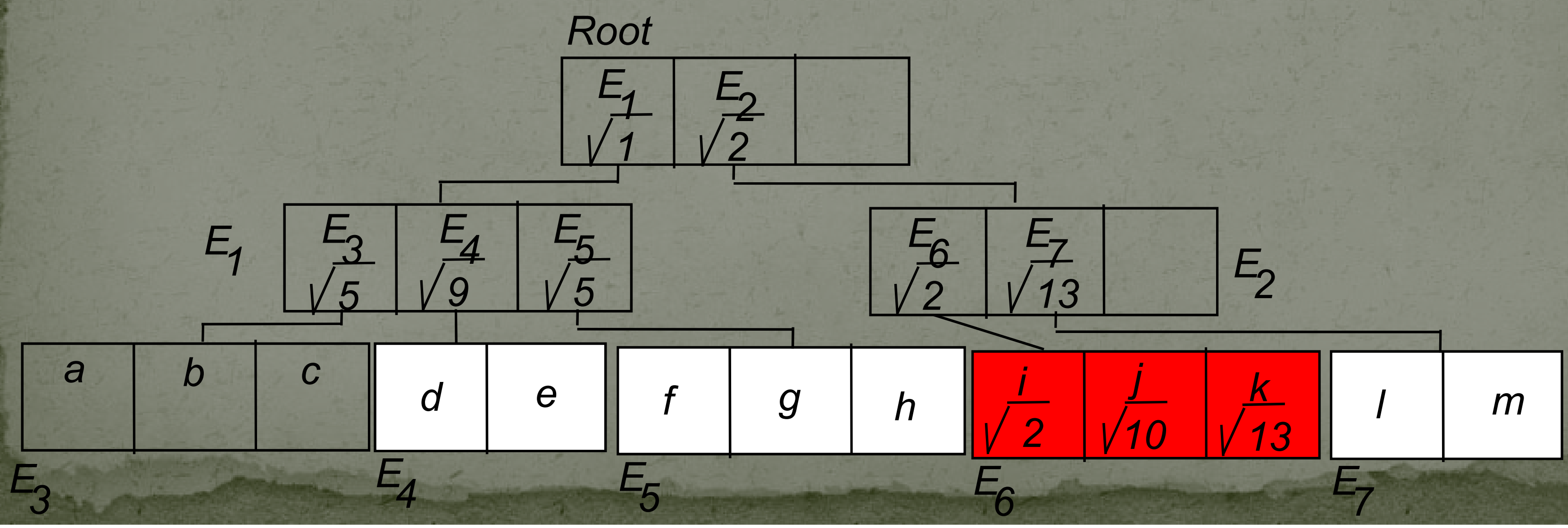
Action	Heap						
Visit Root	$E_1 \sqrt{1}$	$E_2 \sqrt{2}$					
follow E_1	$E_2 \sqrt{2}$	$E_3 \sqrt{5}$	$E_5 \sqrt{5}$	$E_4 \sqrt{9}$			
follow E_2	$E_6 \sqrt{2}$	$E_3 \sqrt{5}$	$E_5 \sqrt{5}$	$E_4 \sqrt{9}$	$E_7 \sqrt{13}$		



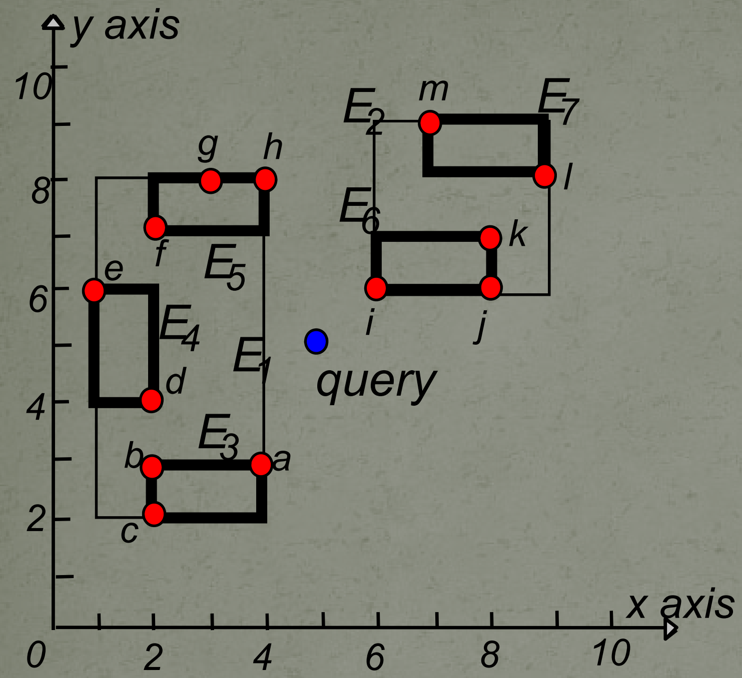
NN Query Example



Action	Heap						
Visit Root	$E_1\sqrt{1}$	$E_2\sqrt{2}$					
follow E_1	$E_2\sqrt{2}$	$E_3\sqrt{5}$	$E_5\sqrt{5}$	$E_4\sqrt{9}$			
follow E_2	$E_6\sqrt{2}$	$E_3\sqrt{5}$	$E_5\sqrt{5}$	$E_4\sqrt{9}$	$E_7\sqrt{13}$		
follow E_6	$i\ \sqrt{2}$	$E_3\sqrt{5}$	$E_5\sqrt{5}$	$E_4\sqrt{9}$	$j\ \sqrt{10}$	$E_7\sqrt{13}$	$k\ \sqrt{13}$

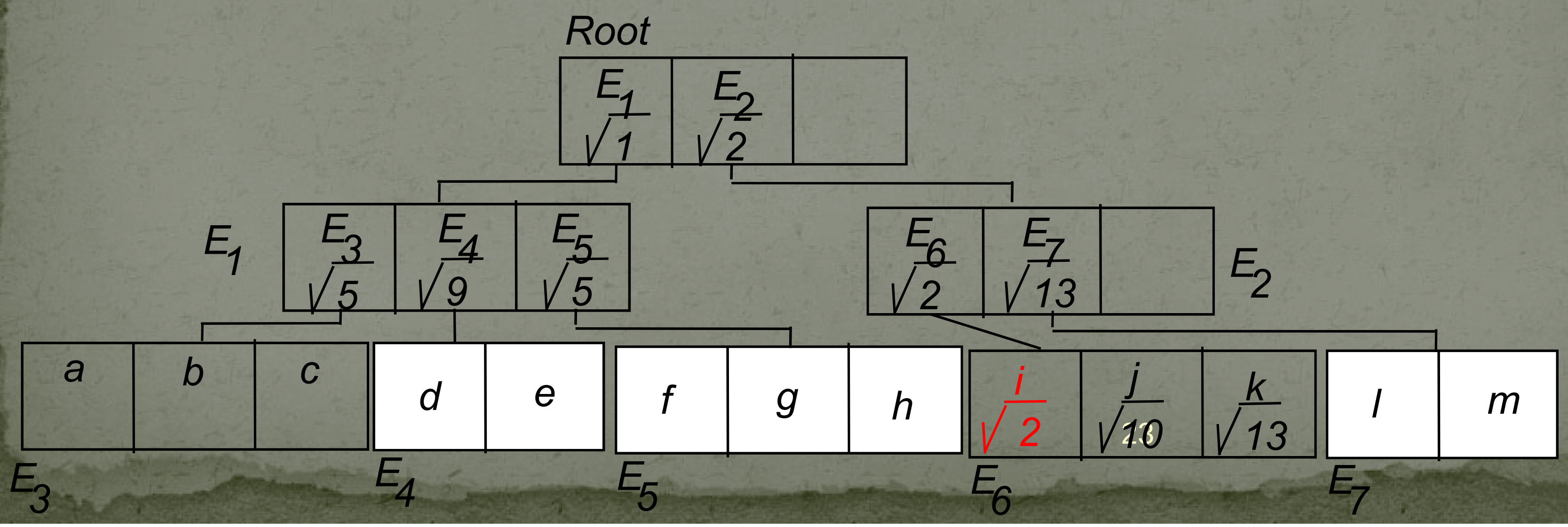


NN Query Example



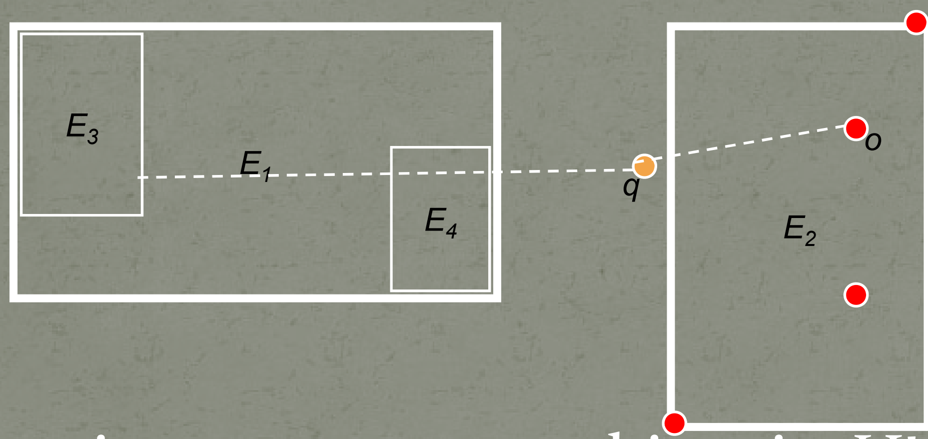
Action	Heap						
Visit Root	E_1 $\sqrt{1}$	E_2 $\sqrt{2}$					
follow E_1	E_2 $\sqrt{2}$	E_3 $\sqrt{5}$	E_5 $\sqrt{5}$	E_4 $\sqrt{9}$			
follow E_2	E_6 $\sqrt{2}$	E_3 $\sqrt{5}$	E_5 $\sqrt{5}$	E_4 $\sqrt{9}$	E_7 $\sqrt{13}$		
follow E_6	i $\sqrt{2}$	E_3 $\sqrt{5}$	E_5 $\sqrt{5}$	E_4 $\sqrt{9}$	j $\sqrt{10}$	E_7 $\sqrt{13}$	k $\sqrt{13}$

Report i and terminate



Pruning 1 in NN Query

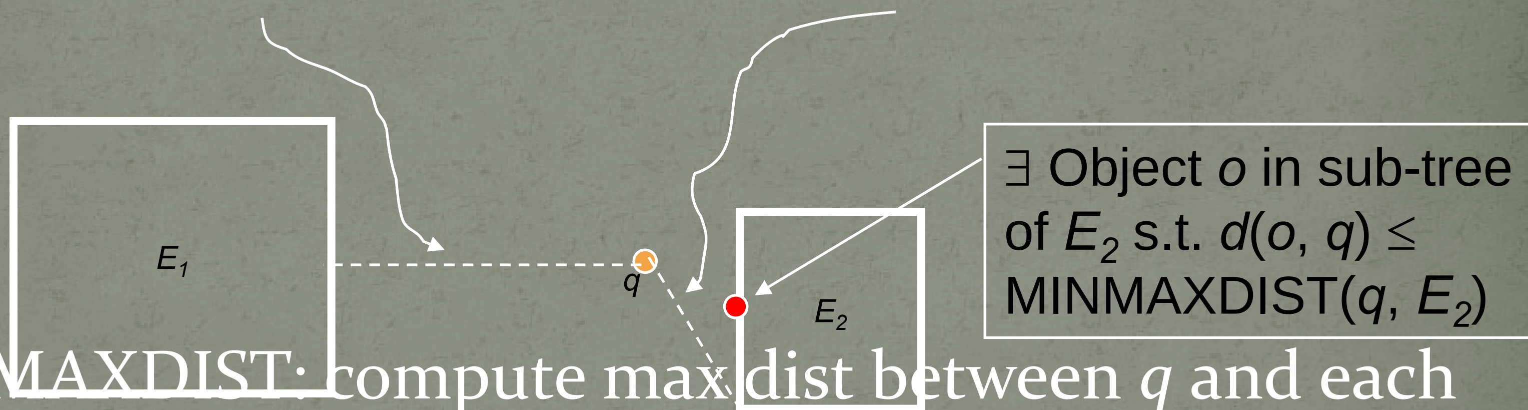
- If we see an object *o*, prune every MBR whose MINDIST > *d*(*o*, *q*).



