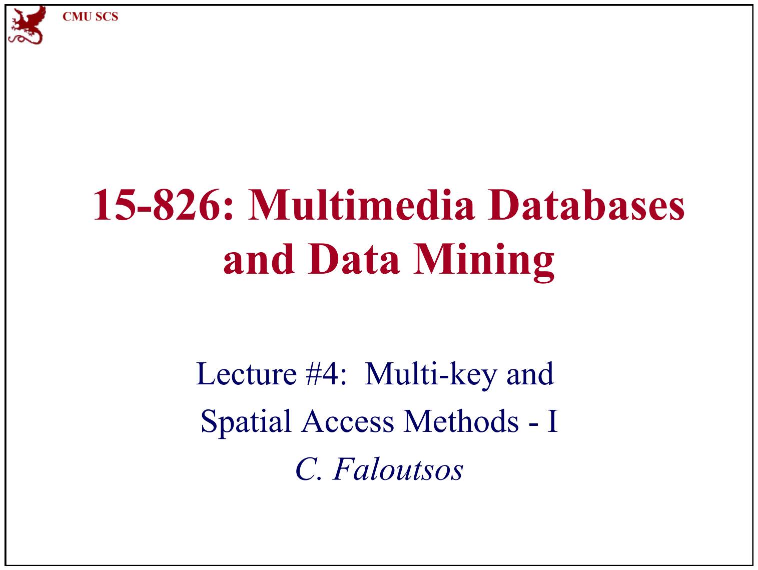


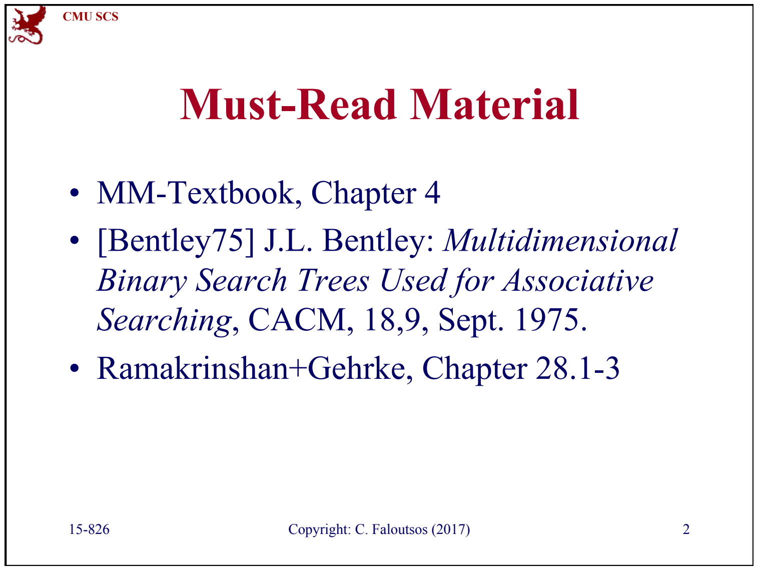
CMU SCS

15-826: Multimedia Databases and Data Mining

Lecture #4: Multi-key and

Spatial Access Methods - I

C. Faloutsos

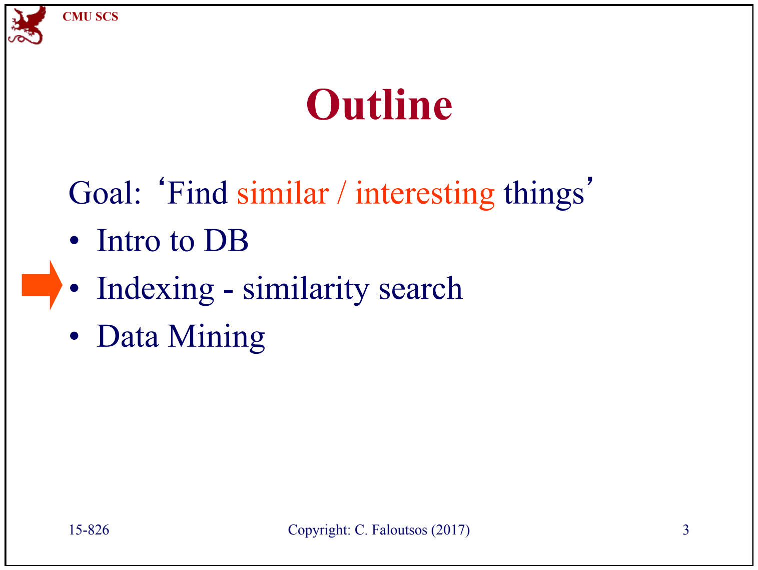


CMU SCS

Must-Read Material

- MM-Textbook, Chapter 4
- [Bentley75] J.L. Bentley: *Multidimensional Binary Search Trees Used for Associative Searching*, CACM, 18,9, Sept. 1975.
- Ramakrinshan+Gehrke, Chapter 28.1-3

