



CMU SCS

## 15-826: Multimedia Databases and Data Mining

Lecture#5: Multi-key and  
Spatial Access Methods - II  
*C. Faloutsos*

---

---

---

---

---

---

---



CMU SCS

## Must-read material

- MM-Textbook, Chapter 5.1
- Ramakrishnan+Gehrke, Chapter 28.4
- J. Orenstein,  
[\*Spatial Query Processing in an Object-Oriented Database System\*](#), Proc. ACM SIGMOD, May, 1986, pp. 326-336, Washington D.C.

15-826 Copyright: C. Faloutsos (2012) 2

---

---

---

---

---

---

---



CMU SCS

## Outline

Goal: 'Find **similar** / **interesting** things'

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

15-826 Copyright: C. Faloutsos (2012) 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 (2012) 4

---

---

---

---

---

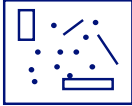
---

---

CMU SCS

## Spatial Access Methods - problem

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



15-826 Copyright: C. Faloutsos (2012) 5

---

---

---

---

---

---

---

CMU SCS

## Spatial Access Methods - problem

- Given a collection of geometric objects (points, lines, polygons, ...)
- organize them on disk, to answer
  - point queries
  - range queries
  - k-nn queries
  - spatial joins ('all pairs' queries)



15-826 Copyright: C. Faloutsos (2012) 6

---

---

---

---

---

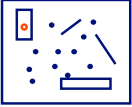
---

---

CMU SCS

## Spatial Access Methods - problem

- Given a collection of geometric objects (points, lines, polygons, ...)
- organize them on disk, to answer
  - point queries
  - range queries
  - k-nn queries
  - spatial joins ('all pairs' queries)



15-826 Copyright: C. Faloutsos (2012) 7

---

---

---

---

---

---

---

CMU SCS

## Spatial Access Methods - problem

- Given a collection of geometric objects (points, lines, polygons, ...)
- organize them on disk, to answer
  - point queries
  - range queries
  - k-nn queries
  - spatial joins ('all pairs' queries)



15-826 Copyright: C. Faloutsos (2012) 8

---

---

---

---

---

---

---

CMU SCS

## Spatial Access Methods - problem

- Given a collection of geometric objects (points, lines, polygons, ...)
- organize them on disk, to answer
  - point queries
  - range queries
  - k-nn queries
  - spatial joins ('all pairs' queries)



15-826 Copyright: C. Faloutsos (2012) 9

---

---

---

---

---

---

---

CMU SCS

## Spatial Access Methods - problem

- Given a collection of geometric objects (points, lines, polygons, ...)
- organize them on disk, to answer
  - point queries
  - range queries
  - k-nn queries
  - spatial joins** ('all pairs' within  $\epsilon$ )



15-826 Copyright: C. Faloutsos (2012) 10

---

---

---

---

---

---

---

---

CMU SCS

## SAMs - motivation

- Q: applications?

