

CMU SCS

## 15-826: Multimedia Databases and Data Mining

Lecture #6: Spatial Access Methods  
Part III: R-trees  
*C. Faloutsos*

---

---

---

---

---

---

---

---

CMU SCS

## Must-read material

- Textbook, Chapter 5.2
- Ramakrishnan+Gehrke, Chapter 28.6
- Guttman, A. (June 1984). *R-Trees: A Dynamic Index Structure for Spatial Searching*. Proc. ACM SIGMOD, Boston, Mass.
- Ibrahim Kamel and Christos Faloutsos, *Hilbert R-tree: An improved R-tree using fractals* Proc. of VLDB Conference, Santiago, Chile, Sept. 12-15, 1994, pp. 500-509.

15-826 Copyright: C. Faloutsos (2009) #2

---

---

---

---

---

---

---

---

CMU SCS

## Outline

Goal: 'Find similar / interesting things'

- Intro to DB
- ➔ • Indexing - similarity search
- Data Mining

15-826 Copyright: C. Faloutsos (2009) #3

---

---

---

---

---

---

---

---

CMU SCS

## Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
- spatial access methods
  - problem defn
  - z-ordering
  - ➔ – R-trees
  - ...
- text
- ...

15-826 Copyright: C. Faloutsos (2009) #4

---

---

---

---

---

---

---

---

CMU SCS

## Indexing - more detailed outline

- R-trees
  - ➔ – main idea; file structure
  - algorithms: insertion/split
  - deletion
  - search: range, nn, spatial joins
  - performance analysis
  - variations (packed; hilbert;...)

15-826 Copyright: C. Faloutsos (2009) #5

---

---

---

---

---

---

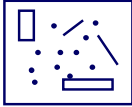
---

---

CMU SCS

## Reminder: problem

- Given a collection of geometric objects (points, lines, polygons, ...)
- organize them on disk, to answer spatial queries (range, nn, etc)



15-826 Copyright: C. Faloutsos (2009) #6

---

---

---

---

---

---

---

---

CMU SCS

## R-trees

- z-ordering: cuts regions to pieces -> dup. elim.
- how could we avoid that?
- Idea: try to extend/merge B-trees and k-d trees

15-826 Copyright: C. Faloutsos (2009) #7

---

---

---

---

---

---

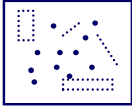
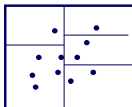
---

---

CMU SCS

## (first attempt: k-d-B-trees)

- [Robinson, 81]: if  $f$  is the fanout, split point-set in  $f$  parts; and so on, recursively

15-826 Copyright: C. Faloutsos (2009) #8

---

---

---

---

---

---


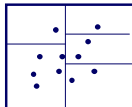
---

---

CMU SCS

## (first attempt: k-d-B-trees)

- But: insertions/deletions are tricky (splits may propagate downwards **and** upwards)
- no guarantee on space utilization

15-826 Copyright: C. Faloutsos (2009) #9

---

---

---

---

---

---


---

---

CMU SCS

## R-trees

- [Guttman 84] Main idea: allow parents to overlap!



Antonin (Toni) Guttman  
<http://www.baymoon.com/~tg2/>

15-826 Copyright: C. Faloutsos (2009) #10

---

---

---

---

---

---


---

---

CMU SCS

## R-trees

- [Guttman 84] Main idea: allow parents to overlap!
  - => guaranteed 50% utilization
  - => easier insertion/split algorithms.
  - (only deal with Minimum Bounding Rectangles - **MBRs**)



15-826 Copyright: C. Faloutsos (2009) #11

---

---

---

---

---

---

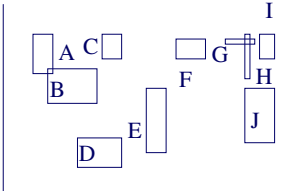
---

---

CMU SCS

## R-trees

- eg., w/ fanout 4: group nearby rectangles to parent MBRs; each group -> disk page



15-826 Copyright: C. Faloutsos (2009) #12

---

---

---

---

---

---

---

---

CMU SCS

## R-trees

- eg., w/ fanout 4:

15-826 Copyright: C. Faloutsos (2009) #13

---

---

---

---

---

---

---

---

CMU SCS

## R-trees

- eg., w/ fanout 4:

15-826 Copyright: C. Faloutsos (2009) #14

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - format of nodes

- {(MBR; obj-ptr)} for leaf nodes

15-826 Copyright: C. Faloutsos (2009) #15

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - format of nodes

- {(MBR; node-ptr)} for non-leaf nodes

15-826 Copyright: C. Faloutsos (2009) #16

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - range search?

