CS709a: Algorithms and Complexity

Focus: Spatial Data Structures and Algorithms

Instructor: Dan Coming
dan.coming@dri.edu

Thursdays 4:00-6:45pm
Office hours after class
(or by appointment)

Today

- Presentation on UnitTest++
- Calendar
- More details for Project 1
- Additional operations on spatial data structures
- Research paper selections?

Tentative Calendar

- 2/12 – Paper selection due
- 2/19 – Paper Presenter: Joe
- 2/26 – Paper Presenter: Matt
  - Present Project 1 in class (Project 1 Due 2/25)
- 3/5 – Paper Presenter: Ray
- 3/12 – Paper Presenter: Mark
  - Present Project 2 in class (Project 2 Due 3/11),
- 3/14-22 Spring Break
- 3/26 – Paper Presenter: Scott
- 4/2 – Paper Presenter: Cody
  - class day likely to be moved
  - Present Project 3 in class (Project 3 Due 4/1)
- 4/9 – Paper Presenter: James
- 4/16 – Paper Presenter: Steve
- 4/23 – Paper Presenter: Roger
- 4/30
- 5/7-13 Finals Week
  - Final Projects and Presentations Due

Project 1 – More details

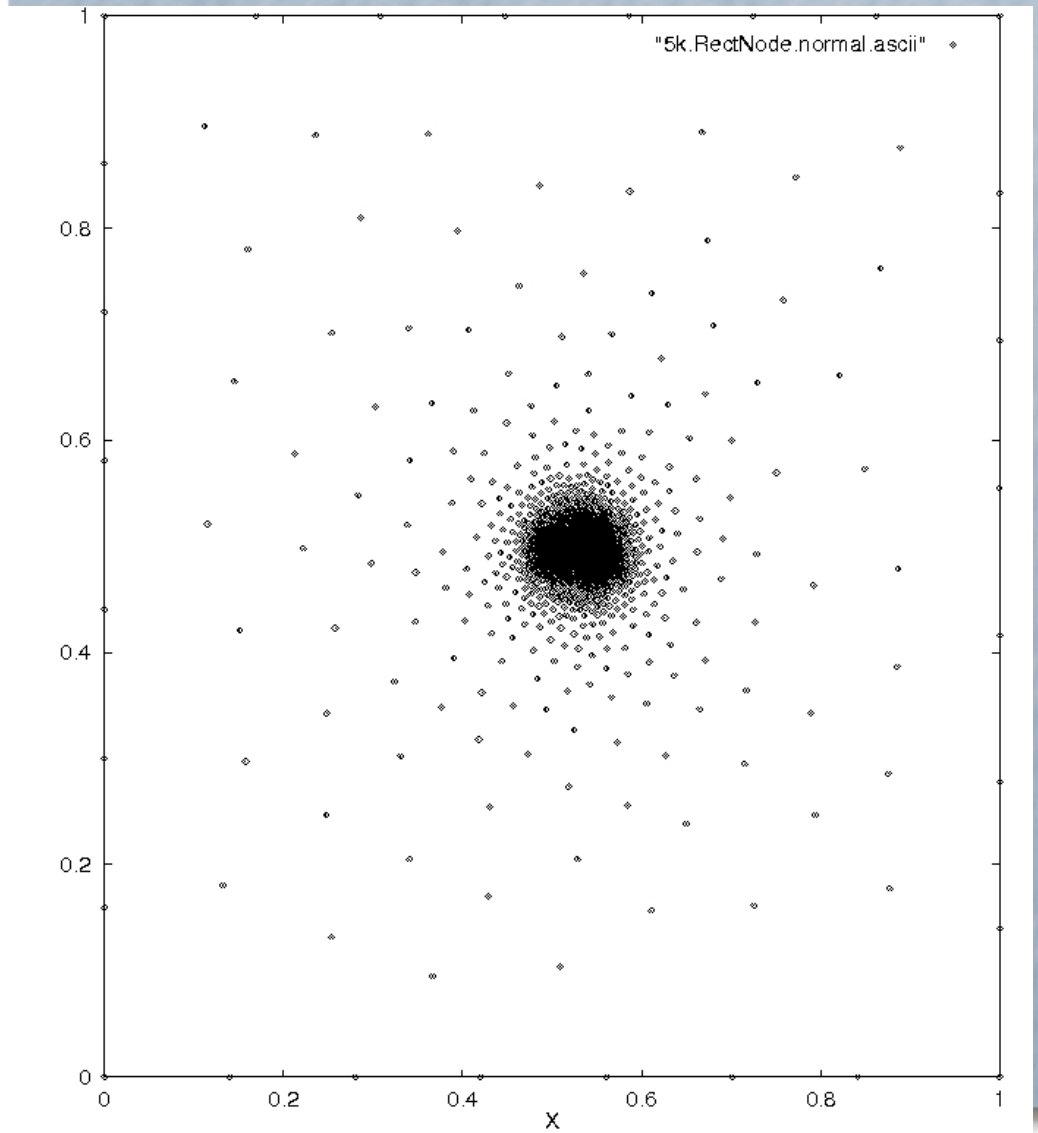
- Template - <int dimensions, class point_t, class T>
  - point_t must have an operator[ ] to get dimension x_i
- Basic operations:
  - constructor(const point_t& min, const point_t&max)
  - copy constructor and operator=, destructor
  - bool empty() const, void clear()
  - max_depth() const (1 if grid)
  - build(std::InputIterator<point_t> first,
    std::InputIterator<point_t> last)

Project 1 – More details

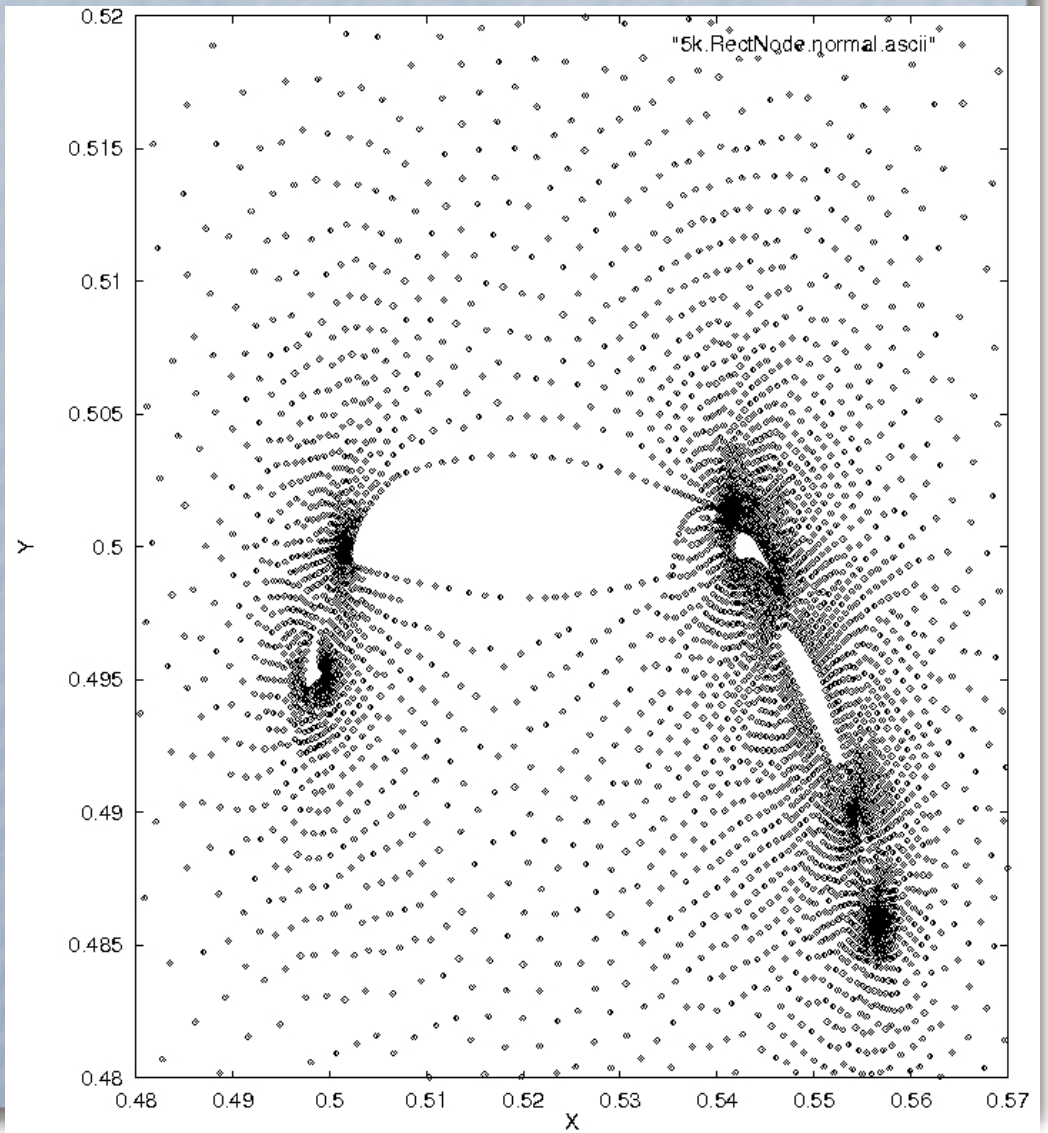
- Data structure specific, consider how to specify grid spacing, splitting heuristic, etc..
- Queries (set results, return the number found)
  - Inclusive and non-inclusive versions
  - int range_search_inclusive(const point_t& min,
    const point_t& max,
    std::vector<point_t>& results)const
  - int radius_search_noninclusive(const point_t& center,
    const T & radius,
    std::vector<point_t>& results)const