15-826 Copyright: C. Faloutsos (2012) 11

---

---

---

---

---

---

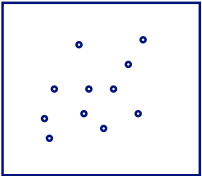
---

---

CMU SCS

## SAMs - motivation

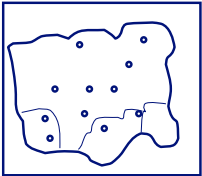
traditional DB



age

salary

GIS



15-826 Copyright: C. Faloutsos (2012) 12

---

---

---

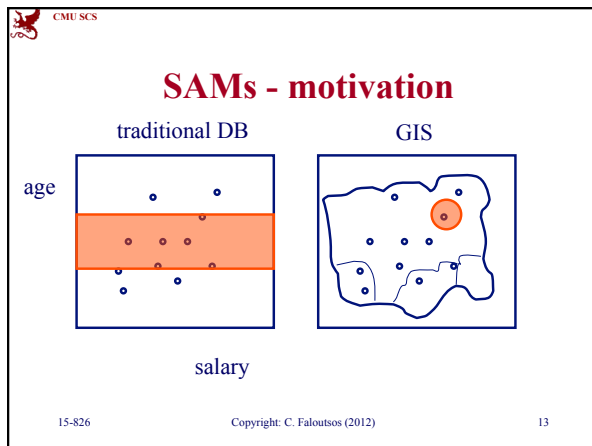
---

---

---

---

---



---

---

---

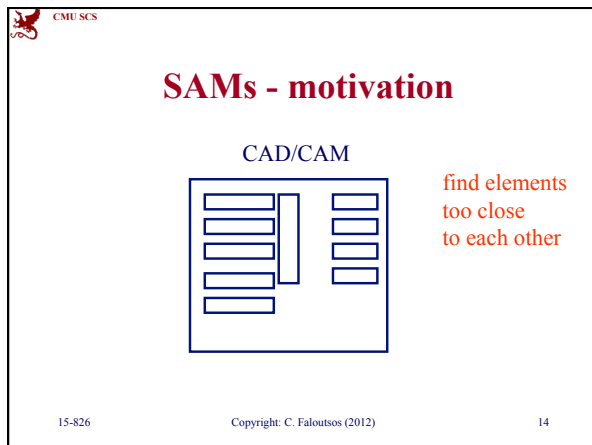
---

---

---

---

---



---

---

---

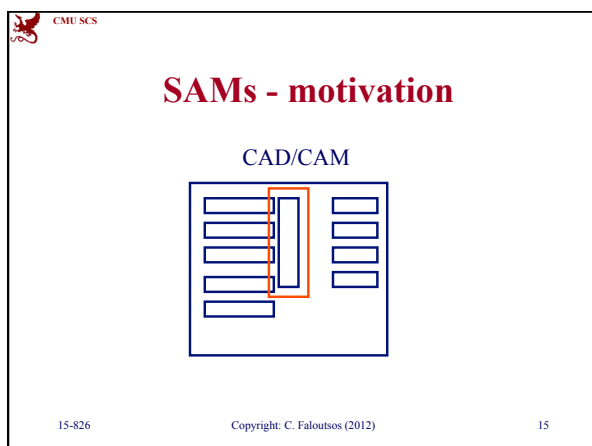
---

---

---

---

---



---

---

---

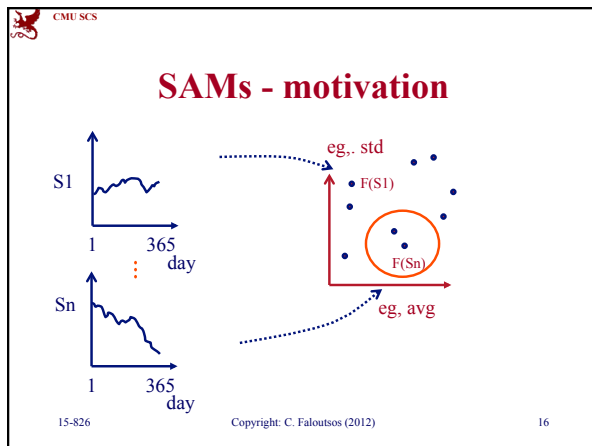
---

---

---

---

---




---

---

---

---

---

---

---

---

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 (2012) 17

---

---

---

---

---

---

---

---

CMU SCS

### SAMs: solutions

- z-ordering
- R-trees
- (grid files)

Q: how would you organize, e.g.,  $n$ -dim points, on disk? ( $C$  points per disk page)

15-826 Copyright: C. Faloutsos (2012) 18

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering

Q: how would you organize, e.g.,  $n$ -dim points, on disk? ( $C$  points per disk page)  
 Hint: reduce the problem to 1-d points (!!)

Q1: why?  
 A:  
 Q2: how?



15-826 Copyright: C. Faloutsos (2012) 19

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering

Q: how would you organize, e.g.,  $n$ -dim points, on disk? ( $C$  points per disk page)  
 Hint: reduce the problem to 1-d points (!!)

Q1: why?  
 A: B-trees!  
 Q2: how?



15-826 Copyright: C. Faloutsos (2012) 20

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering

Q2: how?  
 A: assume finite granularity; z-ordering = bit-shuffling = N-trees = Morton keys = geo-coding = ...



15-826 Copyright: C. Faloutsos (2012) 21

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering

Q2: how?  
 A: assume finite granularity (e.g.,  $2^{32} \times 2^{32}$ ; 4x4 here)  
 Q2.1: how to map n-d cells to 1-d cells?



15-826 Copyright: C. Faloutsos (2012) 22

---

---

---

---

---

---

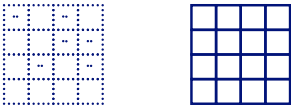
---

---

CMU SCS

## z-ordering

Q2.1: how to map  $n$ -d cells to 1-d cells?



15-826 Copyright: C. Faloutsos (2012) 23

---

---

---

---

---

---

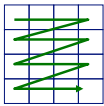
---

---

CMU SCS

## z-ordering

Q2.1: how to map  $n$ -d cells to 1-d cells?  
 A: row-wise  
 Q: is it good?



15-826 Copyright: C. Faloutsos (2012) 24

---

---

---

---

---

---

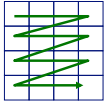
---

---

CMU SCS

## z-ordering

Q: is it good?  
A: great for 'x' axis; bad for 'y' axis



15-826 Copyright: C. Faloutsos (2012) 25

---

---

---

---

---

---

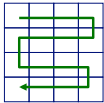
---

---

CMU SCS

## z-ordering

Q: How about the 'snake' curve?



15-826 Copyright: C. Faloutsos (2012) 26

---

---

---

---

---

---

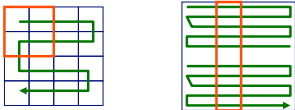
---

---

CMU SCS

## z-ordering

Q: How about the 'snake' curve?  
A: still problems:



$2^{32}$

$2^{32}$

15-826 Copyright: C. Faloutsos (2012) 27

---

---

---

---

---

---

---

---

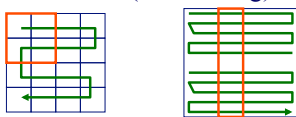
CMU SCS

## z-ordering

Q: Why are those curves 'bad'?

