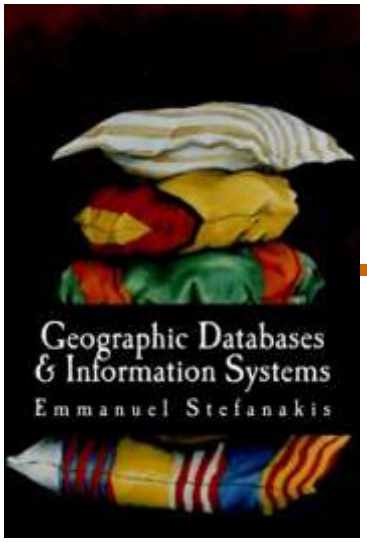


Stefanakis, E., 2014. *Geographic Databases and Information Systems*. CreateSpace Independent Publ. [In English], pp.386.

Get a copy from Amazon

Chapter 13

Spatial Data Structures

Emmanuel Stefanakis

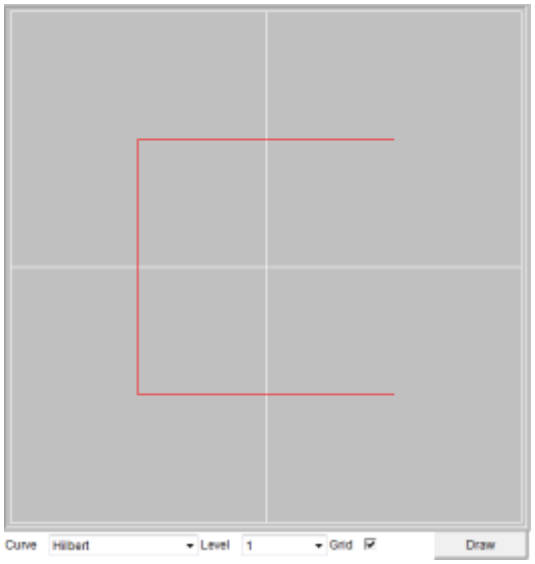
http://www2.unb.ca/~estef/

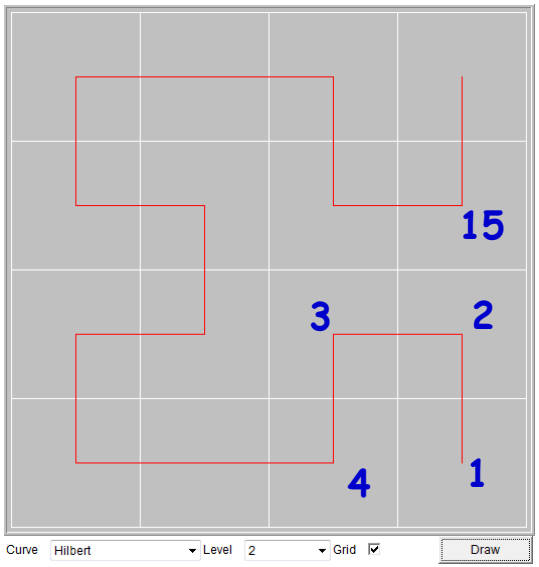
Spatial Indices

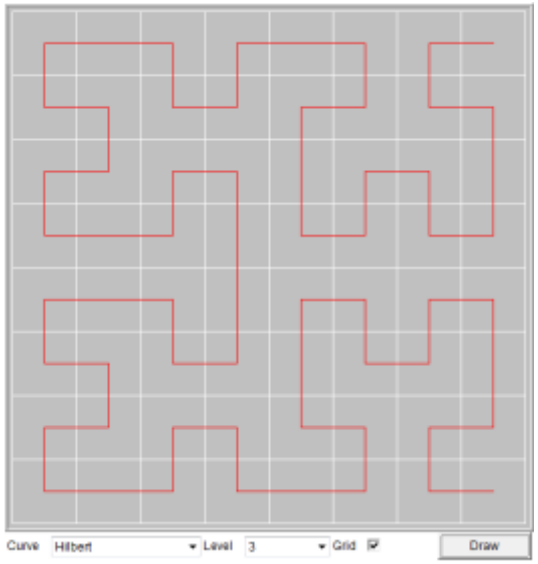
- May we apply binary search in **geometries** ?
  - Not trivial…
  - Why ?
  - Data must be sorted !
    - 2D or higher spaces cannot be sorted

Spatial Indices

- 2D Space can be linearized by space filling curves…
  - e.g., Hilbert curve



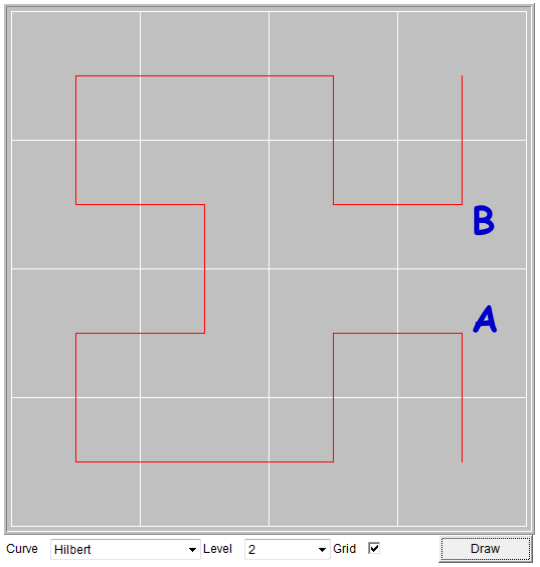


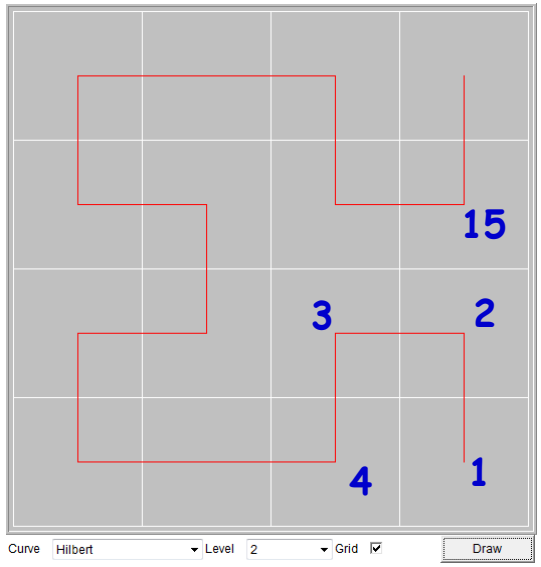


http://www.cs.utexas.edu/users/vbb/misc/sfc/Oindex.html

Spatial Indices

- However,
  - Linearization is not necessarily sorting
    - points A, B are very close to each other,
    - …but very far in the linearization.





Common Spatial Queries

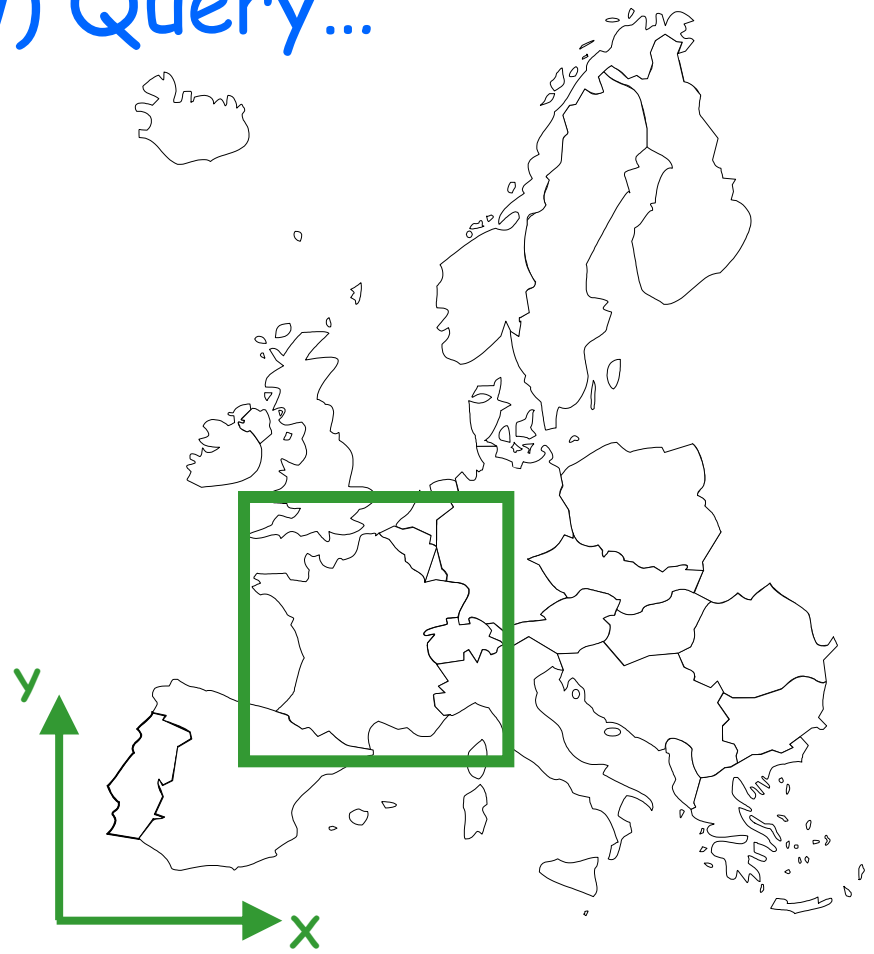
(a) Point Query…

"This is France"