15-826 Copyright: C. Faloutsos (2017) 2



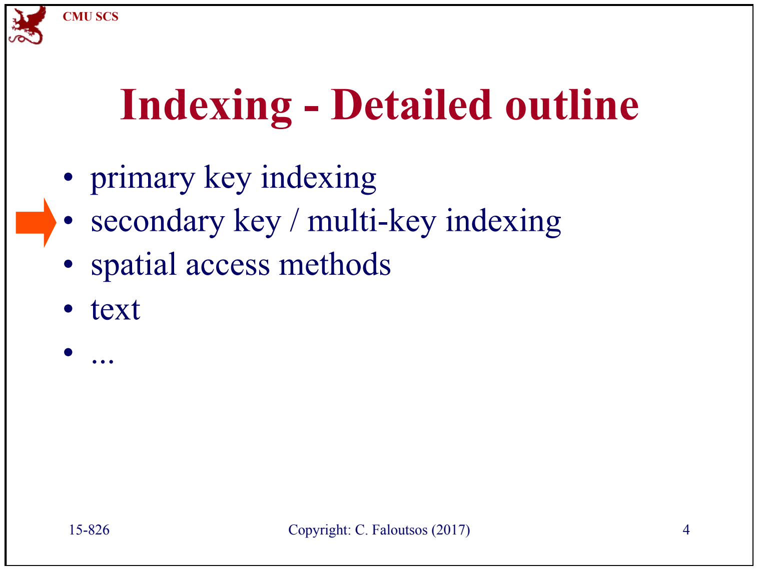
CMU SCS

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
- text
- ...

15-826 Copyright: C. Faloutsos (2017) 4

CMU SCS

Sec. key indexing

- attributes w/ duplicates (eg., EMPLOYEES, with 'job-code')
- Query types:
  - exact match
  - partial match
    - 'job-code'= 'PGM' and 'dept'='R&D'
  - range queries
    - 'job-code'='ADMIN' and salary < 50K

CMU SCS

Sec. key indexing

- Query types - cont'd
  - boolean
    - 'job-code'='ADMIN' or salary>20K
  - nn
    - salary ~ 30K

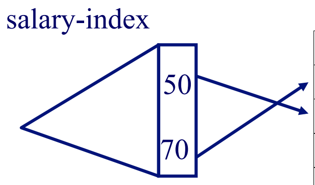
CMU SCS

Solution?

CMU SCS

Solution?

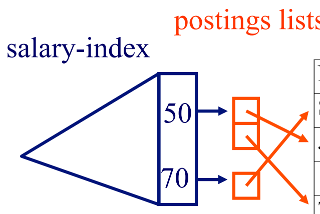
- Inverted indices (usually, w/ B-trees)
- Q: how to handle duplicates?



Name	Job-code	Salary	Dept
Smith	PGM	70	R&D
Jones	ADMIN	50	R&D
….			
Tomson	ENG	50	SALES

Solution

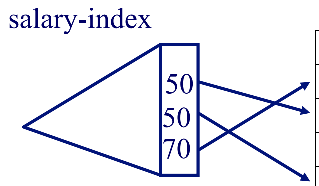
- A#1: eg., with postings lists



Name	Job-code	Salary	Dept
Smith	PGM	70	R&D
Jones	ADMIN	50	R&D
….			
Tomson	ENG	50	SALES

Solution

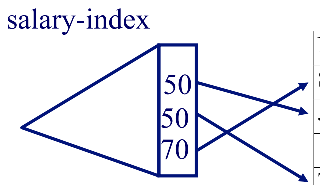
- A#2: modify B-tree code, to handle dup’s



Name	Job-code	Salary	Dept
Smith	PGM	70	R&D
Jones	ADMIN	50	R&D
….			
Tomson	ENG	50	SALES

How to handle Boolean Queries?