Project 1 Datasets on Course Website

Computational Fluid Dynamics Wing Simulation



Range Search Result

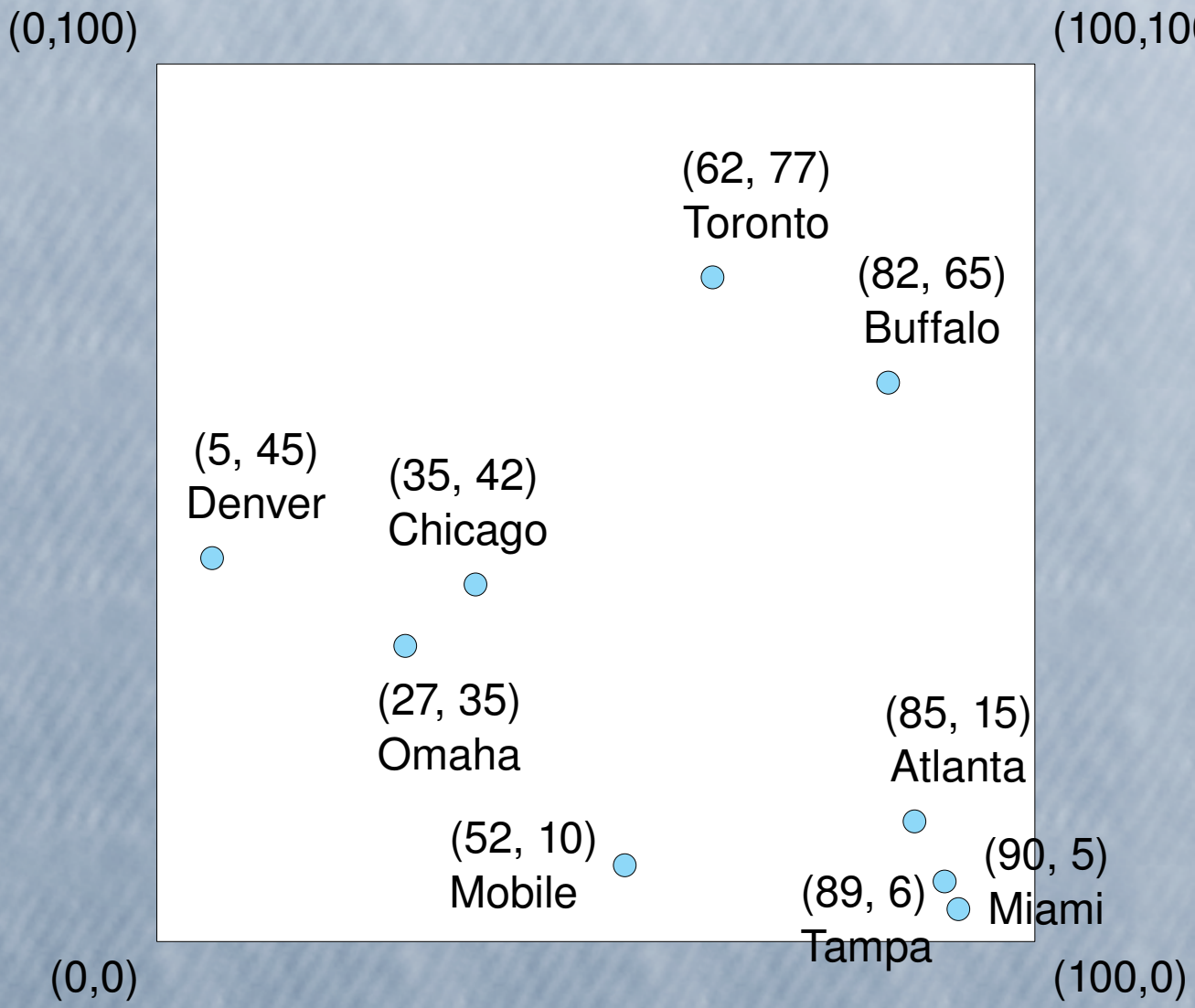


More Query Operations

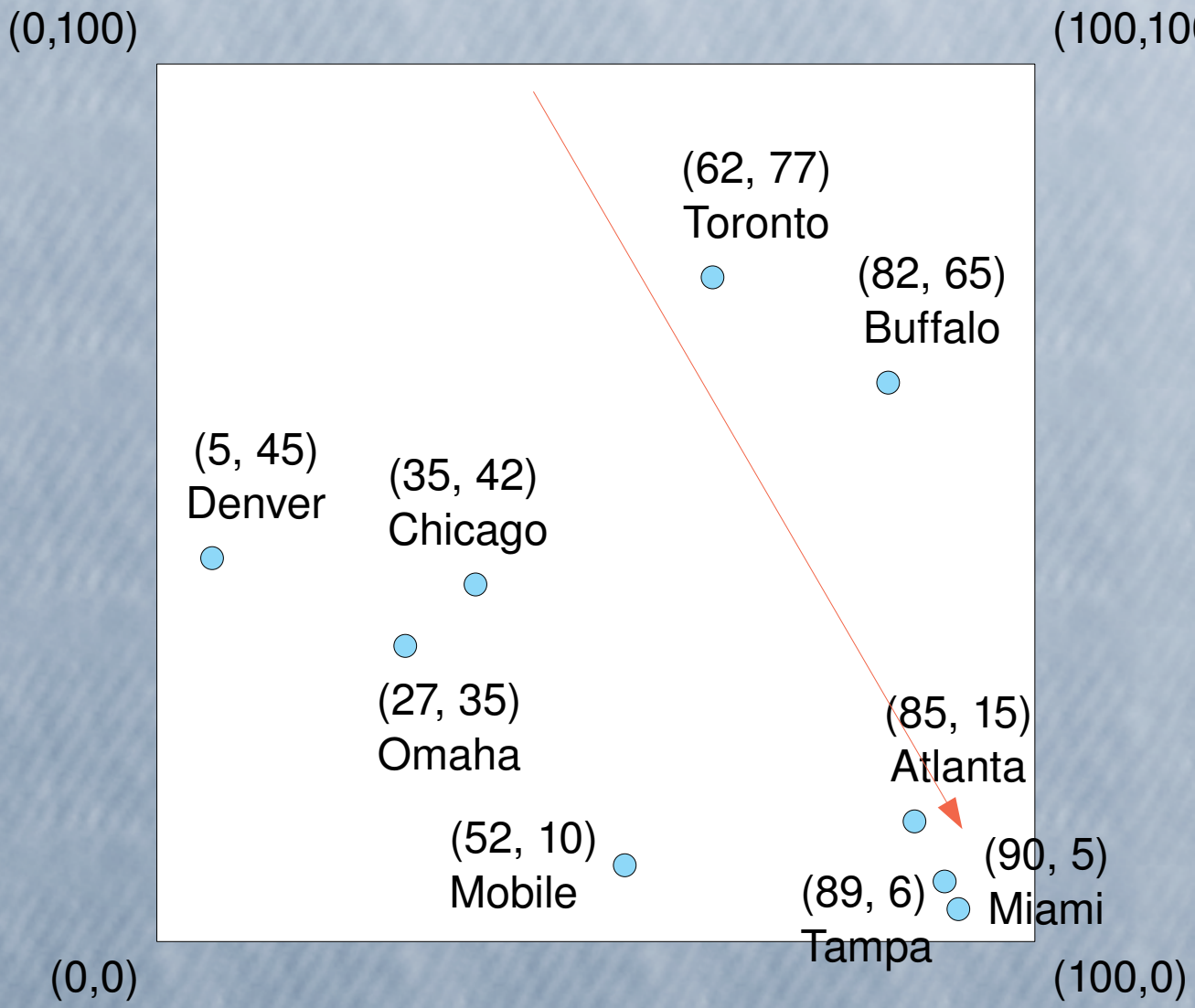
- Single search, single result: Ray trace
- Single search, multiple result: Range search
- Multiple search, single result: Nearest of all pairs
- Multiple search, multiple result: All pairs within some distance of each other (collision detection)

Ray Trace on Point Data

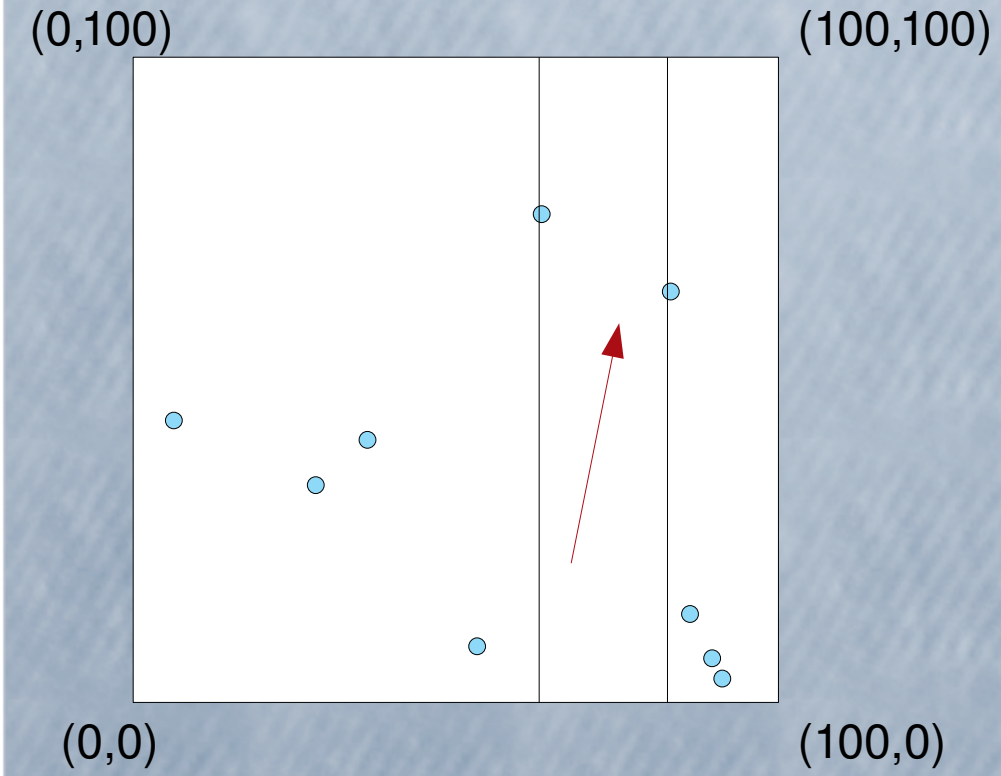
Ray Trace on Point Data

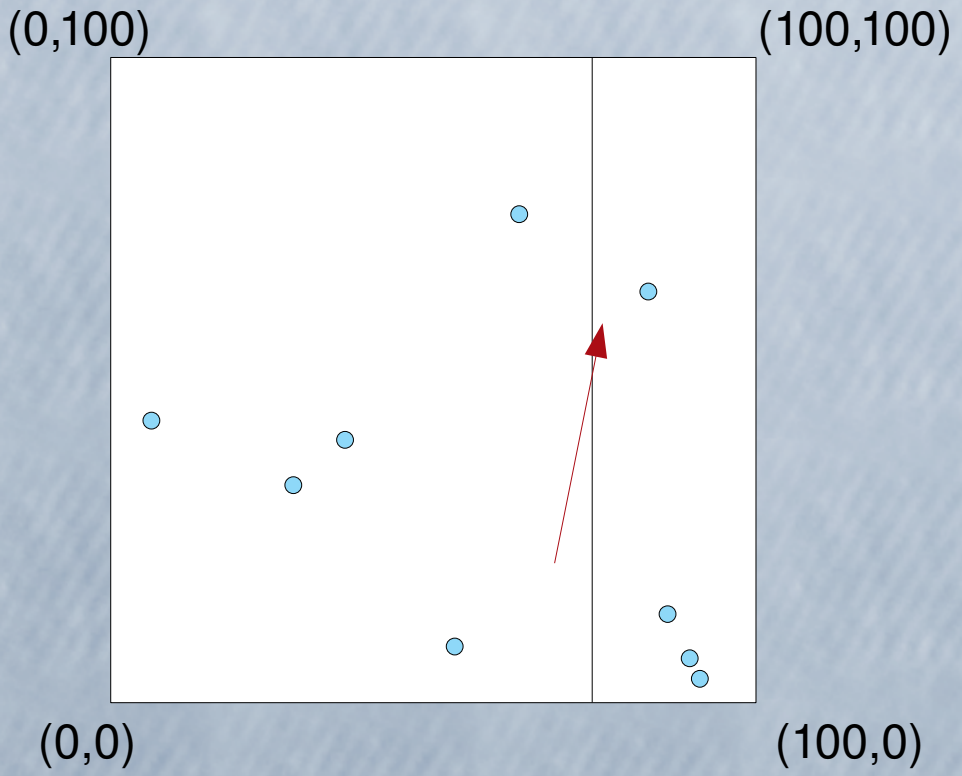


Ray Trace on Point Data



Split On Points or Between Points?





- Empty space culling – Ray Tracing

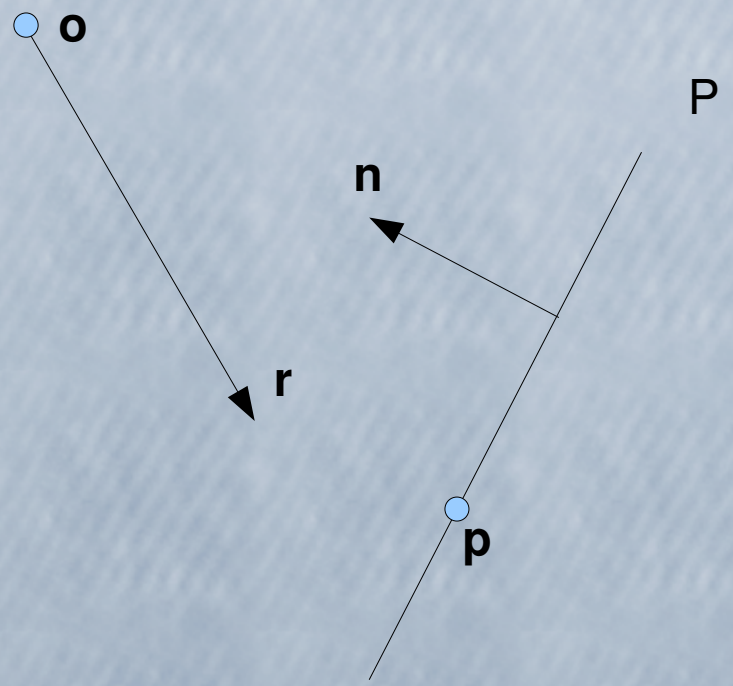
Ray-Plane Intersection

- Plane P: $\mathbf{p} \cdot \mathbf{n} = d$
- Ray: $\mathbf{o} + t\,\mathbf{r}$
- Intersection:

