


CMU SCS

15-826: Multimedia Databases and Data Mining

Lecture#5: Multi-key and
Spatial Access Methods – II – z-ordering
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 (2017) 2




CMU SCS

Outline

Goal: ‘Find **similar / interesting** things’

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

15-826 Copyright: C. Faloutsos (2017) 3



CMU SCS

Indexing - Detailed outline

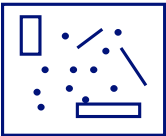
- primary key indexing
- secondary key / multi-key indexing
- ➔ • spatial access methods
 - problem dfn
 - z-ordering
 - R-trees
 - ...
- text
- ...

15-826 Copyright: C. Faloutsos (2017) 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??)

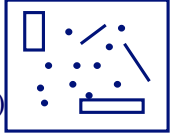


15-826 Copyright: C. Faloutsos (2017) 5

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)

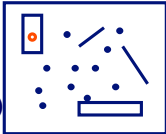


15-826 Copyright: C. Faloutsos (2017) 6

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)

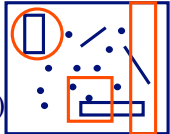


15-826 Copyright: C. Faloutsos (2017) 7

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)

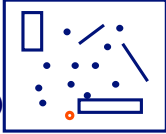


15-826 Copyright: C. Faloutsos (2017) 8

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)

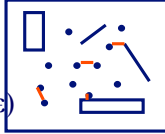


15-826 Copyright: C. Faloutsos (2017) 9

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 (2017) 10

CMU SCS

SAMs - motivation

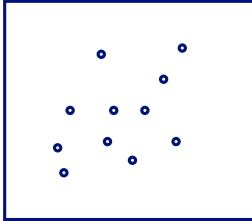
- Q: applications?