- Side notice: at most one object in *H*!

Pruning 2 using MINMAXDIST

- Prune even before we see an object!
- Prune E_1 if exists E_2 s.t. MINDIST(q, E_1) > MINMAXDIST(q, E_2).



- MINMAXDIST: compute max dist between q and each edge of E_2, then take min.

NN Full-Blown Algorithm

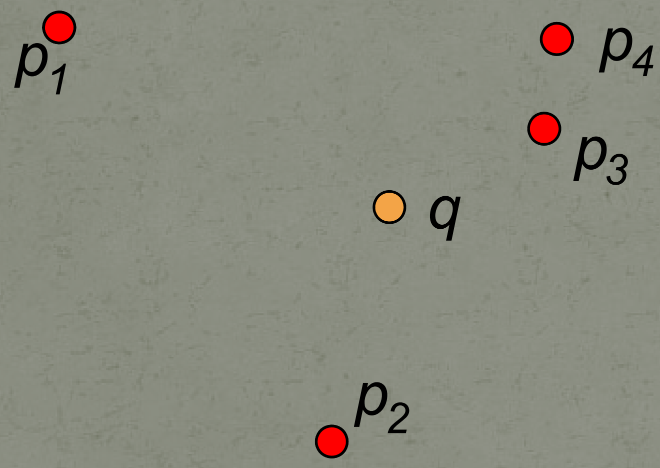
- Keep a heap H of index entries and objects, ordered by MINDIST.
- Initially, H contains the root.
- Set $\delta = +\infty$.
- While $H \neq \phi$
  - Extract the element e with minimum MINDIST.
  - If it is an object, return it as NN.
  - For every entry *se* in PAGE(e) whose MINDIST $\leq \delta$
    - Insert *se* into H.
    - Decrease δ to MINMAXDIST(q, *se*) if possible.
- End while

Content

- The R-tree
  - Range Query
  - Aggregation Query
- NN Query
- **RNN Query**
- Closest Pair Query
- Close Pair Query
- Skyline Query

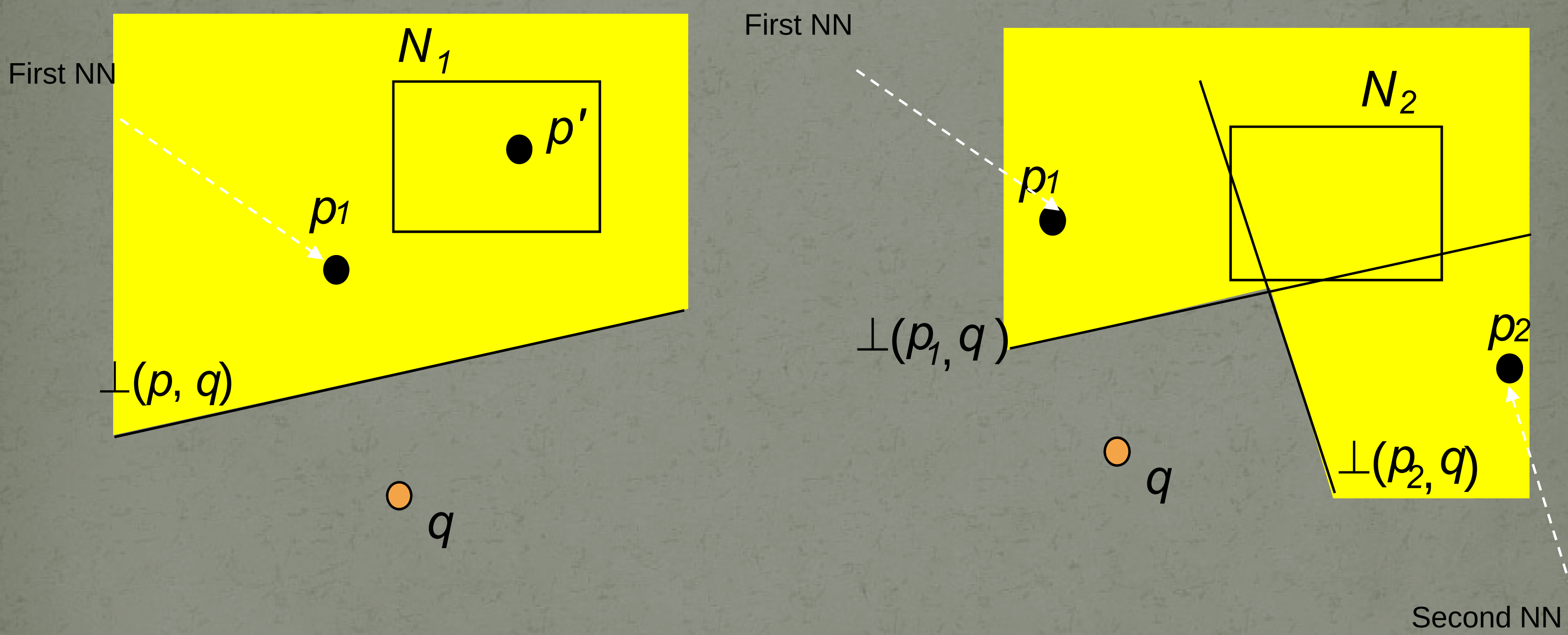
Reverse NN: Definition

- Given a set of points, and a query location *q*.
- Find the points whose NN is *q*.



- RNN(q)={p_1, p_2}, NN(q)= p_3.

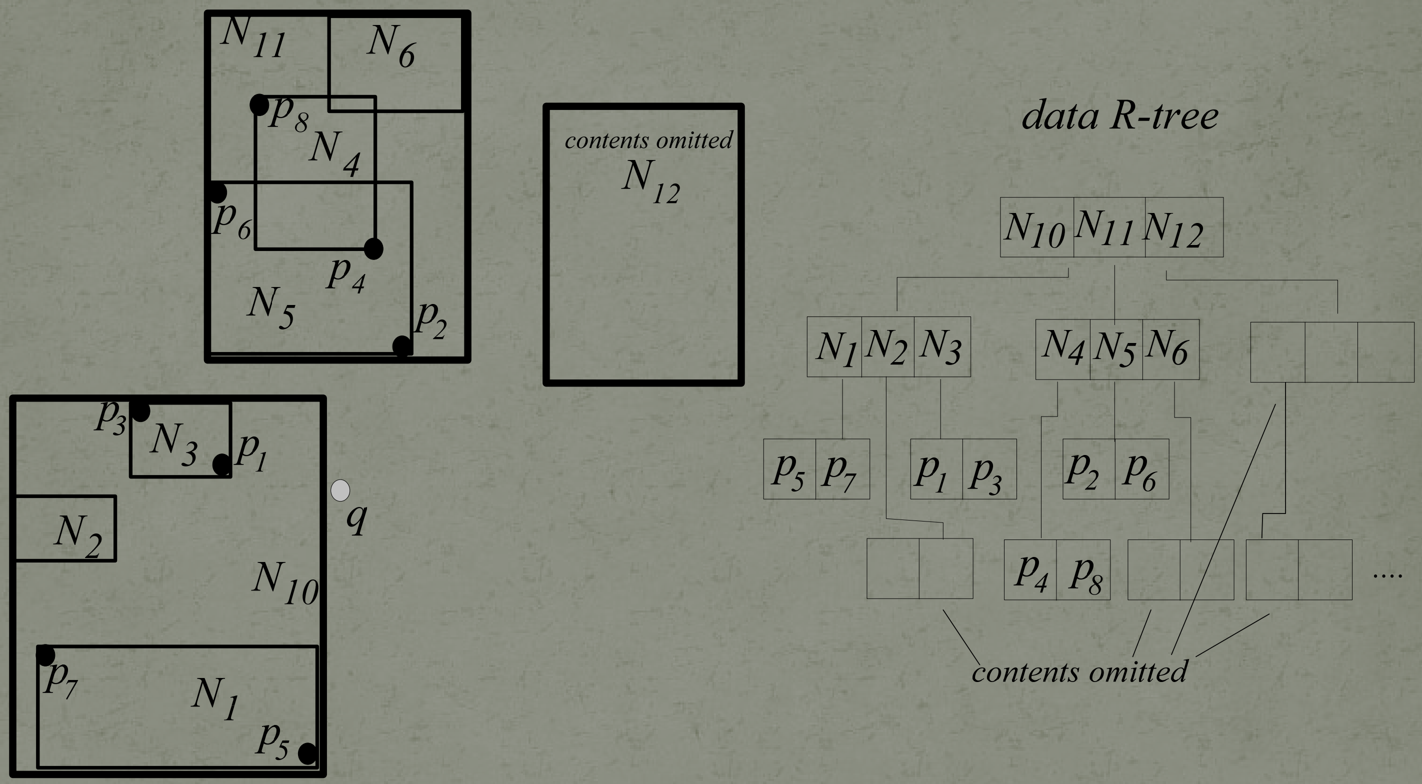
Half-plane pruning



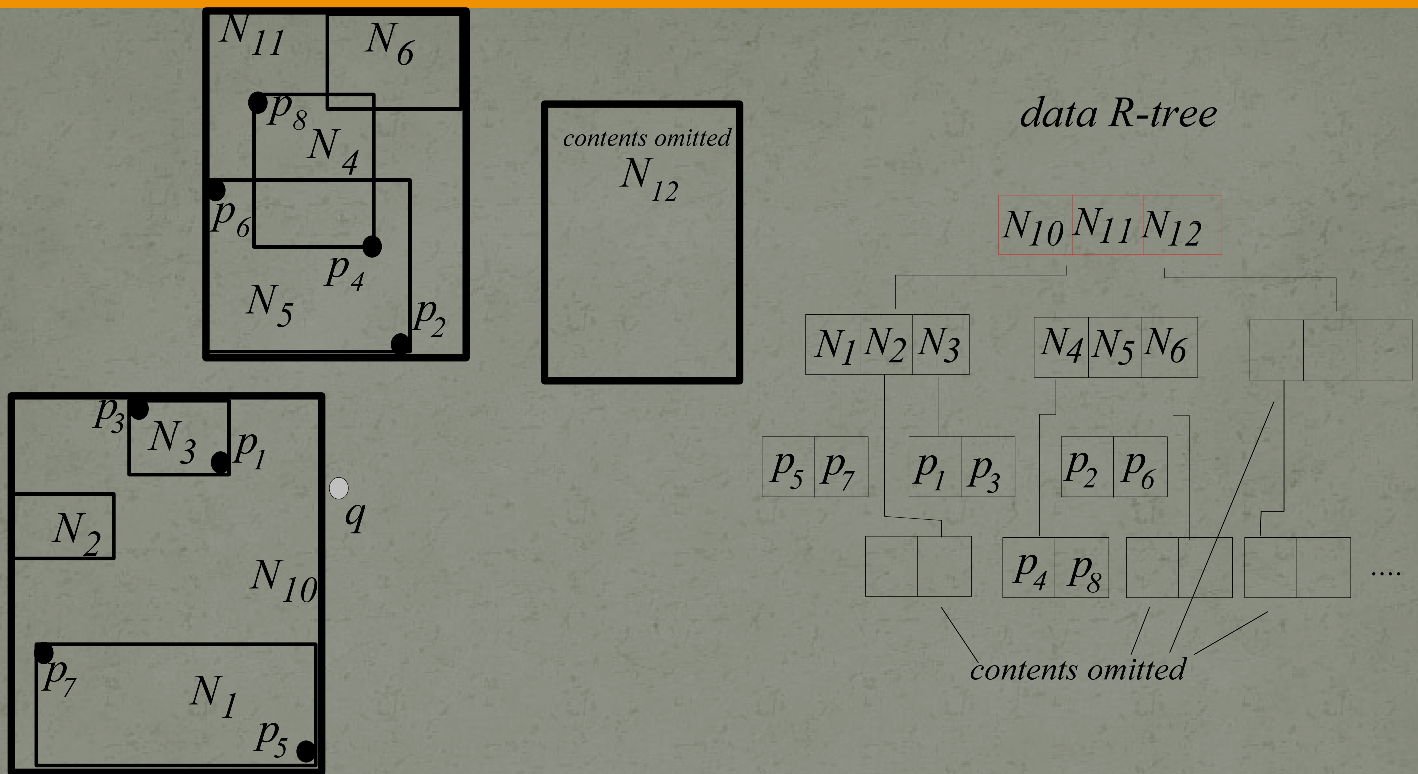
(TPL) Algorithm

- Two logical steps:
  - Filter step: Find the set S_{cnd} of candidate points
    - Find NN;
    - Prune space;
    - Find NN in unpruned space;
    - …
    - Till no more object left.
  - Refinement step: eliminate false positives
    - For each point p in S_{cnd}, check whether its NN is not q.
- The two steps are combined in a single tree traversal, by keeping all pruned MBRs/objects in S_{rfn}.