Common Spatial Queries

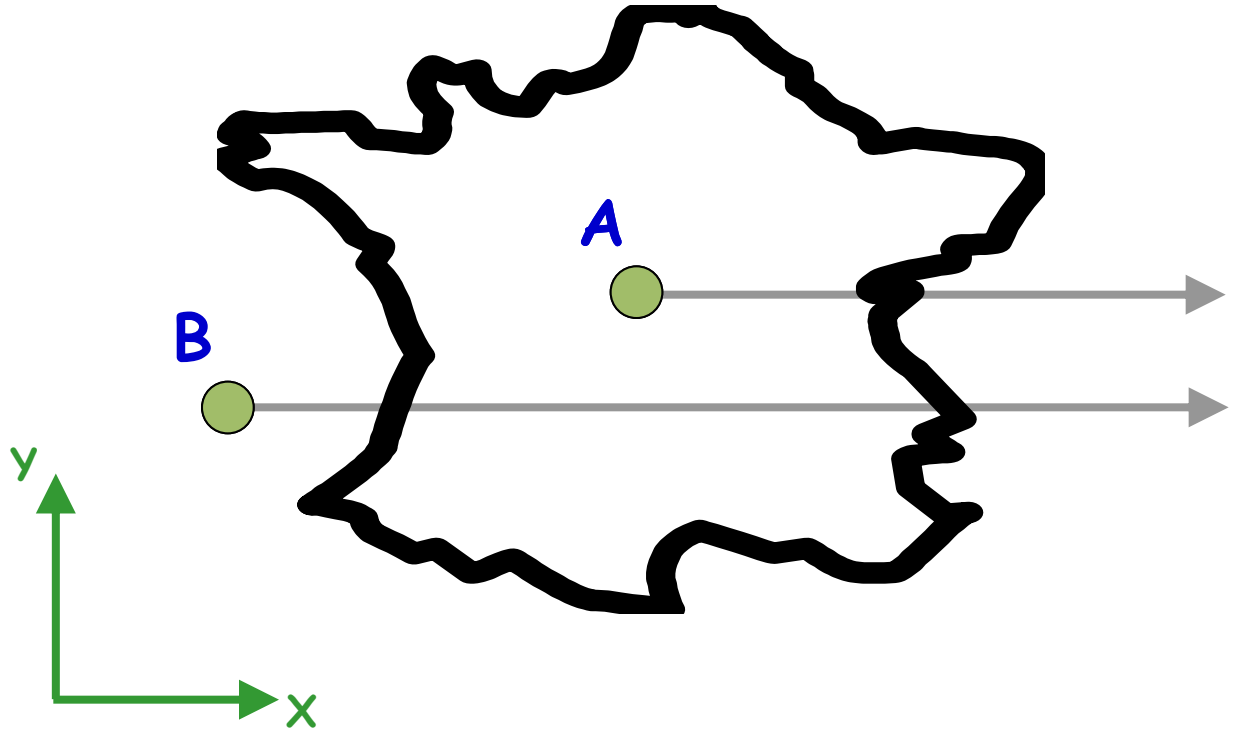
(b) Range (window) Query…

Zoom-In operation



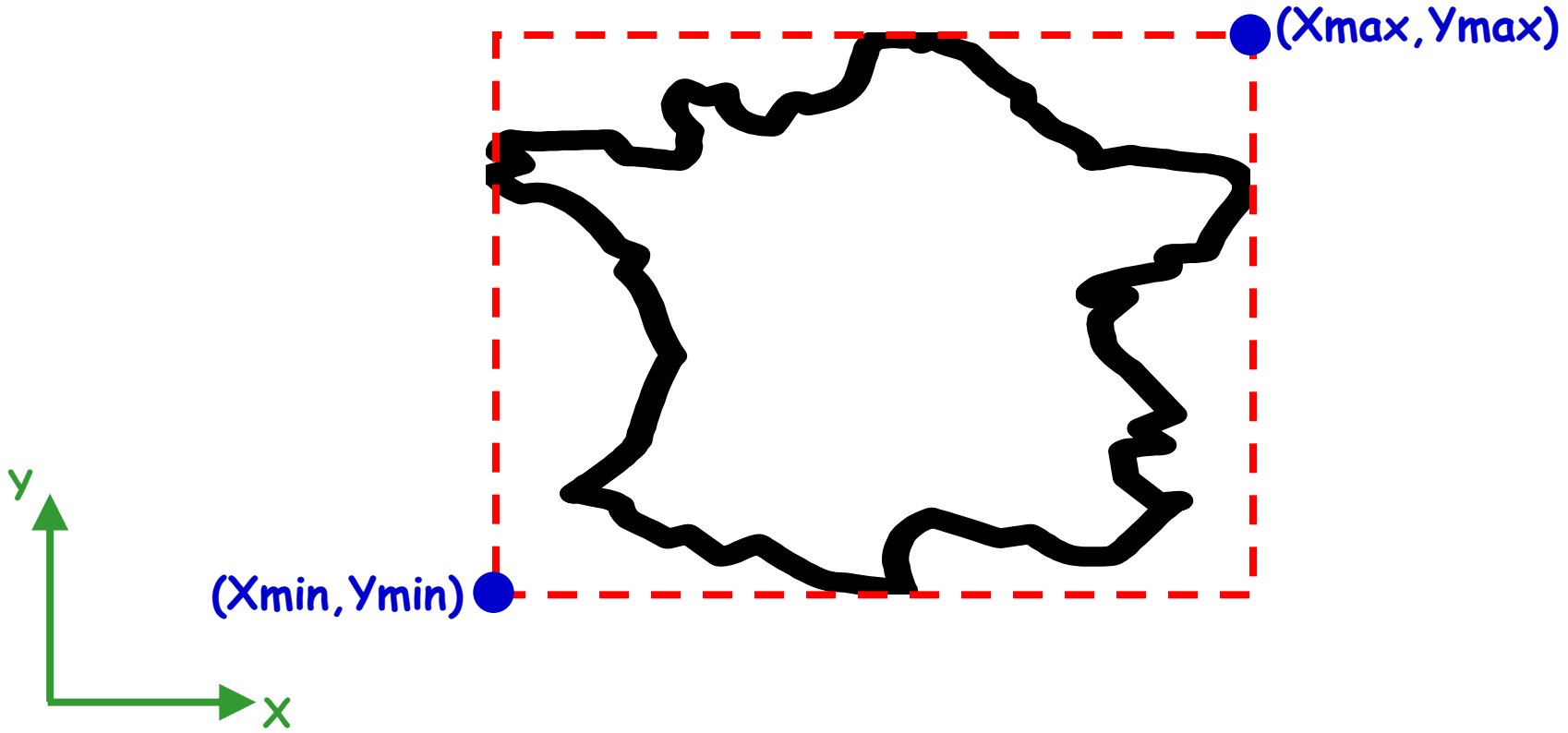
Speed up Point Queries

- Step 1: Represent each polygon by its
  - Minimum Bounding Rectangle (**MBR**)



Speed up Point Queries

- Step 1: Represent each polygon by its
  - Minimum Bounding Rectangle (**MBR**)



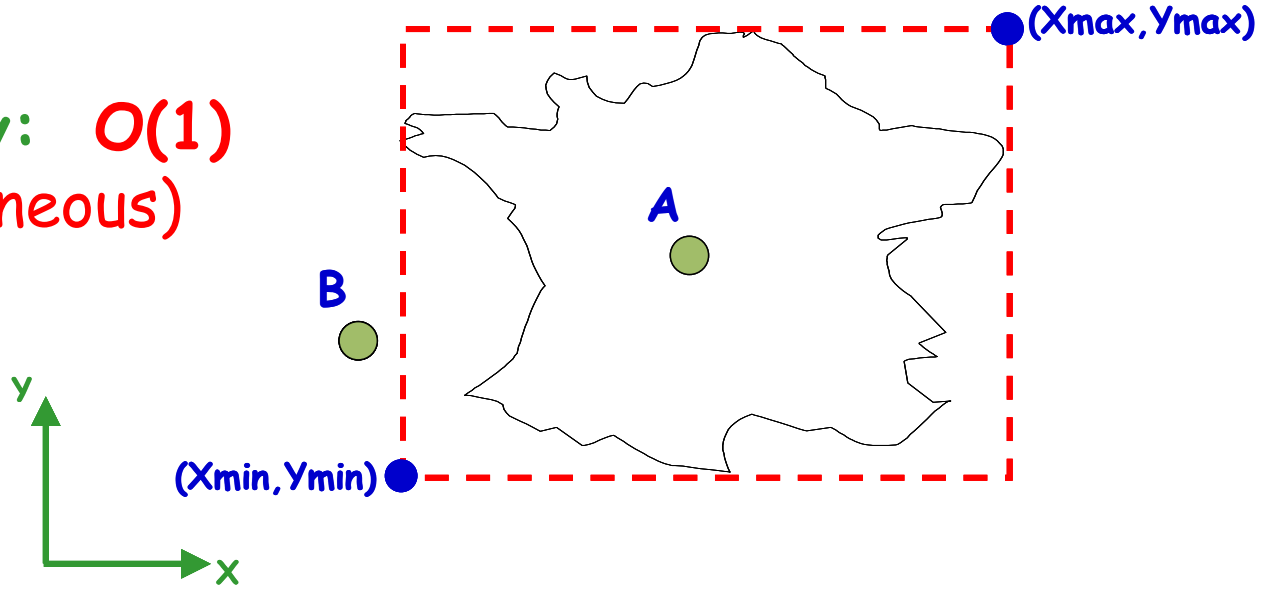
Speed up Point Queries

- Step 2: Test the point **against** the MBR first → very fast **filter**

  $xA \in (Xmin,Xmax)$ and $yA \in (Ymin,Ymax)$ → IN (?)

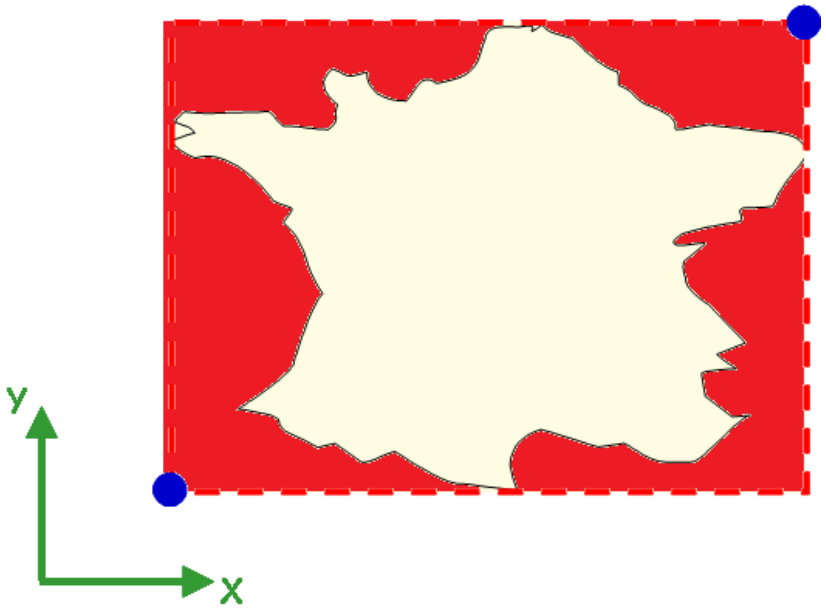
  $xB \notin (Xmin,Xmax)$ and $yB \in (Ymin,Ymax)$ → OUT (!)

Complexity: O(1)
(instantaneous)



Speed up Point Queries

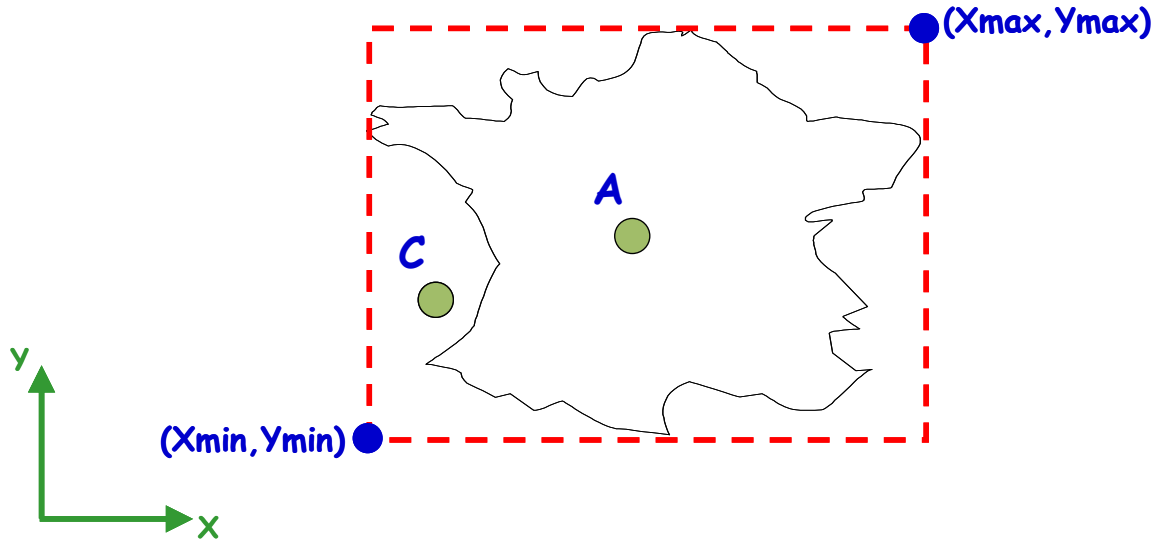
- Attention:
  - Step 2 is just a **filter** to put off all points that lie outside MBR
  - … due to the **dead-zone**