$$(\mathbf{o} + t\,\mathbf{r}) \cdot \mathbf{n} = d$$

$$\mathbf{o} \cdot \mathbf{n} + t\,\mathbf{r} \cdot \mathbf{n} = d$$

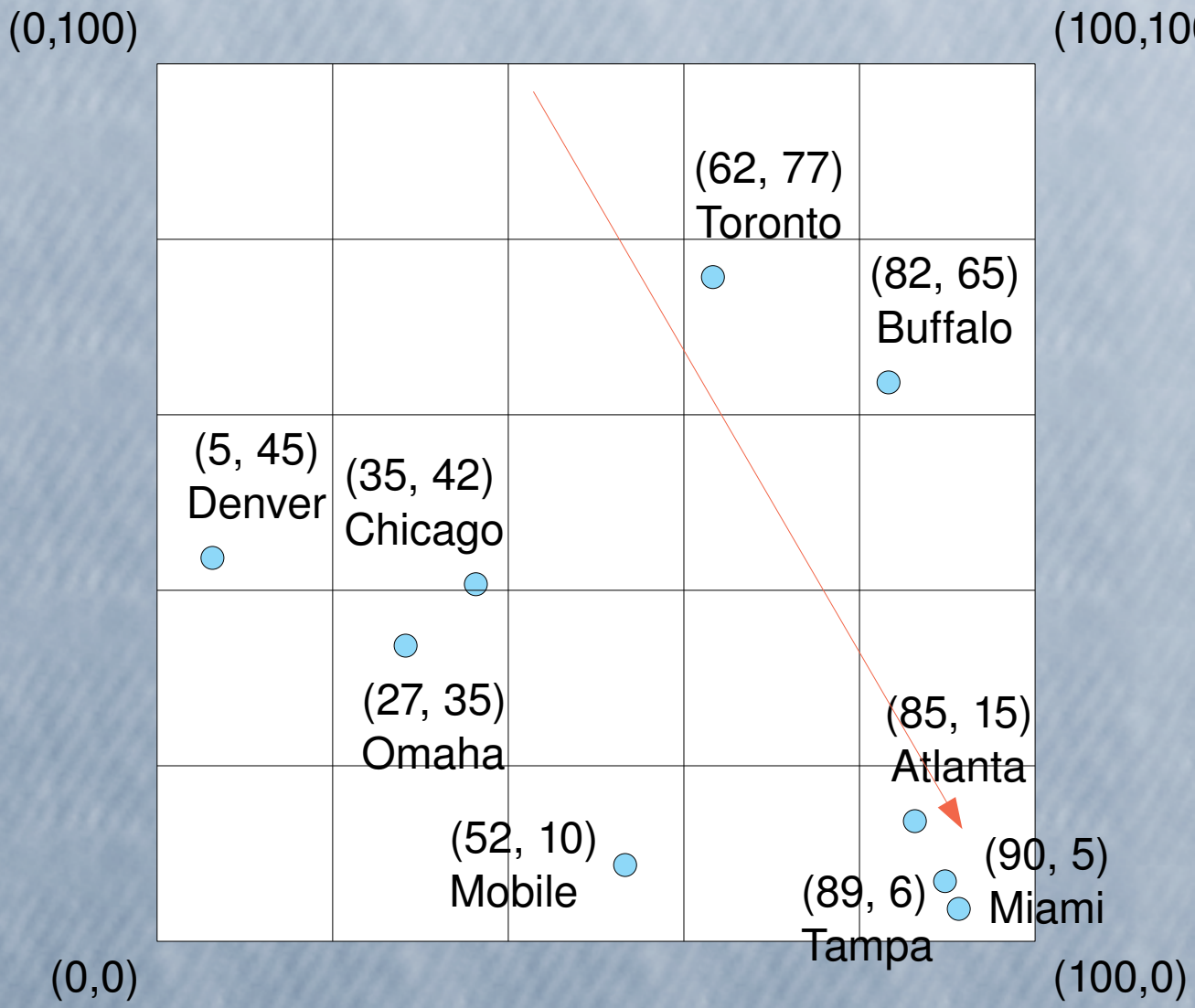
$$t = (\mathrm{d} - \mathbf{o} \cdot \mathbf{n}) / (\mathbf{r} \cdot \mathbf{n})$$



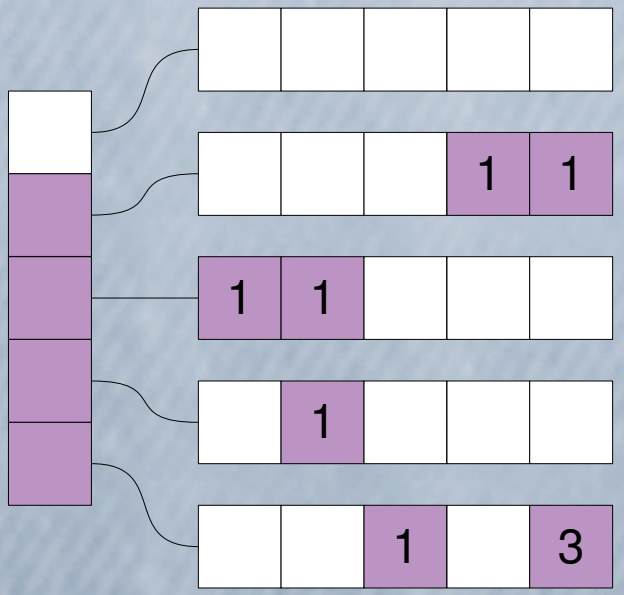
Brute Force Ray Trace

```
function RayTrace(Point origin, Point direction
                  real &r_time, hit_data & r_hit
  for each city in Cities
    for i = 0 to d – 1
      real time; hit_data hit;
      if(RayTrace(origin, direction, city, time, hit)
        if(time < rtime)
          rtime = time;
          r_hit = hit;
        fi
      fi
    end
  end
end
```

Regular Grids

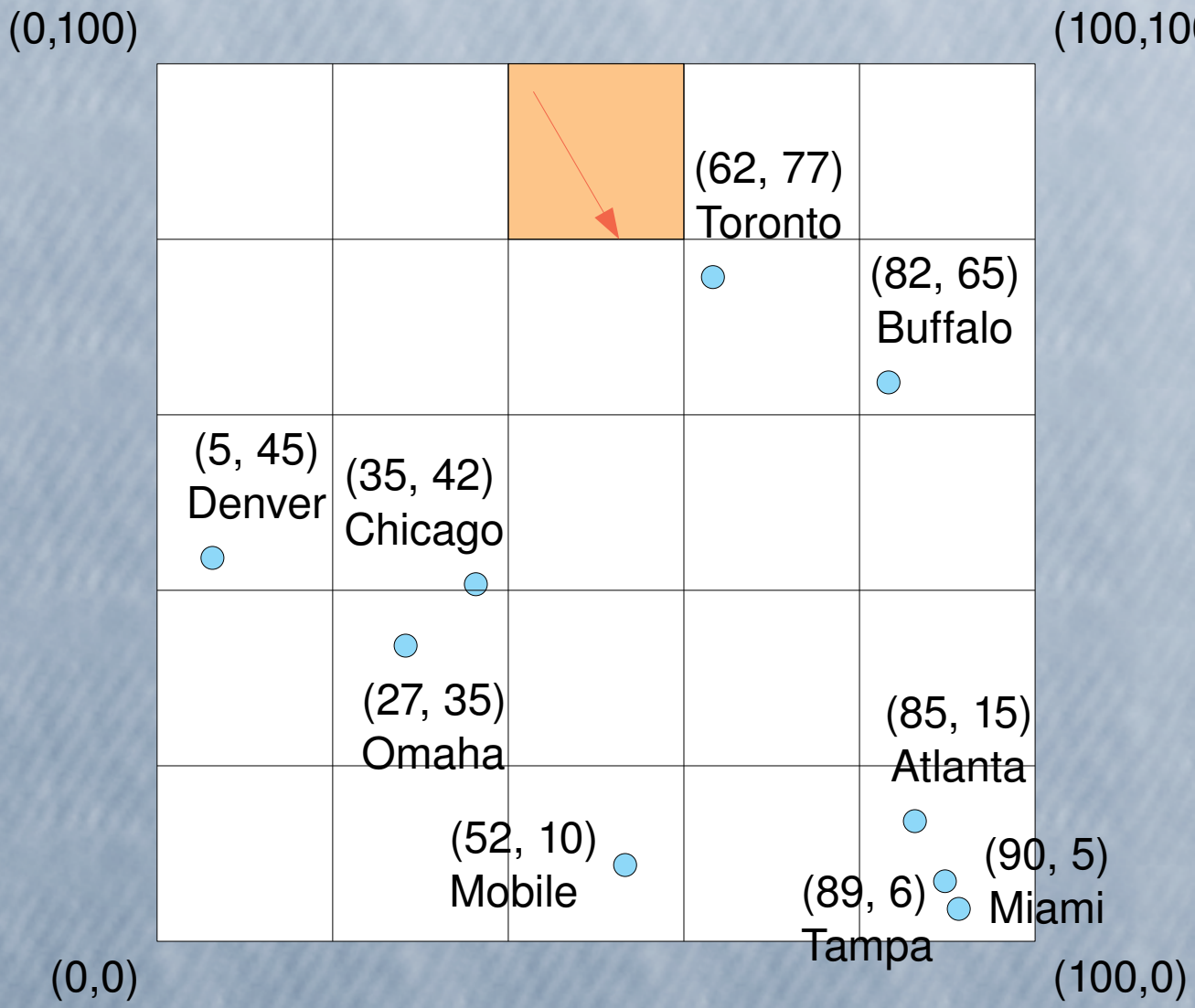


Matrix as ND Array

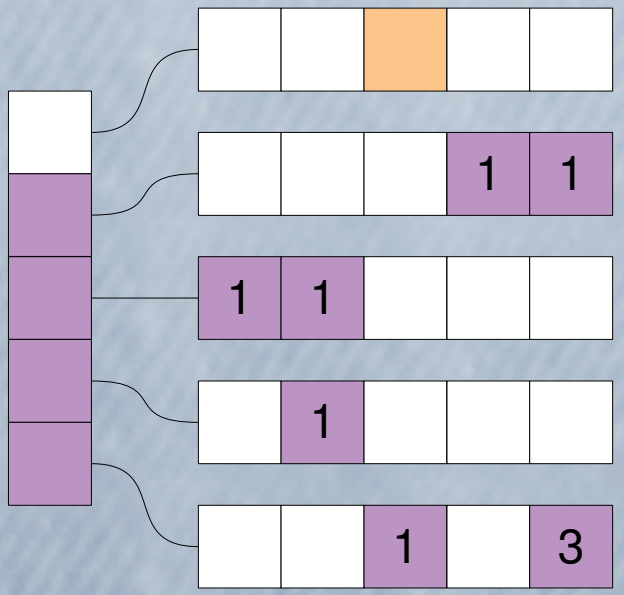


(Could also use sparse matrix representations)