15-826 Copyright: C. Faloutsos (2017) 11

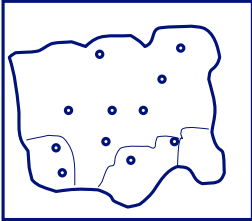
CMU SCS

SAMs - motivation

traditional DB



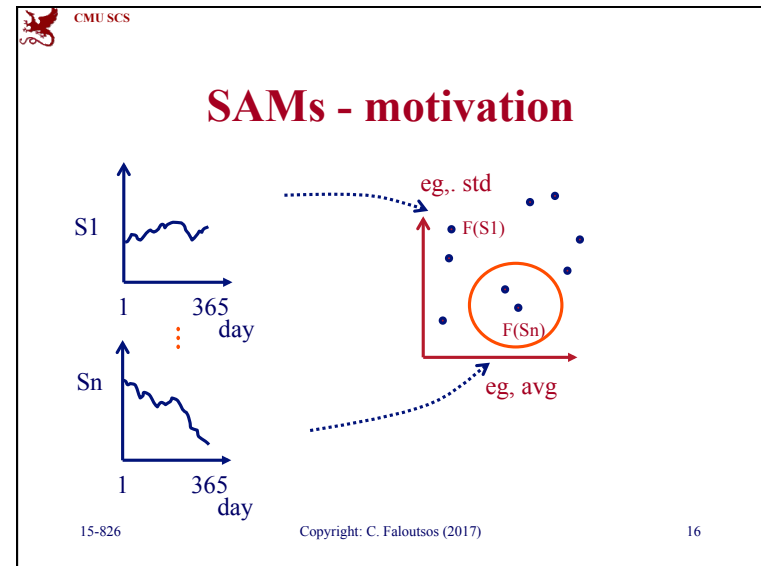
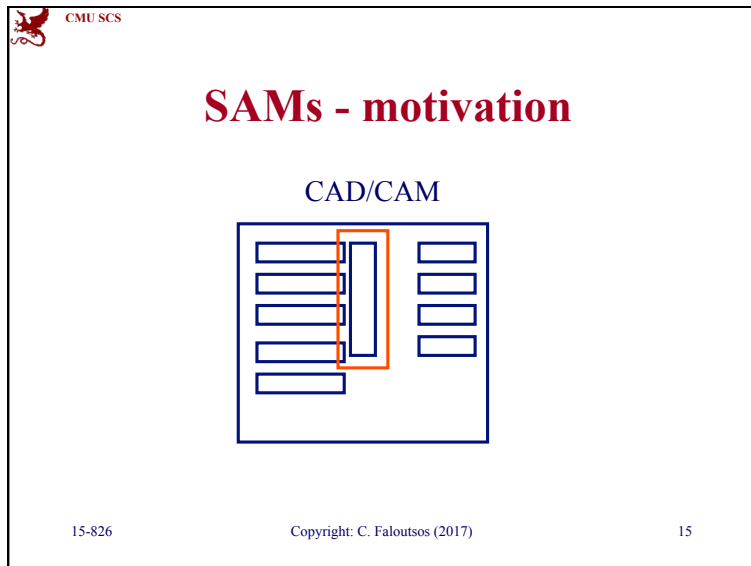
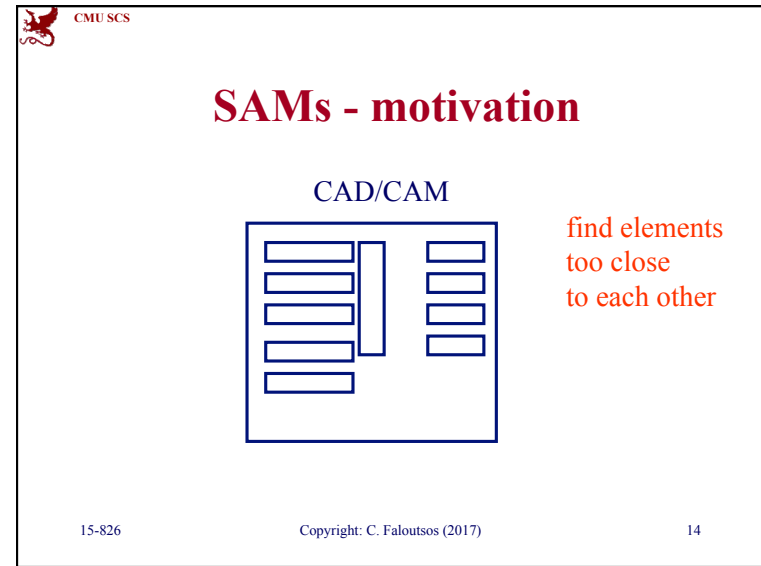
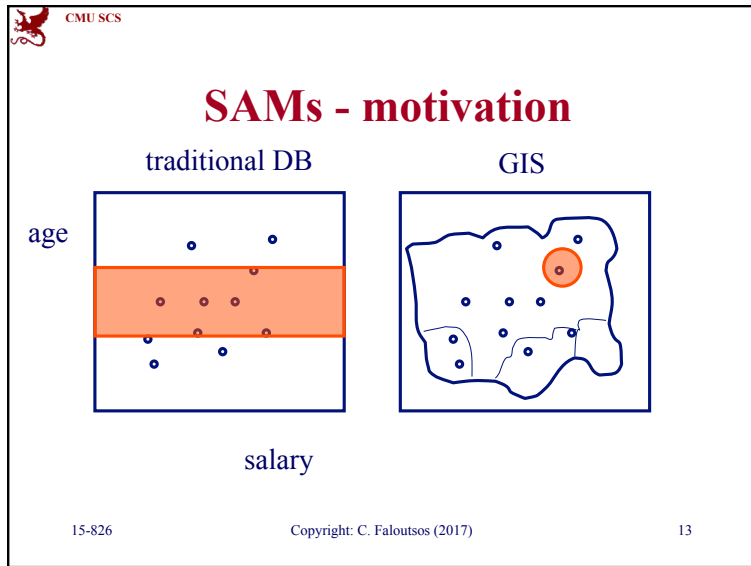
GIS



age

salary

15-826 Copyright: C. Faloutsos (2017) 12



CMU SCS

Indexing - Detailed outline

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

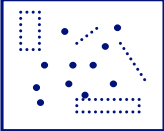
15-826 Copyright: C. Faloutsos (2017) 17

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 (2017) 18

CMU SCS

z-ordering

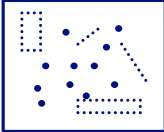
Q: how would you organize, e.g., n -dim points, on disk? (C points per disk page)

Hint: reduce the problem to 1-d points (!!)