Speed up Point Queries

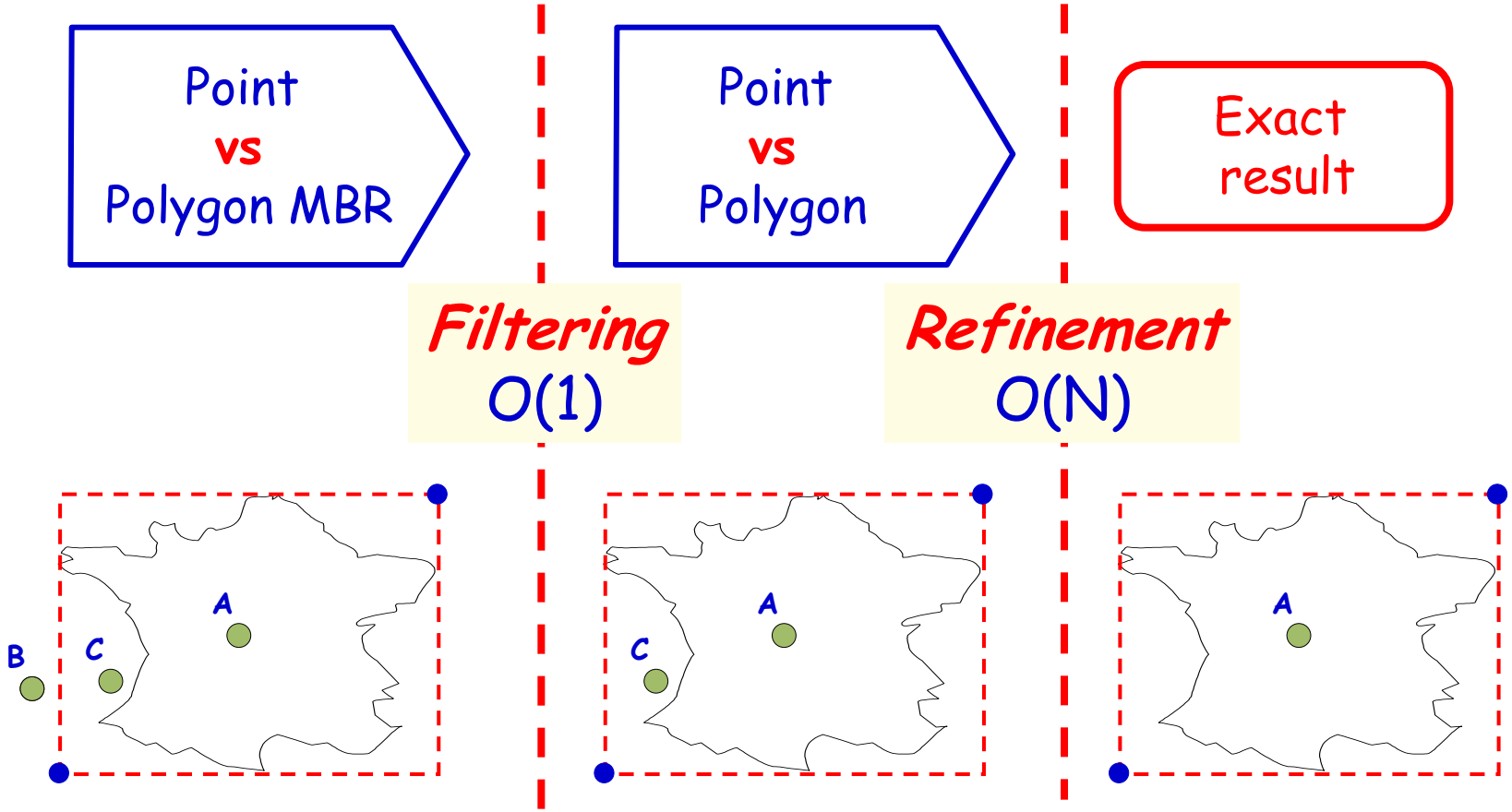
- Step 3: Perform the **refinement**
  - all points that satisfy the MBR criterion must be tested against the polygon

$xA \in (Xmin,Xmax)$ and $yA \in (Ymin,Ymax) \rightarrow$ **IN**
$xC \in (Xmin,Xmax)$ and $yC \in (Ymin,Ymax) \rightarrow$ **OUT**



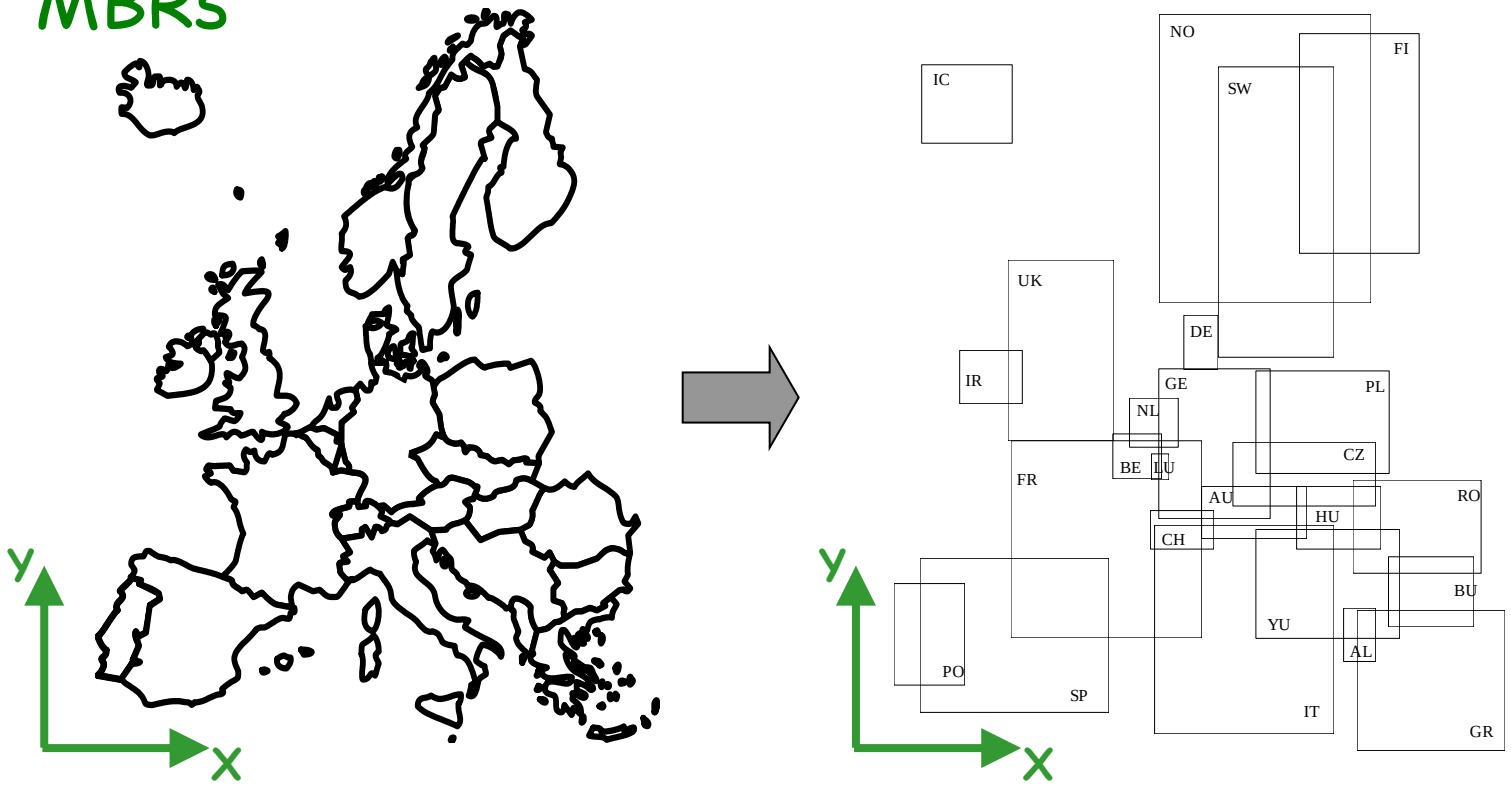
Speed up Point Queries

- **Two-tier** query model… **(1 point vs 1 poly)**



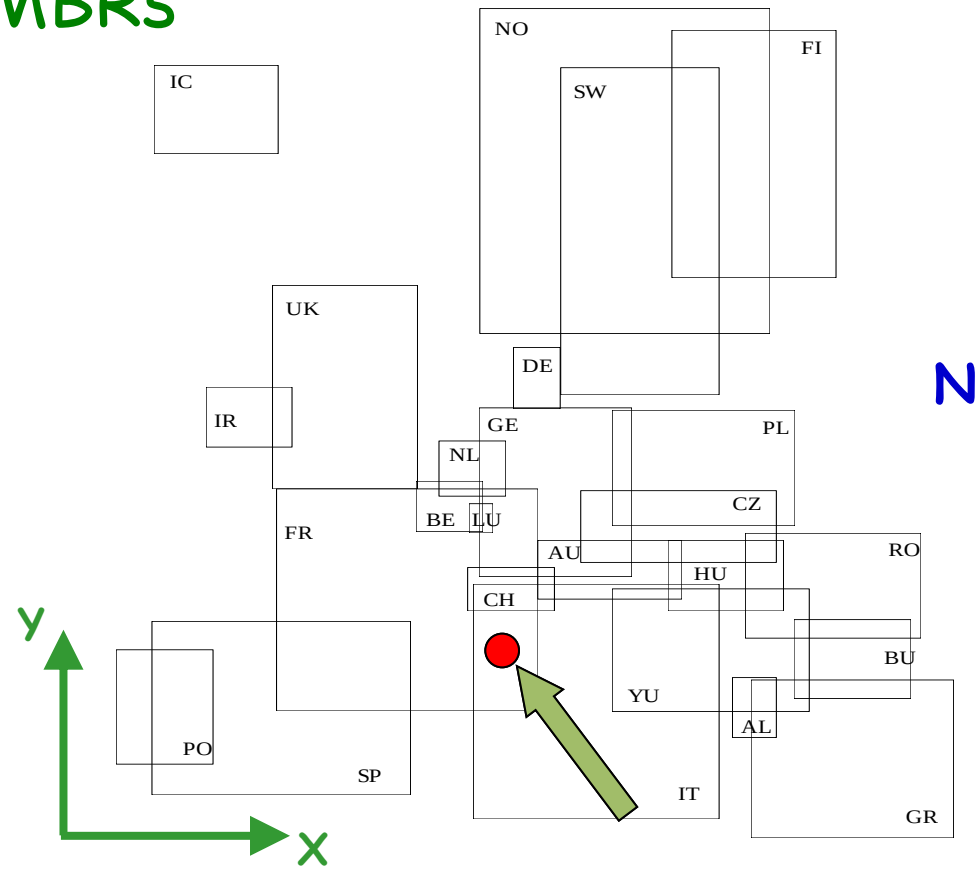
Speed up Point Queries

- Assume the map of **M** polygons…
  - Step 1: Represent all polygons by their MBRs



Speed up Point Queries

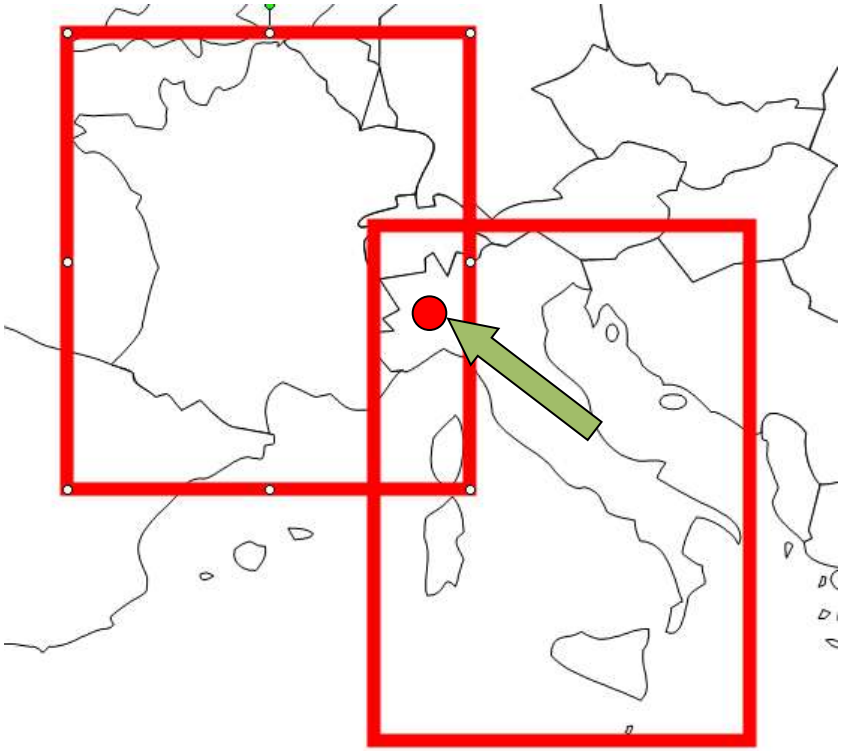
- Step 2: Test the point against the set of MBRs