A: no distance preservation ( $\sim$  clustering)

Q: solution?



$2^{32}$

$2^{32}$

15-826 Copyright: C. Faloutsos (2012) 28

---

---

---

---

---

---

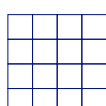
---

---

CMU SCS

## z-ordering

Q: solution? (w/ good clustering, and easy to compute, for 2-d and  $n$ -d?)



15-826 Copyright: C. Faloutsos (2012) 29

---

---

---

---

---

---

---

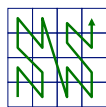
---

CMU SCS

## z-ordering

Q: solution? (w/ good clustering, and easy to compute, for 2-d and  $n$ -d?)

A: z-ordering/bit-shuffling/linear-quadtrees



'looks' better:

- few long jumps;
- scoops out the whole quadrant before leaving it
- a.k.a. space filling curves

15-826 Copyright: C. Faloutsos (2012) 30

---

---

---

---

---

---

---

---

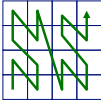
CMU SCS

## z-ordering

z-ordering/bit-shuffling/linear-quadtrees

Q: How to generate this curve ( $z = f(x,y)$ )?

A: 3 (equivalent) answers!



15-826 Copyright: C. Faloutsos (2012) 31

---

---

---

---

---

---

---

---

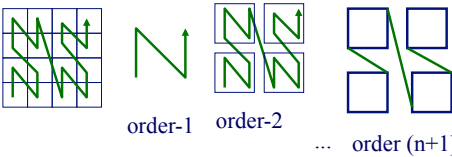
CMU SCS

## z-ordering

**z-ordering**/bit-shuffling/linear-quadtrees

Q: How to generate this curve ( $z = f(x,y)$ )?

A1: 'z' (or 'N') shapes, RECURSIVELY



order-1   order-2   ...   order (n+1)

15-826 Copyright: C. Faloutsos (2012) 32

---

---

---

---

---

---

---

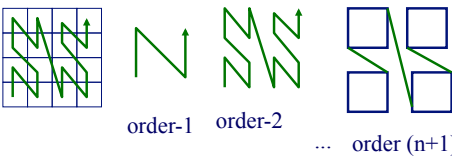
---

CMU SCS

## z-ordering

Notice:

- self similar (we'll see about fractals, soon)
- method is hard to use:  $z = ? f(x,y)$



order-1   order-2   ...   order (n+1)

15-826 Copyright: C. Faloutsos (2012) 33

---

---

---

---

---

---

---

---

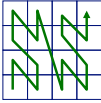
CMU SCS

## z-ordering

z-ordering/**bit-shuffling**/linear-quadtrees

Q: How to generate this curve ( $z = f(x,y)$ )?

A: 3 (equivalent) answers!



Method #2?

15-826 Copyright: C. Faloutsos (2012) 34

---

---

---

---

---

---

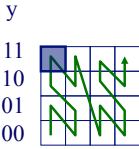
---

---

CMU SCS

## z-ordering

**bit-shuffling**



x

0 0

y

1 1

15-826 Copyright: C. Faloutsos (2012) 35

---

---

---

---

---

---

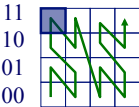
---

---

CMU SCS

## z-ordering

**bit-shuffling**



x

0 0

y

1 1

$z = (0101)_2 = 5$

15-826 Copyright: C. Faloutsos (2012) 36

---

---

---

---

---

---

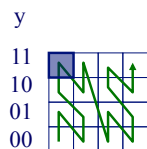
---

---

CMU SCS

## z-ordering

**bit-shuffling**



$$\begin{array}{cc} x & y \\ 0 & 0 \end{array} \quad \begin{array}{cc} & 1 \\ & 1 \end{array}$$

$$z = (0101)_2 = 5$$

How about the reverse:  
 $(x,y) = g(z)$  ?

15-826 Copyright: C. Faloutsos (2012) 37

---

---

---

---

---

---

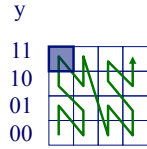
---

---

CMU SCS

## z-ordering

**bit-shuffling**



$$\begin{array}{cc} x & y \\ 0 & 0 \end{array} \quad \begin{array}{cc} & 1 \\ & 1 \end{array}$$

$$z = (0101)_2 = 5$$

How about  $n$ -d spaces?

15-826 Copyright: C. Faloutsos (2012) 38

---

---

---

---

---

---

---

---

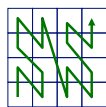
CMU SCS

## z-ordering

z-ordering/bit-shuffling/**linear-quadtrees**

Q: How to generate this curve ( $z = f(x,y)$ )?

A: 3 (equivalent) answers!



Method #3?