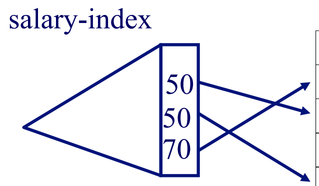
- eg., ‘sal=50 AND job-code=PGM’?



Name	Job-code	Salary	Dept
Smith	PGM	70	R&D
Jones	ADMIN	50	R&D
….			
Tomson	ENG	50	SALES

How to handle Boolean Queries?

– from indices, find lists of qual. record-ids
– merge lists (or check real records)



Name	Job-code	Salary	Dept
Smith	PGM	70	R&D
Jones	ADMIN	50	R&D
….			
Tomson	ENG	50	SALES

Sec. key indexing

- easily solved in commercial DBMS:

```
create index sal-index on
 EMPLOYEE (salary);
select * from EMPLOYEE
   where salary > 50 and
   job-code = 'ADMIN'
```

Sec. key indexing

- can create combined indices:

```
create index sj on
 EMPLOYEE( salary, job-code);
```

Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
  - main memory: quad-trees
  - main memory: k-d-trees
- spatial access methods
- text
- ...

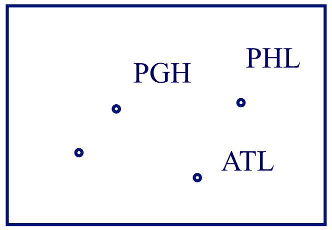
Quad-trees

- problem: find cities within 100mi from Pittsburgh
- assumption: all fit in main memory
- Q: how to answer such queries quickly?

CMU SCS

Quad-trees

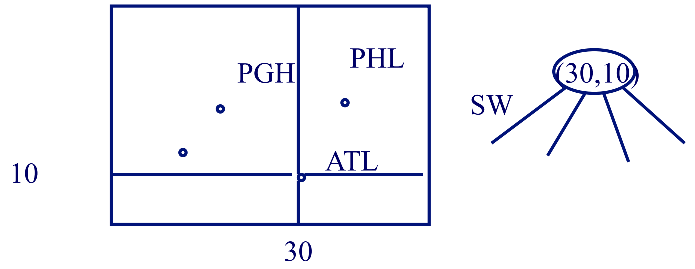
- A: recursive decomposition of space, e.g.:



CMU SCS

Quad-trees

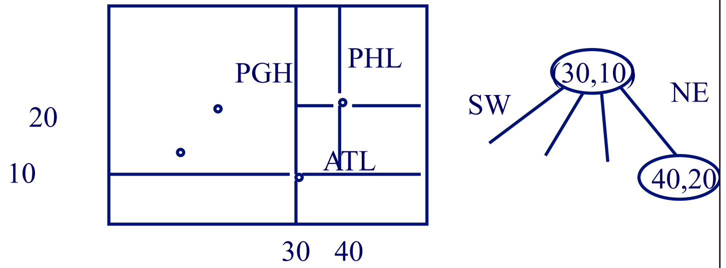
- A: recursive decomposition of space, e.g.:



CMU SCS

Quad-trees

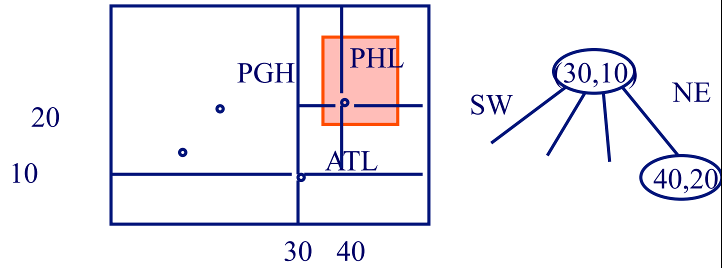
- A: recursive decomposition of space, e.g.:



CMU SCS

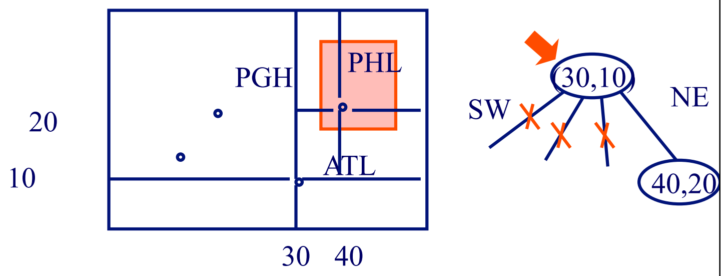
Quad-trees - search?