N Polygons → N MBRs

Speed up Point Queries

- Step 2: Filtering step
  - MBRs may overlap → multiple candidates



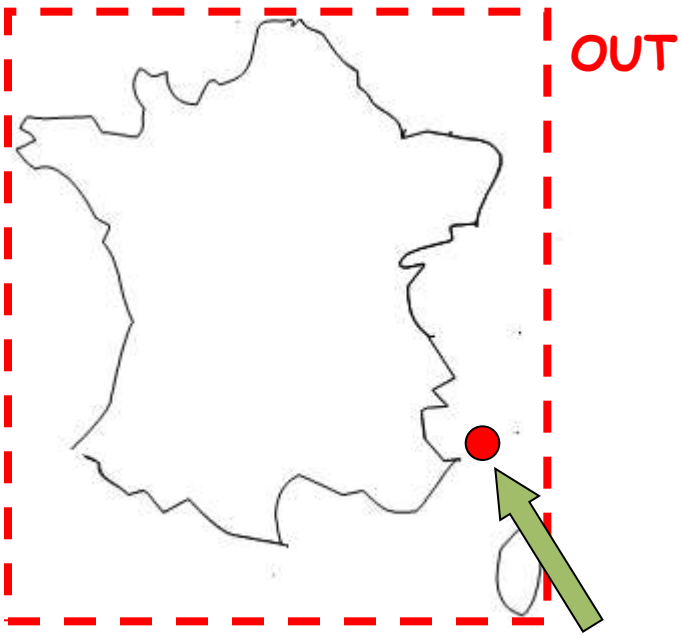
1 point **vs** M MBRs

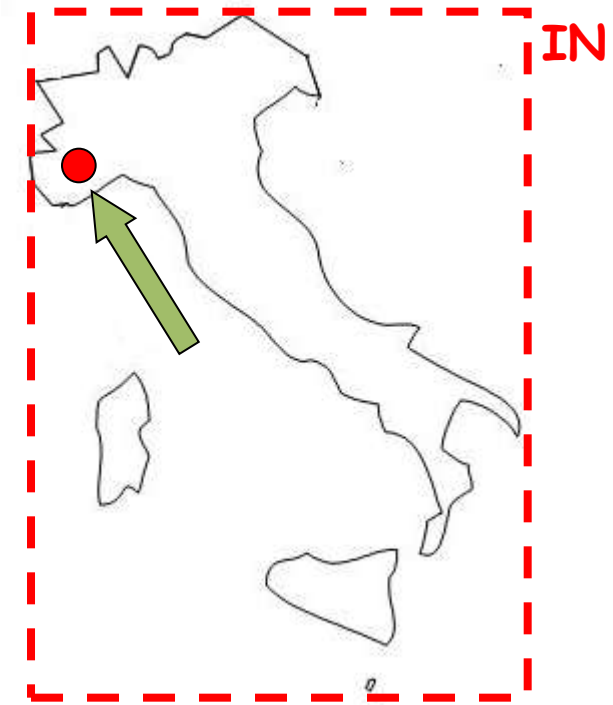
Complexity:

O(M)

Speed up Point Queries

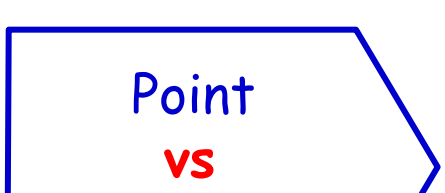
- Step 3: Perform the refinement
  - Test the point against the actual geometries of the candidate MBRs





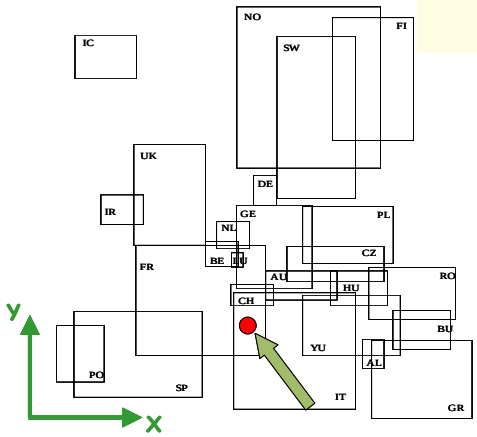
Speed up Point Queries

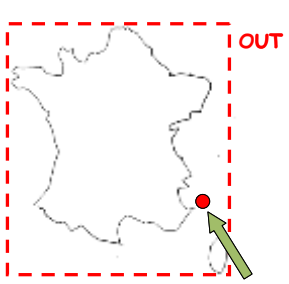
- Two-tier query model… **(1 point vs M polys)**

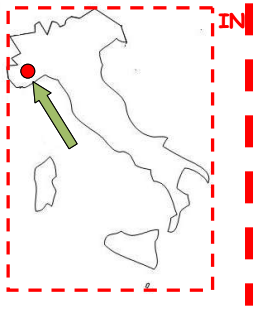


Filtering
O(M)

Refinement
O(kN)



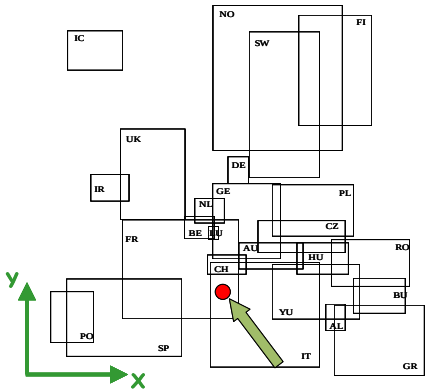




Speed up Point Queries

- Overall Complexity … **(1 point vs M polys)**
  - Map: **M** polygons with **N** vertices/polygon

1,000,000 **1,000**



10^9 **> >** $10^6 \sim 10^6 + k10^3$

Serial Search on Pure Geometries: **O(MxN)**

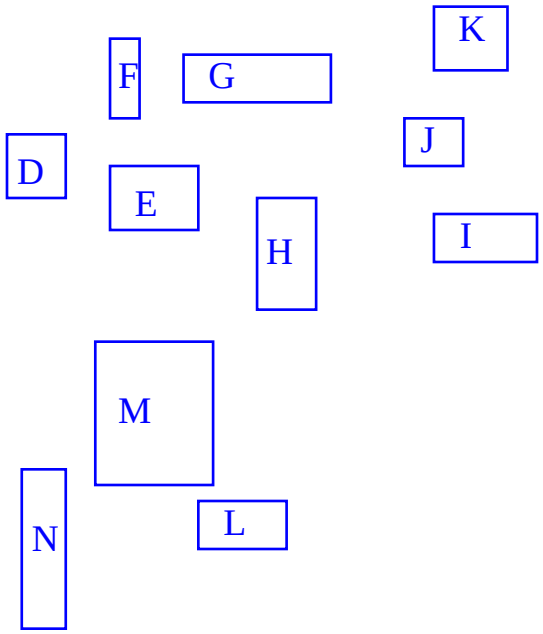
MBR abstraction and two-tier search: **O(M+kN)**

Make it even faster !

- Further speed up if…
  - MBRs are **structured** efficiently
  - Specifically,
    - MBRs that are close together…
      - may be **grouped** into larger MBRs
    - …end up with a hierarchy of MBRs

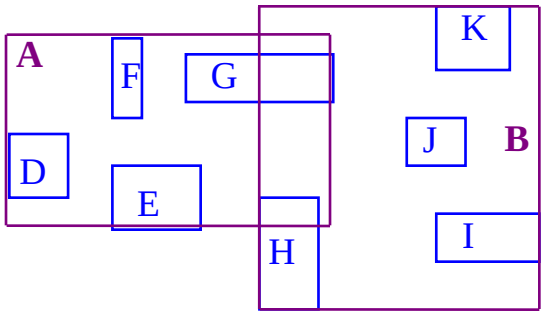
Make it even faster !

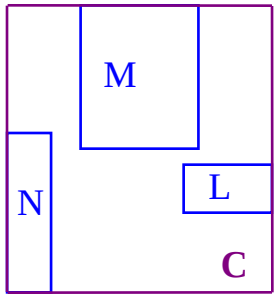
- Example…
  - Assume a set of MBRs

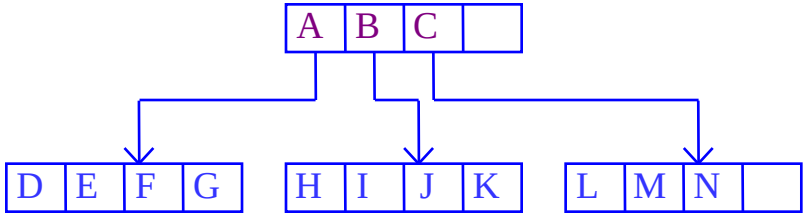


Make it even faster !

- Example…
  - Group them by 4 and generate a **hierarchy**

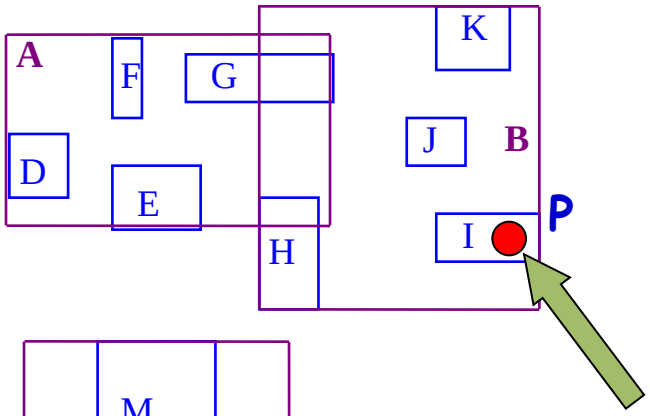


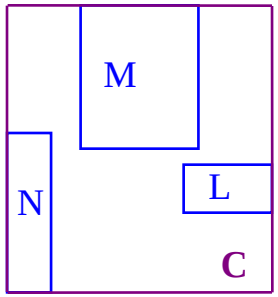




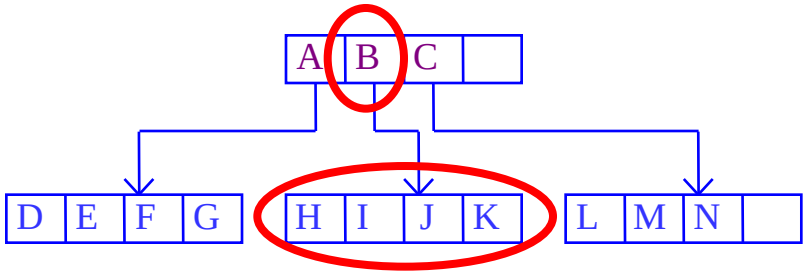
Make it even faster !