Example

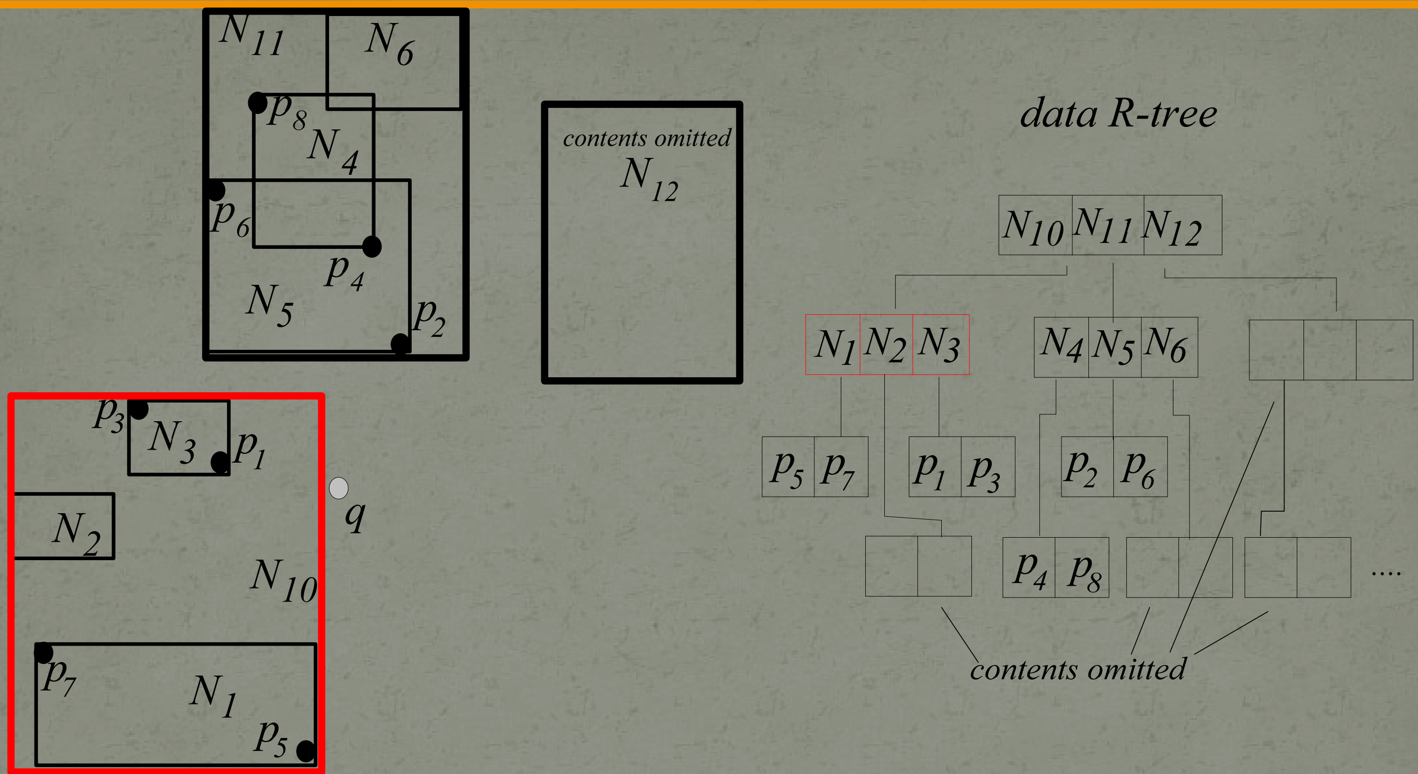

N11
N6
p8
N4
p6
p4
N5
p2
contents omitted
N12
p3
N3
p1
q
N2
N10
p7
N1
p5
data R-tree
N10 N11 N12
N1 N2 N3
N4 N5 N6
p5 p7
p1 p3
p2 p6
p4 p8
....
contents omitted

Example



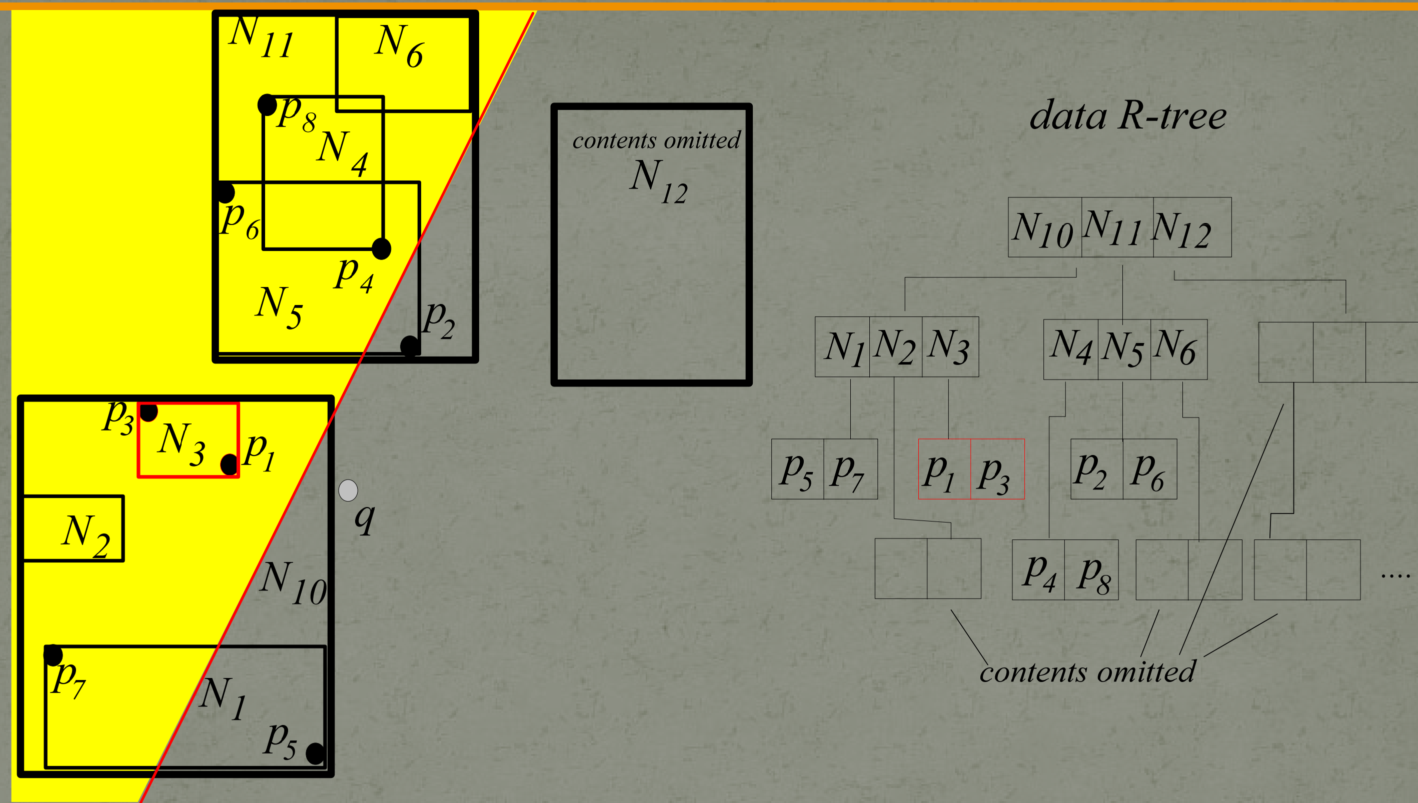
Action	Heap	S_{cnd}	S_{rfn}
visit root	$\{N_{10},N_{11},N_{12}\}$	$\varnothing$	$\varnothing$

Example



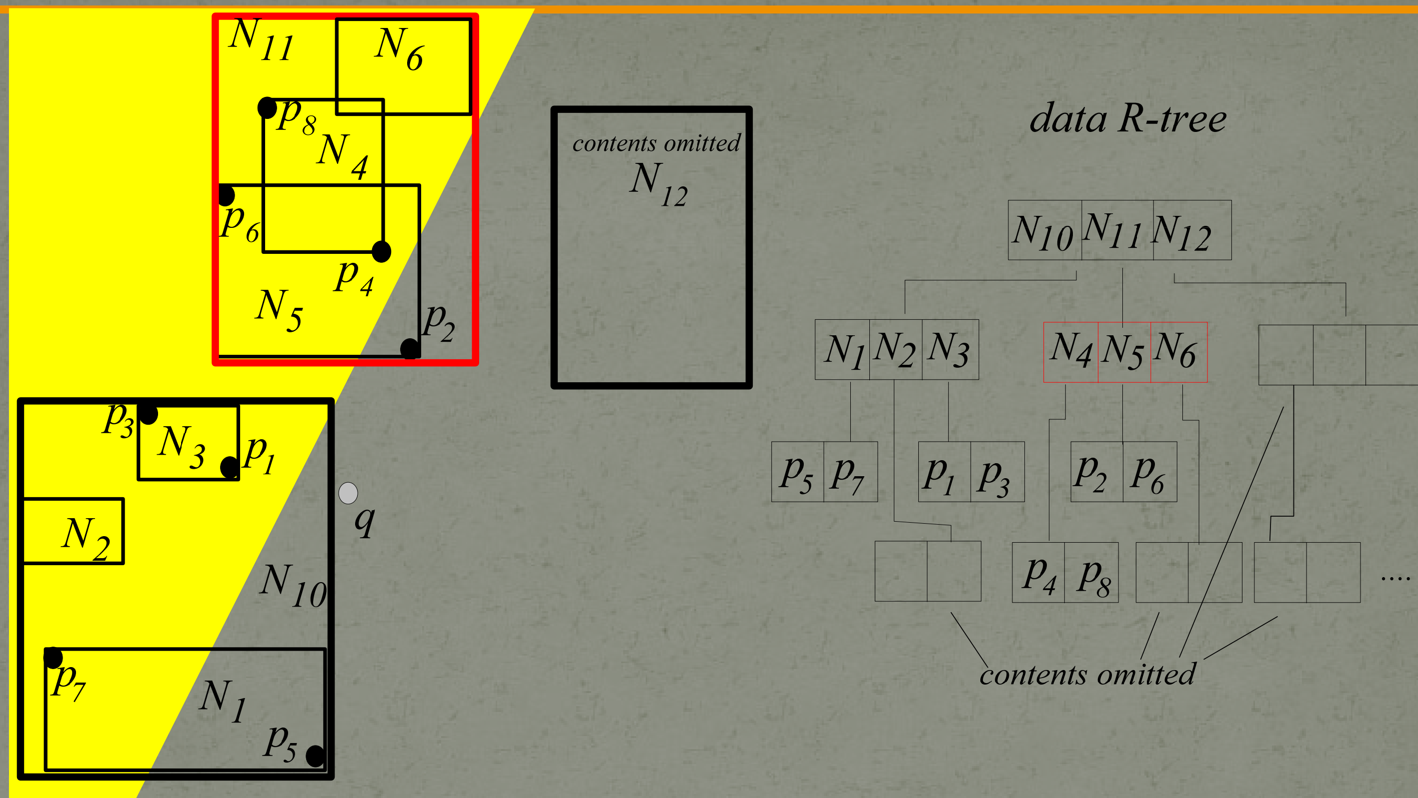
Action	Heap	S_{cnd}	S_{rfn}
visit N_{10}	$\{N_3,N_{11},N_2,N_1,N_{12}\}$	$\varnothing$	$\varnothing$