Q1: why?

A:

Q2: how?



15-826 Copyright: C. Faloutsos (2017) 19

CMU SCS

z-ordering

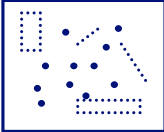
Q: how would you organize, e.g., n -dim points, on disk? (C points per disk page)

Hint: reduce the problem to 1-d points (!!)

Q1: why?

A: B-trees!

Q2: how?

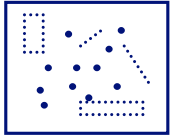


15-826 Copyright: C. Faloutsos (2017) 20

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 (2017) 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?

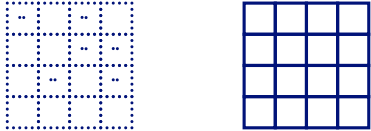


15-826 Copyright: C. Faloutsos (2017) 22

CMU SCS

z-ordering

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

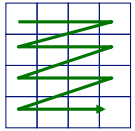


15-826 Copyright: C. Faloutsos (2017) 23

CMU SCS

z-ordering

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

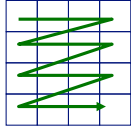


15-826 Copyright: C. Faloutsos (2017) 24

CMU SCS

z-ordering

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

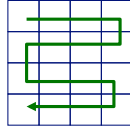


15-826 Copyright: C. Faloutsos (2017) 25

CMU SCS

z-ordering

Q: How about the 'snake' curve?

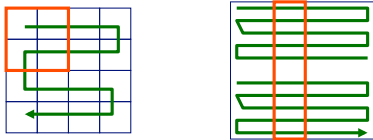


15-826 Copyright: C. Faloutsos (2017) 26

CMU SCS

z-ordering

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



2^{32}

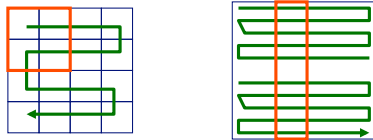
2^{32}

15-826 Copyright: C. Faloutsos (2017) 27

CMU SCS

z-ordering

Q: Why are those curves 'bad' ?
 A: no distance preservation (~ clustering)
 Q: solution?



2^{32}

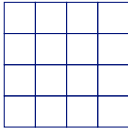
2^{32}

15-826 Copyright: C. Faloutsos (2017) 28

CMU SCS

z-ordering

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



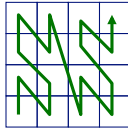
15-826 Copyright: C. Faloutsos (2017) 29

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 (2017) 30

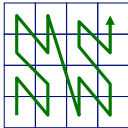
CMU SCS

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 (2017) 31

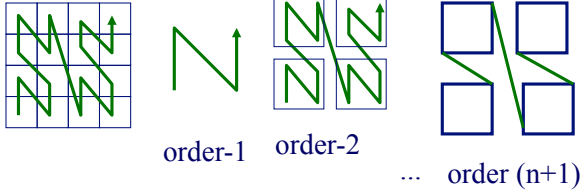
CMU SCS

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 (2017) 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 (2017) 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 (2017) 34

CMU SCS

z-ordering

bit-shuffling

x y
0 0 1 1

15-826 Copyright: C. Faloutsos (2017) 35

CMU SCS

z-ordering

bit-shuffling

x y
0 0 1 1

$z = (0101)_2 = 5$

15-826 Copyright: C. Faloutsos (2017) 36

CMU SCS

z-ordering

bit-shuffling

y

11

10

01

00

x

00

01

10

11

x y

00 11

 / \
 \
 /

$z = (0101)_2 = 5$

How about the reverse:

$(x,y) = g(z) ?$

15-826 Copyright: C. Faloutsos (2017) 37

CMU SCS

z-ordering

bit-shuffling

y

11

10

01

00

x

00

01

10

11

x y

00 11

 / \
 \
 /

$z = (0101)_2 = 5$

How about n -d spaces?