- A point query…
  - tests the point against A,B,C first
  - then proceeds to some polygon MBRs (not all)



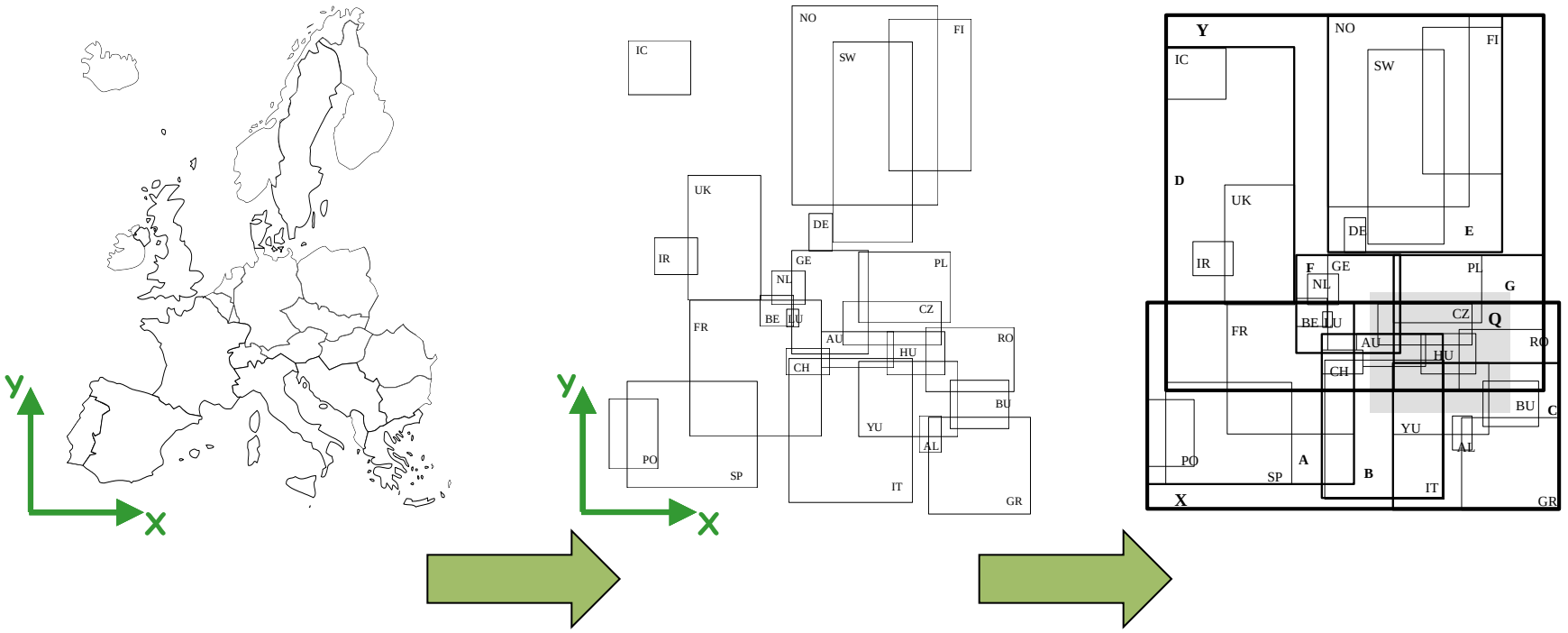


- The point P intersects B
- P against {H,I,J,K} only



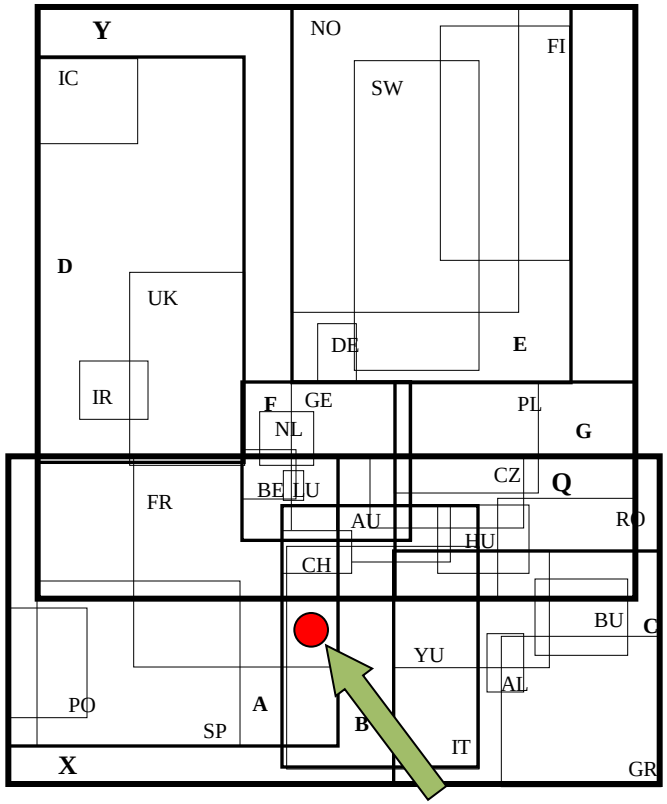
Make it even faster !

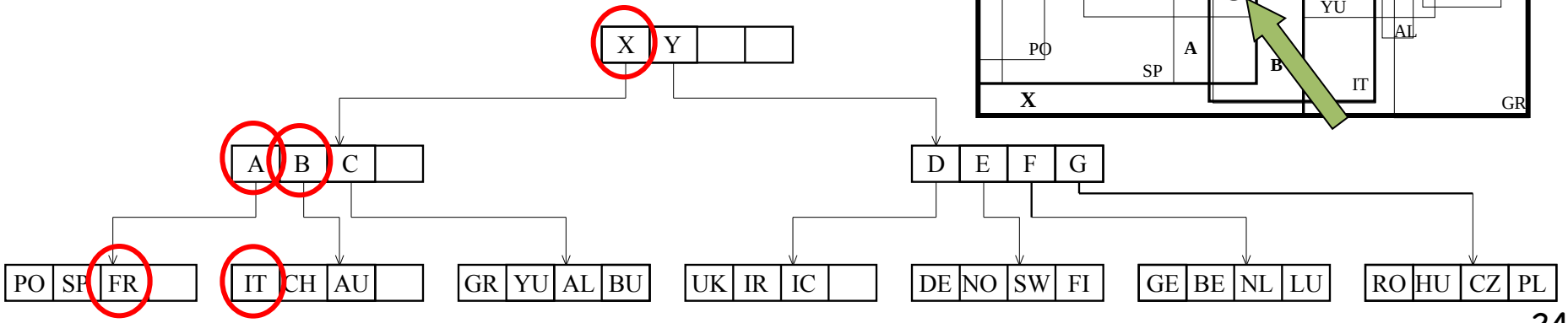
- The map of Europe…



Make it even faster !

- Build the hierarchical structure…
  - **R-tree** (Guttman, 1984)
  - Complexity:
    - Approaches …
      - $O(\log_2 N)$ + Refinements





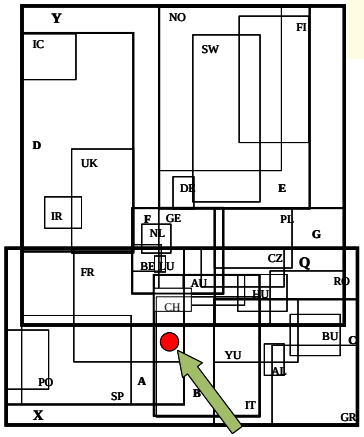
Speed up Point Queries

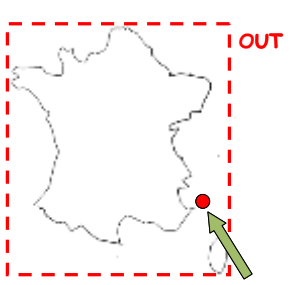
- Two-tier query model… **(1 point vs M polys)**

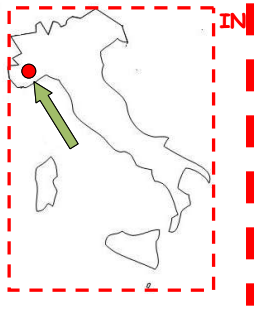
Point **vs** MBR hierarchy → Point **vs** k Candidate Polygons → Exact result

Filtering $O(\log_2 M)$

Refinement $O(kN)$



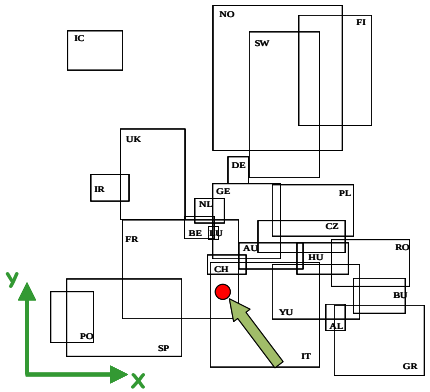




From Polygons to MBRs

- Overall Complexity …
  - Map: **M** polygons with **N** vertices/polygon

1,000,000 | **1,000**



10^9 **> >** $10^6 \sim 10^6 + k10^3$

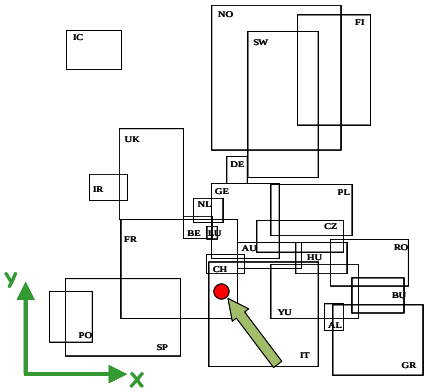
Serial Search on Pure Geometries: **O(MxN)**

MBR abstraction and two-tier search: **O(M+kN)**

From MBRs to MBR Hierarchy

- Overall Complexity …
  - Map: **M** polygons with **N** vertices/polygon

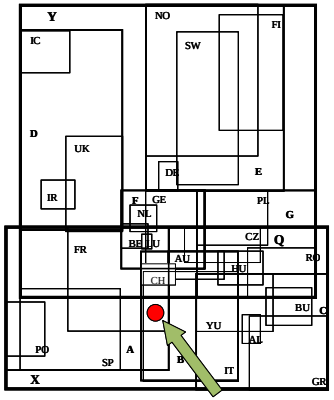
1,000,000



$10^6 + k10^3 \sim 10^6$

MBR abstraction and two-tier search: **O(M+kN)**

1,000



$10^3 \sim 20 + k10^3$

$\log_2 10^6 = 20$

>>

MBR hierarchy and two-tier search: **O($\log_2$M+kN)**

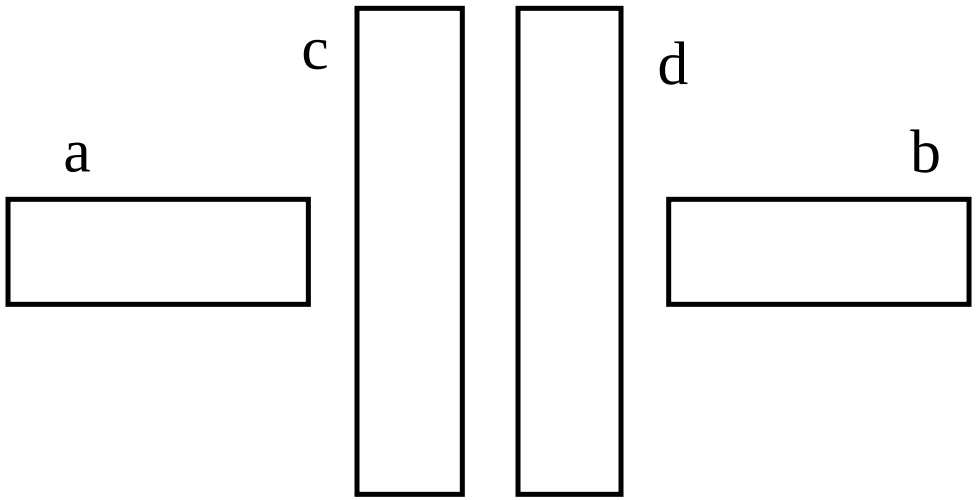
R-tree Index Structure

- Based on the grouping of MBRs…
  - To form a hierarchy
  - … **approaches** logarithmic complexity

- How the MBRs are grouped together?
  - Two objectives (**contradictory**)
    - Minimize the dead zone of the groups
    - Minimize the overlap between groups

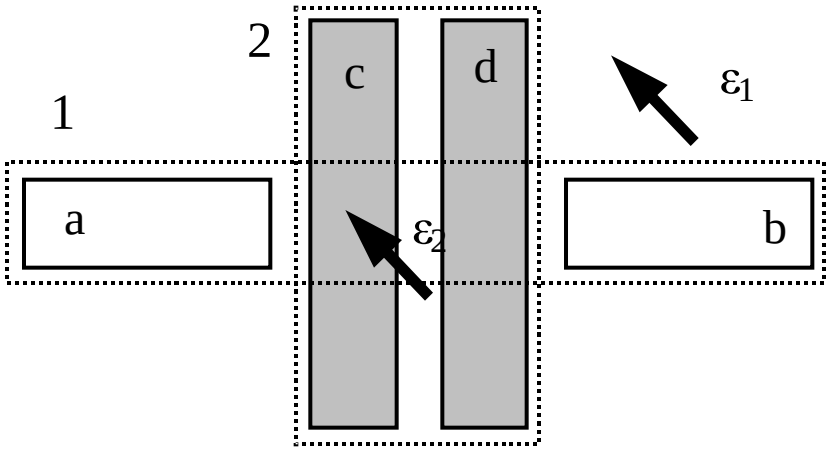
R-tree Index Structure

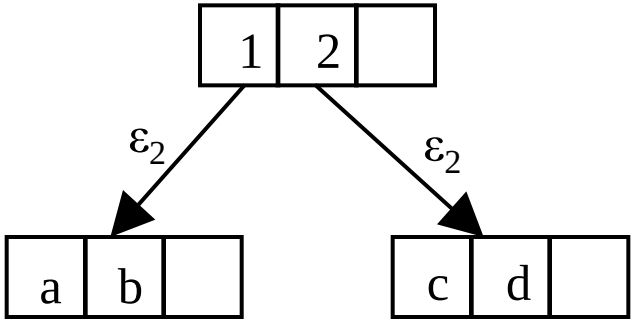
- Assume the following 4 objects…
  - to be grouped into pairs



