


15-826: Multimedia Databases and Data Mining

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



Must-read material

- Textbook, Chapter 5.1
- Ramakrinshan+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 (2009) 2



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

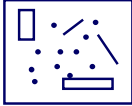
- primary key indexing
- secondary key / multi-key indexing
- ➔ spatial access methods
 - problem defn
 - z-ordering
 - R-trees
 - ...
- text
- ...

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

CMU SCS

Spatial Access Methods - problem

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

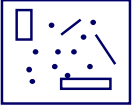


15-826 Copyright: C. Faloutsos (2009) 5

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)

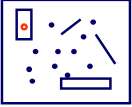


15-826 Copyright: C. Faloutsos (2009) 6

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)

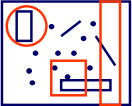


15-826 Copyright: C. Faloutsos (2009) 7

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 (2009) 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)

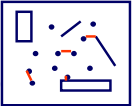


15-826 Copyright: C. Faloutsos (2009) 9

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 ϵ)



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

CMU SCS

SAMs - motivation


- Q: applications?

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

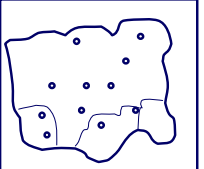
CMU SCS

SAMs - motivation

traditional DB



GIS



age

salary

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

CMU SCS

SAMs - motivation

traditional DB GIS

age

salary

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

CMU SCS

SAMs - motivation

CAD/CAM

find elements too close to each other

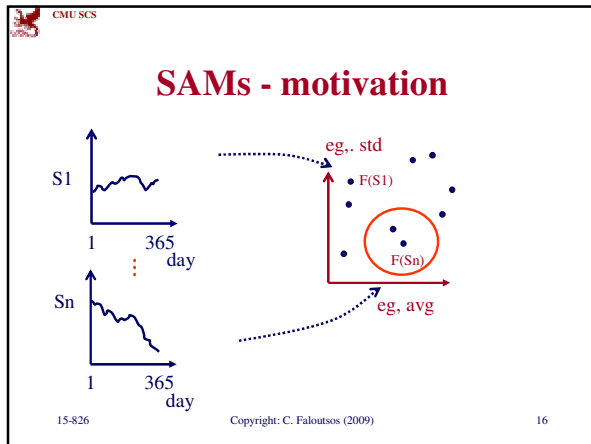
15-826 Copyright: C. Faloutsos (2009) 14

CMU SCS

SAMs - motivation

CAD/CAM

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



- ### Indexing - Detailed outline
- primary key indexing
 - secondary key / multi-key indexing
 - spatial access methods
 - problem defn
 - ➔ - z-ordering
 - R-trees
 - ...
 - text
 - ...


- ### 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)
-
- The diagram shows a square grid of points. A path is drawn through the points, starting from the top-left and moving in a zig-zag pattern towards the bottom-right, illustrating a traversal or ordering scheme for the points.

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 (2009) 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 (2009) 20

CMU SCS

z-ordering

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



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

CMU SCS

z-ordering

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

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

CMU SCS

z-ordering

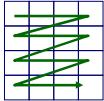
Q2.1: how to map n -d cells to 1-d cells?
 A: row-wise
 Q: is it good?

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

CMU SCS

z-ordering

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

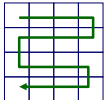


15-826 Copyright: C. Faloutsos (2009) 25

CMU SCS

z-ordering

Q: How about the 'snake' curve?




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

CMU SCS

z-ordering

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



2³²

2³²

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

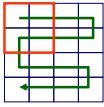
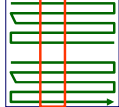
CMU SCS

z-ordering

Q: Why are those curves 'bad'?

A: no distance preservation (~ clustering)

Q: solution?

2^{32}

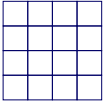
2^{32}

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

CMU SCS

z-ordering

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



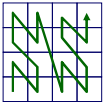
15-826 Copyright: C. Faloutsos (2009) 29

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 (2009) 30

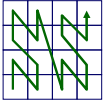
CMU SCS

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 (2009) 31

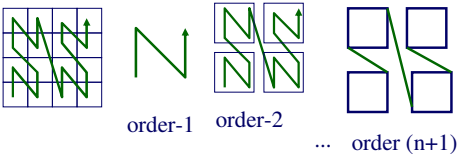
CMU SCS

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)

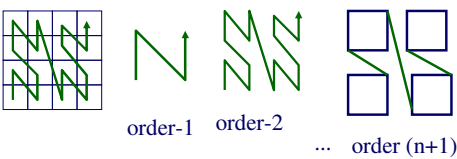
15-826 Copyright: C. Faloutsos (2009) 32

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 (2009) 33

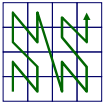
CMU SCS

z-ordering

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

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

A: 3 (equivalent) answers!