15-826 Copyright: C. Faloutsos (2017) 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 (2017) 39

CMU SCS

z-ordering

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

W E

1

0

0 1

N

S

15-826 Copyright: C. Faloutsos (2017) 40

CMU SCS

z-ordering

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

15-826 Copyright: C. Faloutsos (2017) 41

CMU SCS

z-ordering

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

15-826 Copyright: C. Faloutsos (2017) 42

CMU SCS

z-ordering

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

method#1: 14
 method#2: $\text{shuffle}(11;10) = (1110)_2 = 14$

15-826 Copyright: C. Faloutsos (2017) 43

CMU SCS

z-ordering

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

method#1: 14
 method#2: $\text{shuffle}(11;10) = (1110)_2 = 14$
 method#3: $EN;ES = \dots = 14$

15-826 Copyright: C. Faloutsos (2017) 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
 - ...

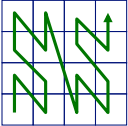
15-826 Copyright: C. Faloutsos (2017) 45

CMU SCS

z-ordering - usage & algo's

Q1: How to store on disk?
A:

Q2: How to answer range queries etc



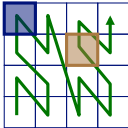
15-826 Copyright: C. Faloutsos (2017) 46

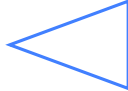
CMU SCS

z-ordering - usage & algo's

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

SF





z	cname	etc
5	SF	
12	PGH	

15-826 Copyright: C. Faloutsos (2017) 47

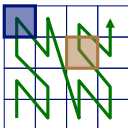
CMU SCS

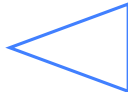
z-ordering - usage & algo's

MAJOR ADVANTAGES w/ B-tree:

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

SF





z	cname	etc
5	SF	
12	PGH	

15-826 Copyright: C. Faloutsos (2017) 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 (2017) 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 (2017) 50

CMU SCS

z-ordering - usage & algo's

Q2: range queries?

SF

z	cname	etc
5	SF	
12	PGH	

15-826 Copyright: C. Faloutsos (2017) 51

CMU SCS

z-ordering - usage & algo's

Q2: range queries?

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

SF

z	cname	etc
5	SF	
12	PGH	

15-826 Copyright: C. Faloutsos (2017) 52

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 (2017) 53

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 (2017) 54

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 (2017) 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

12-15

9,11-15

15-826 Copyright: C. Faloutsos (2017) 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 (2017) 57

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 (2017) 58

CMU SCS

z-ordering - usage & algo's

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

PGH

z	cname	etc
5	SF	
12	PGH	

15-826 Copyright: C. Faloutsos (2017) 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 ...

PGH

z	cname	etc
5	SF	
12	PGH	

15-826 Copyright: C. Faloutsos (2017) 60

CMU SCS

z-ordering - usage & algo's

... ask a range query.

SF

PGH

nn wrt z-value

15-826 Copyright: C. Faloutsos (2017) 61

CMU SCS

z-ordering - usage & algo's

... ask a range query.

SF

PGH

nn wrt z-value

15-826 Copyright: C. Faloutsos (2017) 62

CMU SCS

z-ordering - usage & algo's

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

SF

PGH

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

15-826 Copyright: C. Faloutsos (2017) 63

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 (2017) 64

CMU SCS

z-ordering - regions

Q: z-value for a region?

$z_B = ??$

$z_C = ??$

15-826 Copyright: C. Faloutsos (2017) 65

CMU SCS

z-ordering - regions

Q: z-value for a region?

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

$z_B = ??$

$z_C = ??$

15-826 Copyright: C. Faloutsos (2017) 66

CMU SCS

z-ordering - regions "don't care"

Q: z-value for a region?