Example



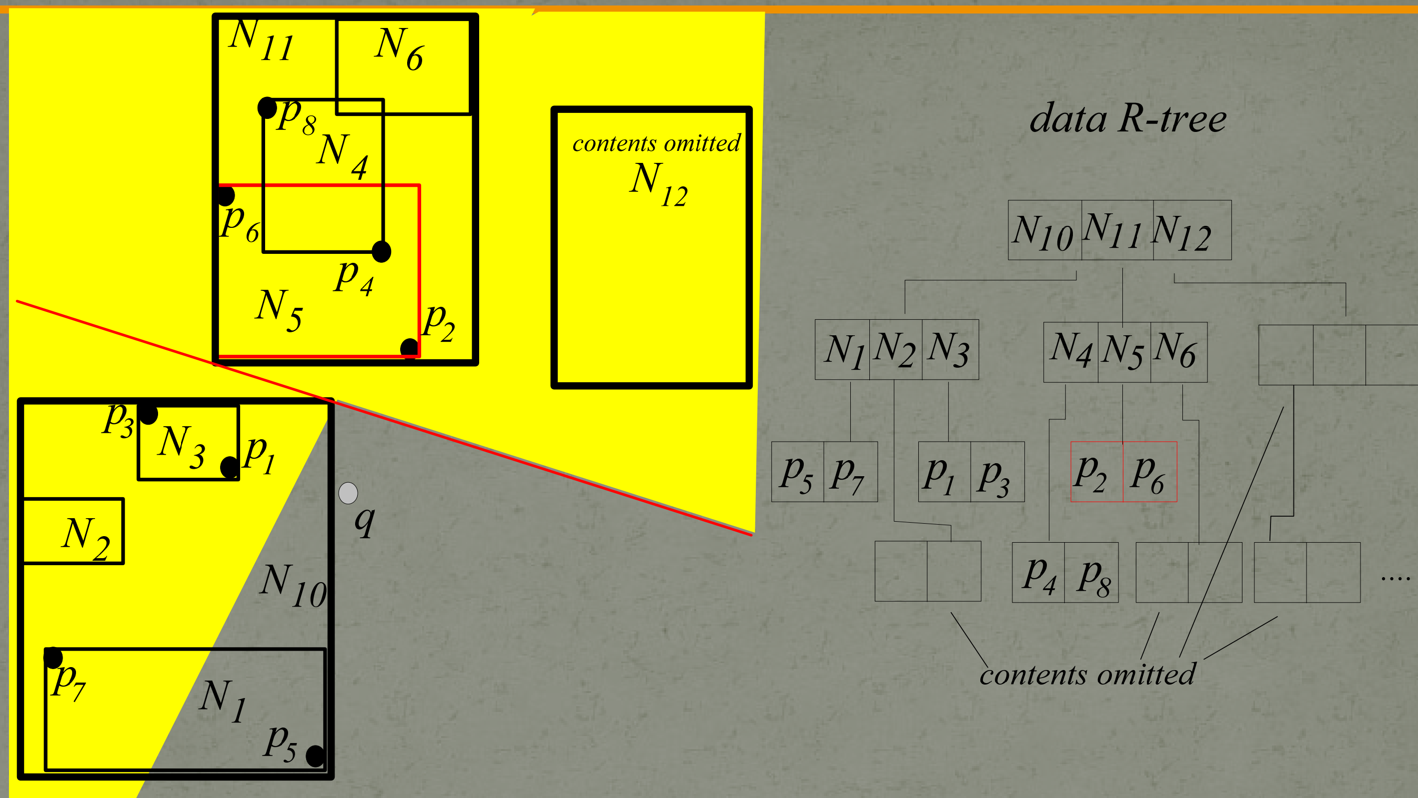
Action	**Heap**	S_{cnd}	S_{rfn}
visit N_3	$\{N_{11}, N_2, N_1, N_{12}\}$	$\{p_1\}$	$\{p_3\}$

Example



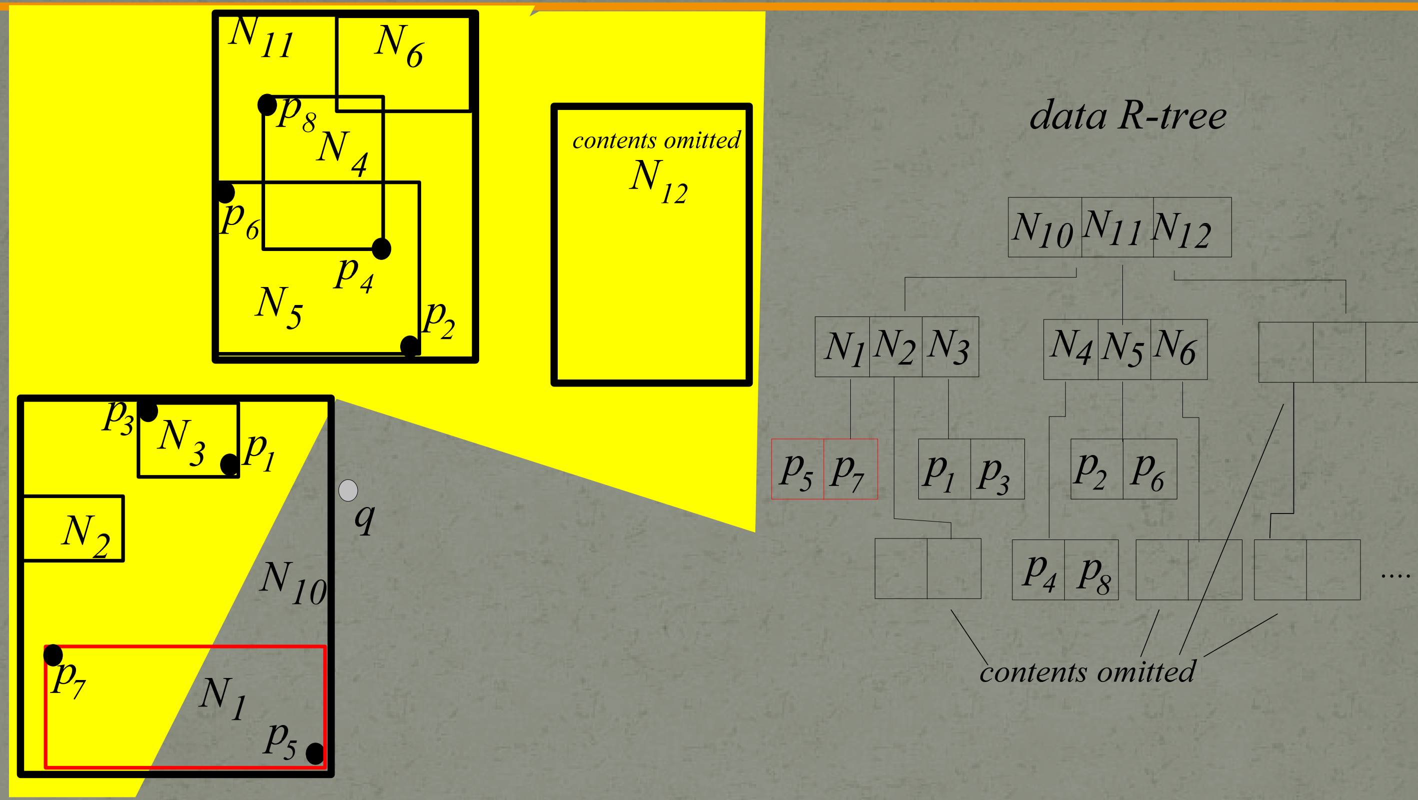
Action	**Heap**	S_{cnd}	S_{rfn}
visit N_{11}	$\{N_5,N_2,N_1,N_{12}\}$	$\{p_1\}$	$\{p_3,N_4,N_6\}$

Example



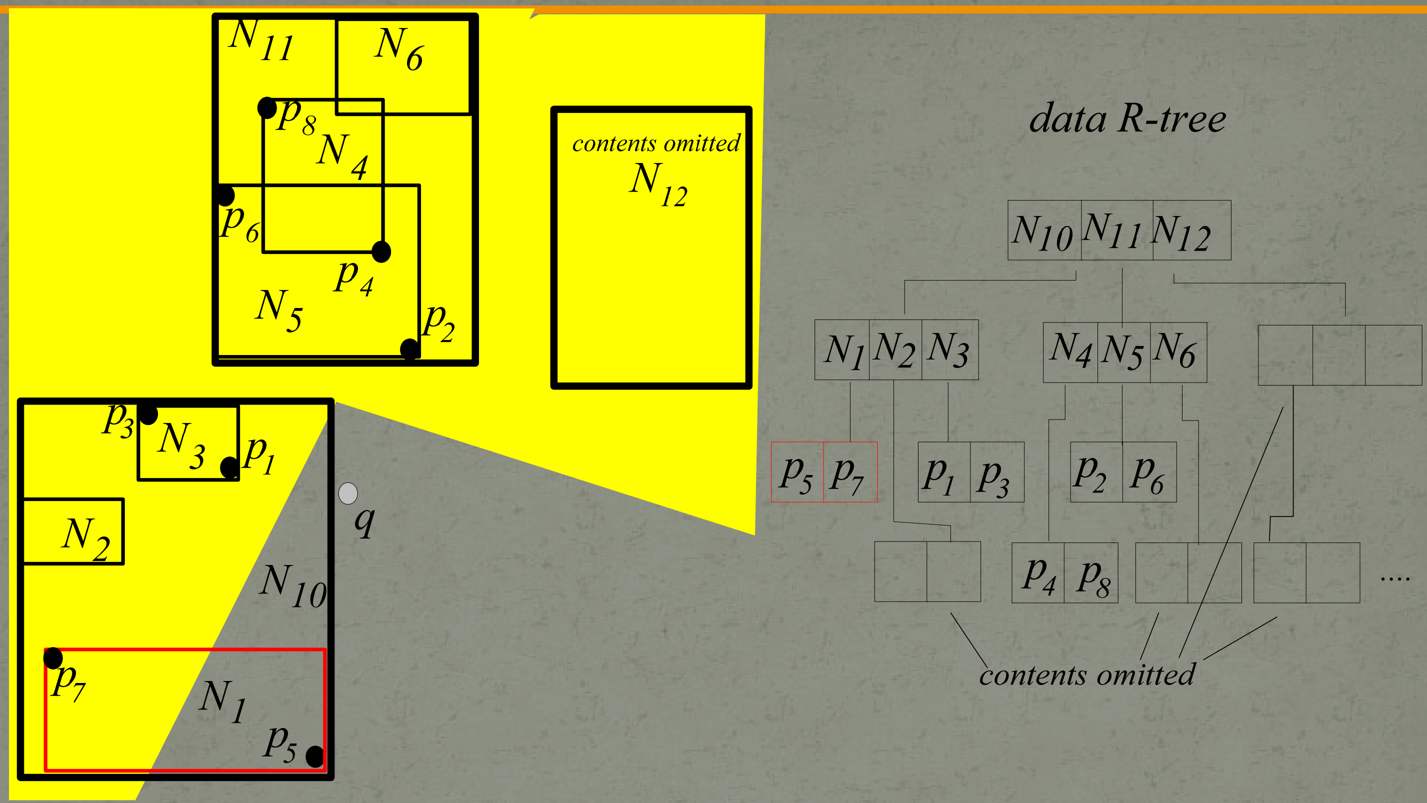
Action	**Heap**	**S_{cnd}**	**S_{rfn}**
visit N_5	$\{N_2, N_1, N_{12}\}$	$\{p_1, p_2\}$	$\{p_3, N_4, N_6, p_6\}$

Example (end of filter step)



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

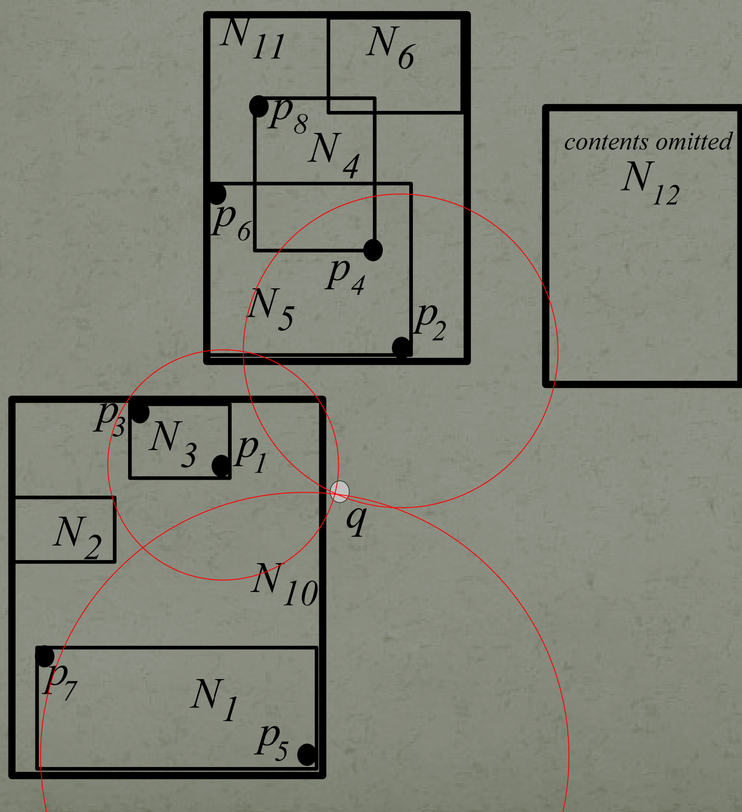
Example (end of filter step)