- find cities with (35<x<45, 15<y<25):



Quad-trees - search?

- find cities with (35<x<45, 15<y<25):



Quad-trees - search?

- pseudocode:

```
range-query( tree-ptr, range)
  if (tree-ptr == NULL) exit;
  if (tree-ptr->point within range){
    print tree-ptr->point}
  for each quadrant {
    if ( range intersects quadrant ) {
      range-query( tree-ptr->quadrant-ptr, range);
  }
```

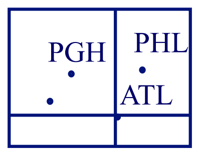
Quad-trees - k-nn search?

- k-nearest neighbor algo - more complicated:
  - find 'good' neighbors and put them in a stack
  - go to the most promising quadrant, and update the stack of neighbors
  - until we hit the leaves

Quad-trees - discussion

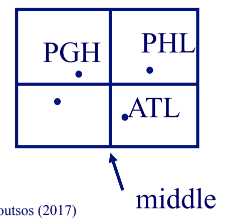
- great for 2- and 3-d spaces
- several variations, like fixed decomposition:

'adaptive'



'fixed'

z-ordering (later)



CMU SCS

Quad-trees - discussion

- but: unsuitable for higher-d spaces (why?)

CMU SCS

Quad-trees - discussion

- but: unsuitable for higher-d spaces (why?)
- A: 2^d pointers, per node!
- Q: how to solve this problem?
- A: k-d-trees!

CMU SCS

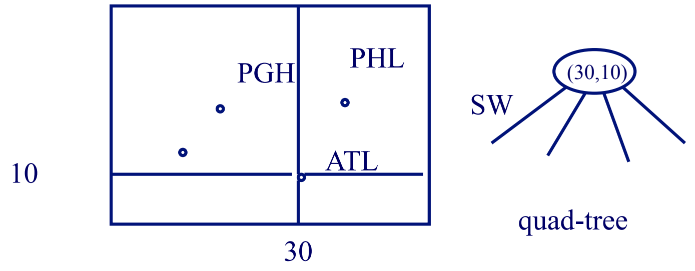
Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
  - main memory: quad-trees
  - main memory: k-d-trees
- spatial access methods
- text
- ...

CMU SCS

k-d-trees

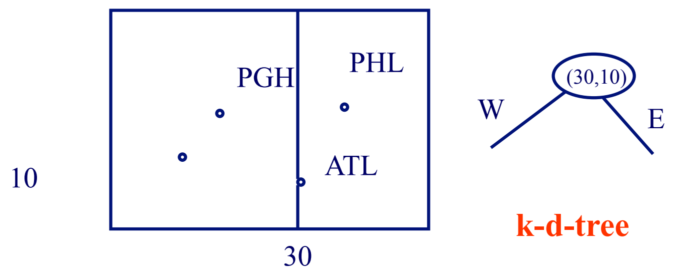
- Binary trees, with alternating ‘discriminators’



CMU SCS

k-d-trees

- Binary trees, with alternating 'discriminators'

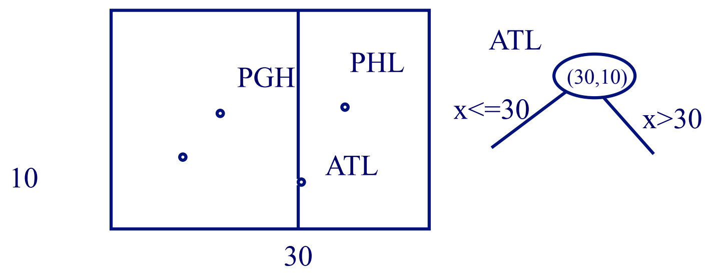


15-826 Copyright: C. Faloutsos (2017) 29

CMU SCS

k-d-trees

- Binary trees, with alternating 'discriminators'

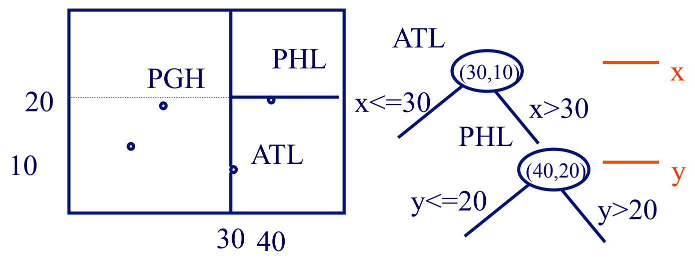


15-826 Copyright: C. Faloutsos (2017) 30

CMU SCS

k-d-trees

- Binary trees, with alternating 'discriminators'