15-826 Copyright: C. Faloutsos (2012) 39

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering

**linear-quadtrees** : assign N->1, S->0 e.t.c.

15-826 Copyright: C. Faloutsos (2012) 40

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering

... and repeat recursively. Eg.:  $z_{\text{blue-cell}} =$   
 $WN; WN = (0101)_2 = 5$

15-826 Copyright: C. Faloutsos (2012) 41

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering

Drill: z-value of magenta cell, with the three methods?

15-826 Copyright: C. Faloutsos (2012) 42

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering

Drill: z-value of magenta cell, with the three methods?

W E

1

0

0 1

method#1: 14

method#2:  $\text{shuffle}(11;10) = (1110)_2 = 14$

15-826 Copyright: C. Faloutsos (2012) 43

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering

Drill: z-value of magenta cell, with the three methods?

W E

1

0

0 1

method#1: 14

method#2:  $\text{shuffle}(11;10) = (1110)_2 = 14$

method#3: EN;ES = ... = 14

15-826 Copyright: C. Faloutsos (2012) 44

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - Detailed outline

- spatial access methods
  - z-ordering
    - main idea - 3 methods
    - use w/ B-trees; algorithms (range, knn queries ...)
    - non-point (eg., region) data
    - analysis; variations
  - R-trees
  - ...

15-826 Copyright: C. Faloutsos (2012) 45

---

---

---

---

---

---

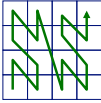
---

---

CMU SCS

## z-ordering - usage & algo's

Q1: How to store on disk?  
A:  
Q2: How to answer range queries etc



15-826 Copyright: C. Faloutsos (2012) 46

---

---

---

---

---

---

---

---

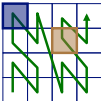
CMU SCS

## z-ordering - usage & algo's

Q1: How to store on disk?  
A: treat z-value as primary key; feed to B-tree

PGH

SF



| z  | cname | etc |
|----|-------|-----|
| 5  | SF    |     |
| 12 | PGH   |     |
|    |       |     |

15-826 Copyright: C. Faloutsos (2012) 47

---

---

---

---

---

---

---

---

CMU SCS

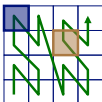
## z-ordering - usage & algo's

MAJOR ADVANTAGES w/ B-tree:

- already inside commercial systems (no coding/debugging!)
- concurrency & recovery is ready

PGH

SF



| z  | cname | etc |
|----|-------|-----|
| 5  | SF    |     |
| 12 | PGH   |     |
|    |       |     |

15-826 Copyright: C. Faloutsos (2012) 48

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - usage & algo's

Q2: queries? (eg.: *find city at (0,3)* )?

SF

| z  | cname | etc |
|----|-------|-----|
| 5  | SF    |     |
| 12 | PGH   |     |
|    |       |     |

15-826 Copyright: C. Faloutsos (2012) 49

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - usage & algo's

Q2: queries? (eg.: *find city at (0,3)* )?

A: find z-value; search B-tree

SF

| z  | cname | etc |
|----|-------|-----|
| 5  | SF    |     |
| 12 | PGH   |     |
|    |       |     |

15-826 Copyright: C. Faloutsos (2012) 50

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - usage & algo's

Q2: range queries?

SF

| z  | cname | etc |
|----|-------|-----|
| 5  | SF    |     |
| 12 | PGH   |     |
|    |       |     |

15-826 Copyright: C. Faloutsos (2012) 51

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - usage & algo's

Q2: range queries?

A: compute ranges of z-values; use B-tree

SF

9,11-15

| z  | cname | etc |
|----|-------|-----|
| 5  | SF    |     |
| 12 | PGH   |     |
|    |       |     |

15-826 Copyright: C. Faloutsos (2012) 52

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - usage & algo's

Q2': range queries - how to reduce # of qualifying of ranges?

SF

9,11-15

| z  | cname | etc |
|----|-------|-----|
| 5  | SF    |     |
| 12 | PGH   |     |
|    |       |     |

15-826 Copyright: C. Faloutsos (2012) 53

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - usage & algo's

Q2': range queries - how to reduce # of qualifying of ranges?

A: Augment the query!

SF

9,11-15 -> 8-15

| z  | cname | etc |
|----|-------|-----|
| 5  | SF    |     |
| 12 | PGH   |     |
|    |       |     |

15-826 Copyright: C. Faloutsos (2012) 54

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - usage & algo's

Q2'': range queries - how to break a query into ranges?



15-826 Copyright: C. Faloutsos (2012) 55

---

---

---

---

---

---

---

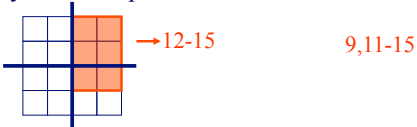
---

CMU SCS

## z-ordering - usage & algo's

Q2'': range queries - how to break a query into ranges?

A: recursively, quadtree-style; decompose only non-full quadrants



15-826 Copyright: C. Faloutsos (2012) 56

---

---

---

---

---

---

---

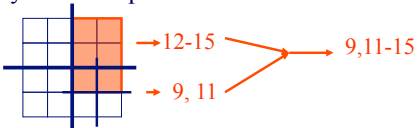
---

CMU SCS

## z-ordering - usage & algo's

Q2'': range queries - how to break a query into ranges?