Regular Grids

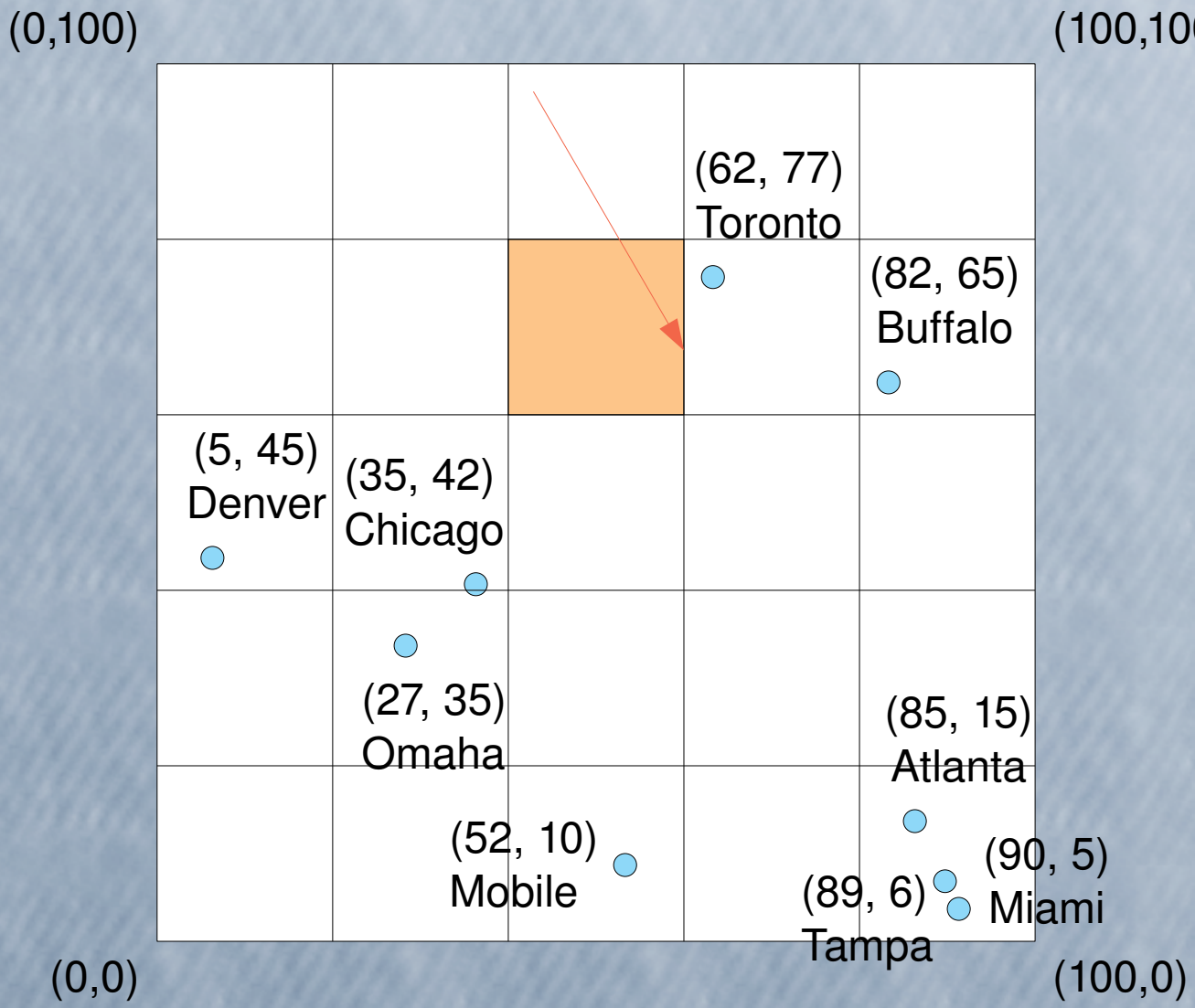


Matrix as ND Array

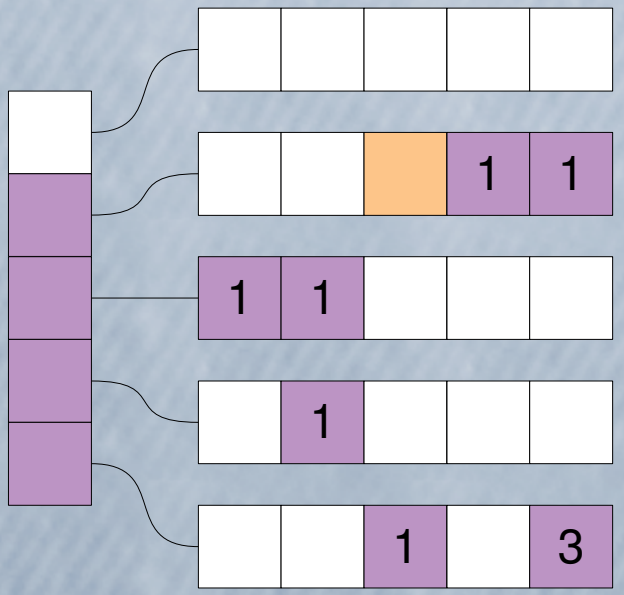


(Could also use sparse matrix representations)

Regular Grids

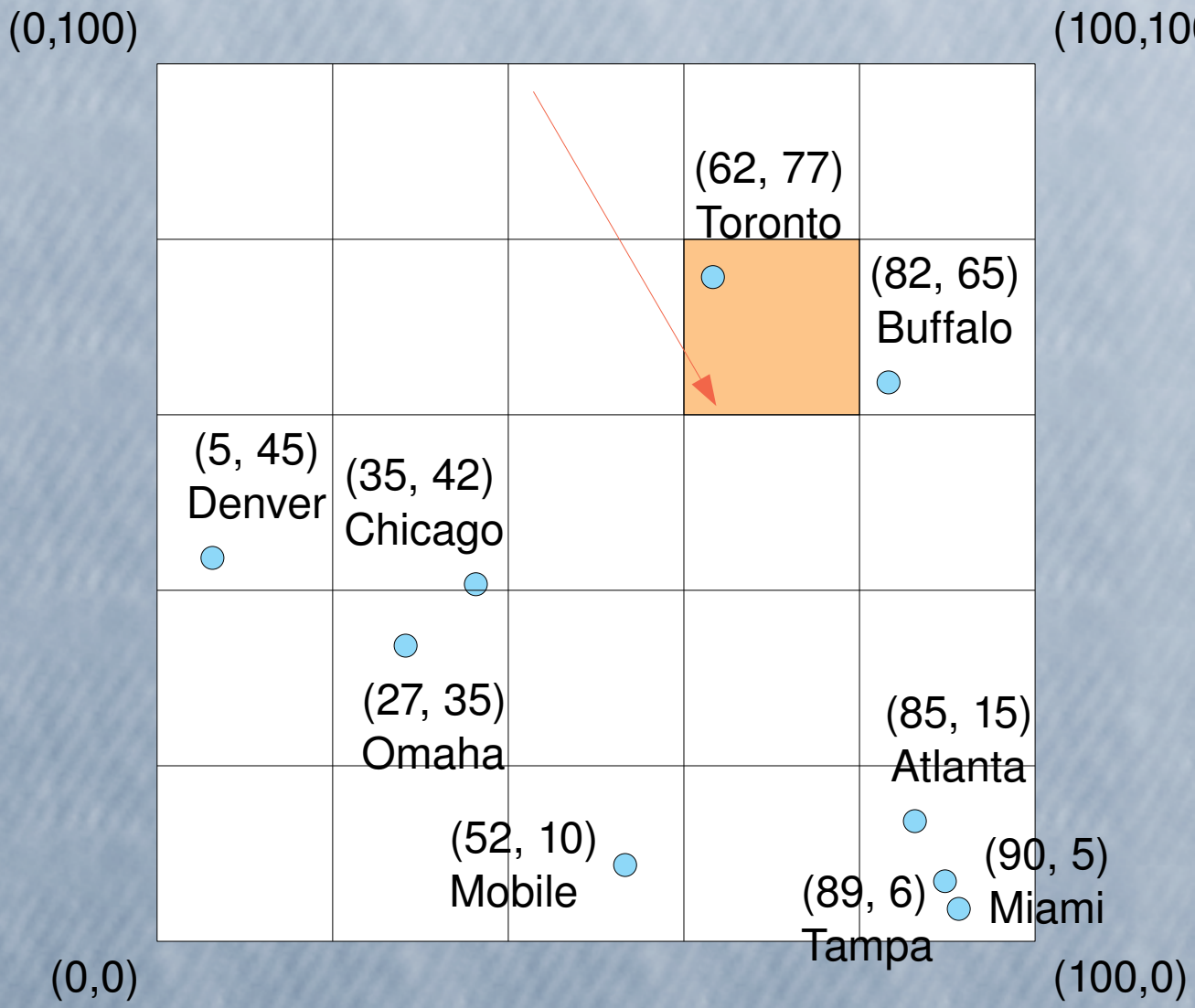


Matrix as ND Array

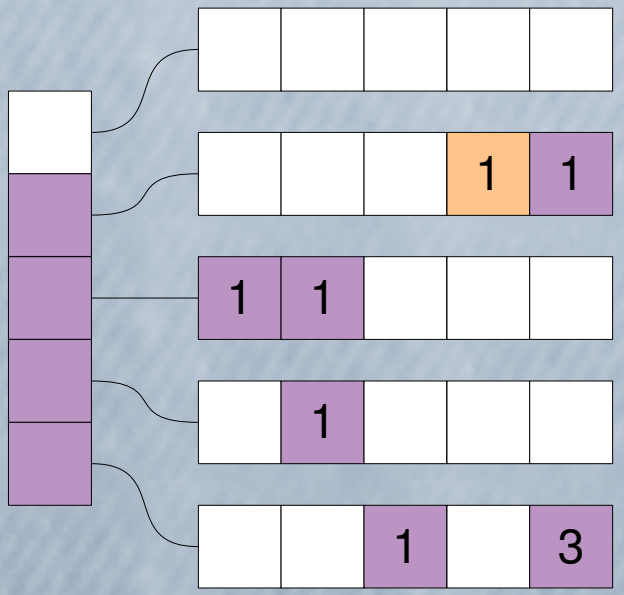


(Could also use sparse matrix representations)

Regular Grids

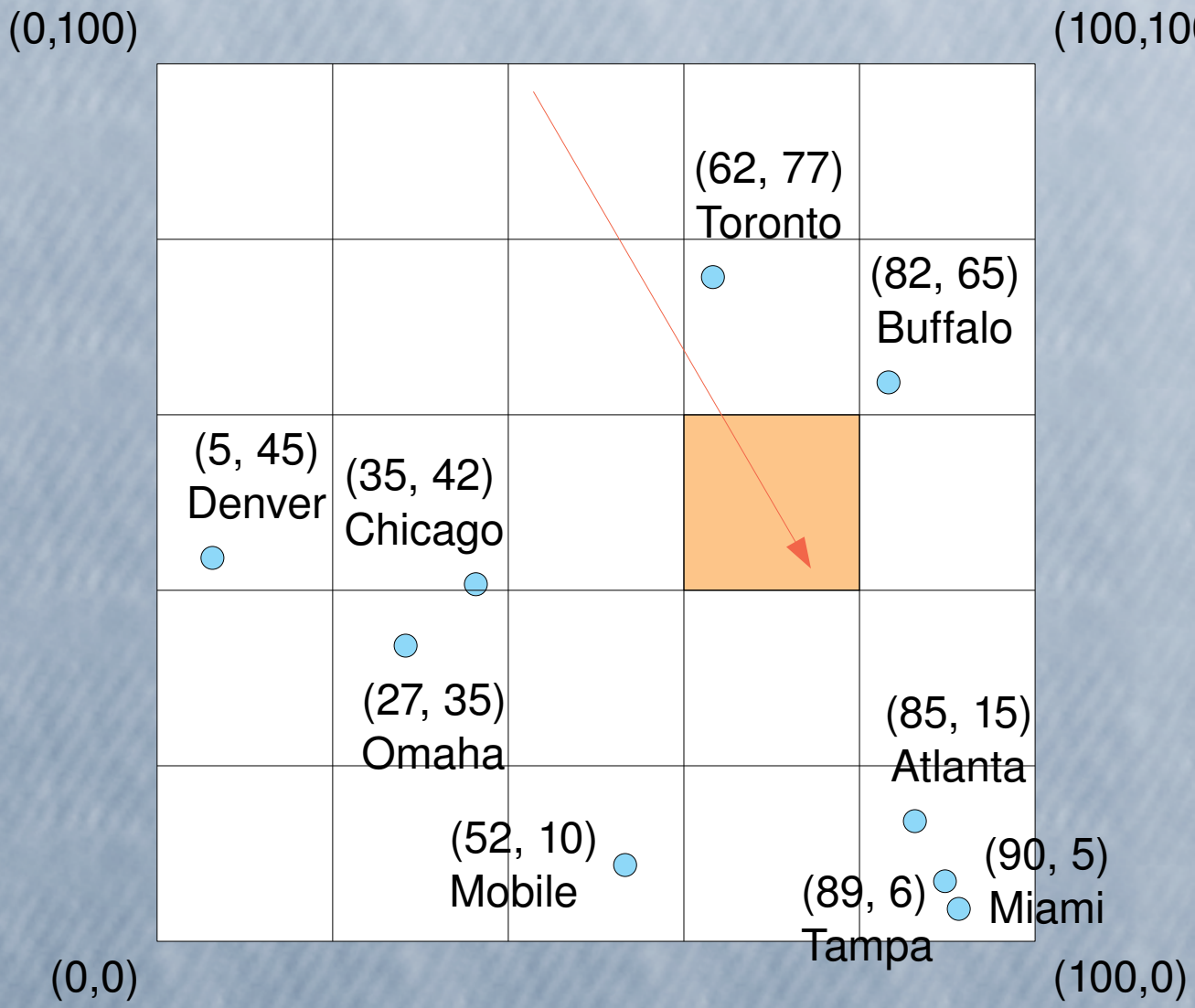


Matrix as ND Array

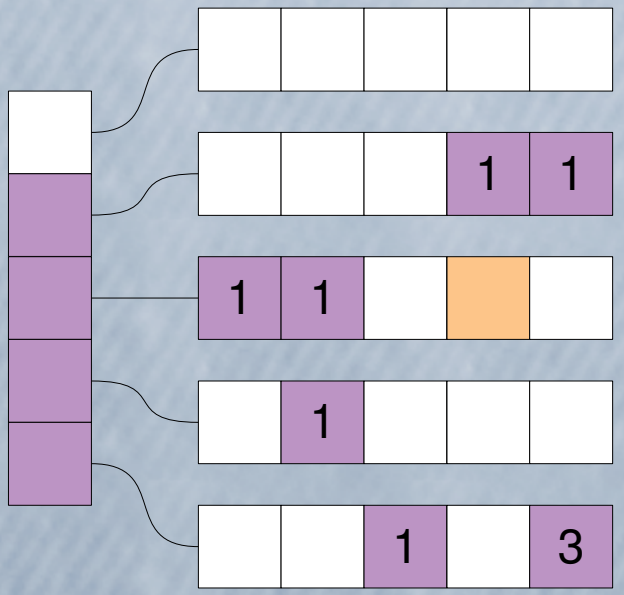


(Could also use sparse matrix representations)