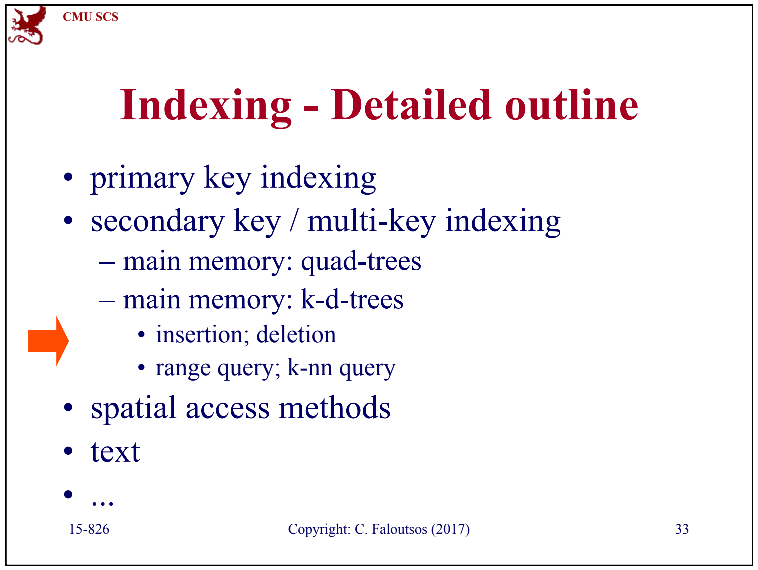
15-826 Copyright: C. Faloutsos (2017) 31

CMU SCS

(Several demos/applets, e.g.)

- http://donar.umiacs.umd.edu/quadtree/points/kdtree.html

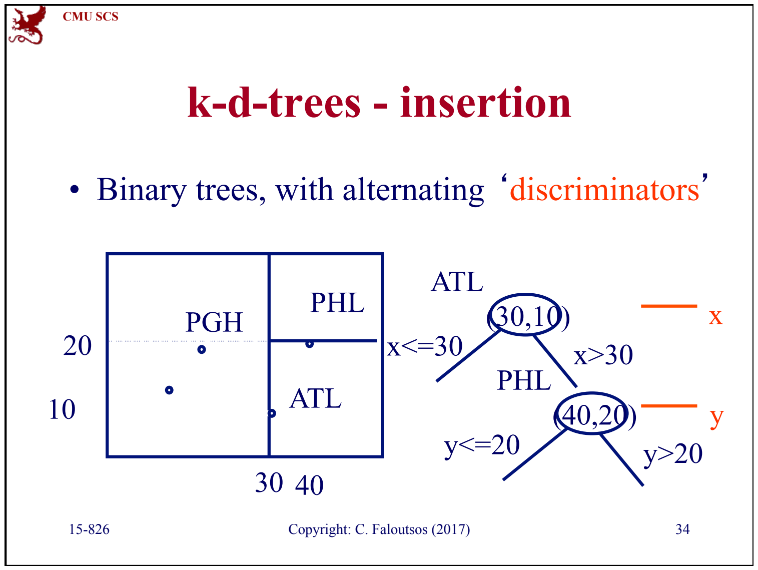
15-826 Copyright: C. Faloutsos (2017) 32


CMU SCS

Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
  - main memory: quad-trees
  - main memory: k-d-trees
    - insertion; deletion
    - range query; k-nn query
- spatial access methods
- text
- ...