A: recursively, quadtree-style; decompose only non-full quadrants



15-826 Copyright: C. Faloutsos (2012) 57

---

---

---

---

---

---

---

---



# z-ordering - Detailed outline

- spatial access methods
  - z-ordering
    - main idea - 3 methods
    - use w/ B-trees; algorithms (range, knn queries ...)
    - non-point (eg., region) data
    - analysis; variations
  - R-trees
  - ...



15-826 Copyright: C. Faloutsos (2012) 58

---

---

---

---

---

---

# z-ordering - usage & algo's

Q3: k-nn queries? (say, 1-nn)?

| z  | cname | etc |
|----|-------|-----|
| 5  | SF    |     |
| 12 | PGH   |     |
|    |       |     |

15-826 Copyright: C. Faloutsos (2012) 59

---

---

---

---

---

---

CMU SCS

**z-ordering - usage & algo's**

Q3: k-nn queries? (say, 1-nn)?  
A: traverse B-tree; find nn wrt z-values and ...

SF      PGH

The diagram illustrates the mapping from spatial data to a B-tree. On the left, a 4x4 grid labeled 'SF' contains a green zig-zag path representing a space-filling curve. An orange-shaded square highlights a specific region. A large blue triangle points from this grid to a B-tree structure on the right. The B-tree has three levels: a root node labeled 'z cname etc' and two leaf nodes. The first leaf node contains '5' and 'SF', while the second leaf node contains '12' and 'PGH'. Empty slots are shown in the third row of the tree.

| z  | cname | etc |
|----|-------|-----|
| 5  | SF    |     |
| 12 | PGH   |     |
|    |       |     |

15-826 Copyright: C. Faloutsos (2012)

---

---

---

---

---

---

CMU SCS

## z-ordering - usage & algo's

... ask a range query.

15-826 Copyright: C. Faloutsos (2012) 61

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - usage & algo's

... ask a range query.

15-826 Copyright: C. Faloutsos (2012) 62

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - usage & algo's

Q4: all-pairs queries? ( *all pairs of cities within 10 miles from each other?* )

(we'll see 'spatial joins' later: *find all PA counties that intersect a lake*)

15-826 Copyright: C. Faloutsos (2012) 63

---

---

---

---

---

---

---

---



# z-ordering - Detailed outline

- spatial access methods
  - z-ordering
    - main idea - 3 methods
    - use w/ B-trees; algorithms (range, knn queries ...)
    - non-point (eg., region) data
    - analysis; variations
  - R-trees
  - ...

15-826 Copyright: C. Faloutsos (2012) 64

---

---

---

---

---

---

# z-ordering - regions

Q: z-value for a region?

A

B

C

$z_B = ??$

$z_C = ??$

15-826 Copyright: C. Faloutsos (2012) 65

---

---

---

---

---

---

# z-ordering - regions

Q: z-value for a region?

A: 1 or more z-values; by quadtree decomposition

A

B

$z_B = ??$

$z_C = ??$

C

15-826

Copyright: C. Faloutsos (2012)

66

---

---

---

---

---

---

**z-ordering - regions** “don't care”

Q: z-value for a region?

$z_B = 11^{**}$

$z_C = ??$

15-826 Copyright: C. Faloutsos (2012) 67

---

---

---

---

---

---

---

---

**z-ordering - regions** “don't care”

Q: z-value for a region?

$z_B = 11^{**}$

$z_C = \{0010; 1000\}$

15-826 Copyright: C. Faloutsos (2012) 68

---

---

---

---

---

---

---

---

**z-ordering - regions**

Q: How to store in B-tree?

Q: How to search (range etc queries)

15-826 Copyright: C. Faloutsos (2012) 69

---

---

---

---

---

---

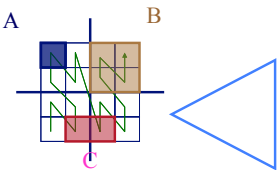
---

---

**z-ordering - regions**

Q: How to store in B-tree? A: sort ( $* < 0 < 1$ )

Q: How to search (range etc queries)



| z    | obj-id | etc |
|------|--------|-----|
| 0010 | C      |     |
| 0101 | A      |     |
| 1000 | C      |     |
| 11** | B      |     |

15-826 Copyright: C. Faloutsos (2012) 70

---

---

---

---

---

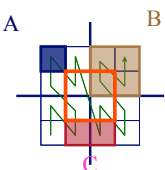
---

---

---

**z-ordering - regions**

Q: How to search (range etc queries) - eg 'red' range query



| z    | obj-id | etc |
|------|--------|-----|
| 0010 | C      |     |
| 0101 | A      |     |
| 1000 | C      |     |
| 11** | B      |     |