15-826 Copyright: C. Faloutsos (2009) #17

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - range search?

15-826 Copyright: C. Faloutsos (2009) #18

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - range search

Observations:

- every parent node completely covers its 'children'
- a child MBR may be covered by more than one parent - it is stored under ONLY ONE of them. (ie., no need for dup. elim.)

15-826 Copyright: C. Faloutsos (2009) #19

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - range search

Observations - cont'd

- a point query may follow multiple branches.
- everything works for **any** dimensionality

15-826 Copyright: C. Faloutsos (2009) #20

---

---

---

---

---

---

---

---

CMU SCS

## Indexing - more detailed outline

- R-trees
  - main idea; file structure
  - ➔ - algorithms: insertion/split
  - deletion
  - search: range, nn, spatial joins
  - performance analysis
  - variations (packed; hilbert;...)

15-826 Copyright: C. Faloutsos (2009) #21

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion

- eg., rectangle 'X'

15-826 Copyright: C. Faloutsos (2009) #22

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion

- eg., rectangle 'X'

15-826 Copyright: C. Faloutsos (2009) #23

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion

- eg., rectangle 'Y'

15-826 Copyright: C. Faloutsos (2009) #24

---

---

---

---

---

---

---

---



CMU SCS

### R-trees - insertion

- eg., rectangle 'Y': extend suitable parent.

15-826 Copyright: C. Faloutsos (2009) #25

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion

- eg., rectangle 'Y': extend suitable parent.
- Q: how to measure 'suitability'?

15-826 Copyright: C. Faloutsos (2009) #26

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion

- eg., rectangle 'Y': extend suitable parent.
- Q: how to measure 'suitability'?
- A: by increase in area (volume) (more details: later, under 'performance analysis')
- Q: what if there is no room? how to split?

15-826 Copyright: C. Faloutsos (2009) #27

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - insertion

- eg., rectangle 'W'

15-826 Copyright: C. Faloutsos (2009) #28

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - insertion

- eg., rectangle 'W' - focus on 'P1' - how to split?

15-826 Copyright: C. Faloutsos (2009) #29

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - insertion

- eg., rectangle 'W' - focus on 'P1' - how to split?

- (A1: plane sweep, until 50% of rectangles)
- A2: 'linear' split
- ➔ A3: quadratic split
- A4: exponential split

15-826 Copyright: C. Faloutsos (2009) #30

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion & split

- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed'

15-826 Copyright: C. Faloutsos (2009) #31

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion & split

- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed'
- Q: how to measure 'closeness'?

15-826 Copyright: C. Faloutsos (2009) #32

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion & split

- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed'
- Q: how to measure 'closeness'?
- A: by increase of area (volume)

15-826 Copyright: C. Faloutsos (2009) #33

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion & split

- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed'

15-826 Copyright: C. Faloutsos (2009) #34

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion & split

- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed'

15-826 Copyright: C. Faloutsos (2009) #35

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion & split

- pick two rectangles as 'seeds';
- assign each rectangle 'R' to the 'closest' 'seed'
- smart idea: pre-sort rectangles according to delta of closeness (ie., schedule easiest choices first!)

15-826 Copyright: C. Faloutsos (2009) #36

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion - pseudocode

- decide which parent to put new rectangle into ('closest' parent)
- if overflow, split to two, using (say,) the quadratic split algorithm
  - propagate the split upwards, if necessary
- update the MBRs of the affected parents.

15-826 Copyright: C. Faloutsos (2009) #37

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - insertion - observations

- **many** more split algorithms exist (next!)