R-tree Index Structure

- 1st Option…
  - Minimize the dead zone of the groups

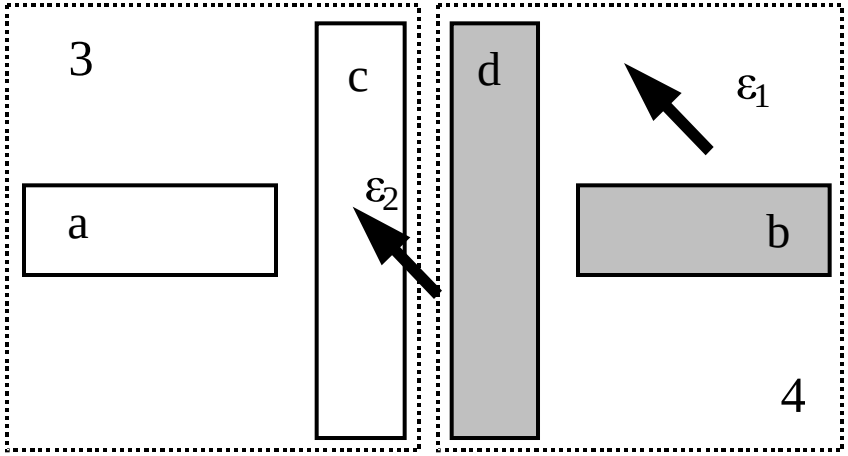


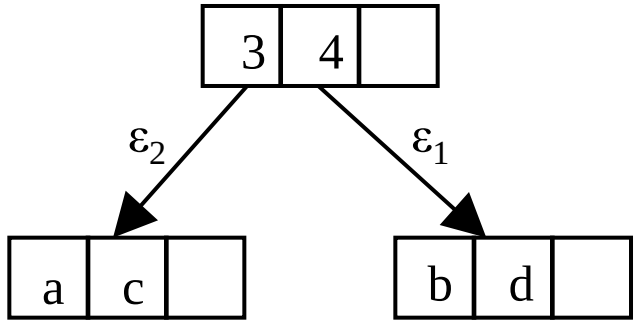


Point query ε_2 forces
to descend both sub-trees

R-tree Index Structure

- 2nd Option…
  - Minimize the overlap between groups





Point query ε_1 forces descending one sub-trees (although it lies in dead zone)

R-tree Index Structure

- Many versions…
  - R+tree
  - Packed R-tree
  - R*tree
  - …

They apply various heuristics in grouping the MBRs

- Cost Models…

They may accurately calculate the search cost (complexity)

$$C(Q) = 1 + \sum_{j=1}^{h-1} \left\{ N_j \cdot \prod_{i=1}^{n} \left(s_j + Q_i \right) \right\}$$

$$C(R_1, R_2) = \sum_{j=1}^{h-1} \left\{ N_{R_2,j} \cdot N_{R_1,j} \cdot \prod_{k=1}^{n} \left(s_{R_1,j,k} + s_{R_2,j,k} \right) + N_{R_2,j} \cdot N_{R_1,j+1} \cdot \prod_{k=1}^{n} \left(s_{R_1,j+1,k} + s_{R_2,j,k} \right) \right\}$$

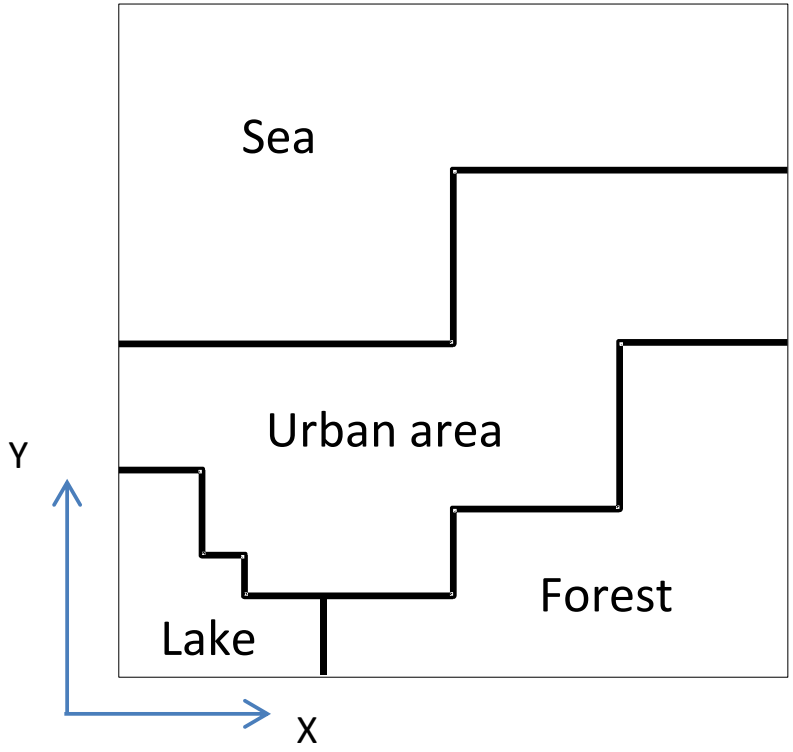
Spatial Data Structures

Quadtree, k-d-tree, Grid-file

Emmanuel Stefanakis

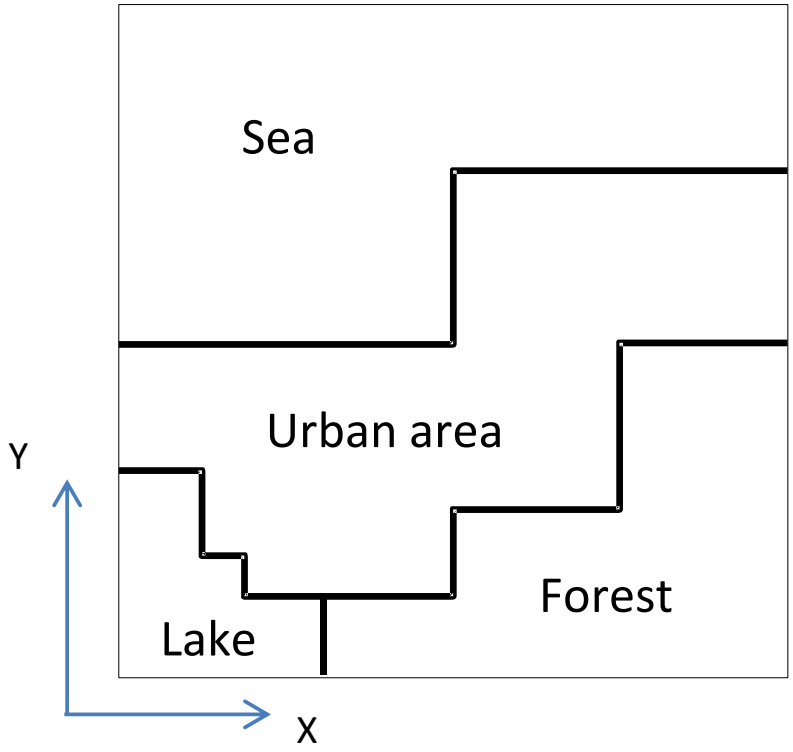
estef@unb.ca

Raster File



s	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s	u	u	u	u	u	u	u	u
s	s	s	s	s	s	s	s	u	u	u	u	u	u	u	u
s	s	s	s	s	s	s	s	u	u	u	u	u	u	u	u
s	s	s	s	s	s	s	s	u	u	u	u	u	u	u	u
u	u	u	u	u	u	u	u	u	u	u	u	f	f	f	f
u	u	u	u	u	u	u	u	u	u	u	u	f	f	f	f
u	u	u	u	u	u	u	u	u	u	u	u	f	f	f	f
l	l	u	u	u	u	u	u	u	u	u	u	f	f	f	f
l	l	u	u	u	u	u	u	f	f	f	f	f	f	f	f
l	l	l	u	u	u	u	u	f	f	f	f	f	f	f	f
l	l	l	l	l	f	f	f	f	f	f	f	f	f	f	f
l	l	l	l	l	f	f	f	f	f	f	f	f	f	f	f

Raster File



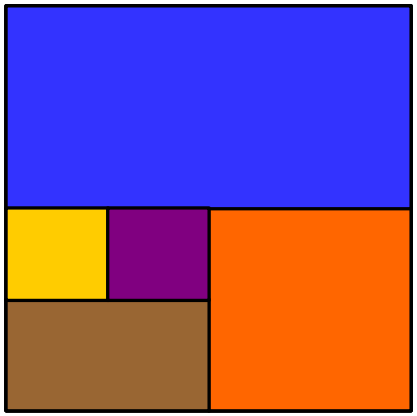
s	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s	s	s	s	s	s	s	s	s
s	s	s	s	s	s	s	s	u	u	u	u	u	u	u	u
s	s	s	s	s	s	s	s	u	u	u	u	u	u	u	u
s	s	s	s	s	s	s	s	u	u	u	u	u	u	u	u
s	s	s	s	s	s	s	s	u	u	u	u	u	u	u	u
u	u	u	u	u	u	u	u	u	u	u	u	f	f	f	f
u	u	u	u	u	u	u	u	u	u	u	u	f	f	f	f
u	u	u	u	u	u	u	u	u	u	u	u	f	f	f	f
l	l	u	u	u	u	u	u	u	u	u	u	f	f	f	f
l	l	u	u	u	u	u	u	f	f	f	f	f	f	f	f
l	l	l	u	u	u	u	u	f	f	f	f	f	f	f	f
l	l	l	l	l	f	f	f	f	f	f	f	f	f	f	f
l	l	l	l	l	f	f	f	f	f	f	f	f	f	f	f

Run-length encoding:

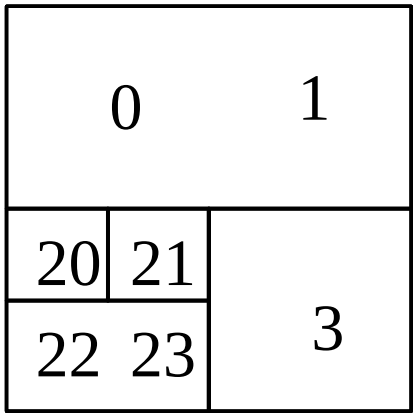
72s, 8u, 8s, 8u, 8s, 8u, 8s, 20u, 4f, 12u, 4f, 12u, 6f, 10u, 6f, 6u, 8f, 3l, 5u, 8f, 5l, 11f, 5l, 11f.