Regular Grids

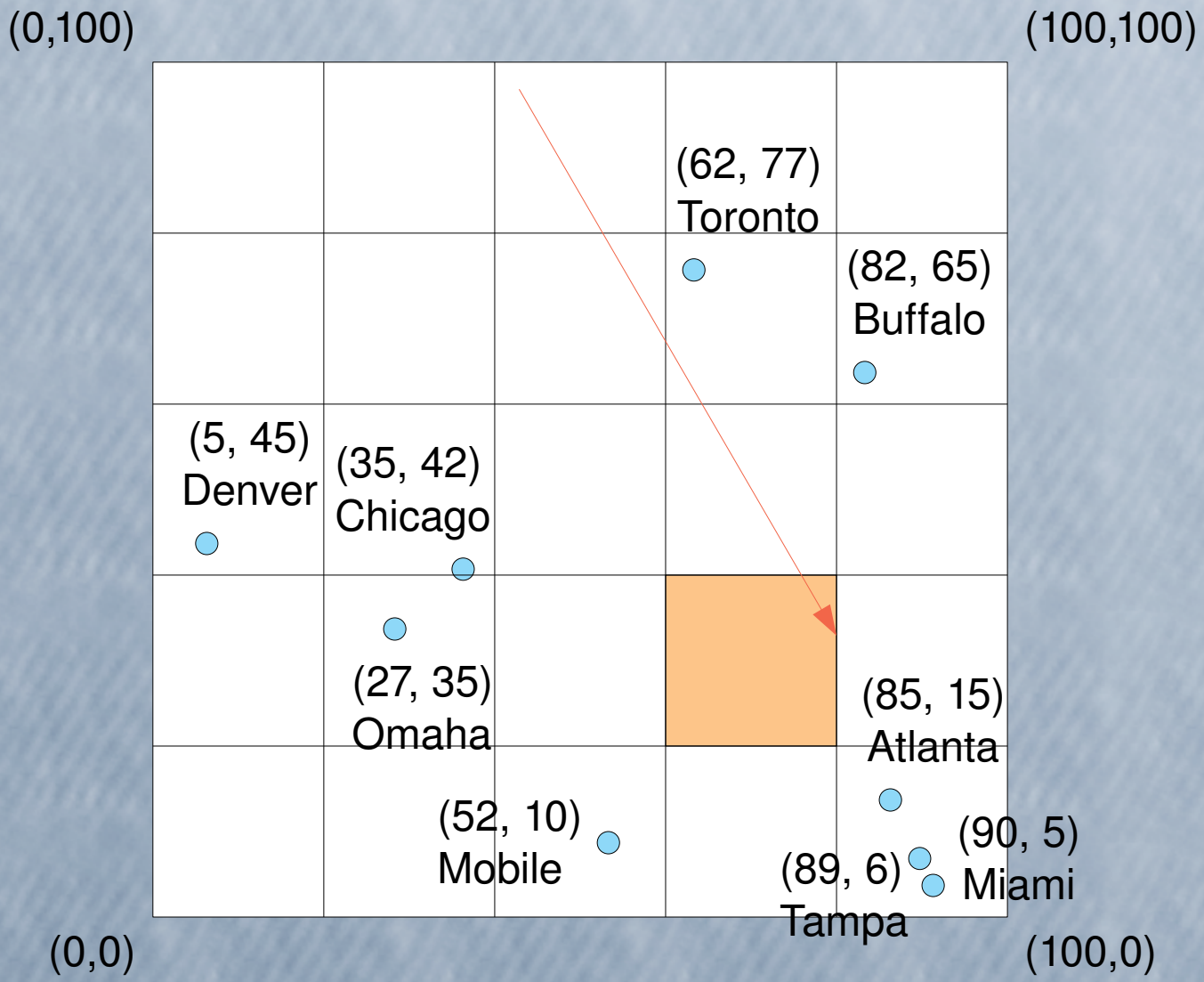


Matrix as ND Array

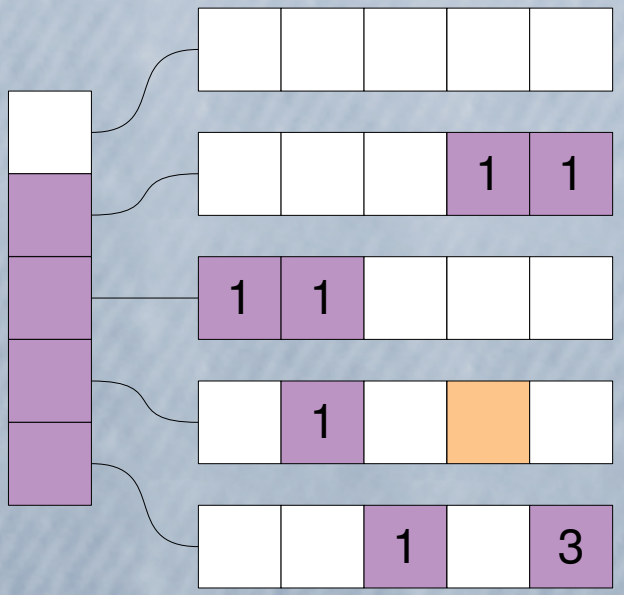


(Could also use sparse matrix representations)

Regular Grids



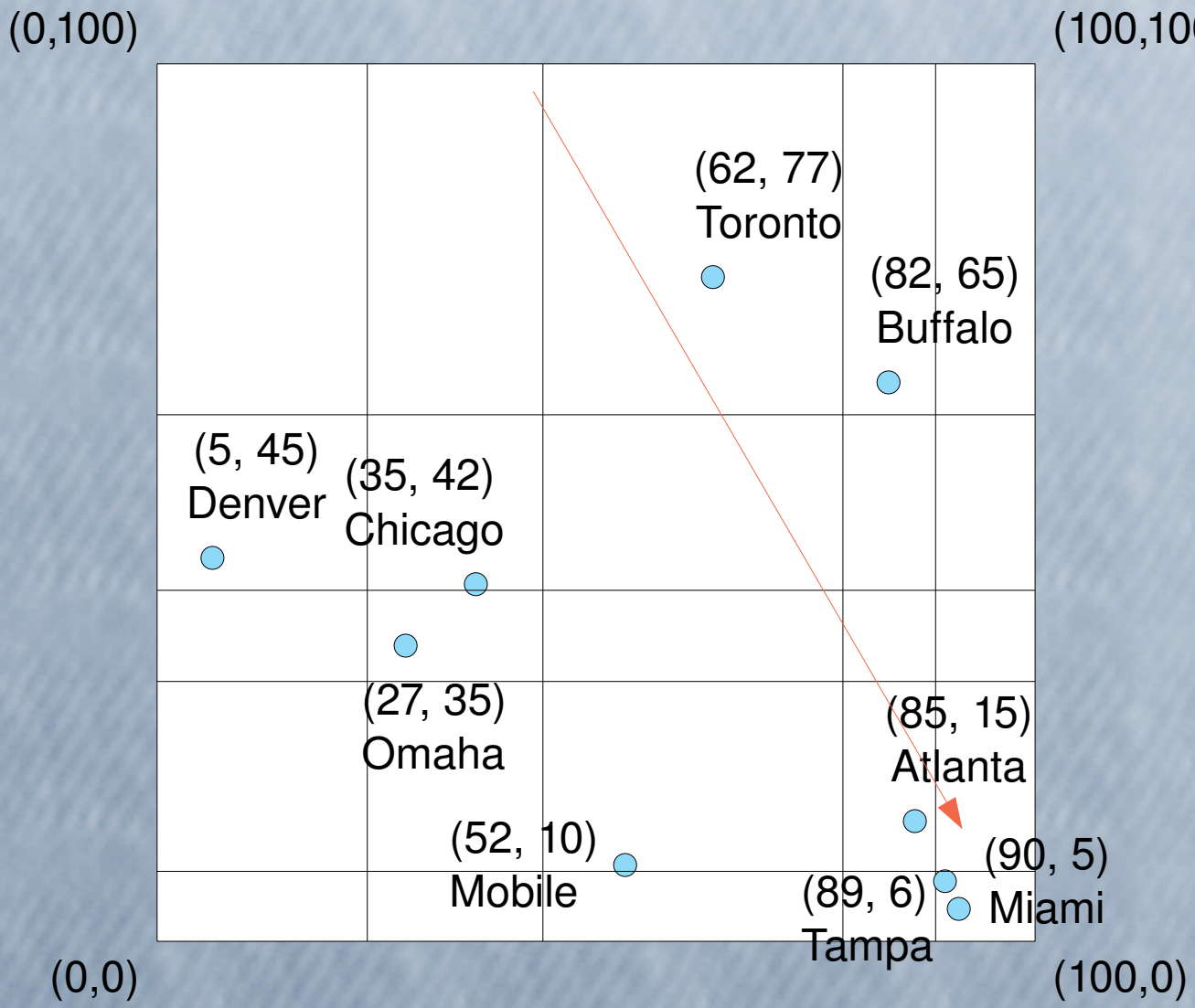
Matrix as ND Array



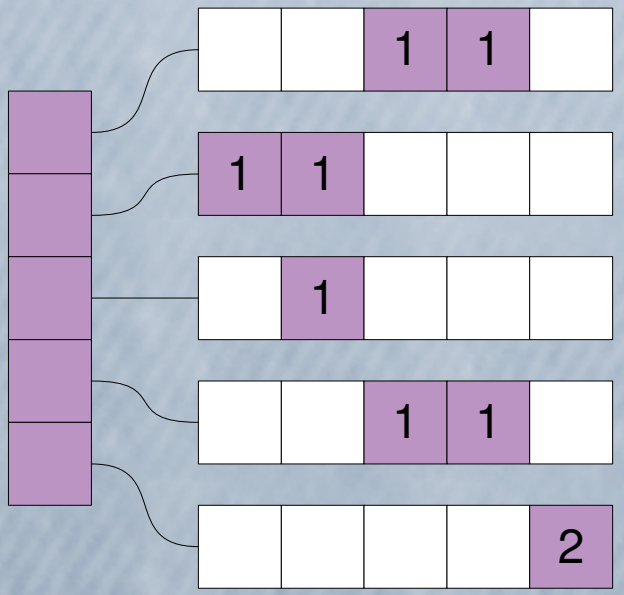
(Could also use sparse matrix representations)

Is there a repeating access pattern? Why?