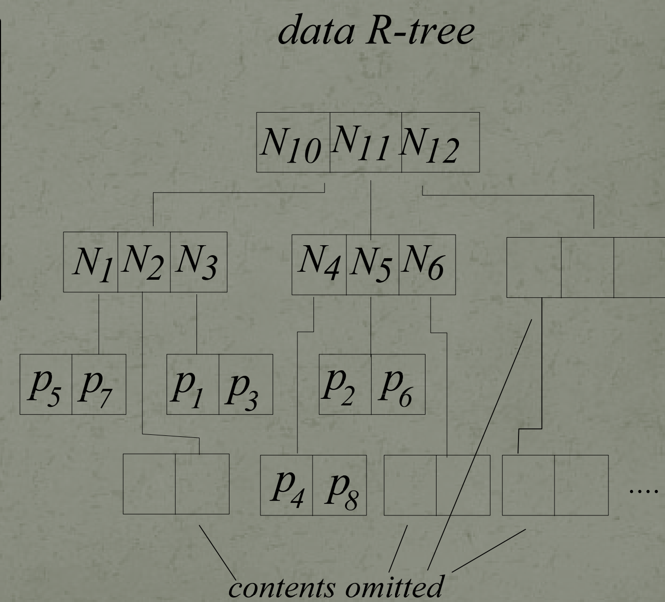


Action	**Heap**	S_{cnd}	S_{rfn}
	$\varnothing$	$\{p_1,p_2,p_5\}$	$\{p_3,N_4,N_6,p_6,N_2,p_7,N_{12}\}$

Example (refinement)

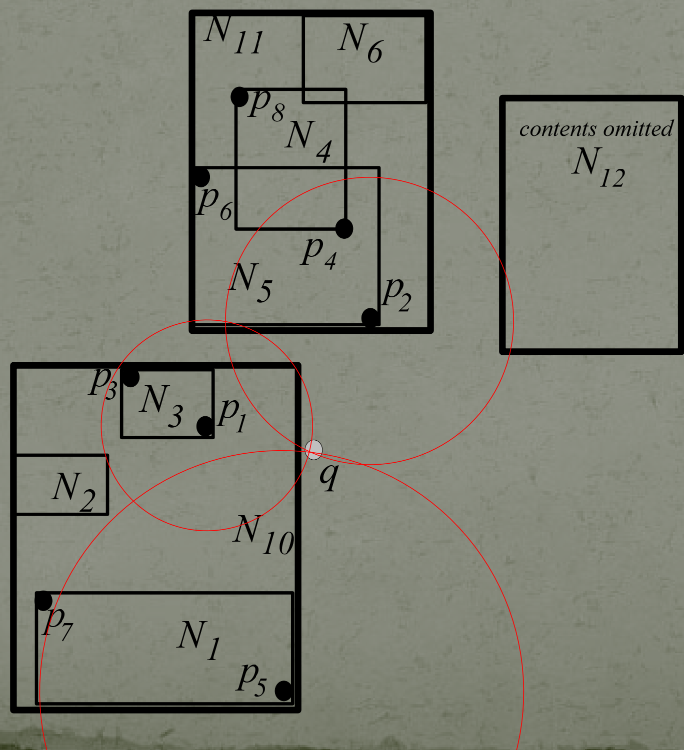
Action	S_{cnd}	S_{rfn}
	$\{p_1, p_2, p_5\}$	$\{p_3, p_6, p_7, N_4, N_6, N_2, N_{12}\}$
invalidate p_1	$\{$~~p_1~~$, p_2, p_5\}$	$\{N_4, N_6, N_2, N_{12}\}$
remove N_6, N_2	$\{$~~p_1~~$, p_2, p_5\}$	$\{N_4, N_{12}\}$

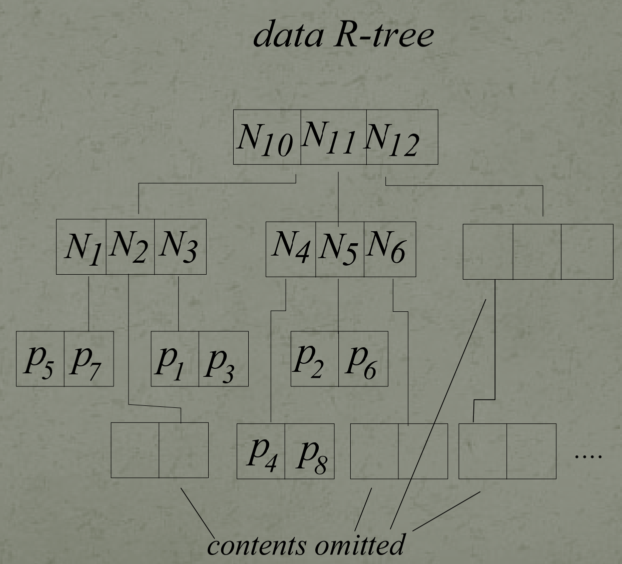




Example (refinement)

Action	S_{cnd}	S_{rfn}
$_2$	$\{\cancel{p_1}, p_2, p_5\}$	$\{N_4, N_{12}\}$
access N_4	$\{\cancel{p_1}, p_2, p_5\}$	$\{p_4, p_8, N_{12}\}$
invalidate p_2	$\{\cancel{p_1}, \cancel{p_2}, p_5\}$	$\{N_{12}\}$
remove N_{12}	$\{\cancel{p_1}, \cancel{p_2}, p_5\}$	$\varnothing$



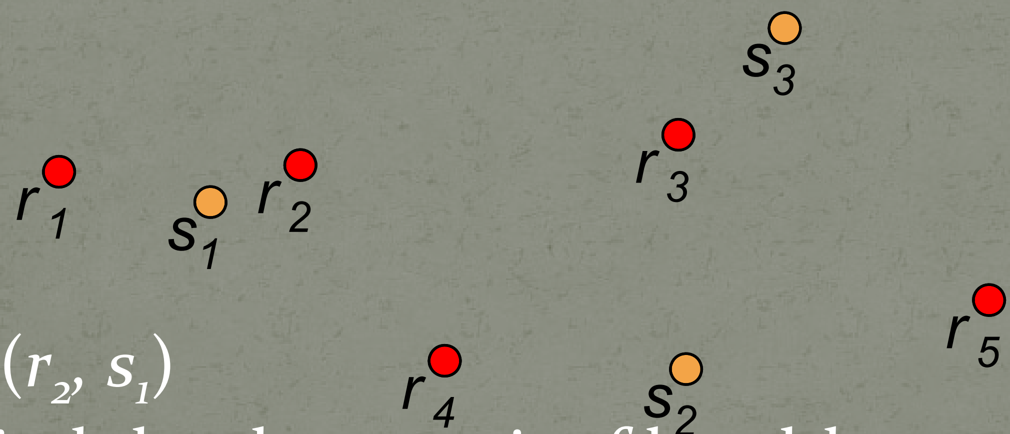


Content

- The R-tree
  - Range Query
  - Aggregation Query
- NN Query
- RNN Query
- **Closest Pair Query**
- Close Pair Query
- Skyline Query

Closest Pair (CP) Query

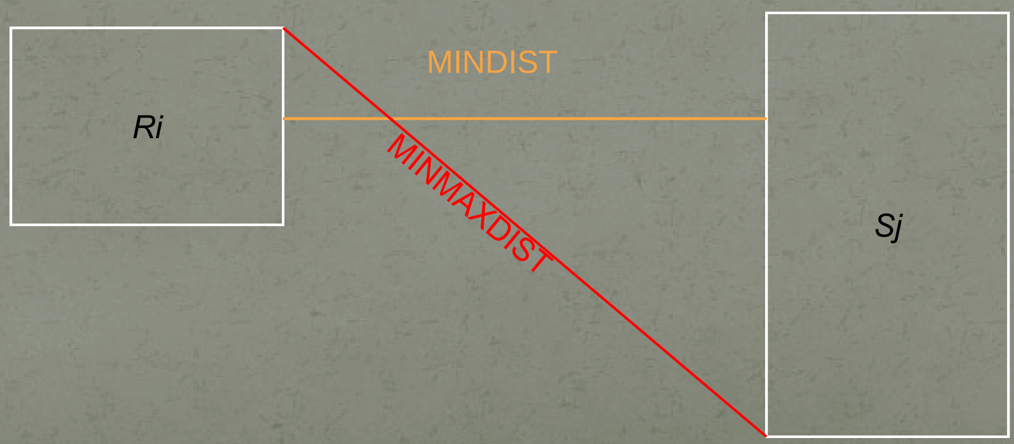
- Given two sets of objects *R* and *S*,
- Find the pair of objects ($r \in R$, $s \in S$) with minimum distance.



- CP = (r_2, s_1)
- E.g. find the closest pair of hotel-bar.

CP Solution Idea

- Assume *R* and *S* are indexed by R-trees with same height.
- Similar to the NN query algorithm.
- MINDIST, MINMAXDIST for a pair of MBRs:



CP Basic Algorithm

- Keep a heap H of pairs of index entries and pairs of objects, ordered by MINDIST.
- Initially, H contains the pair of roots.
- While $H \neq \phi$
  - Extract the pair (e_R, e_S) with minimum MINDIST.
  - If it is a pair of objects, return it as CP.
  - For every entry se_R in PAGE(e_R) and every entry se_S in PAGE(e_S)
    - Insert(e_R, e_S) into H.
- End while

CP Full-Blown Algorithm

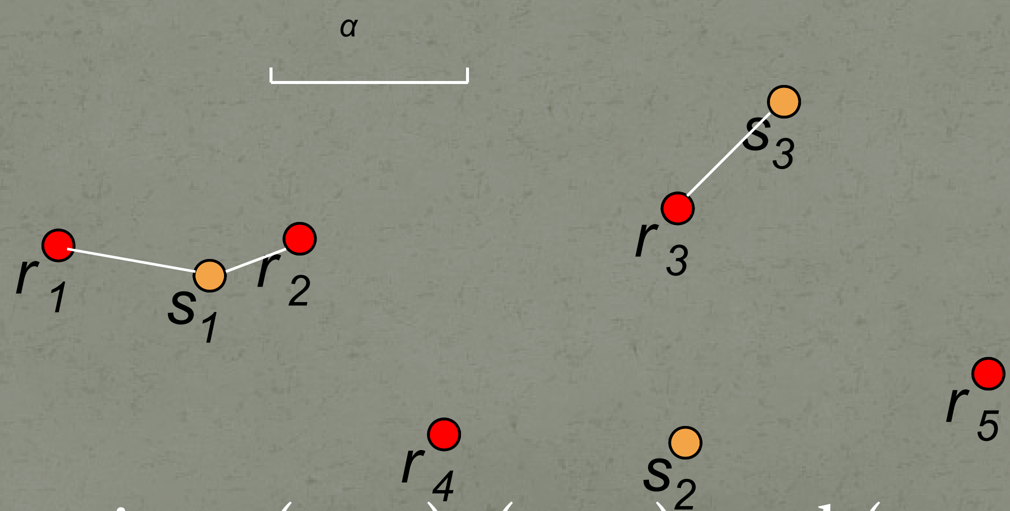
- Keep a priority queue H of pairs of index entries and pairs of objects, ordered by MINDIST.
- Initially, H contains the pair of roots.
- Set $\delta = +\infty$.
- While $H \neq \phi$
  - Extract the pair (e_R, e_S) with minimum MINDIST.
  - If it is a pair of objects, return it as CP.
  - For every entry se_R in PAGE(e_R) and every entry se_S in PAGE(e_S) whose MINDIST≤δ
    - Insert(se_R, se_S) into H.
    - Decrease δ to MINMAXDIST(se_R, se_S) if possible.
- End while

Content

- The R-tree
  - Range Query
  - Aggregation Query
- NN Query
- RNN Query
- Closest Pair Query
- **Close Pair Query**
- Skyline Query

Close Pair Query

- Given two sets of objects R and S, plus a threshold α,
- Find every pair of objects ($r \in R$, $s \in S$) with distance $<\alpha$.