15-826 Copyright: C. Faloutsos (2012) 71

---

---

---

---

---

---

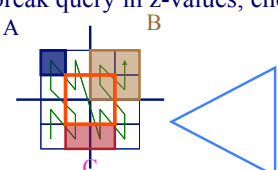
---

---

**z-ordering - regions**

Q: How to search (range etc queries) - eg 'red' range query

A: break query in z-values; check B-tree



| z    | obj-id | etc |
|------|--------|-----|
| 0010 | C      |     |
| 0101 | A      |     |
| 1000 | C      |     |
| 11** | B      |     |

15-826 Copyright: C. Faloutsos (2012) 72

---

---

---

---

---

---

---

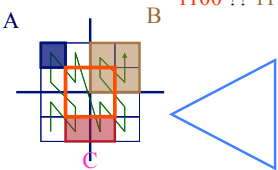
---

CMU SCS

## z-ordering - regions

Almost identical to range queries for point data, except for the “don’t cares” - i.e.,

1100 ?? 11\*\*



| z    | obj-id | etc |
|------|--------|-----|
| 0010 | C      |     |
| 0101 | A      |     |
| 1000 | C      |     |
| 11** | B      |     |

15-826 Copyright: C. Faloutsos (2012) 73

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - regions

Almost identical to range queries for point data, except for the “don’t cares” - i.e.,

$z1 = 1100 \text{ ?? } 11** = z2$

Specifically: does  $z1$  contain/avoid/intersect  $z2$ ?

Q: what is the criterion to decide?

15-826 Copyright: C. Faloutsos (2012) 74

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - regions

$z1 = 1100 \text{ ?? } 11** = z2$

Specifically: does  $z1$  contain/avoid/intersect  $z2$ ?

Q: what is the criterion to decide?

A: **Prefix property:** let  $r1, r2$  be the corresponding regions, and let  $r1$  be the smallest ( $\Rightarrow z1$  has fewest ‘\*’s). Then:

15-826 Copyright: C. Faloutsos (2012) 75

---

---

---

---

---

---

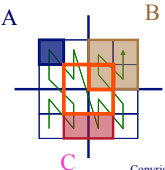
---

---

CMU SCS

## z-ordering - regions

- $r_2$  will either contain completely, or avoid completely  $r_1$ .
- it will contain  $r_1$ , if  $z_2$  is the prefix of  $z_1$



15-826 Copyright: C. Faloutsos (2012) 76

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - regions

Drill (True/False). Given:

- $z_1 = 011001**$
- $z_2 = 01*****$
- $z_3 = 0100****$

T/F  $r_2$  contains  $r_1$   
T/F  $r_3$  contains  $r_1$   
T/F  $r_3$  contains  $r_2$

15-826 Copyright: C. Faloutsos (2012) 77

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - regions

Drill (True/False). Given:

- $z_1 = 011001**$
- $z_2 = 01*****$
- $z_3 = 0100****$

T/F  $r_2$  contains  $r_1$  - TRUE (prefix property)  
T/F  $r_3$  contains  $r_1$  - FALSE (disjoint)  
T/F  $r_3$  contains  $r_2$  - FALSE ( $r_2$  contains  $r_3$ )

15-826 Copyright: C. Faloutsos (2012) 78

---

---

---

---

---

---

---

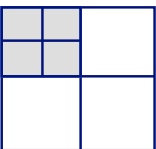
---

CMU SCS

## z-ordering - regions

Drill (True/False). Given:

- $z1 = 011001**$
- $z2 = 01*****$
- $z3 = 0100****$



15-826 Copyright: C. Faloutsos (2012) 79

---

---

---

---

---

---

---

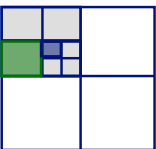
---

CMU SCS

## z-ordering - regions

Drill (True/False). Given:

- $z1 = 011001**$
- $z2 = 01*****$
- $z3 = 0100****$



T/F r2 contains r1 - TRUE (prefix property)  
T/F r3 contains r1 - FALSE (disjoint)  
T/F r3 contains r2 - FALSE (r2 contains r3)

15-826 Copyright: C. Faloutsos (2012) 80

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - regions

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



15-826 Copyright: C. Faloutsos (2012) 81

---

---

---

---

---

---

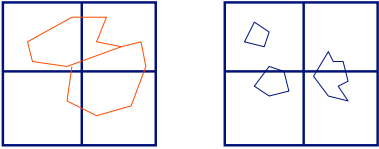
---

---

CMU SCS

## z-ordering - regions

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



15-826 Copyright: C. Faloutsos (2012) 82

---

---

---

---

---

---

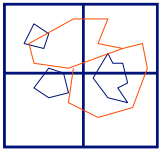
---

---

CMU SCS

## z-ordering - regions

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



15-826 Copyright: C. Faloutsos (2012) 83

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - regions

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

Naive algorithm:  $O(N * M)$   
Something faster?