15-826 Copyright: C. Faloutsos (2017) 33

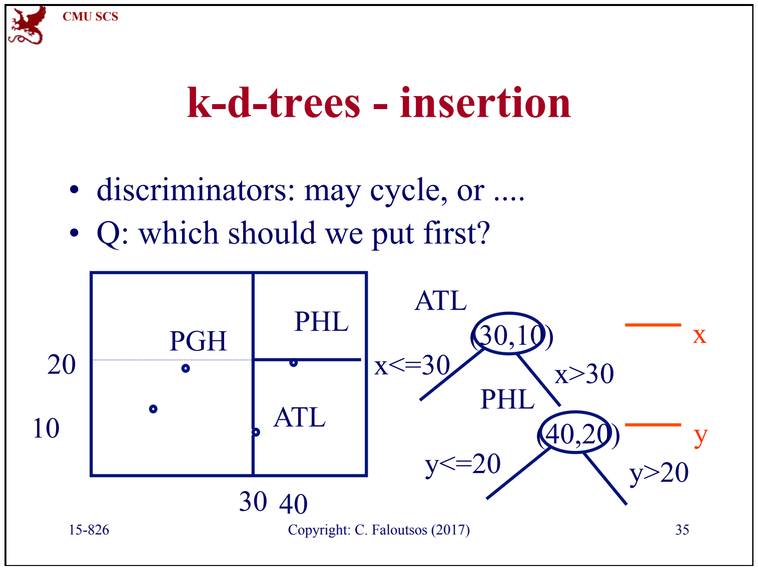

CMU SCS

k-d-trees - insertion

- Binary trees, with alternating ‘discriminators’

PGH
PHL
ATL
20
10
30
40
ATL
(30,10)
x<=30
x>30
PHL
(40,20)
y<=20
y>20
x
y

15-826 Copyright: C. Faloutsos (2017) 34

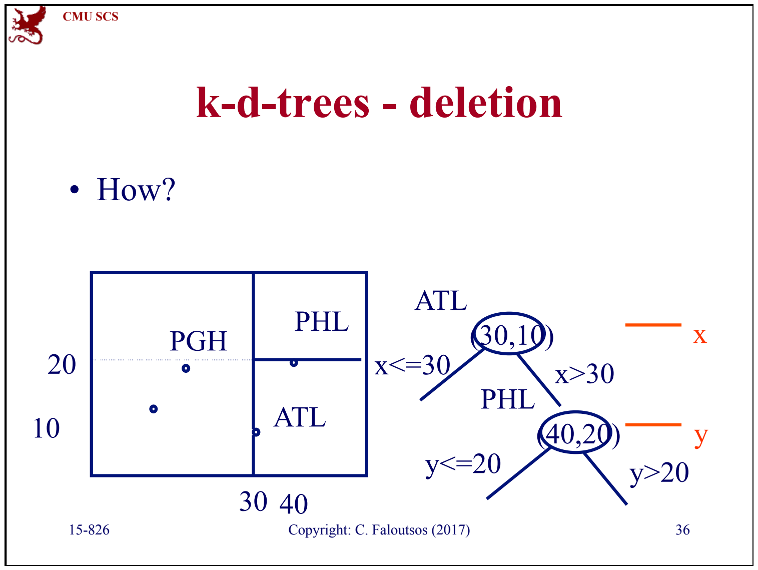

CMU SCS

k-d-trees - insertion

- discriminators: may cycle, or ....
- Q: which should we put first?

PGH
PHL
ATL
20
10
30
40
ATL
(30,10)
x<=30
x>30
PHL
(40,20)
y<=20
y>20
x
y

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


CMU SCS

k-d-trees - deletion

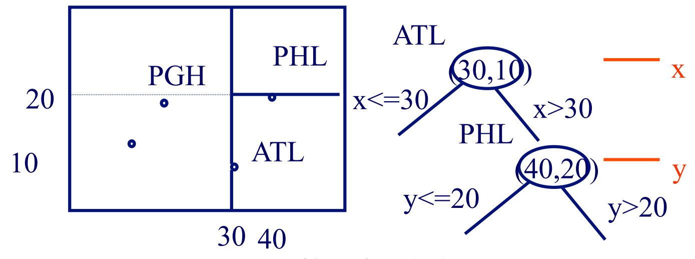
- How?

PGH
PHL
ATL
20
10
30
40
ATL
(30,10)
x<=30
x>30
PHL
(40,20)
y<=20
y>20
x
y

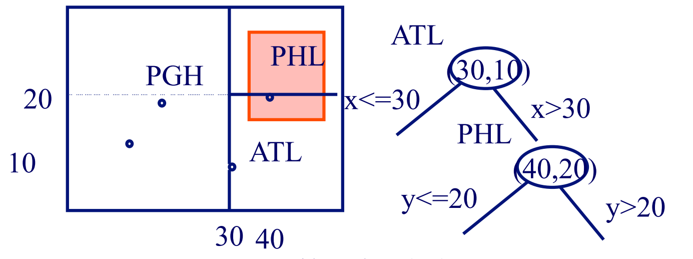
15-826 Copyright: C. Faloutsos (2017) 36

k-d-trees - deletion

- Tricky! 'delete-and-promote' (or 'mark as deleted')