- Close pairs = (r_1, s_1), (r_2, s_1), and (r_3, s_3).

Close Pair Solution Idea

- Observation: if $d(r, s) < \alpha$, $\forall mbr_R$, mbr_S that contain r and s, respectively, we have: MINDIST(mbr_R, mbr_S) $< \alpha$.
- Solution idea:
  - start with the pair of root nodes,
  - Join pairs of index entries whose MINDIST $< \alpha$,
  - Till we reach leaf level.

Close Pair Algorithm

- Push the pair of root nodes into *stack*.
- While *stack* $\neq \phi$
  - Pop a pair (e_R, e_S) from *stack*.
  - For every entry se_R in PAGE(e_R) and se_S in PAGE(e_S) where MINDIST(se_R, se_S) $<\alpha$
    - Push (se_R, se_S) into stack if se_R is an index entry;
    - Otherwise report (se_R, se_S) as one close pair.
- End while

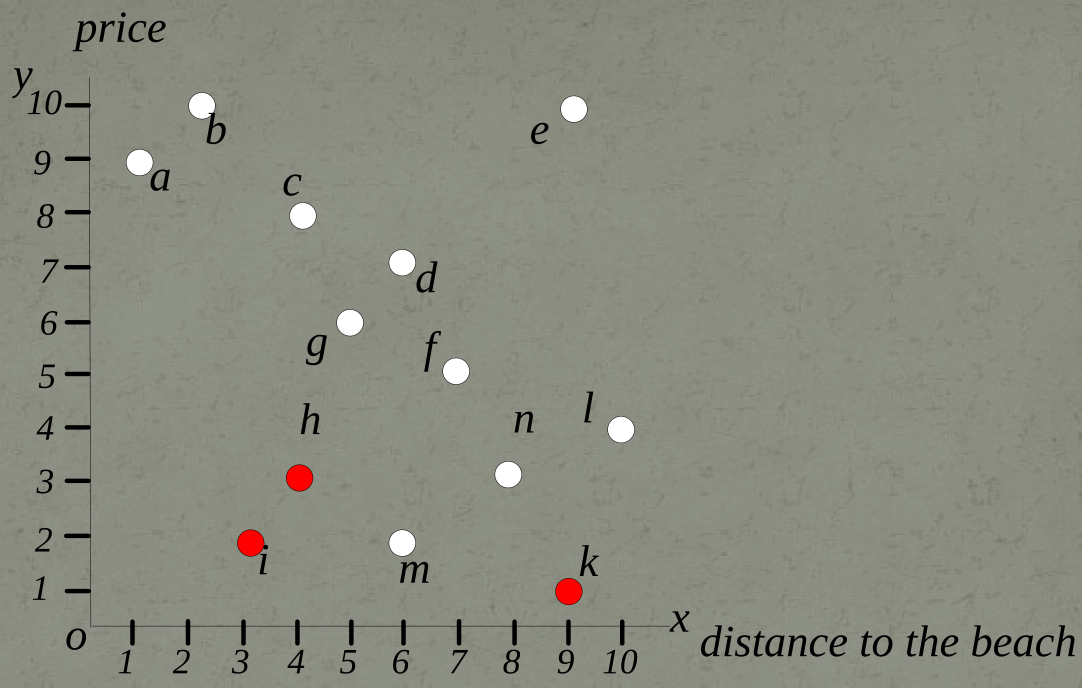
Content

- The R-tree
  - Range Query
  - Aggregation Query
- NN Query
- RNN Query
- Closest Pair Query
- Close Pair Query
- **Skyline Query**

Skyline of Manhattan

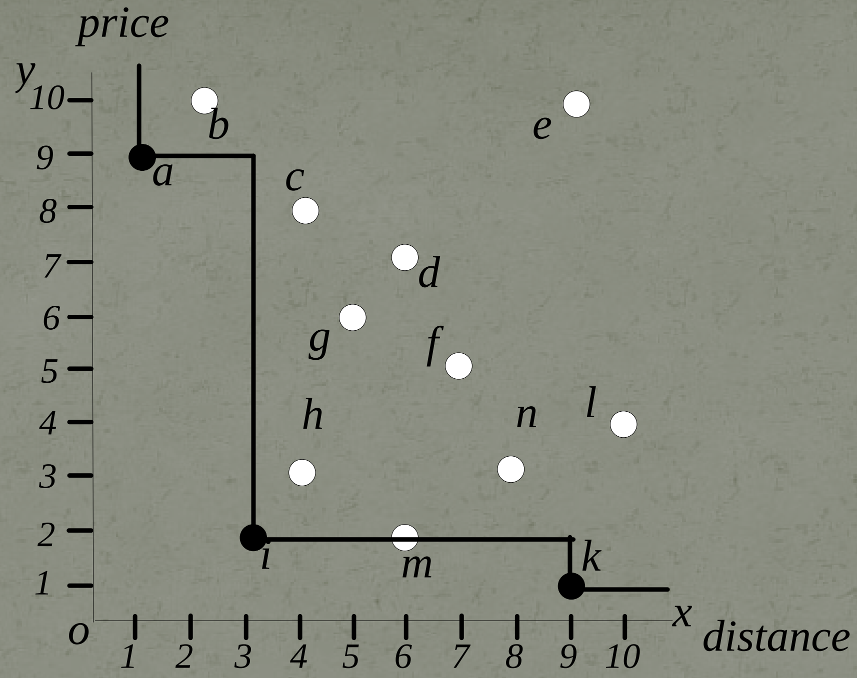
- Which buildings can we see?
  - *Higher or nearer*

Finding A Hotel Close to the Beach



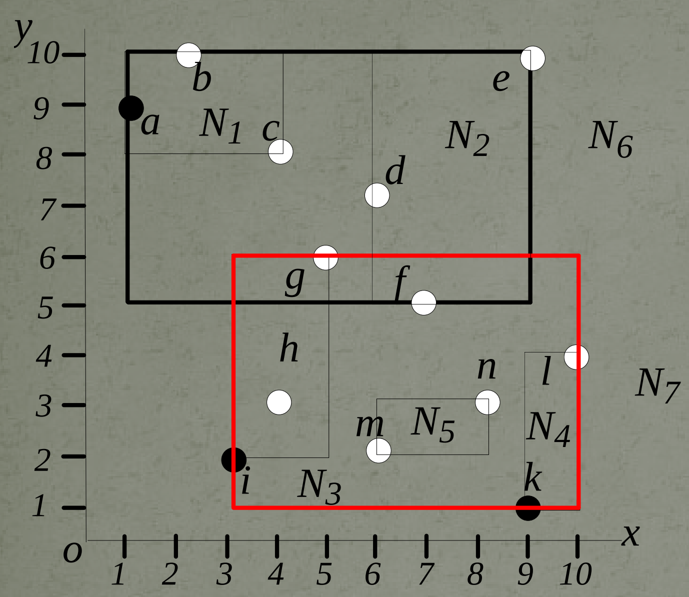
- Which one is better?
  - *i* or *h*? (*i*, because its price and distance dominate those of *h*)
  - *i* or *k*?

Skyline Queries

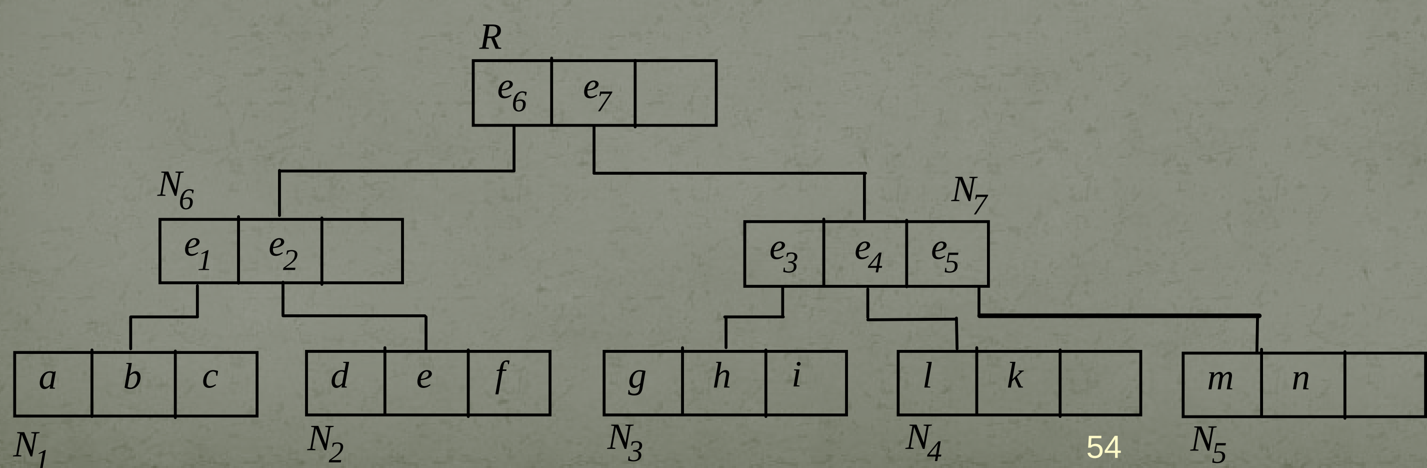


- Retrieve points not dominated by any other point.

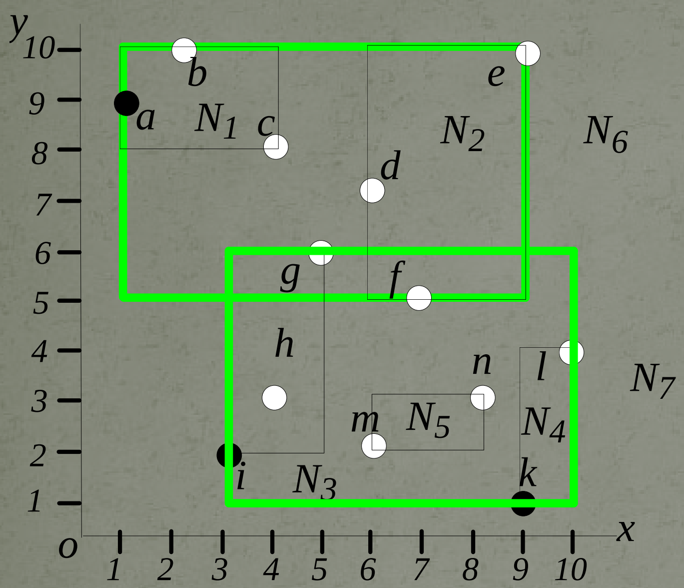
Branched and Bound Skyline (BBS)



- Assume all points are indexed in an R-tree.
- mindist(MBR) = the L_1 distance between its lower-left corner and the origin.