15-826 Copyright: C. Faloutsos (2009) #38

---

---

---

---

---

---

---

---

CMU SCS

### Indexing - more detailed outline

- R-trees
  - main idea; file structure
  - algorithms: insertion/split
  - ➔ – deletion
  - search: range, nn, spatial joins
  - performance analysis
  - variations (packed; hilbert;...)

15-826 Copyright: C. Faloutsos (2009) #39

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - deletion

- delete rectangle
- if underflow
  - ??

15-826 Copyright: C. Faloutsos (2009) #40

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - deletion

- delete rectangle
- if underflow
  - temporarily delete all siblings (!);
  - delete the parent node and
  - re-insert them

15-826 Copyright: C. Faloutsos (2009) #41

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - deletion

- variations: later (eg. Hilbert R-trees w/ 2-to-1 merge)

15-826 Copyright: C. Faloutsos (2009) #42

---

---

---

---

---

---

---

---

CMU SCS

## Indexing - more detailed outline

- R-trees
  - main idea; file structure
  - algorithms: insertion/split
  - deletion
  - ➔ - search: range, nn, spatial joins
  - performance analysis
  - variations (packed; hilbert;...)

15-826 Copyright: C. Faloutsos (2009) #43

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - range search

pseudocode:

- check the root
- for each branch,
- if its MBR intersects the query rectangle
- apply range-search (or print out, if this is a leaf)

15-826 Copyright: C. Faloutsos (2009) #44

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - nn search

15-826 Copyright: C. Faloutsos (2009) #45

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - nn search

- Q: How? (find near neighbor; refine...)

15-826 Copyright: C. Faloutsos (2009) #46

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - nn search

- A1: depth-first search; then, range query

15-826 Copyright: C. Faloutsos (2009) #47

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - nn search

- A1: depth-first search; then, range query

15-826 Copyright: C. Faloutsos (2009) #48

---

---

---

---

---

---

---

---



CMU SCS

### R-trees - nn search

- A1: depth-first search; then, range query

The diagram shows an R-tree structure. The root node is P1, which contains leaf nodes A, B, C, and D. Node P1 also contains a pointer to node P2. Node P2 contains leaf nodes B and D. Node P3 contains leaf nodes G, H, and I. Node P4 contains leaf nodes E and J. A query point  $q$  is shown as a red dot near node P2. The leaf nodes are represented by small rectangles, and the internal nodes are represented by larger rectangles.

15-826 Copyright: C. Faloutsos (2009) #49

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - nn search

- A2: [Roussopoulos+, sigmod95]:
  - priority queue, with promising MBRs, and their best and worst-case distance
- main idea:

15-826 Copyright: C. Faloutsos (2009) #50

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - nn search

consider only P2 and P4, for illustration

The diagram is identical to the one in slide #49, but with a focus on nodes P2 and P4. Node P2 contains leaf nodes B and D, and node P4 contains leaf nodes E and J. The query point  $q$  is shown near P2.

15-826 Copyright: C. Faloutsos (2009) #51

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - nn search

best of P4      => P4 is useless  
 worst of P2      for 1-nn

15-826      Copyright: C. Faloutsos (2009)      #52

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - nn search

- what is really the worst of, say, P2?

15-826      Copyright: C. Faloutsos (2009)      #53

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - nn search

- what is really the worst of, say, P2?
- A: the smallest of the two red segments!

15-826      Copyright: C. Faloutsos (2009)      #54

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - nn search

- variations: [Hjaltason & Samet] incremental nn:
  - build a priority queue
  - scan enough of the tree, to make sure you have the  $k$  nn
  - to find the  $(k+1)$ -th, check the queue, and scan some more of the tree
- ‘optimal’ (but, may need too much memory)

15-826 Copyright: C. Faloutsos (2009) #55

---

---

---

---

---

---

---

---

CMU SCS

## Indexing - more detailed outline

- R-trees
  - main idea; file structure
  - algorithms: insertion/split
  - deletion
  - ➔ – search: range, nn, **spatial joins**
  - performance analysis
  - variations (packed; hilbert;...)