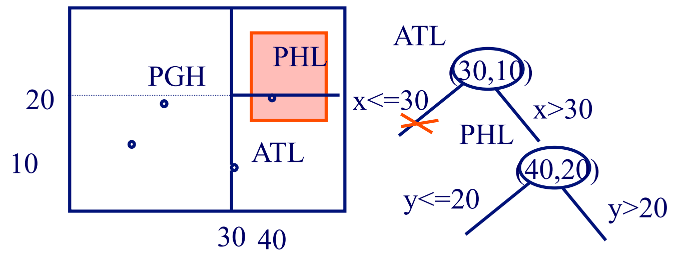


k-d-trees - range query



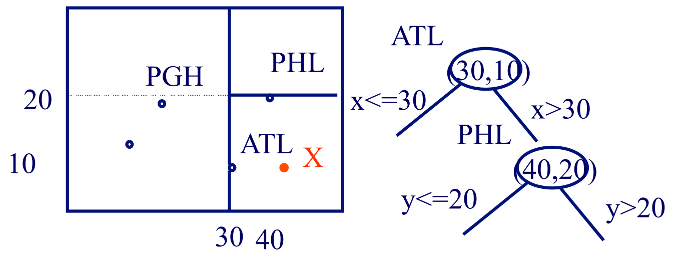
k-d-trees - range query

- similar to quad-trees: check the root; proceed to appropriate child(ren).



k-d-trees - k-nn query

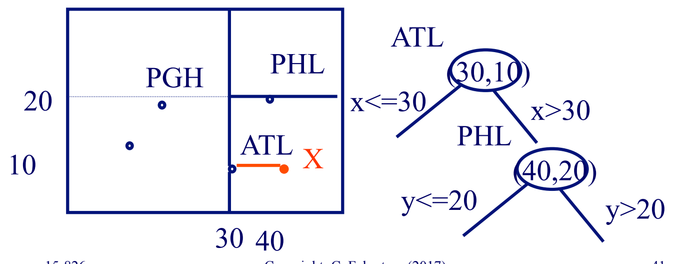
- e.g., 1-nn: closest city to 'X'



CMU SCS

k-d-trees - k-nn query

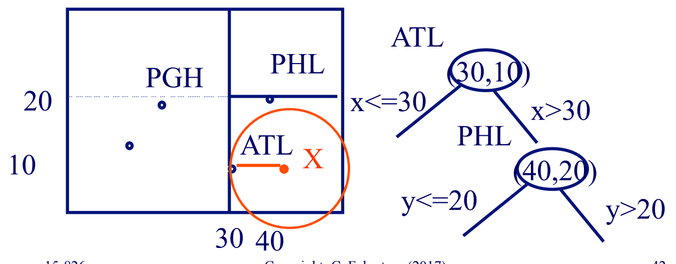
- A: check root; put in stack; proceed to child



CMU SCS

k-d-trees - k-nn query

- A: check root; put in stack; proceed to child



CMU SCS

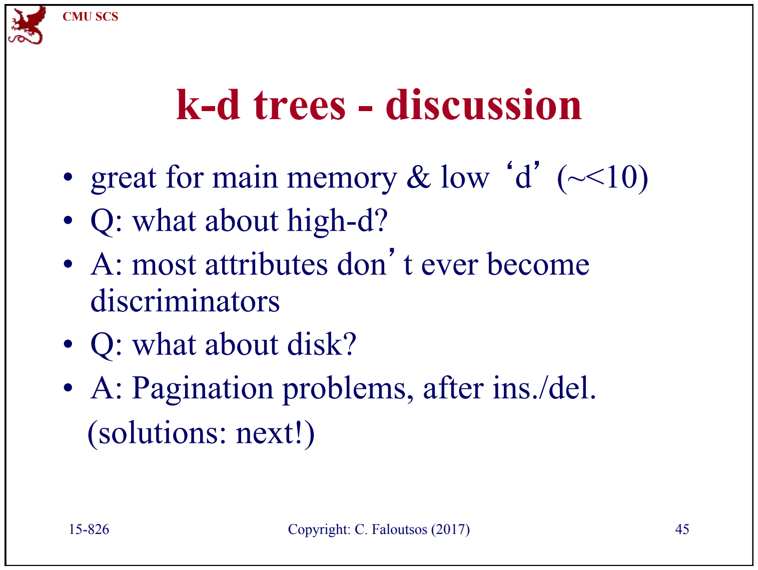
Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
  - main memory: quad-trees
  - main memory: k-d-trees
    - insertion; deletion
    - range query; k-nn query
    - discussion
- spatial access methods
- text
- ...