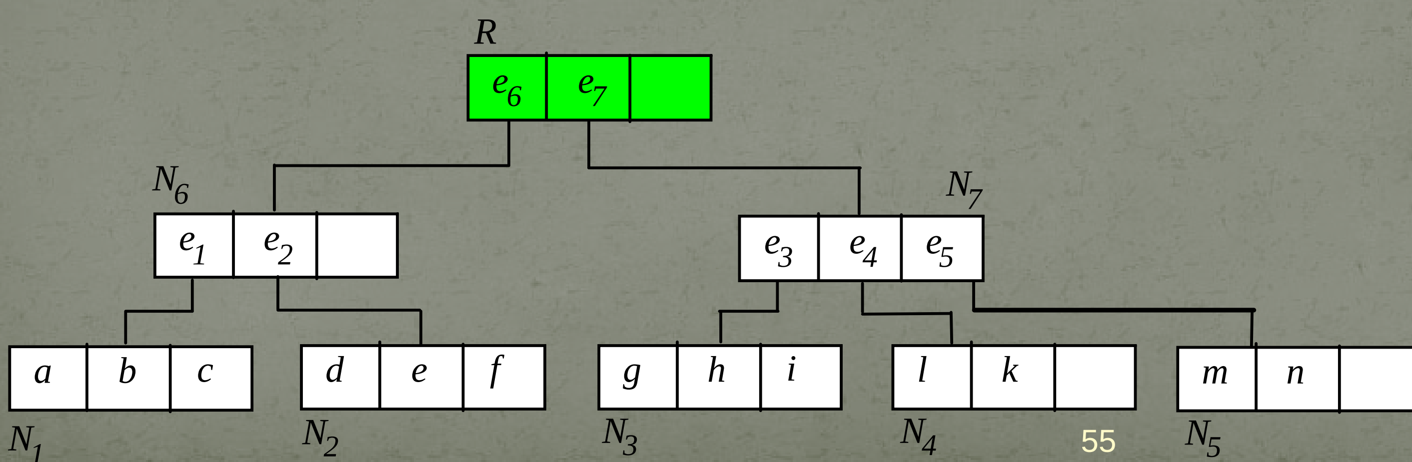


Branched and Bound Skyline (BBS)

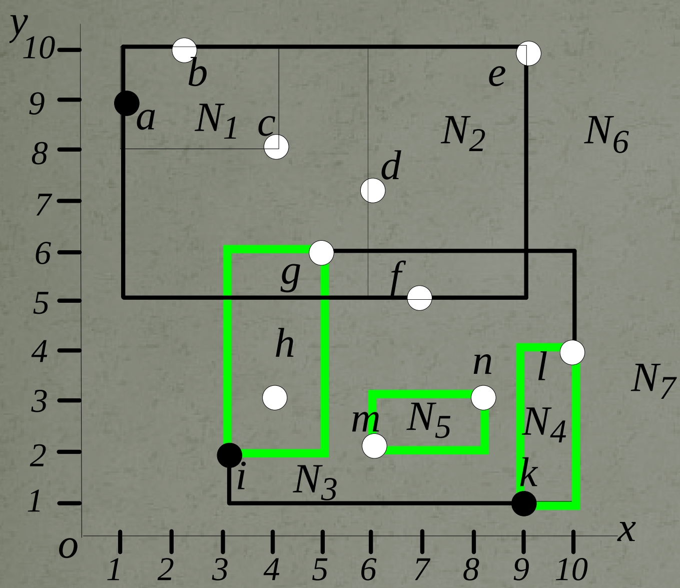


- Each heap entry keeps the mindist of the MBR.

action	heap contents	*S*
access root	<e_7,4><e_6,6>	∅

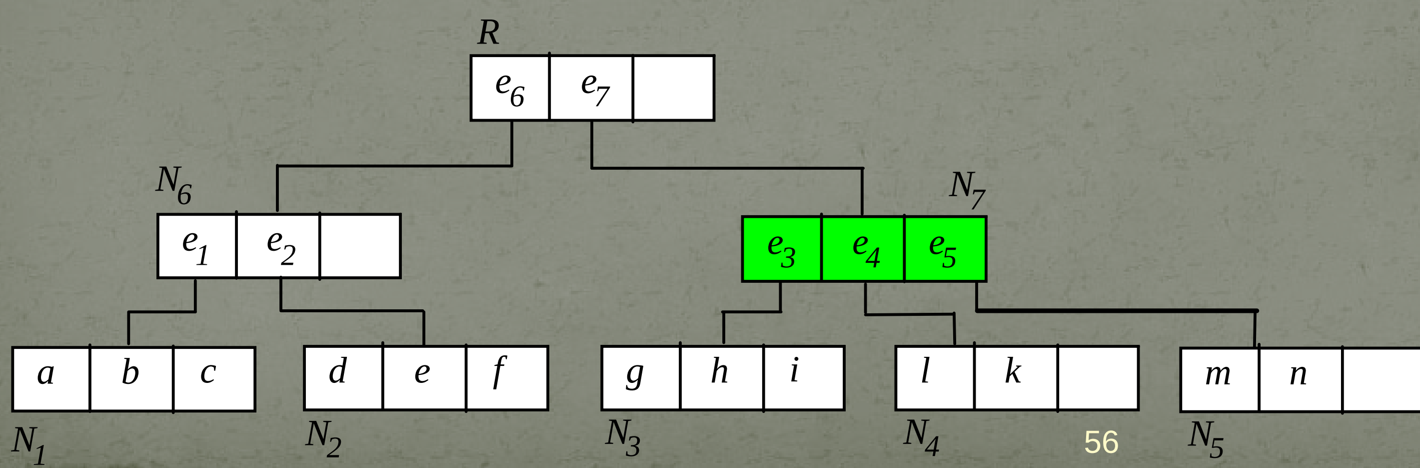


Example of BBS

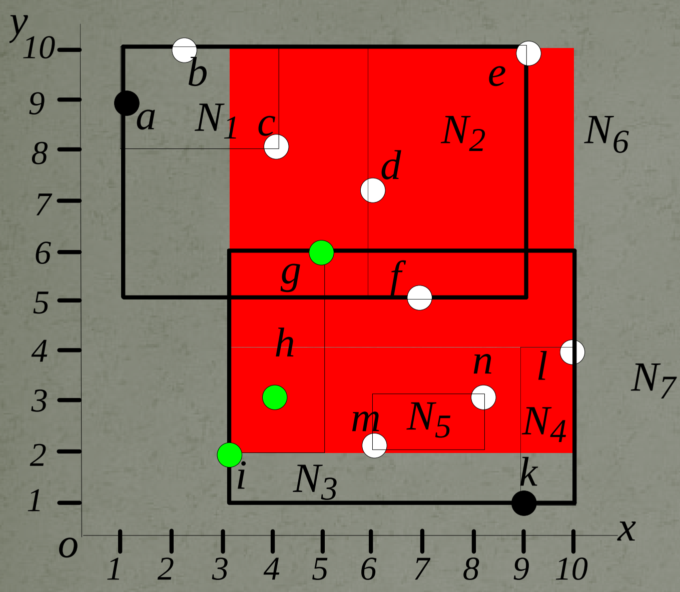


- Process entries in ascending order of their mindists.

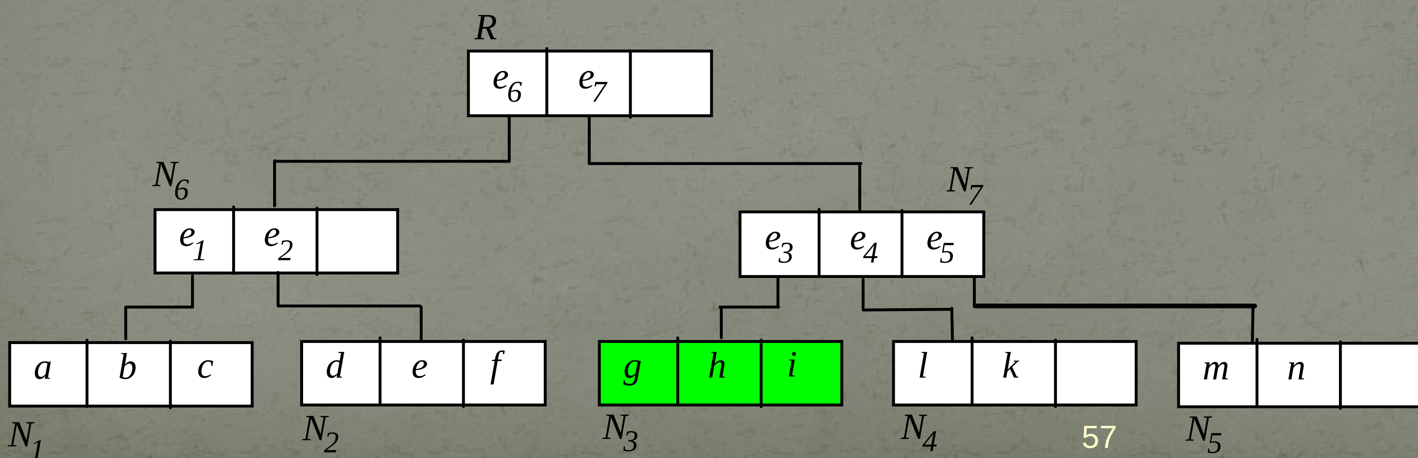
action	**heap contents**	***S***
access root	$<e_7,4><e_6,6>$	∅
expand e_7	$<e_3,5><e_6,6><e_5,8><e_4,10>$	∅



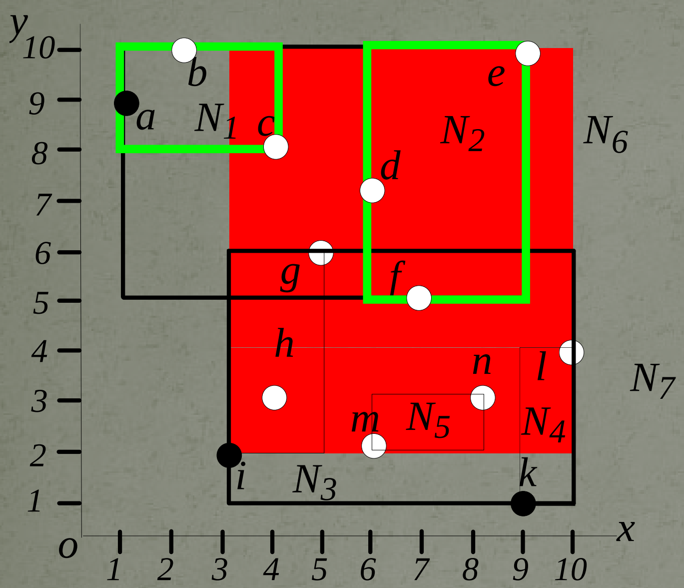
Example of BBS



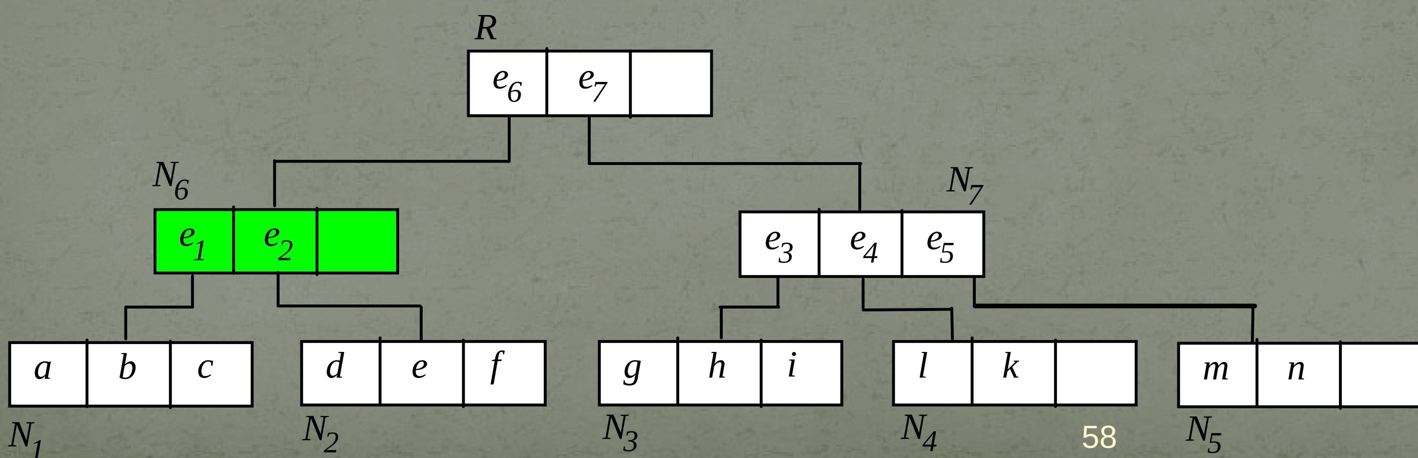
action	heap contents	S
access root	$<e_7,4><e_6,6>$	$\varnothing$
expand e_7	$<e_3,5><e_6,6><e_5,8><e_4,10>$	$\varnothing$
expand e_3	**$<i,5>$**$<e_6,6><e_5,8><e_4,10>$	$\{i\}$