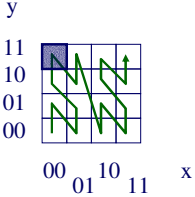
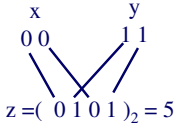
Method #2?

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

CMU SCS

z-ordering

bit-shuffling

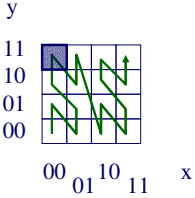
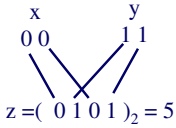
$z = (0101)_2 = 5$

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

CMU SCS

z-ordering

bit-shuffling

$z = (0101)_2 = 5$

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

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

CMU SCS

z-ordering

bit-shuffling

How about n -d spaces?

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

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 (2009) 38

CMU SCS

z-ordering

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

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

CMU SCS

z-ordering

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

1

0

N

S

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

CMU SCS

z-ordering

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

W E

1

0

N

S

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

CMU SCS

z-ordering

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

W E

1

0

N

S

method#1: 14

method#2: shuffle(11;10)=

$(1110)_2 = 14$

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

CMU SCS

z-ordering

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

W E

	0	1	
1			
0			
	0	1	

N
S

method#1: 14
 method#2: shuffle(11;10)=
 (1110)₂ = 14
 method#3: EN;ES = ... = 14

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

CMU SCS

z-ordering - Detailed outline

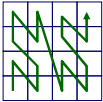
- spatial access methods
 - z-ordering
 - main idea - 3 methods
 - use w/ B-trees; algorithms (range, knn queries ...)
 - non-point (eg., region) data
 - analysis; variations
 - R-trees
 - ...

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

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 (2009) 45

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 (2009) 46

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 (2009) 47

CMU SCS

z-ordering - usage & algo's

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

PGH

SF

z	cname	etc
5	SF	
12	PGH	

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

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 (2009) 49

CMU SCS

z-ordering - usage & algo's

Q2: range queries?

SF

z	cname	etc
5	SF	
12	PGH	

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

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 (2009) 51

CMU SCS

z-ordering - usage & algo's

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

SF PGH 9,11-15

z	cname	etc
5	SF	
12	PGH	

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

CMU SCS

z-ordering - usage & algo's

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

A: Augment the query!

SF PGH 9,11-15 -> 8-15

z	cname	etc
5	SF	
12	PGH	

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

CMU SCS

z-ordering - usage & algo's

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

9,11-15

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

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 (2009) 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 (2009) 56

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 (2009) 57

CMU SCS

z-ordering - usage & algo's

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

SF

PGH

z	cname	etc
5	SF	
12	PGH	

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

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

z	cname	etc
5	SF	
12	PGH	

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

CMU SCS

z-ordering - usage & algo's

... ask a range query.

SF

PGH

nn wrt z-value

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

CMU SCS

z-ordering - usage & algo's

... ask a range query.

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

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 (2009) 62

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 (2009) 63

CMU SCS

z-ordering - regions

Q: z-value for a region?

A B

C

$z_B = ??$

$z_C = ??$

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

CMU SCS

z-ordering - regions

Q: z-value for a region?

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

A B

C

$z_B = ??$

$z_C = ??$

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

CMU SCS

z-ordering - regions

Q: z-value for a region?

W E B

A N

C

$z_B = 11^{**}$ “don't care”

$z_C = ??$

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

CMU SCS

z-ordering - regions

“don't care”

Q: z-value for a region? $z_B = 11^{**}$

$z_C = \{0010; 1000\}$

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

CMU SCS

z-ordering - regions

Q: How to store in B-tree?