Irregular Grids

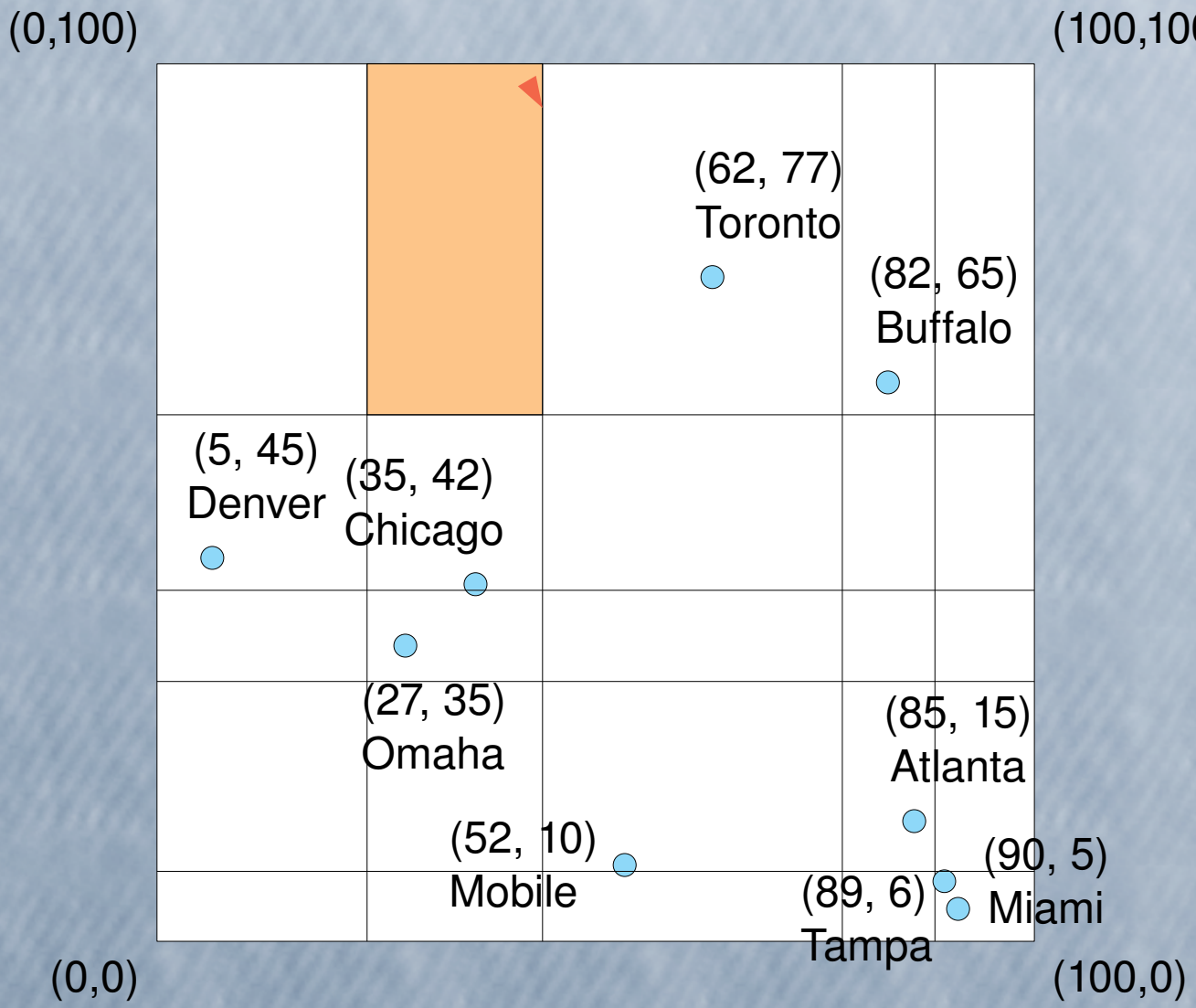


Matrix as ND Array

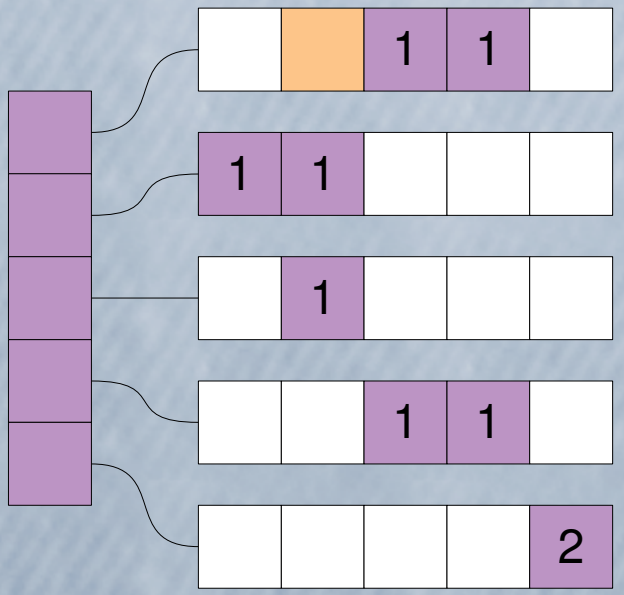


(Could also use sparse matrix representations)

Irregular Grids

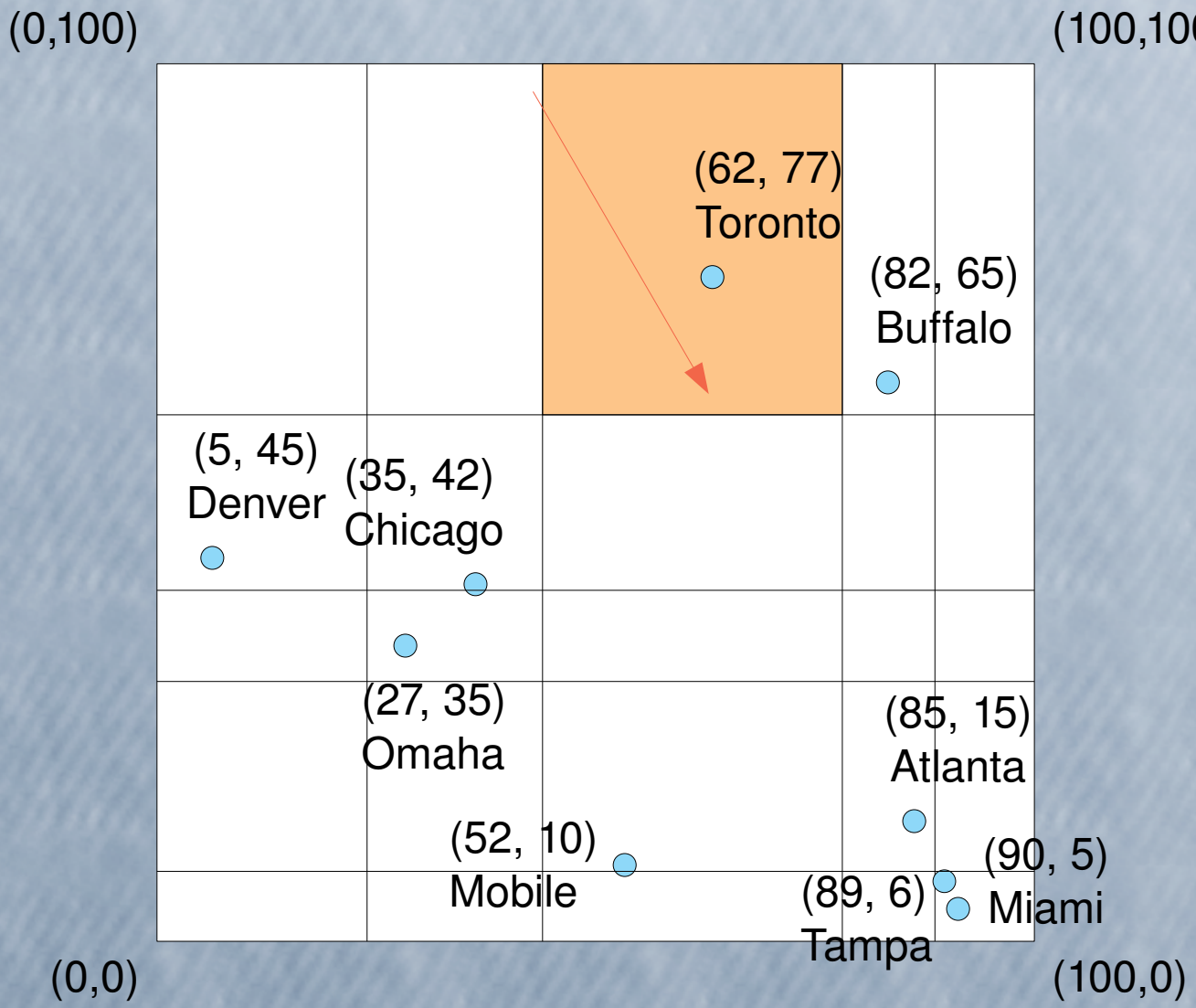


Matrix as ND Array

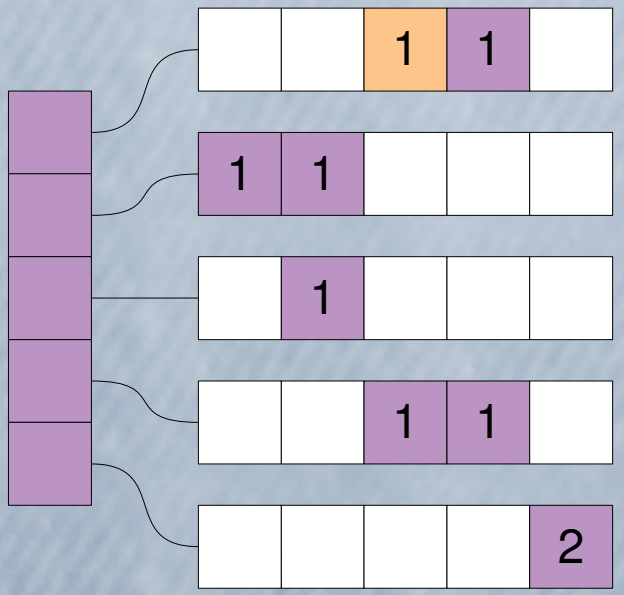


(Could also use sparse matrix representations)

Irregular Grids

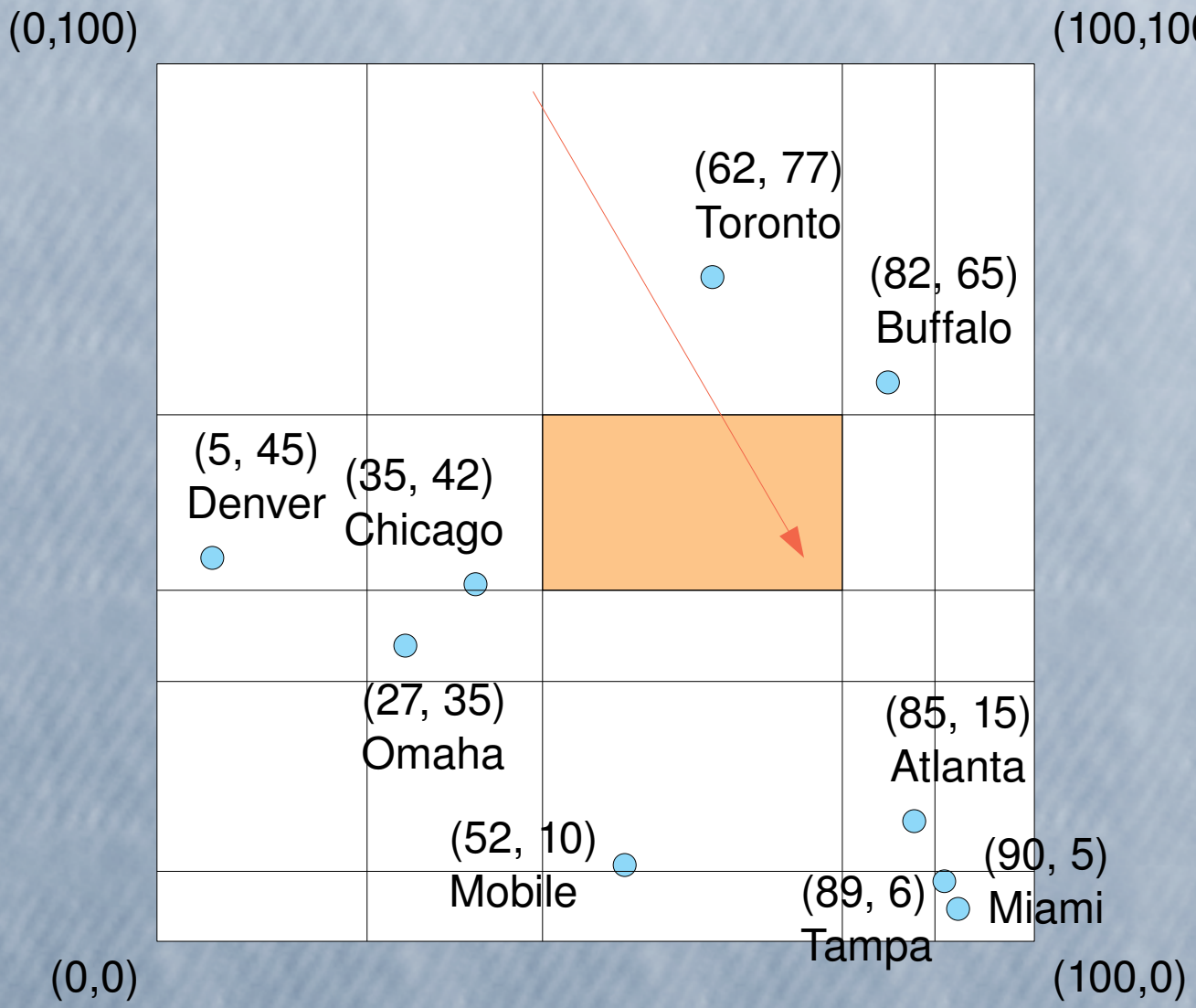


Matrix as ND Array

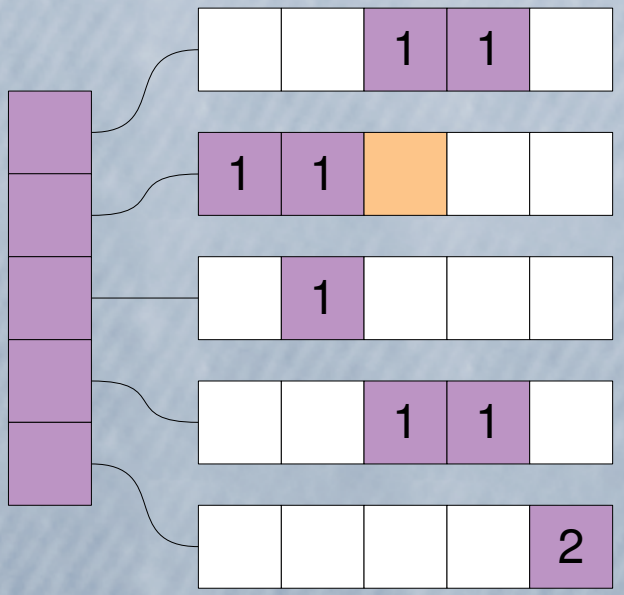


(Could also use sparse matrix representations)