15-826 Copyright: C. Faloutsos (2009) #56

---

---

---

---

---

---

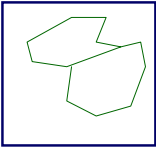
---

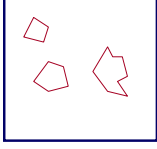
---

CMU SCS

## R-trees - spatial joins

**Spatial joins:** find (quickly) all  
counties intersecting lakes





15-826 Copyright: C. Faloutsos (2009) #57

---

---

---

---

---

---

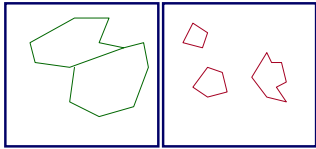
---

---

CMU SCS

### R-trees - spatial joins

Spatial joins: find (quickly) all  
counties intersecting lakes



15-826 Copyright: C. Faloutsos (2009) #58

---

---

---

---

---

---

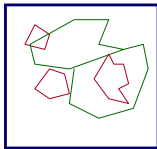
---

---

CMU SCS

### R-trees - spatial joins

Spatial joins: find (quickly) all  
counties intersecting lakes



15-826 Copyright: C. Faloutsos (2009) #59

---

---

---

---

---

---


---

---

CMU SCS

### R-trees - spatial joins

Assume that they are both organized in R-trees:



15-826 Copyright: C. Faloutsos (2009) #60

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - spatial joins

for each parent P1 of tree T1  
 for each parent P2 of tree T2  
 if their MBRs intersect,  
 process them recursively (ie., check their children)

15-826 Copyright: C. Faloutsos (2009) #61

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - spatial joins

Improvements - variations:

- [Seeger+, sigmod 92]: do some pre-filtering; do plane-sweeping to avoid  $N1 * N2$  tests for intersection
- [Lo & Ravishankar, sigmod 94]: 'seeded' R-trees (FYI, many more papers on spatial joins, without R-trees: [Koudas+ Sevcik], e.t.c.)

15-826 Copyright: C. Faloutsos (2009) #62

---

---

---

---

---

---

---

---

CMU SCS

## Indexing - more detailed outline

- R-trees
  - main idea; file structure
  - algorithms: insertion/split
  - deletion
  - search: range, nn, spatial joins
  - ➔ - performance analysis
  - variations (packed; hilbert;...)

15-826 Copyright: C. Faloutsos (2009) #63

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- How many disk (=node) accesses we'll need for
  - range
  - nn
  - spatial joins
- why does it matter?

15-826 Copyright: C. Faloutsos (2009) #64

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- How many disk (=node) accesses we'll need for
  - ➔ - range
  - nn
  - spatial joins
- why does it matter?
- A: because we can design split etc algorithms accordingly; also, do query-optimization

15-826 Copyright: C. Faloutsos (2009) #65

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- A: because we can design split etc algorithms accordingly; also, do query-optimization
- motivating question: on, e.g., split, should we try to minimize the area (volume)? the perimeter? the overlap? or a weighted combination? why?

15-826 Copyright: C. Faloutsos (2009) #66

---

---

---

---

---

---

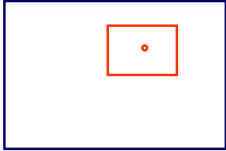
---

---

CMU SCS

### R-trees - performance analysis

- How many disk accesses for range queries?
  - query distribution wrt location?
  - “ “ wrt size?



15-826 Copyright: C. Faloutsos (2009) #67

---

---

---

---

---

---

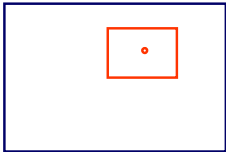
---

---

CMU SCS

### R-trees - performance analysis

- How many disk accesses for range queries?
  - query distribution wrt location? **uniform; (biased)**
  - “ “ wrt size? **uniform**



15-826 Copyright: C. Faloutsos (2009) #68

---

---

---

---

---

---

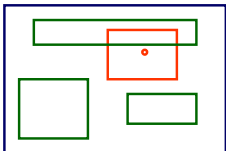
---

---

CMU SCS

### R-trees - performance analysis

- easier case: we know the positions of parent MBRs, eg:



15-826 Copyright: C. Faloutsos (2009) #69

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- How many times will P1 be retrieved (unif. queries)?