Quadtree

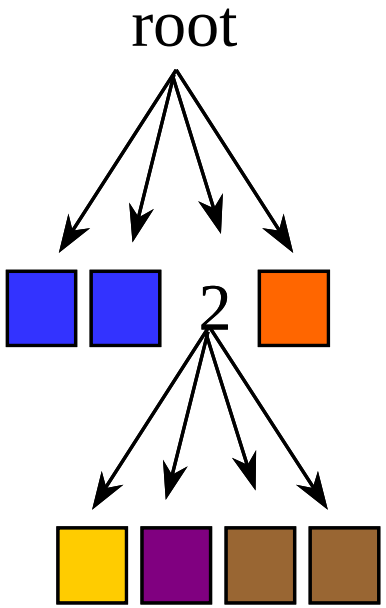
Data

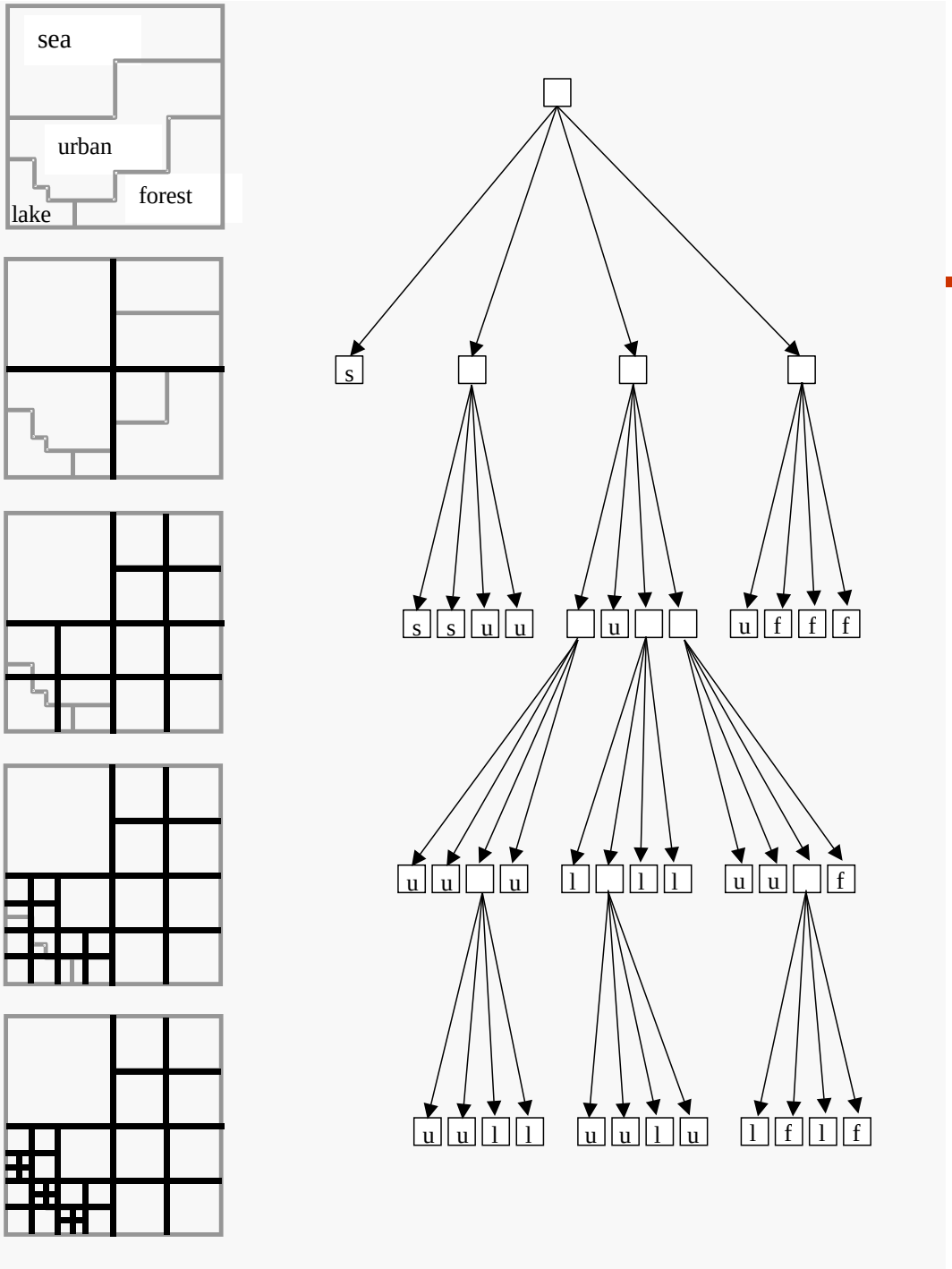


Representation

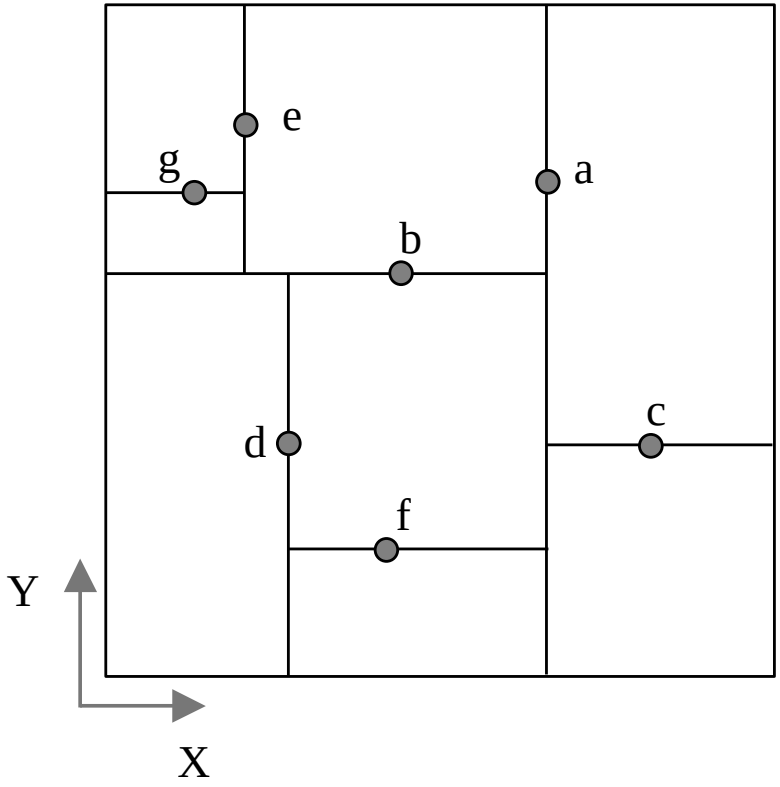


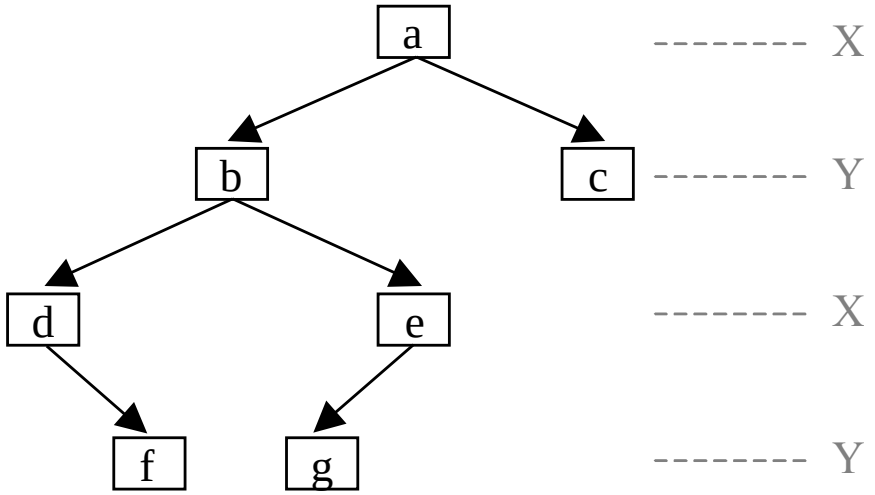
Quadtree

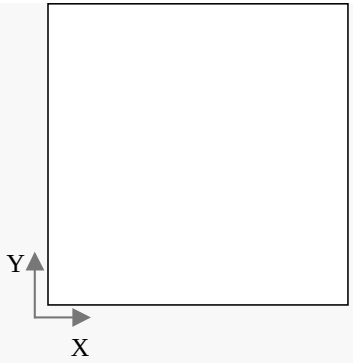



sea
urban
forest
lake
s
s s u u
u
u f f f
u u u
l l l
u u f
u u l l
u u l u
l f l f