15-826 Copyright: C. Faloutsos (2012) 84

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - regions

Spatial joins: find (quickly) all  
counties intersecting lakes

| z    | obj-id | etc |
|------|--------|-----|
| 0010 | ALG    |     |
| ...  | ...    |     |
| 1000 | WAS    |     |
| 11** | ALG    |     |

| z    | obj-id | etc |
|------|--------|-----|
| 0011 | Erie   |     |
| 0101 | Erie   |     |
| ...  |        |     |
| 10** | Ont.   |     |

15-826 Copyright: C. Faloutsos (2012) 85

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - regions

Spatial joins: find (quickly) all  
counties intersecting lakes

Solution: merge the lists of (sorted) z-values,  
 looking for the prefix property

footnote#1: '\*' needs careful treatment  
 footnote#2: need dup. elimination

15-826 Copyright: C. Faloutsos (2012) 86

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - Detailed outline

- spatial access methods
  - z-ordering
    - main idea - 3 methods
    - use w/ B-trees; algorithms (range, knn queries ...)
    - non-point (eg., region) data
    - analysis; variations
  - R-trees
  - ...

➡

15-826 Copyright: C. Faloutsos (2012) 87

---

---

---

---

---

---

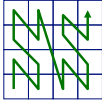
---

---

CMU SCS

## z-ordering - variations

Q: is z-ordering the best we can do?



15-826 Copyright: C. Faloutsos (2012) 88

---

---

---

---

---

---

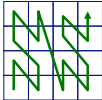
---

---

CMU SCS

## z-ordering - variations

Q: is z-ordering the best we can do?  
A: probably not - occasional long 'jumps'  
Q: then?



15-826 Copyright: C. Faloutsos (2012) 89

---

---

---

---

---

---

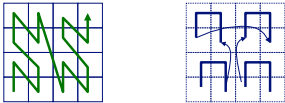
---

---

CMU SCS

## z-ordering - variations

Q: is z-ordering the best we can do?  
A: probably not - occasional long 'jumps'  
Q: then? A1: Gray codes



15-826 Copyright: C. Faloutsos (2012) 90

---

---

---

---

---

---

---

---

CMU SCS

### (Gray codes)

- Ingenious way to spot flickering LED – binary:

|      |     |   |
|------|-----|---|
|      | 000 | 0 |
|      | 001 | 1 |
|      | 010 | 2 |
| 3.5V | 011 | 3 |
| →    | 100 | 4 |
|      | 101 | 5 |
|      | 110 | 6 |
|      | 111 | 7 |

F. Gray. *Pulse code communication*,  
March 17, 1953  
[U.S. Patent 2,632,058](#)

15-826

Copyright: C. Faloutsos (2012)

91

---

---

---

---

---

---

---

---

CMU SCS

### (Gray codes)

- Ingenious way to spot flickering LED

|   |
|---|
| 0 |
| 1 |

15-826

Copyright: C. Faloutsos (2012)

92

---

---

---

---

---

---

---

---

CMU SCS

### (Gray codes)

- Ingenious way to spot flickering LED

|   |    |
|---|----|
| 0 | .0 |
| 1 | .1 |
|   | .. |
|   | .. |

15-826

Copyright: C. Faloutsos (2012)

93

---

---

---

---

---

---

---

---

CMU SCS

### (Gray codes)

- Ingenious way to spot flickering LED

|   |    |
|---|----|
| 0 | .0 |
| 1 | .1 |
|   | .. |
|   | .. |

15-826 Copyright: C. Faloutsos (2012) 94

---

---

---

---

---

---

---

---

CMU SCS

### (Gray codes)

- Ingenious way to spot flickering LED

|   |    |
|---|----|
| 0 | .0 |
| 1 | .1 |
|   | .1 |
|   | .0 |

15-826 Copyright: C. Faloutsos (2012) 95

---

---

---

---

---

---

---

---

CMU SCS

### (Gray codes)

- Ingenious way to spot flickering LED

|   |    |
|---|----|
| 0 | 00 |
| 1 | 01 |
|   | 11 |
|   | 10 |

15-826 Copyright: C. Faloutsos (2012) 96

---

---

---

---

---

---

---

---

CMU SCS

## (Gray codes)

- Ingenious way to spot flickering LED

|   |    |     |   |
|---|----|-----|---|
| 0 | 00 | 000 | 0 |
| 1 | 01 | 001 | 1 |
|   | 11 | 011 | 2 |
|   | 10 | 010 | 3 |
|   |    | 110 | 4 |
|   |    | 111 | 5 |
|   |    | 101 | 6 |
|   |    | 100 | 7 |

15-826 Copyright: C. Faloutsos (2012) 97

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - variations

Q: is z-ordering the best we can do?

A: probably not - occasional long 'jumps'

Q: then? A1: Gray codes – CAN WE DO BETTER?

15-826 Copyright: C. Faloutsos (2012) 98

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - variations

A2: Hilbert curve! (a.k.a. Hilbert-Peano curve)

15-826 Copyright: C. Faloutsos (2012) 99

---

---

---

---

---

---

---

---

CMU SCS

**(break)**



David Hilbert  
(1862-1943)



Giuseppe Peano  
(1858-1932)

15-826 Copyright: C. Faloutsos (2012) 100

---

---

---

---

---

---

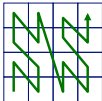
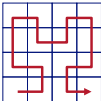
---

---

CMU SCS

**z-ordering - variations**

‘Looks’ better (never long jumps). How to derive it?