$z_B = 11^{**}$ ← "don't care"

$z_C = ??$

15-826 Copyright: C. Faloutsos (2017) 67

CMU SCS

z-ordering - regions "don't care"

Q: z-value for a region?

$z_B = 11^{**}$ ← "don't care"

$z_C = \{0010; 1000\}$

15-826 Copyright: C. Faloutsos (2017) 68

CMU SCS

z-ordering - regions

Q: How to store in B-tree?
 Q: How to search (range etc queries)

A B
C

15-826 Copyright: C. Faloutsos (2017) 69

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	

A B
C

15-826 Copyright: C. Faloutsos (2017) 70

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	

A B
C

15-826 Copyright: C. Faloutsos (2017) 71

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	

A B
C

15-826 Copyright: C. Faloutsos (2017) 72

CMU SCS

z-ordering - regions

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

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

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

15-826 Copyright: C. Faloutsos (2017) 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 (2017) 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 (2017) 75

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 (2017) 76

CMU SCS

z-ordering - regions

Drill (True/False). Given:

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

T/F r2 contains r1
 T/F r3 contains r1
 T/F r3 contains r2

15-826 Copyright: C. Faloutsos (2017) 77

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)

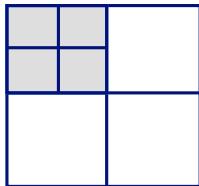
15-826 Copyright: C. Faloutsos (2017) 78

CMU SCS

z-ordering - regions

Drill (True/False). Given:

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



z2

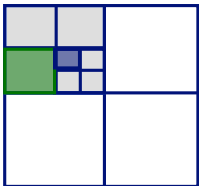
15-826 Copyright: C. Faloutsos (2017) 79

CMU SCS

z-ordering - regions

Drill (True/False). Given:

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



z2

z3

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 (2017) 80

CMU SCS

z-ordering - regions

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

15-826 Copyright: C. Faloutsos (2017) 81

CMU SCS

z-ordering - regions

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

15-826 Copyright: C. Faloutsos (2017) 82

CMU SCS

z-ordering - regions

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

15-826 Copyright: C. Faloutsos (2017) 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 (2017) 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 (2017) 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 (2017) 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
 - ...

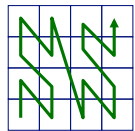
➔

15-826 Copyright: C. Faloutsos (2017) 87

CMU SCS

z-ordering - variations

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

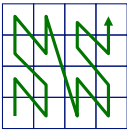


15-826 Copyright: C. Faloutsos (2017) 88

CMU SCS

z-ordering - variations

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

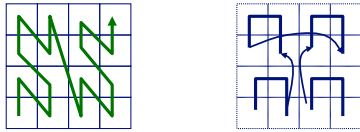


15-826 Copyright: C. Faloutsos (2017) 89

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 (2017) 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](https://www.uspto.gov/patent/publications/details/2632058)

15-826 Copyright: C. Faloutsos (2017) 91

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

0
1

15-826 Copyright: C. Faloutsos (2017) 92

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

0	.0
1	.1
	..
	..

15-826 Copyright: C. Faloutsos (2017) 93

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

0	.0
1	.1
	..
	..

15-826 Copyright: C. Faloutsos (2017) 94

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

0	.0
1	.1
	.1
	.0

15-826 Copyright: C. Faloutsos (2017) 95

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

0	00
1	01
	11
	10

15-826 Copyright: C. Faloutsos (2017) 96

CMU SCS

(Gray codes)

- Ingenious way to spot flickering LED

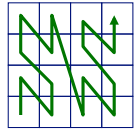
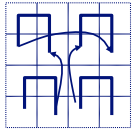
0	00	000	0
1	01	001	1
	11	011	2
	10	010	3
		110	4
		111	5
		101	6
		100	7

15-826 Copyright: C. Faloutsos (2017) 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?