15-826 Copyright: C. Faloutsos (2009) #70

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- How many times will P1 be retrieved (unif. POINT queries)?

15-826 Copyright: C. Faloutsos (2009) #71

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- How many times will P1 be retrieved (unif. POINT queries)? A:  $x1 * x2$

15-826 Copyright: C. Faloutsos (2009) #72

---

---

---

---

---

---

---

---



CMU SCS

### R-trees - performance analysis

- How many times will P1 be retrieved (unif. queries of size  $q1 \times q2$ )?

15-826 Copyright: C. Faloutsos (2009) #73

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- How many times will P1 be retrieved (unif. queries of size  $q1 \times q2$ )?

15-826 Copyright: C. Faloutsos (2009) #74

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- How many times will P1 be retrieved (unif. queries of size  $q1 \times q2$ )?

15-826 Copyright: C. Faloutsos (2009) #75

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- How many times will P1 be retrieved (unif. queries of size  $q_1 \times q_2$ )?

15-826 Copyright: C. Faloutsos (2009) #76

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- How many times will P1 be retrieved (unif. queries of size  $q_1 \times q_2$ )? A:  $(x_1 + q_1) * (x_2 + q_2)$

15-826 Copyright: C. Faloutsos (2009) #77

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- Thus, given a tree with  $N$  nodes ( $i=1, \dots, N$ ) we expect

$$\begin{aligned} \#DiskAccesses(q_1, q_2) &= \sum (x_{i,1} + q_1) * (x_{i,2} + q_2) \\ &= \sum (x_{i,1} * x_{i,2}) + \\ &\quad q_2 * \sum (x_{i,1}) + \\ &\quad q_1 * \sum (x_{i,2}) \\ &\quad q_1 * q_2 * N \end{aligned}$$

15-826 Copyright: C. Faloutsos (2009) #78

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

- Thus, given a tree with  $N$  nodes ( $i=1, \dots, N$ ) we expect

$$\begin{aligned} \#DiskAccesses(q1,q2) &= \sum (x_{i,1} + q1) * (x_{i,2} + q2) \\ &= \sum (x_{i,1} * x_{i,2}) + \text{volume} \\ &\quad q2 * \sum (x_{i,1}) + \text{surface area} \\ &\quad q1 * \sum (x_{i,2}) \\ &\quad q1 * q2 * N \text{ count} \end{aligned}$$

15-826 Copyright: C. Faloutsos (2009) #79

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

Observations:

- for point queries: only volume matters
- for horizontal-line queries: ( $q2=0$ ): vertical length matters
- for large queries ( $q1, q2 \gg 0$ ): the count  $N$  matters

15-826 Copyright: C. Faloutsos (2009) #80

---

---

---

---

---

---

---


---

CMU SCS

### R-trees - performance analysis

Observations (cont'ed)

- overlap: does not seem to matter
- formula: easily extendible to  $n$  dimensions
- (for even more details: [Pagel +, PODS93], [Kamel+, CIKM93])



Berndt-Uwe Pagel

15-826 Copyright: C. Faloutsos (2009) #81

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - performance analysis

Conclusions:

- splits should try to minimize area and perimeter
- ie., we want few, small, square-like parent MBRs
- rule of thumb: shoot for queries with  $q_1=q_2 = 0.1$  (or  $=0.5$  or so).

15-826 Copyright: C. Faloutsos (2009) #82

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - performance analysis

- How many disk (=node) accesses we'll need for
  - ➔ - range
  - nn
  - spatial joins

15-826 Copyright: C. Faloutsos (2009) #83

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - performance analysis

Range queries - how many disk accesses, if we just now that we have  
 -  $N$  points in  $n$ -d space?  
 A: ?

15-826 Copyright: C. Faloutsos (2009) #84

---

---

---

---

---

---


---

---

CMU SCS

### R-trees - performance analysis

Range queries - how many disk accesses, if we just now that we have  
 -  $N$  points in  $n$ -d space?  
 A: can not tell! need to know distribution



15-826 Copyright: C. Faloutsos (2009) #85

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - performance analysis

What are obvious and/or realistic distributions?