Irregular Grids

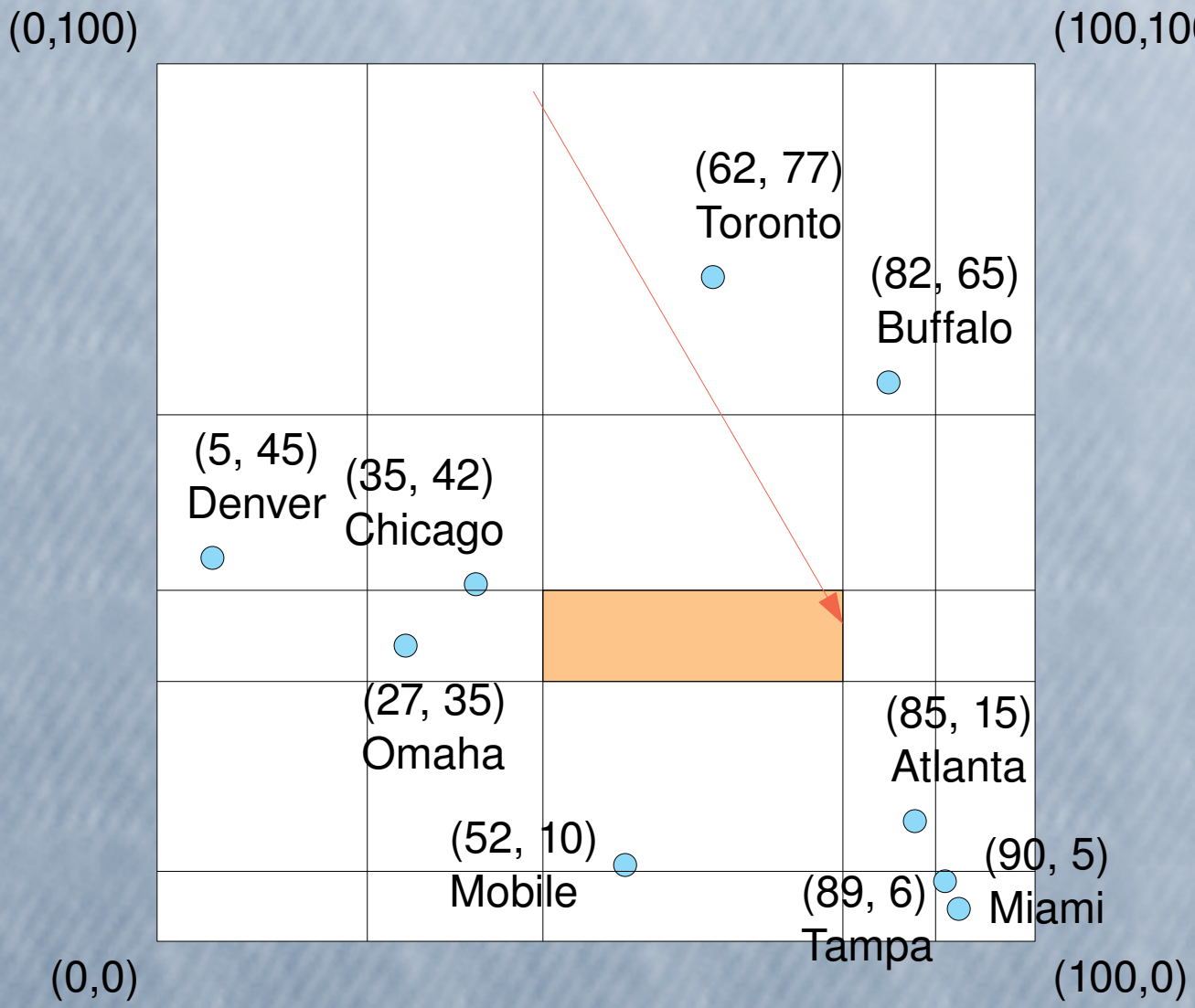


Matrix as ND Array

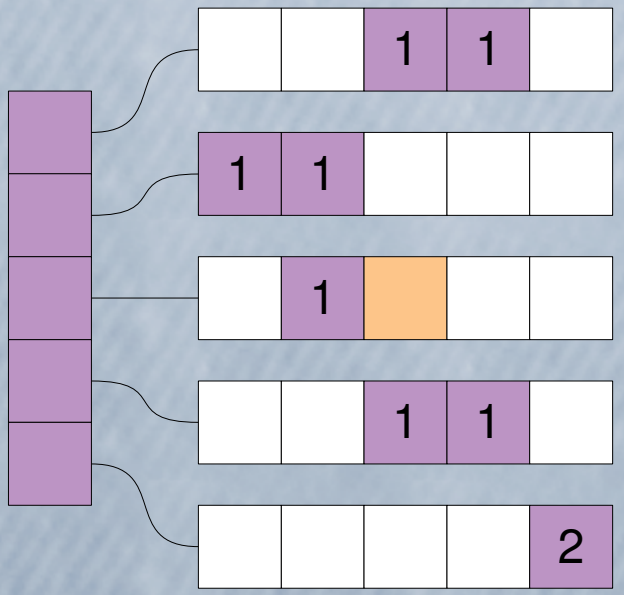


(Could also use sparse matrix representations)

Irregular Grids

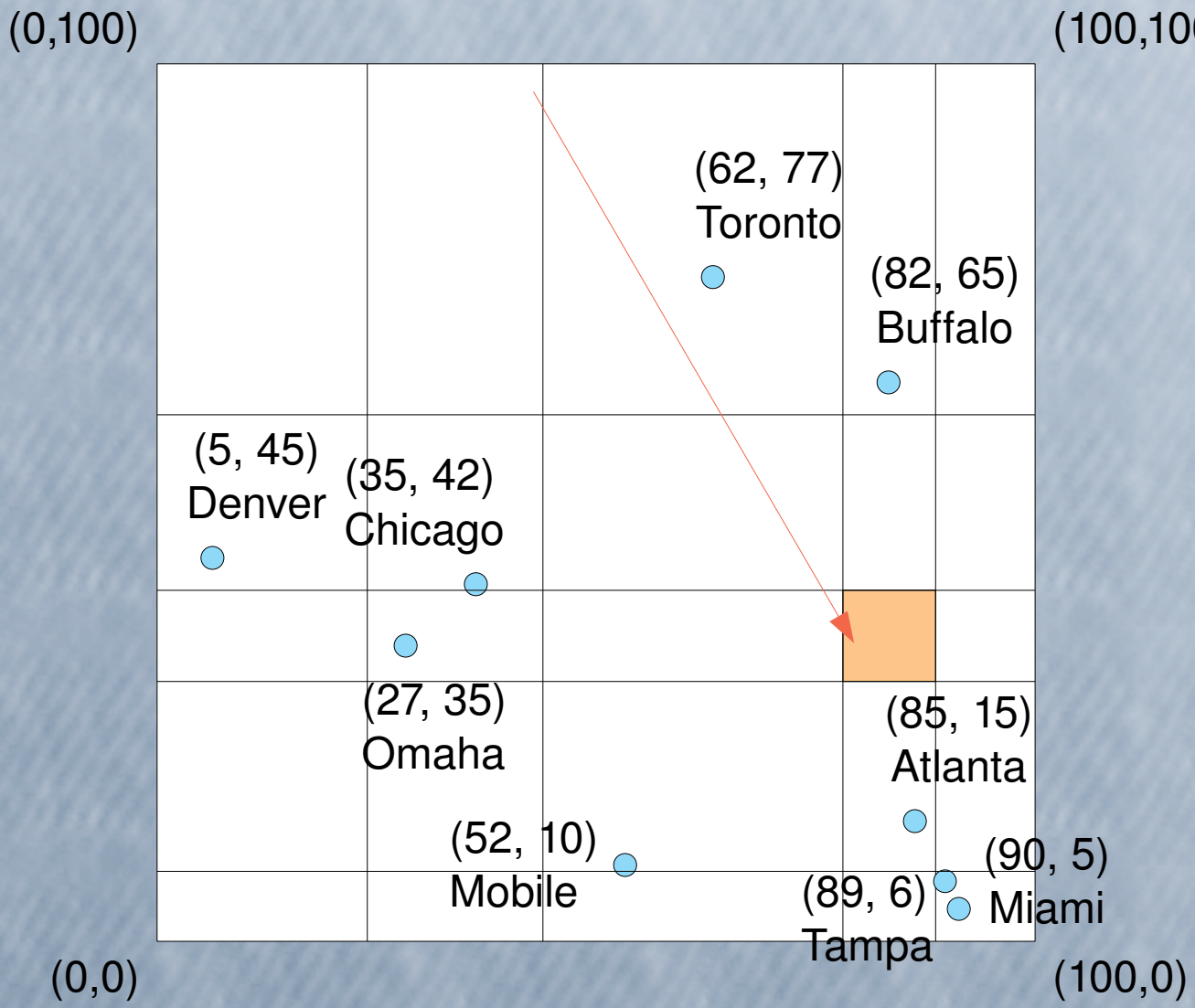


Matrix as ND Array

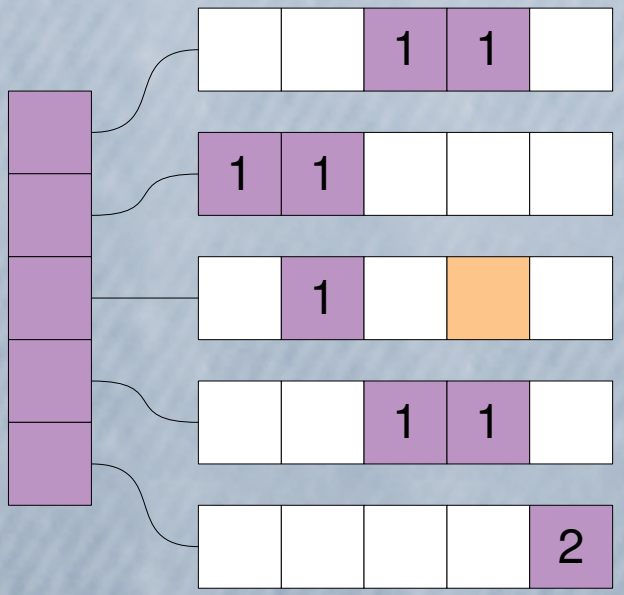


(Could also use sparse matrix representations)