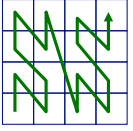
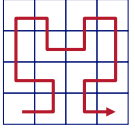



15-826 Copyright: C. Faloutsos (2017) 98

CMU SCS

z-ordering - variations

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

15-826 Copyright: C. Faloutsos (2017) 99

CMU SCS

(break)




David Hilbert
(1862-1943)

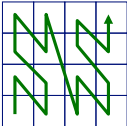
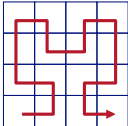
Giuseppe Peano
(1858-1932)

15-826 Copyright: C. Faloutsos (2017) 100

CMU SCS

z-ordering - variations

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


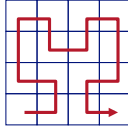
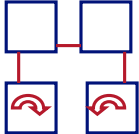



15-826 Copyright: C. Faloutsos (2017) 101

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 (2017) 102

CMU SCS

z-ordering - variations

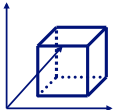
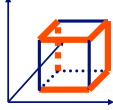
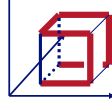
Q: function for the Hilbert curve ($h = f(x,y)$)?
 A: bit-shuffling, followed by post-processing, to account for rotations. Linear on # bits.
 See textbook, for pointers to code/ algorithms (eg., [Jagadish, 90])

15-826 Copyright: C. Faloutsos (2017) 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 (2017) 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
 - ...

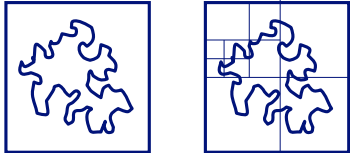
15-826 Copyright: C. Faloutsos (2017) 105

CMU SCS

z-ordering - analysis

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

A: proportional to perimeter (surface etc)

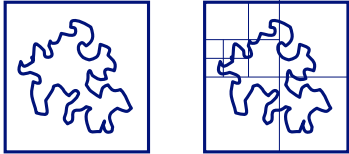


15-826 Copyright: C. Faloutsos (2017) 106

CMU SCS

z-ordering - analysis

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

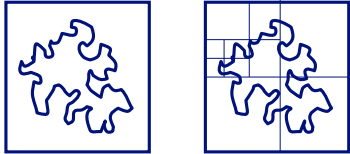


15-826 Copyright: C. Faloutsos (2017) 107

CMU SCS

z-ordering - analysis

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



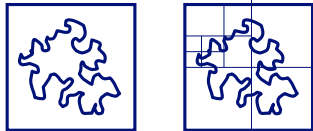
15-826 Copyright: C. Faloutsos (2017) 108

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]

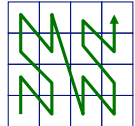
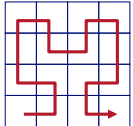


15-826 Copyright: C. Faloutsos (2017) 109

CMU SCS

z-ordering - analysis

Q: how to measure the ‘goodness’ of a curve?

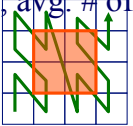
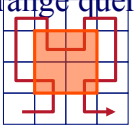
15-826 Copyright: C. Faloutsos (2017) 110

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 (2017) 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 (2017) 112

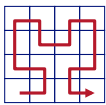
CMU SCS

z-ordering - fun observations

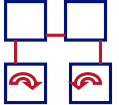
Hilbert and z-ordering curves: “space filling curves”: eventually, they visit every point in n-d space - therefore:



order-1



order-2




... order (n+1)

15-826 Copyright: C. Faloutsos (2017) 113

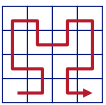
CMU SCS

z-ordering - fun observations

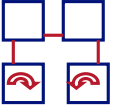
... they show that the plane has as many points as a line (-> headaches for 1900's mathematics/topology). (fractals, again!)



order-1



order-2



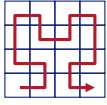
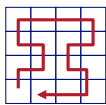

... order (n+1)

15-826 Copyright: C. Faloutsos (2017) 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 (2017) 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 (2017) 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 (2017) 117



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 (2017) 118