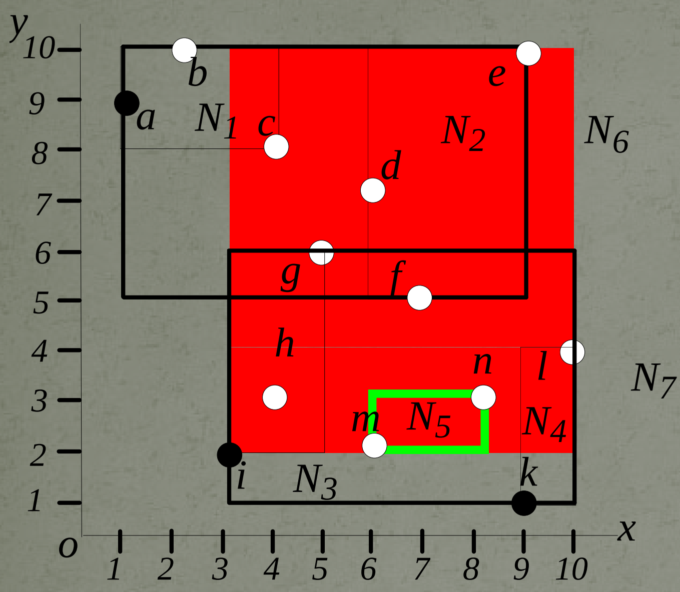
Example of BBS



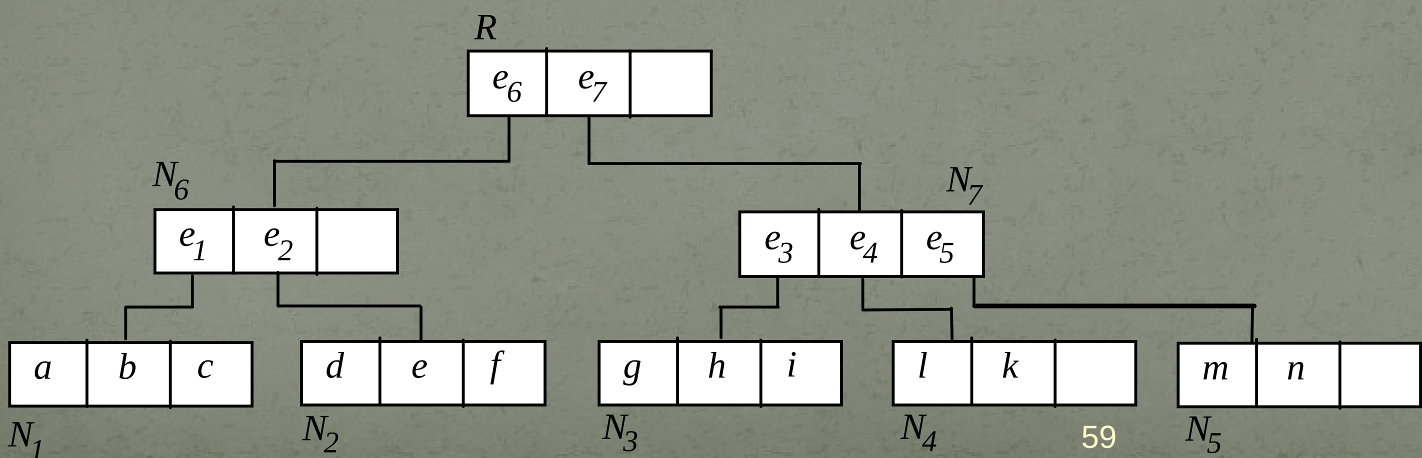
action	heap contents	*S*
access root	$\langle e_7,4\rangle\langle e_6,6\rangle$	$\emptyset$
expand e_7	$\langle e_3,5\rangle\langle e_6,6\rangle\langle e_5,8\rangle\langle e_4,10\rangle$	$\emptyset$
expand e_3	**<i,5>**$\langle e_6,6\rangle\langle e_5,8\rangle\langle e_4,10\rangle$	{*i*}
expand e_6	$\langle e_5,8\rangle\langle e_1,9\rangle\langle e_4,10\rangle$	{*i*}



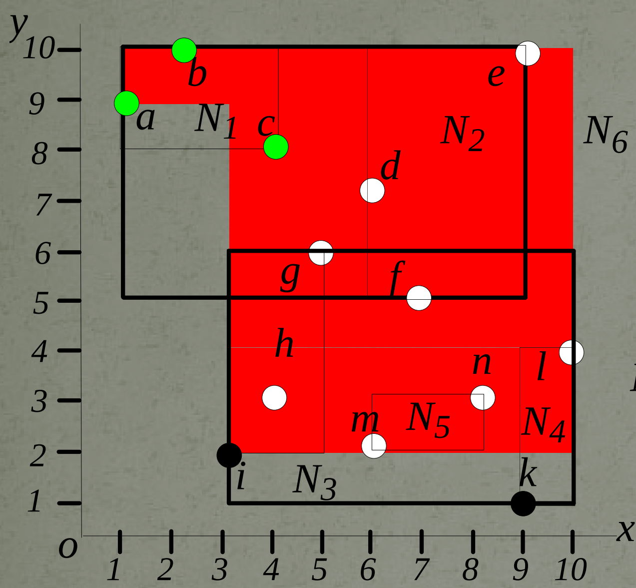
Example of BBS



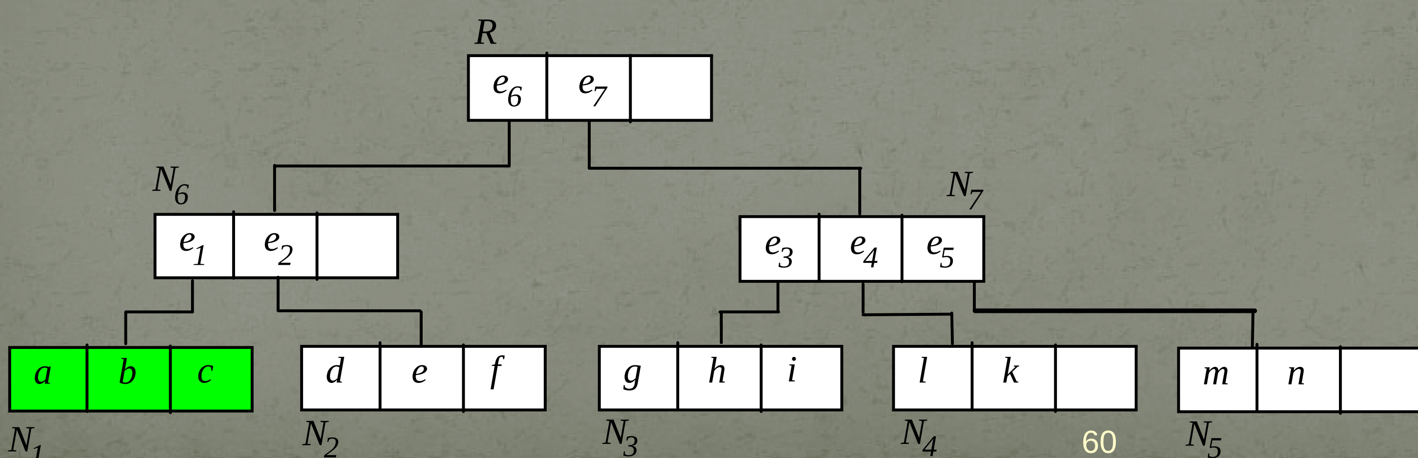
action	heap contents	*S*
access root	$<e_7,4><e_6,6>$	∅
expand e_7	$<e_3,5><e_6,6><e_5,8><e_4,10>$	∅
expand e_3	**<i,5>**$<e_6,6><e_5,8><e_4,10>$	{*i*}
expand e_6	$<e_5,8><e_1,9><e_4,10>$	{*i*}
remove e_5	$<e_1,9><e_4,10>$	{*i*}



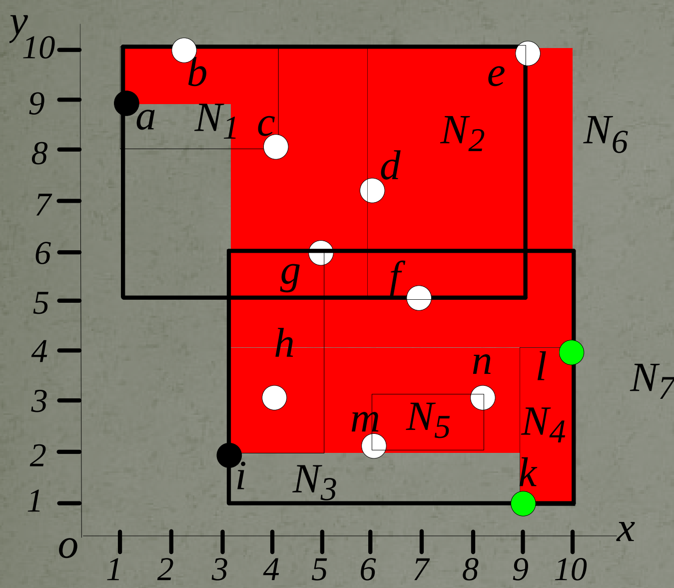
Example of BBS



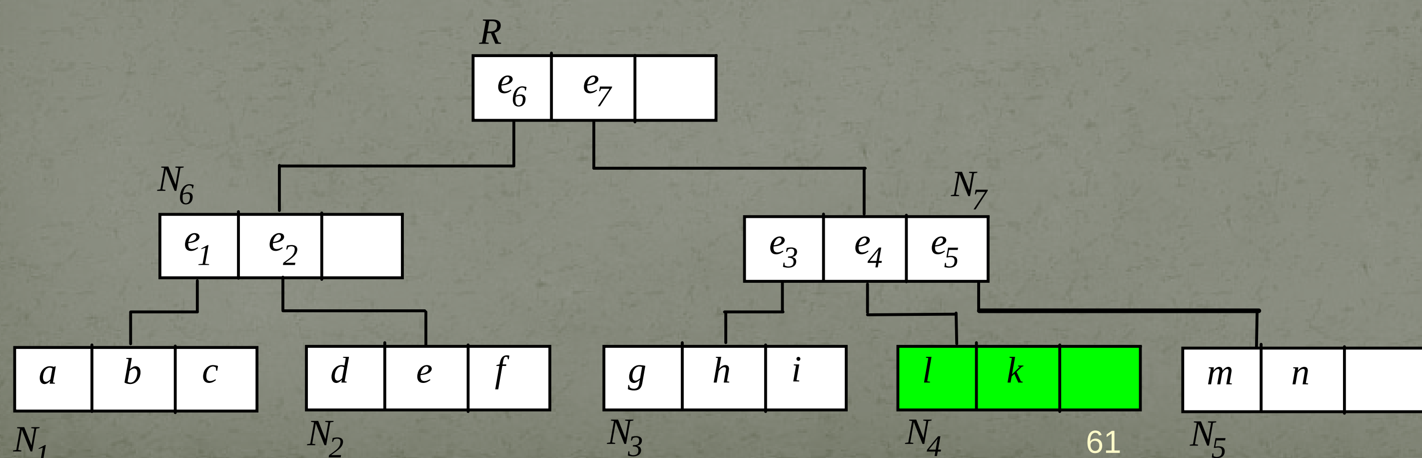
action	heap contents	S
access root	<e_7,4><e_6,6>	∅
expand e_7	<e_3,5><e_6,6><e_5,8><e_4,10>	∅
expand e_3	**<*i*,5>**<e_6,6><e_5,8> <e_4,10>	{*i*}
expand e_6	<e_5,8><e_1,9><e_4,10>	{*i*}
remove e_5	<e_1,9><e_4,10>	{*i*}
expand e_1	**<*a*,10>**<e_4,10>	{*i*,*a*}



Example of BBS



action	heap contents	S
access root	$<e_7,4><e_6,6>$	∅
expand e_7	$<e_3,5><e_6,6><e_5,8><e_4,10>$	∅
expand e_3	$\mathbf{<i,5>}<e_6,6><e_5,8><e_4,10>$	{*i*}
expand e_6	$<e_5,8><e_1,9><e_4,10>$	{*i*}
remove e_5	$<e_1,9><e_4,10>$	{*i*}
expand e_1	$\mathbf{<a,10>}<e_4,10>$	{*i*,*a*}
expand e_4	$\mathbf{<k,10>}$	{*i*,*a*,*k*}



Summary & Reference

- NN Query: N. Roussopoulos, S. Kelley and F. Vincent, "Nearest Neighbor Queries", SIGMOD'95. (It didn't talk about best-first search).
- RNN Query: Y. Tao, D. Papadias and X. Lian, "Reverse kNN Search in Arbitrary Dimensionality", VLDB'04.
- Skyline Query: D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An Optimal and Progressive Algorithm for Skyline Queries", SIGMOD'03.
- Also talked about Closest/Close Pair Queries.