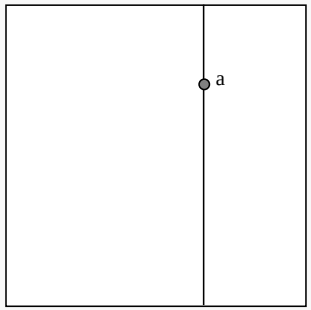
K-d-tree



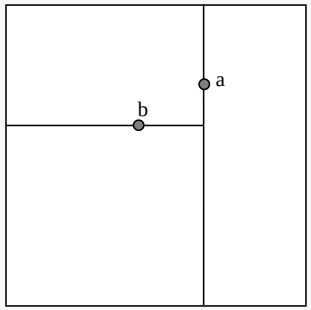



Y
X

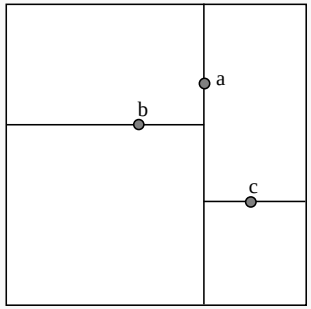
Study area…

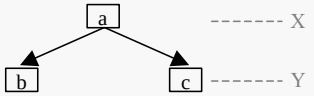

a

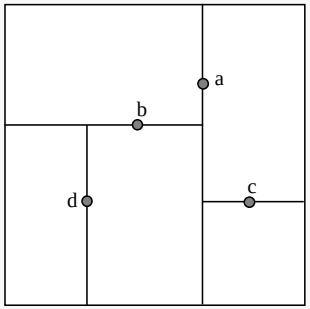

a
X

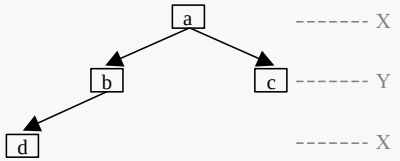

a
b

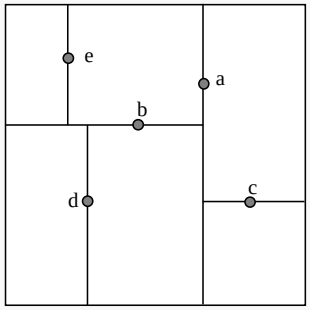

a
X
b
Y

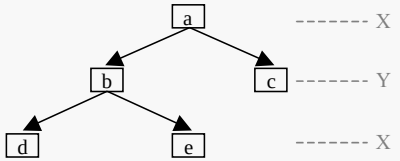

a
b
c

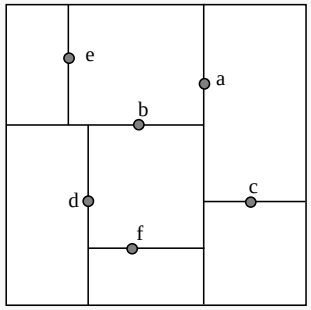

a
X
b
c
Y

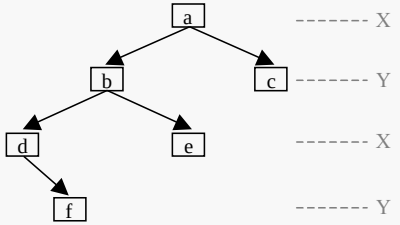

a
b
c
d

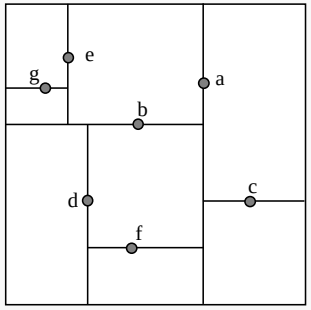

a
b
c
d
X
Y
X

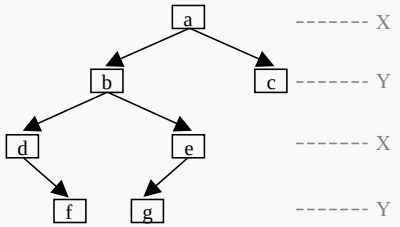

e
a
b
c
d


a
b
c
d
e
X
Y
X

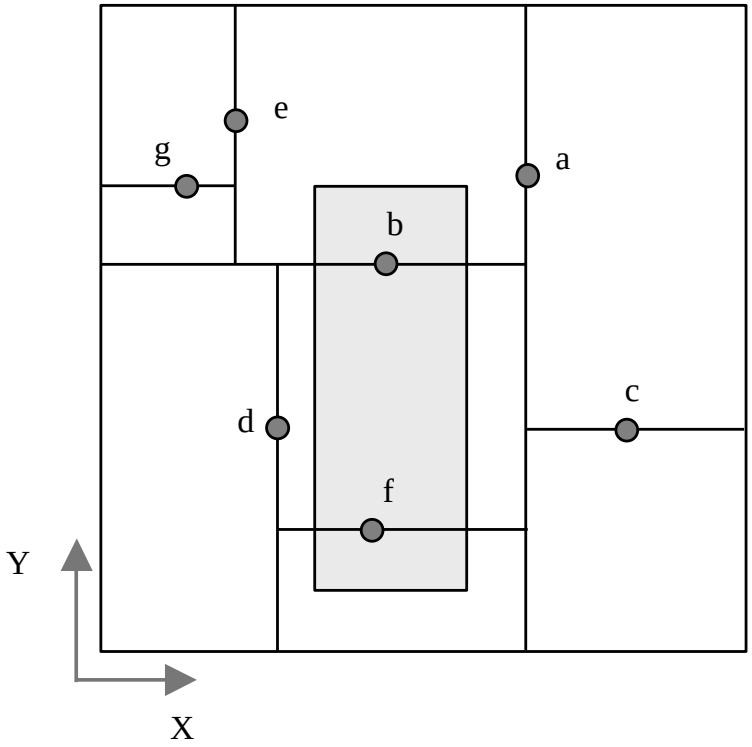

e
a
b
c
d
f

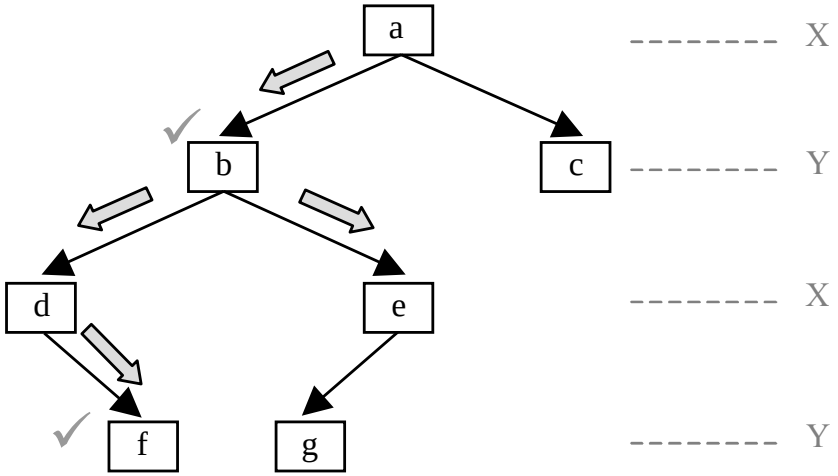

a
b
c
d
e
f
X
Y
X
Y


g
e
a
b
c
d
f

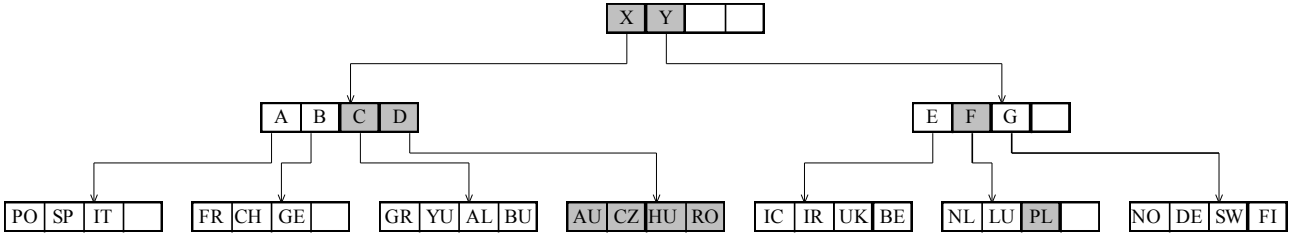

a
b
c
d
e
f
g
X
Y
X
Y

K-d-tree (Search)

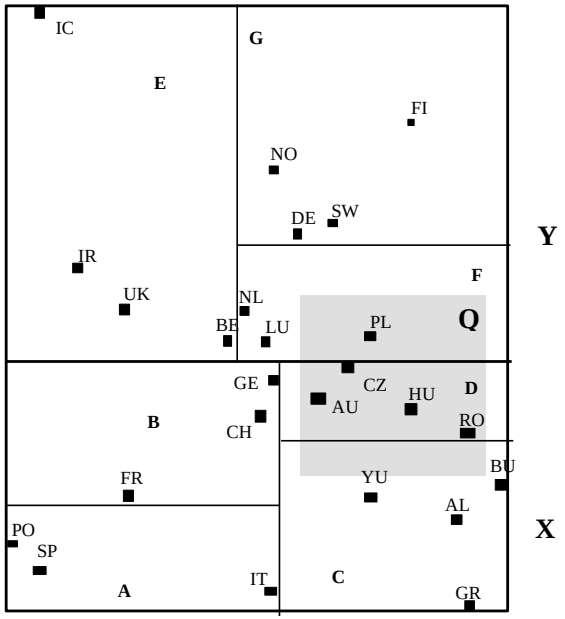




K-d-B-tree (balanced)

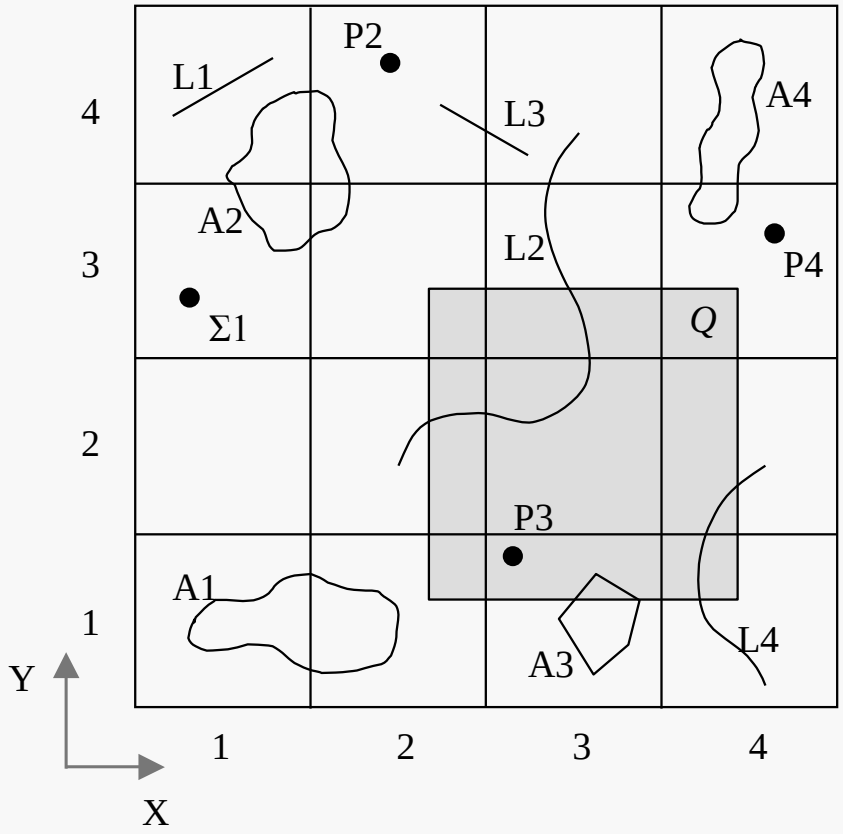


(a)



(b)

Grid-file



(a)

cell(1,4): {L1,A2}
cell(2,4): {P2,A2,L3}
cell(3,4): {L2,L3}
cell(4,4): {A4}

cell(1,3): {P1,A2}
cell(2,3): {A2}
cell(3,3): {L2}
cell(4,3): {P4,A4}

cell(1,2): { ∅}
cell(2,2): {L2}
cell(3,2): {L2}
cell(4,2): {L4}

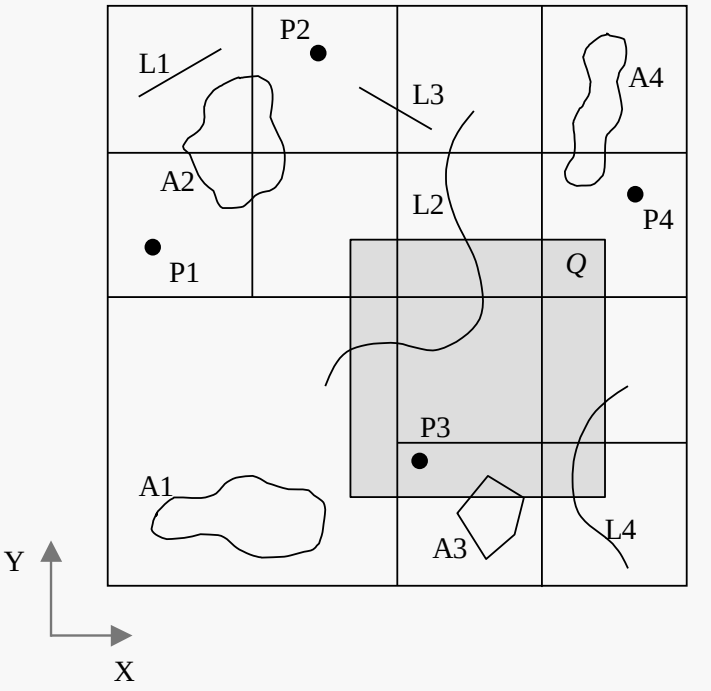
cell(1,1): {A1}
cell(2,1): {A1}
cell(3,1): {P3,A3}
cell(4,1): {L4}

```
Q ⇒ cells:  (2,1),(3,1),(4,1) ,
            (2,2),(3,2),(4,2) ,
            (2,3),(3,3),(4,3)  ⇒
  ⇒ candidate objects:      {A1},{P3,A3},{L4} ,
                           {L2},{L2},{L4} ,
                           {A2},{L2},{P4,A4} =
                       =  {P3,P4,L2,L4,A1,A2,A3,A4} ⇒
  ⇒ result: {P3,L2,L4,A3}
```

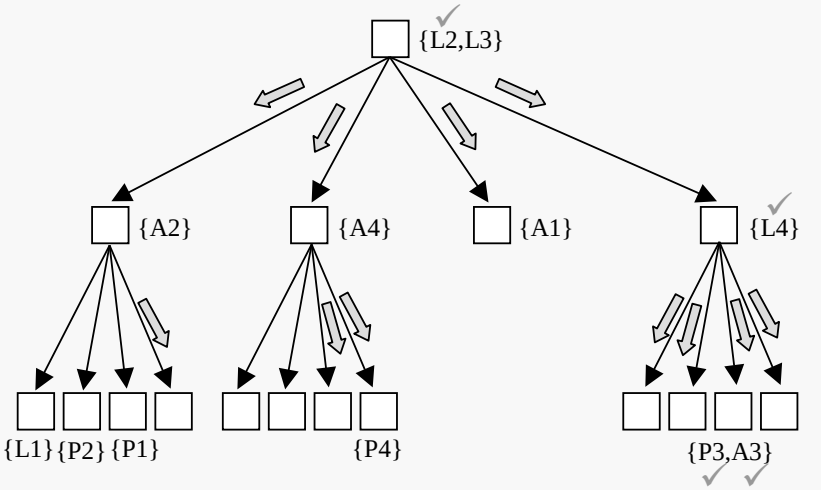
(b)

Quadtree

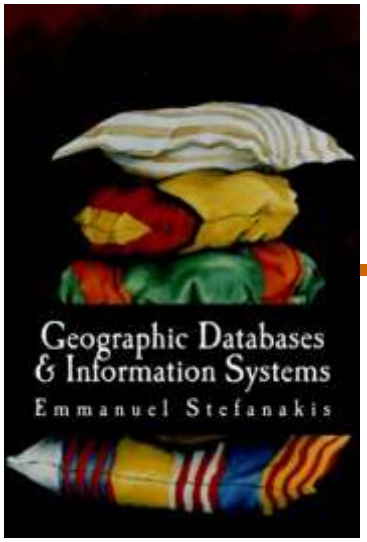
for vectors



(a)



(b)



Stefanakis, E., 2014. *Geographic Databases and Information Systems*. CreateSpace Independent Publ. [In English], pp.386.

Get a copy from Amazon

Chapter 13

Spatial Data Structures

Emmanuel Stefanakis

http://www2.unb.ca/~estef/