Q: How to search (range etc queries)

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

CMU SCS

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 (2009) 69

CMU SCS

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 (2009) 70

CMU SCS

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 (2009) 71

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 (2009) 72

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 (2009) 73

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 (2009) 74

CMU SCS

z-ordering - regions

- $r2$ will either contain completely, or avoid completely $r1$.
- it will contain $r1$, if $z2$ is the prefix of $z1$

$1100 \text{ ?? } 11^{**}$

region of $z1$:
completely contained in
region of $z2$

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

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 (2009) 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 - 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 (2009) 77

CMU SCS

z-ordering - regions

Drill (True/False). Given:

- $z_1 = 011001^{**}$
- $z_2 = 01^{*****}$
- $z_3 = 0100^{****}$

z_2

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

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 (2009) 79

CMU SCS

z-ordering - regions

Spatial joins: find (quickly) all
 counties intersecting lakes

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

CMU SCS

z-ordering - regions

Spatial joins: find (quickly) all
 counties intersecting lakes

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

CMU SCS

z-ordering - regions

Spatial joins: find (quickly) all
 counties intersecting lakes

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

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 (2009) 83

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 (2009) 84

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 (2009) 85

CMU SCS

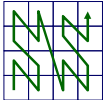
z-ordering - Detailed outline

- 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
 - ...

CMU SCS

z-ordering - variations

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



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

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 (2009) 88

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 (2009) 89

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED – binary:

	000	0
	001	1
	010	2
	011	3
3.5V	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 (2009) 90

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

0	
1	

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

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

0	.0
1	.1
	..
	..

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

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

0	.0
1	.1
	..
	..

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

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

0	.0	
1	.1	
	.1	
	.0	

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

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

0	00	
1	01	
	11	
	10	

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

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

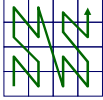
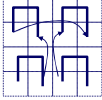
0	00	000	0
1	01	001	1
	11	011	2
	10	010	3
		110	4
		111	5
		101	6
		100	7

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

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?

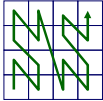
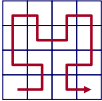



15-826 Copyright: C. Faloutsos (2009) 97

CMU SCS

z-ordering - variations

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

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

CMU SCS

(break)



David Hilbert
(1862-1943)



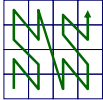
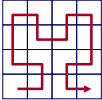
Giuseppe Peano
(1858-1932)

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

CMU SCS

z-ordering - variations

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


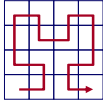
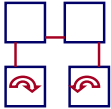



15-826 Copyright: C. Faloutsos (2009) 100

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 (2009) 101

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 (2009) 102


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 (2009) 103

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
 - ...

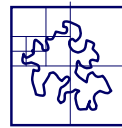
15-826 Copyright: C. Faloutsos (2009) 104

CMU SCS

z-ordering - analysis

Q: How many pieces ('quad-tree blocks') per region?
 A: proportional to perimeter (surface etc)



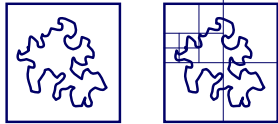


15-826 Copyright: C. Faloutsos (2009) 105

CMU SCS

z-ordering - analysis

(How long is the coastline, say, of England?
Paradox: The answer changes with the yardstick -> fractals ...)

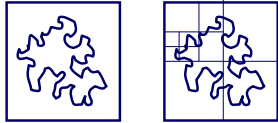


15-826 Copyright: C. Faloutsos (2009) 106

CMU SCS

z-ordering - analysis

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

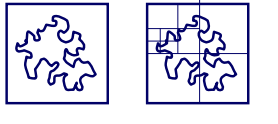


15-826 Copyright: C. Faloutsos (2009) 107

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]

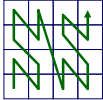
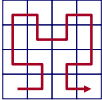


15-826 Copyright: C. Faloutsos (2009) 108

CMU SCS

z-ordering - analysis

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

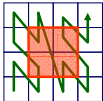
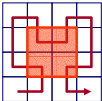
15-826 Copyright: C. Faloutsos (2009) 109

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 (2009) 110

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 (2009) 111

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 (2009) 112

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 (2009) 113

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 (2009) 114

CMU SCS

z-ordering - fun observations

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

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

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) 116

CMU SCS

Conclusions

- z-ordering is a great idea (n-d points -> 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 (2009) 117