15-826 Copyright: C. Faloutsos (2009) #86

---

---

---

---

---

---

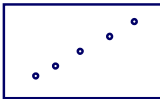
---

---

CMU SCS

### R-trees - performance analysis

What are obvious and/or realistic distributions?  
 A: uniform  
 A: Gaussian / mixture of Gaussians  
 A: self-similar / fractal. Fractal dimension  $\sim$  intrinsic dimension



15-826 Copyright: C. Faloutsos (2009) #87

---

---

---

---

---

---


---

---

CMU SCS

## R-trees - performance analysis

Formulas for range queries and k-nn queries: use fractal dimension [Kamel+, PODS94], [Korn+ ICDE2000] [Kriegel+, PODS97]  
 Formulas for spatial joins of regions: open research question



15-826 Copyright: C. Faloutsos (2009) #88

---

---

---

---

---

---

---

---

CMU SCS

## Indexing - more detailed outline

- R-trees
  - main idea; file structure
  - algorithms: insertion/split
  - deletion
  - search: range, nn, spatial joins
  - performance analysis
  - ➔ - variations (packed; hilbert;...)

15-826 Copyright: C. Faloutsos (2009) #89

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - variations

Guttman's R-trees sparked **much** follow-up work

- ➔ can we do better splits?
  - what about static datasets (no ins/del/upd)?
  - what about other bounding shapes?

15-826 Copyright: C. Faloutsos (2009) #90

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - variations

Guttman's R-trees sparked much follow-up work

- can we do better splits?
  - i.e, defer splits?

15-826 Copyright: C. Faloutsos (2009) #91

---

---

---

---

---

---



---

---

CMU SCS

## R-trees - variations

A: R\*-trees [Beckmann+, SIGMOD90]

Norbert Beckmann  
 Hans Peter Kriegel →   
 Ralf Schneider  
 Bernhard Seeger → 

15-826 Copyright: C. Faloutsos (2009) #92

---

---

---

---

---

---

---


---

CMU SCS

## R-trees - variations

A: R\*-trees [Beckmann+, SIGMOD90]

- defer splits, by forced-reinsert, i.e.: instead of splitting, temporarily delete some entries, shrink overflowing MBR, and re-insert those entries
- Which ones to re-insert?
- How many?



15-826 Copyright: C. Faloutsos (2009) #93

---

---

---

---

---

---

---


---

CMU SCS

## R-trees - variations

A: R\*-trees [Beckmann+, SIGMOD90]

- defer splits, by forced-reinsert, i.e.: instead of splitting, temporarily delete some entries, shrink overflowing MBR, and re-insert those entries
- Which ones to re-insert?
- How many? A: 30%



15-826 Copyright: C. Faloutsos (2009) #94

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - variations

Q: Other ways to defer splits?

15-826 Copyright: C. Faloutsos (2009) #95

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - variations

Q: Other ways to defer splits?

A: Push a few keys to the closest sibling node (closest = ??)

15-826 Copyright: C. Faloutsos (2009) #96

---

---

---

---

---

---

---

---



CMU SCS

### R-trees - variations

R\*-trees: Also try to minimize area AND perimeter, in their split.  
 Performance: higher space utilization; faster than plain R-trees. One of the **most successful** R-tree variants.

15-826 Copyright: C. Faloutsos (2009) #97

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - variations

Guttman's R-trees sparked **much** follow-up work

- can we do better splits?
- ➔ what about static datasets (no ins/del/upd)?
  - Hilbert R-trees
- what about other bounding shapes?

15-826 Copyright: C. Faloutsos (2009) #98

---

---

---

---

---

---

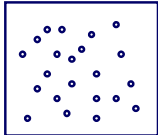
---

---

CMU SCS

### R-trees - variations

- what about static datasets (no ins/del/upd)?
- Q: Best way to pack points?