CMU SCS

k-d trees - discussion

- great for main memory & low ‘d’ (~<10)
- Q: what about high-d?
- A:
- Q: what about disk
- A:



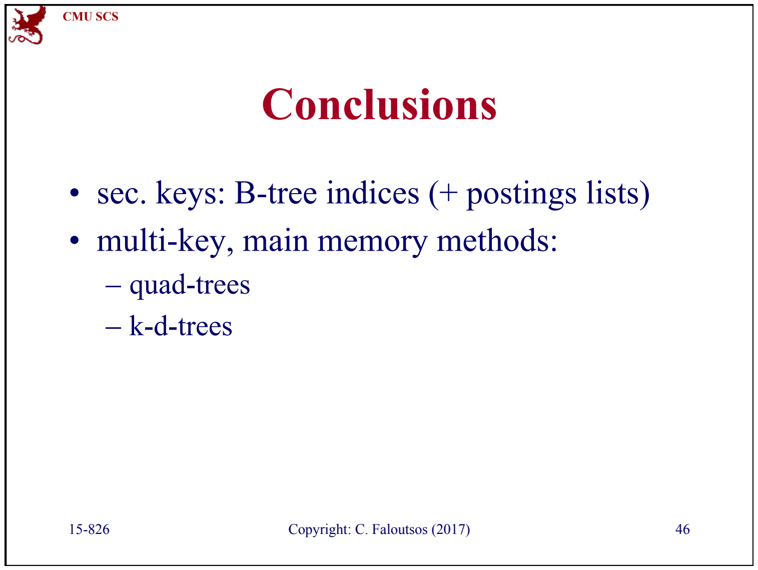
CMU SCS

k-d trees - discussion

- great for main memory & low ‘d’ (~<10)
- Q: what about high-d?
- A: most attributes don’t ever become discriminators
- Q: what about disk?
- A: Pagination problems, after ins./del.

(solutions: next!)

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

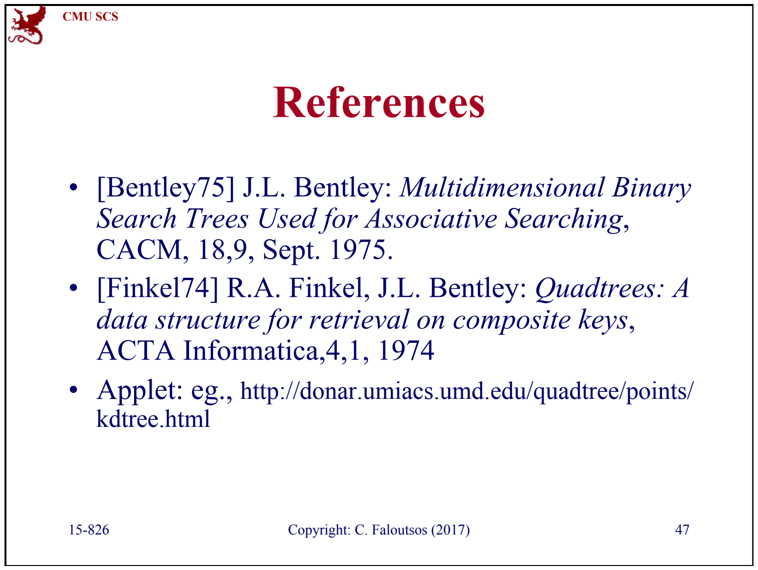


CMU SCS

Conclusions

- sec. keys: B-tree indices (+ postings lists)
- multi-key, main memory methods:
  - quad-trees
  - k-d-trees

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



CMU SCS

References

- [Bentley75] J.L. Bentley: *Multidimensional Binary Search Trees Used for Associative Searching*, CACM, 18,9, Sept. 1975.
- [Finkel74] R.A. Finkel, J.L. Bentley: *Quadtrees: A data structure for retrieval on composite keys*, ACTA Informatica,4,1, 1974
- Applet: eg., http://donar.umiacs.umd.edu/quadtree/points/kdtree.html

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