15-826 Copyright: C. Faloutsos (2012) 101

---

---

---

---

---

---

---

---

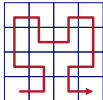
CMU SCS

**z-ordering - variations**

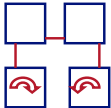
‘Looks’ better (never long jumps). How to derive it?



order-1



order-2



... order (n+1)

15-826 Copyright: C. Faloutsos (2012) 102

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - variations

Q: function for the Hilbert curve ( $h = f(x,y)$ )?

A: bit-shuffling, followed by post-processing, to account for rotations. Linear on # bits.

See textbook, for pointers to code/ algorithms (eg., [Jagadish, 90])

15-826 Copyright: C. Faloutsos (2012) 103

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - variations

Q: how about Hilbert curve in 3-d? n-d?

A: Exists (and is not unique!). Eg., 3-d, order-1 Hilbert curves (Hamiltonian paths on cube)



#1



#2

15-826 Copyright: C. Faloutsos (2012) 104

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - Detailed outline

- spatial access methods
  - z-ordering
    - main idea - 3 methods
    - use w/ B-trees; algorithms (range, knn queries ...)
    - non-point (eg., region) data
    - analysis; variations
  - R-trees
  - ...

15-826 Copyright: C. Faloutsos (2012) 105

---

---

---

---

---

---

---

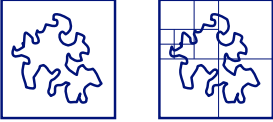
---

CMU SCS

### z-ordering - analysis

Q: How many pieces ('quad-tree blocks') per region?

A: proportional to perimeter (surface etc)



15-826 Copyright: C. Faloutsos (2012) 106

---

---

---

---

---

---

---

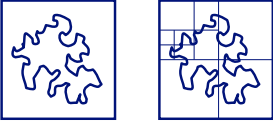
---

CMU SCS

### z-ordering - analysis

(How long is the coastline, say, of England?)

Paradox: The answer changes with the yard-stick -> fractals ...)



15-826 Copyright: C. Faloutsos (2012) 107

---

---

---

---

---

---

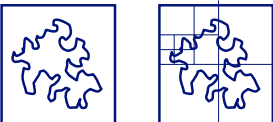
---

---

CMU SCS

### z-ordering - analysis

Q: Should we decompose a region to full detail (and store in B-tree)?



15-826 Copyright: C. Faloutsos (2012) 108

---

---

---

---

---

---

---

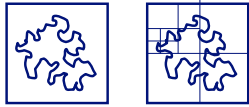
---

CMU SCS

### z-ordering - analysis

Q: Should we decompose a region to full detail (and store in B-tree)?

A: NO! approximation with 1-3 pieces/z-values is best [Orenstein90]



15-826 Copyright: C. Faloutsos (2012) 109

---

---

---

---

---

---

---

---

CMU SCS

### z-ordering - analysis

Q: how to measure the 'goodness' of a curve?



15-826 Copyright: C. Faloutsos (2012) 110

---

---

---

---

---

---

---

---

CMU SCS

### z-ordering - analysis

Q: how to measure the 'goodness' of a curve?

A: e.g., avg. # of runs, for range queries



4 runs  
(#runs ~ #disk accesses on B-tree)

3 runs

15-826 Copyright: C. Faloutsos (2012) 111

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - analysis

Q: So, is Hilbert really better?  
 A: 27% fewer runs, for 2-d (similar for 3-d)

Q: are there formulas for #runs, #of quadtree blocks etc?  
 A: Yes ([Jagadish; Moon+ etc] see textbook)

15-826 Copyright: C. Faloutsos (2012) 112

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - fun observations

Hilbert and z-ordering curves: “space filling curves”: eventually, they visit every point in n-d space - therefore:





order-1      order-2      ... order (n+1)

15-826 Copyright: C. Faloutsos (2012) 113

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - fun observations

... they show that the plane has as many points as a line (-> headaches for 1900's mathematics/topology). (fractals, again!)





order-1      order-2      ... order (n+1)

15-826 Copyright: C. Faloutsos (2012) 114

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - fun observations

Observation #2: Hilbert (like) curve for video encoding [Y. Matias+, CRYPTO '87]:  
Given a frame, visit its pixels in randomized hilbert order; compress; and transmit




15-826 Copyright: C. Faloutsos (2012) 115

---

---

---

---

---

---

---

---

CMU SCS

## z-ordering - fun observations

In general, Hilbert curve is great for preserving distances, clustering, vector quantization etc

15-826 Copyright: C. Faloutsos (2012) 116

---

---

---

---

---

---

---

---

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 (2012) 117

---

---

---

---

---

---

---

---

CMU/SCS

## Conclusions

- z-ordering is a great idea (n-d points  $\rightarrow$  1-d points; feed to B-trees)
- used by TIGER system  
<http://www.census.gov/geo/www/tiger/>
- and (most probably) by other GIS products
- works great with low-dim points

15-826

Copyright: C. Faloutsos (2012)

118

---

---

---

---

---

---

---