15-826 Copyright: C. Faloutsos (2009) #99

---

---

---

---

---

---

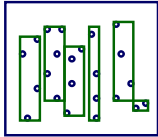
---

---

CMU SCS

### R-trees - variations

- what about static datasets (no ins/del/upd)?
- Q: Best way to pack points?
- A1: plane-sweep  
great for queries on 'x';  
terrible for 'y'



15-826 Copyright: C. Faloutsos (2009) #100

---

---

---

---

---

---

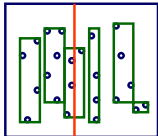
---

---

CMU SCS

### R-trees - variations

- what about static datasets (no ins/del/upd)?
- Q: Best way to pack points?
- A1: plane-sweep  
great for queries on 'x';  
bad for 'y'



15-826 Copyright: C. Faloutsos (2009) #101

---

---

---

---

---

---

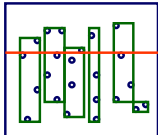
---

---

CMU SCS

### R-trees - variations

- what about static datasets (no ins/del/upd)?
- Q: Best way to pack points?
- A1: plane-sweep  
great for queries on 'x';  
terrible for 'y'
- Q: how to improve?



15-826 Copyright: C. Faloutsos (2009) #102

---

---

---

---

---

---

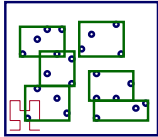
---

---

CMU SCS

### R-trees - variations

- A: plane-sweep on HILBERT curve!



15-826 Copyright: C. Faloutsos (2009) #103

---

---

---

---

---

---

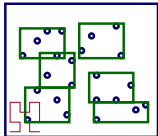
---

---

CMU SCS

### R-trees - variations

- A: plane-sweep on HILBERT curve!
- In fact, it can be made dynamic (how?), as well as to handle regions (how?)



15-826 Copyright: C. Faloutsos (2009) #104

---

---

---

---

---

---

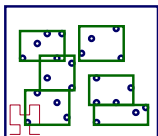
---

---

CMU SCS

### R-trees - variations

- Dynamic ('Hilbert R-tree):
  - each point has an 'h'-value (hilbert value)
  - insertions: like a B-tree on the h-value
  - but also store MBR, for searches



15-826 Copyright: C. Faloutsos (2009) #105

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - variations

- Data structure of a node?

15-826 Copyright: C. Faloutsos (2009) #106

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - variations

- Data structure of a node?

~B-tree

15-826 Copyright: C. Faloutsos (2009) #107

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - variations

- Data structure of a node?

~ R-tree

15-826 Copyright: C. Faloutsos (2009) #108

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - variations

Guttman's R-trees sparked **much** follow-up work

- can we do better splits?
- what about static datasets (no ins/del/upd)?
  - Hilbert R-trees - main idea
  - ➔ – handling regions
  - performance/discusion
- what about other bounding shapes?

15-826 Copyright: C. Faloutsos (2009) #109

---

---

---

---

---

---

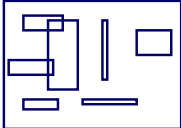
---

---

CMU SCS

### R-trees - variations

- What if we have regions, instead of points?
- I.e., how to impose a linear ordering ('h-value') on rectangles?



15-826 Copyright: C. Faloutsos (2009) #110

---

---

---

---

---

---

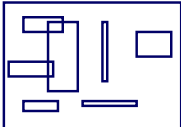
---

---

CMU SCS

### R-trees - variations

- What if we have regions, instead of points?
- I.e., how to impose a linear ordering ('h-value') on rectangles?
- A1: h-value of center
- A2: h-value of 4-d point (center, x-radius, y-radius)
- A3: ...



15-826 Copyright: C. Faloutsos (2009) #111

---

---

---

---

---

---

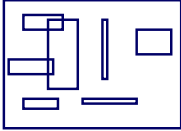
---

---

CMU SCS

### R-trees - variations

- What if we have regions, instead of points?
- I.e., how to impose a linear ordering ('h-value') on rectangles?
- **A1: h-value of center**
- A2: h-value of 4-d point (center, x-radius, y-radius)
- A3: ...



15-826 Copyright: C. Faloutsos (2009) #112

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - variations

- with h-values, we can have deferred splits, 2-to-3 splits (3-to-4, etc)
- experimentally: faster than R\*-trees (reference: [Kamel Faloutsos vldb 94])

15-826 Copyright: C. Faloutsos (2009) #113

---

---

---

---

---

---

---

---

CMU SCS

### R-trees - variations

Guttman's R-trees sparked **much** follow-up work

- can we do better splits?
- what about static datasets (no ins/del/upd)?
- ➔ what about other bounding shapes?

15-826 Copyright: C. Faloutsos (2009) #114

---

---

---

---

---

---

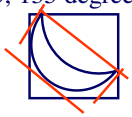
---

---

CMU SCS

### R-trees - variations

- what about other bounding shapes? (and why?)
- A1: arbitrary-orientation lines (cell-tree, [Guenther])
- A2: P-trees (polygon trees) (MB polygon: 0, 90, 45, 135 degree lines)



15-826 Copyright: C. Faloutsos (2009) #115

---

---

---

---

---

---

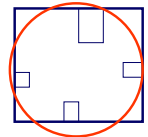
---

---

CMU SCS

### R-trees - variations

- A3: L-shapes; holes (hB-tree)
- A4: TV-trees [Lin+, VLDB-Journal 1994]
- A5: SR-trees [Katayama+, SIGMOD97] (used in Informedia)



15-826 Copyright: C. Faloutsos (2009) #116

---

---

---

---

---

---

---

---

CMU SCS

### Indexing - Detailed outline

- spatial access methods
  - problem defn
  - z-ordering
  - R-trees
  - ➡ - misc topics
    - grid files
    - dimensionality curse
    - metric trees
    - other nn methods
- text, ...

15-826 Copyright: C. Faloutsos (2009) #117

---

---

---

---

---

---

---

---

CMU SCS

## R-trees - conclusions

- Popular method; like multi-d B-trees
- guaranteed utilization
- good search times (for low-dim. at least)
- Informix (-> IBM-DB2) ships DataBlade with R-trees

15-826 Copyright: C. Faloutsos (2009) #118

---

---

---

---

---

---

---

---

CMU SCS

## References

- Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, Bernhard Seeger: *The R\*-Tree: An Efficient and Robust Access Method for Points and Rectangles*. ACM SIGMOD 1990: 322-331
- ➔ • Guttman, A. (June 1984). *R-Trees: A Dynamic Index Structure for Spatial Searching*. Proc. ACM SIGMOD, Boston, Mass.

15-826 Copyright: C. Faloutsos (2009) #119

---

---

---

---

---

---

---

---

CMU SCS

## References

- Jagadish, H. V. (May 23-25, 1990). Linear Clustering of Objects with Multiple Attributes. ACM SIGMOD Conf., Atlantic City, NJ.
- Ibrahim Kamel, Christos Faloutsos: *On Packing R-trees*, CIKM, 1993
- Lin, K.-I., H. V. Jagadish, et al. (Oct. 1994). "The TV-tree - An Index Structure for High-dimensional Data." VLDB Journal 3: 517-542.

15-826 Copyright: C. Faloutsos (2009) #120

---

---

---

---


---

---

---

---



 CMU SCS

### References, cont'd

- Pagel, B., H. Six, et al. (May 1993). *Towards an Analysis of Range Query Performance*. Proc. of ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Washington, D.C.
- Robinson, J. T. (1981). The k-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes. Proc. ACM SIGMOD.
- Roussopoulos, N., S. Kelley, et al. (May 1995). Nearest Neighbor Queries. Proc. of ACM-SIGMOD, San Jose, CA.

15-826 Copyright: C. Faloutsos (2009) #121

---

---

---


---

---

---

---

---

 CMU SCS

### Other resources

- Code, papers, datasets etc:  
[www.rtreportal.org/](http://www.rtreportal.org/)
- Java applets and more info:  
[donar.umiacs.umd.edu/quadtrees/points/rtrees.html](http://donar.umiacs.umd.edu/quadtrees/points/rtrees.html)

15-826 Copyright: C. Faloutsos (2009) #122

---

---

---

---

---

---

---

---