Irregular Grids

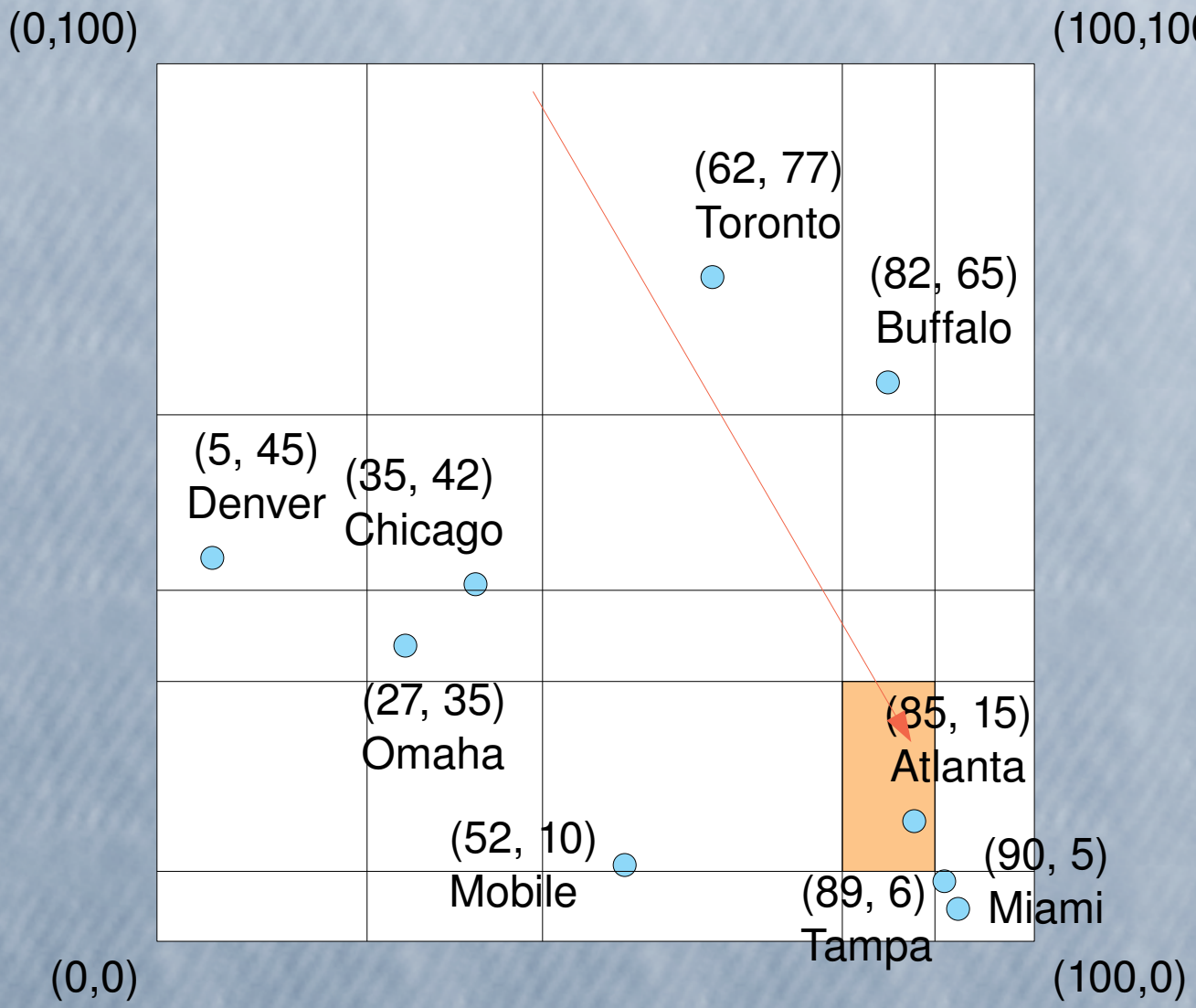


Matrix as ND Array

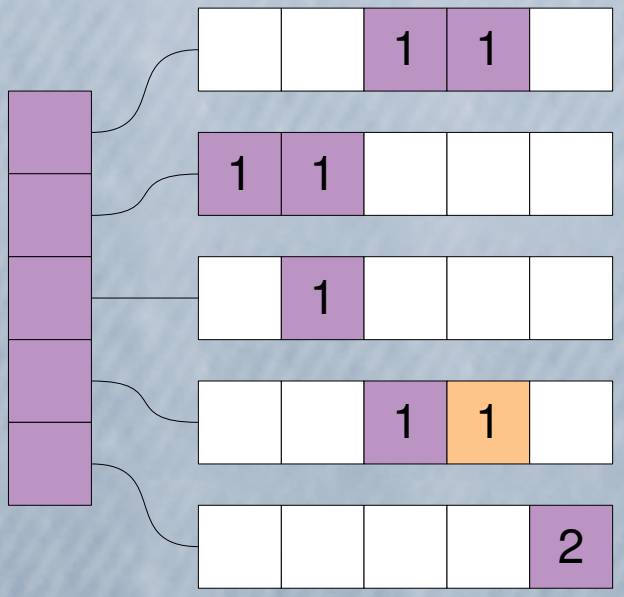


(Could also use sparse matrix representations)

Irregular Grids



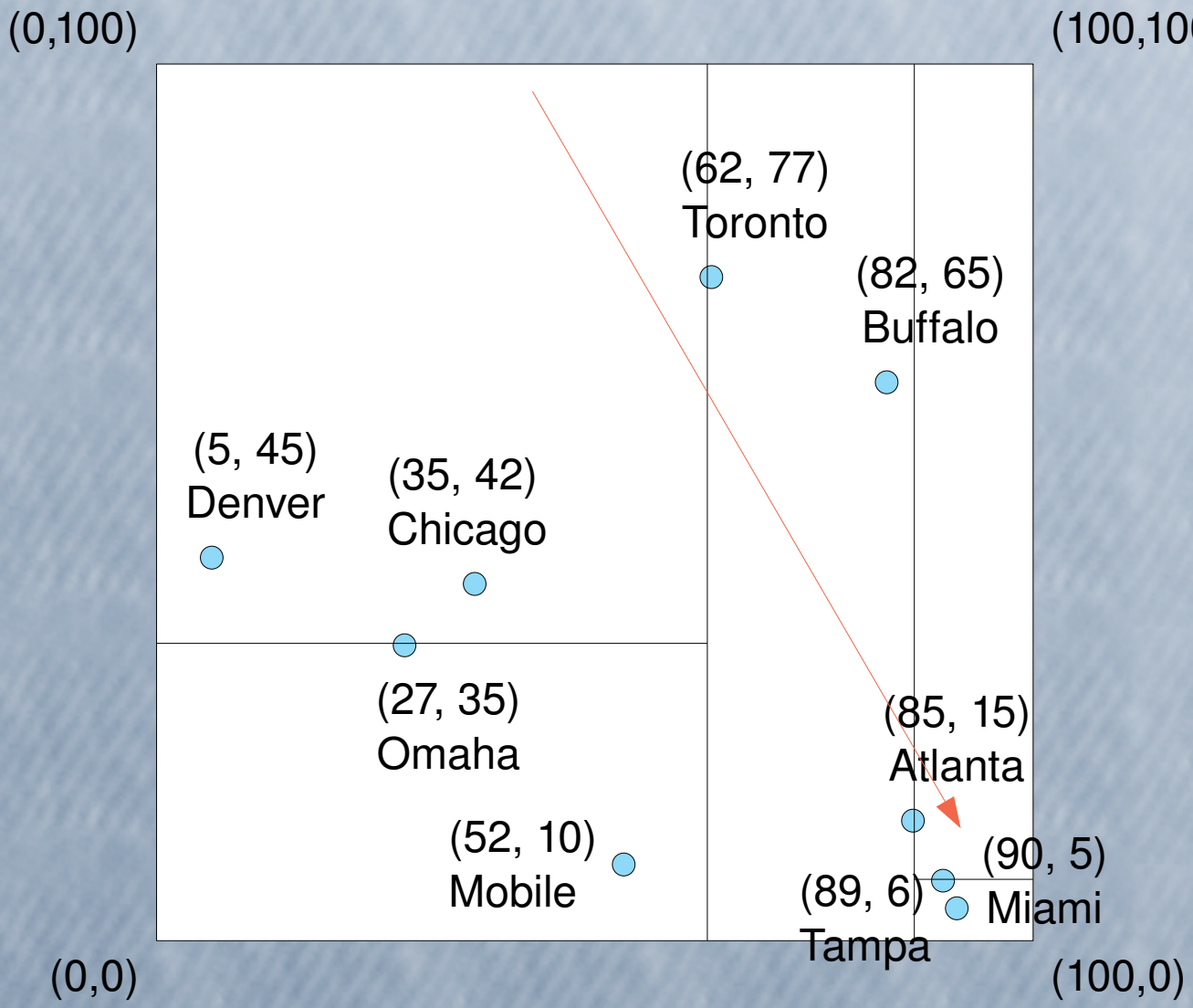
Matrix as ND Array



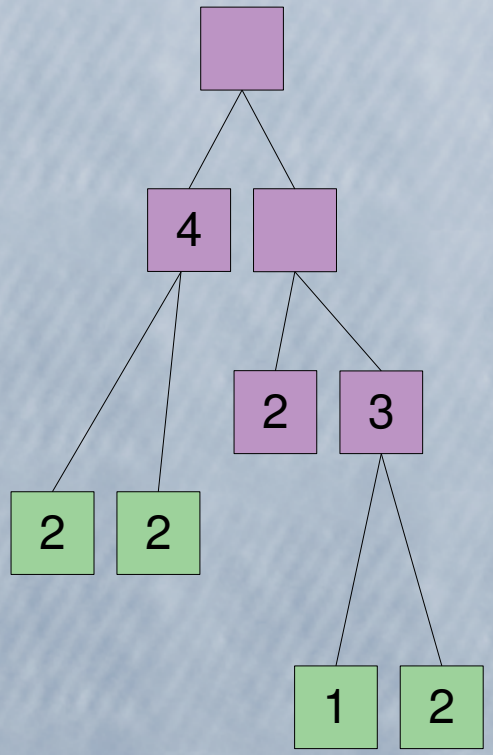
(Could also use sparse matrix representations)

Is there a repeating access pattern? Why?

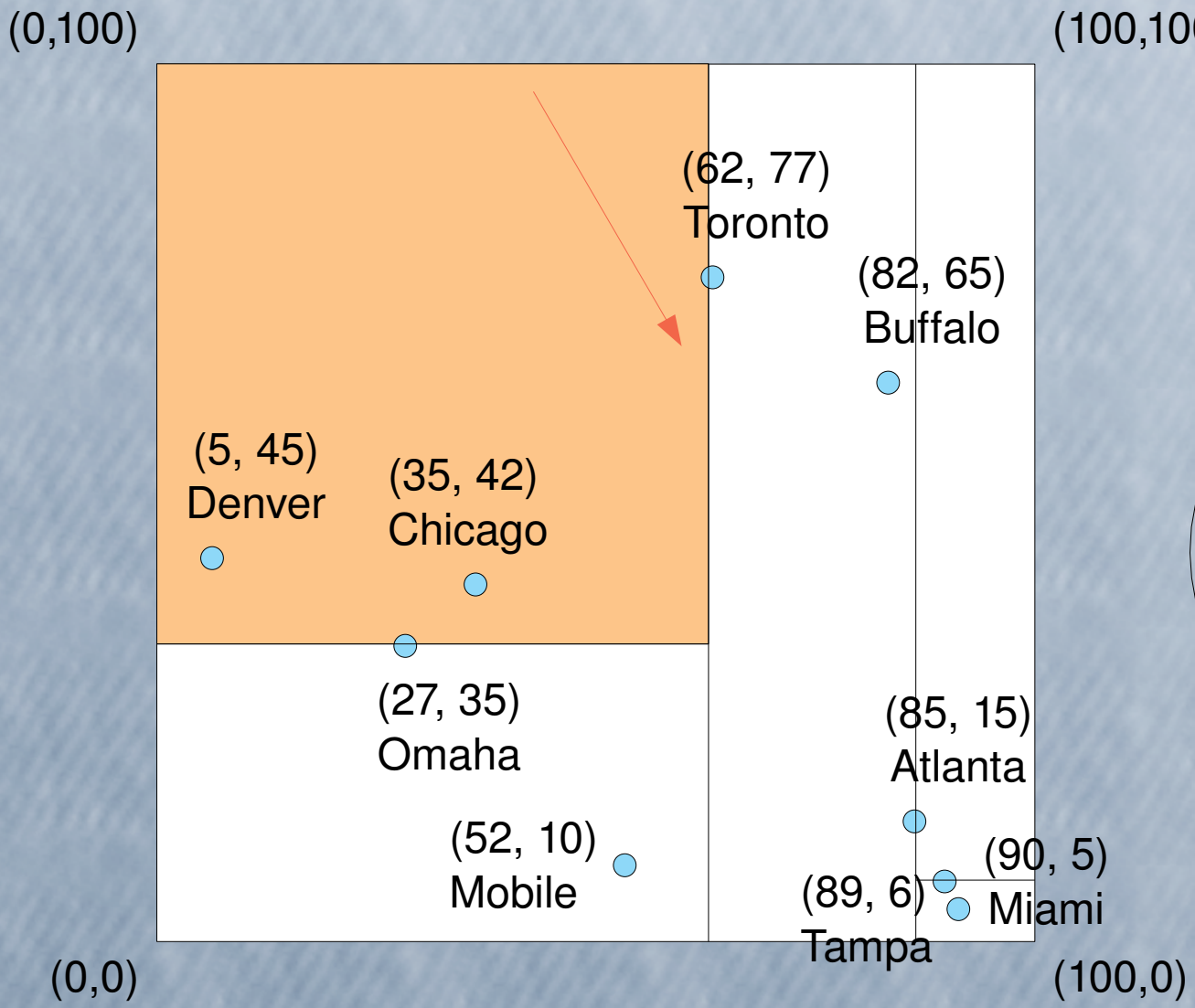
Multidimensional Range Trees



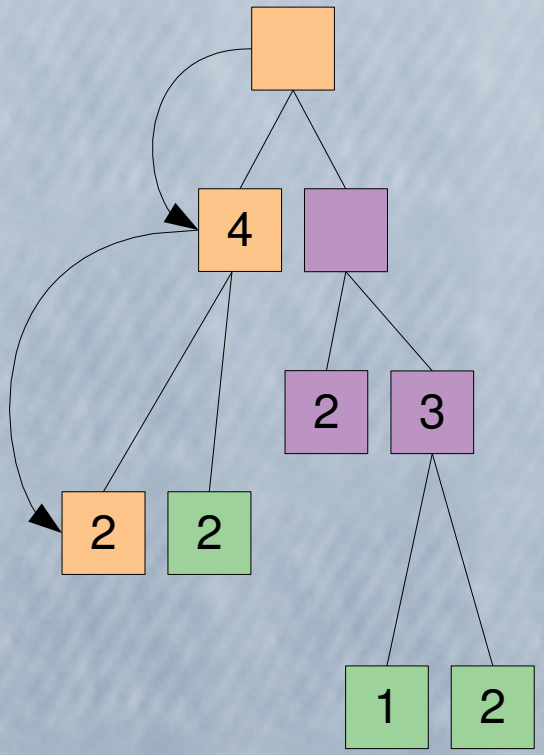
Multidimensional
Tree Data Structure



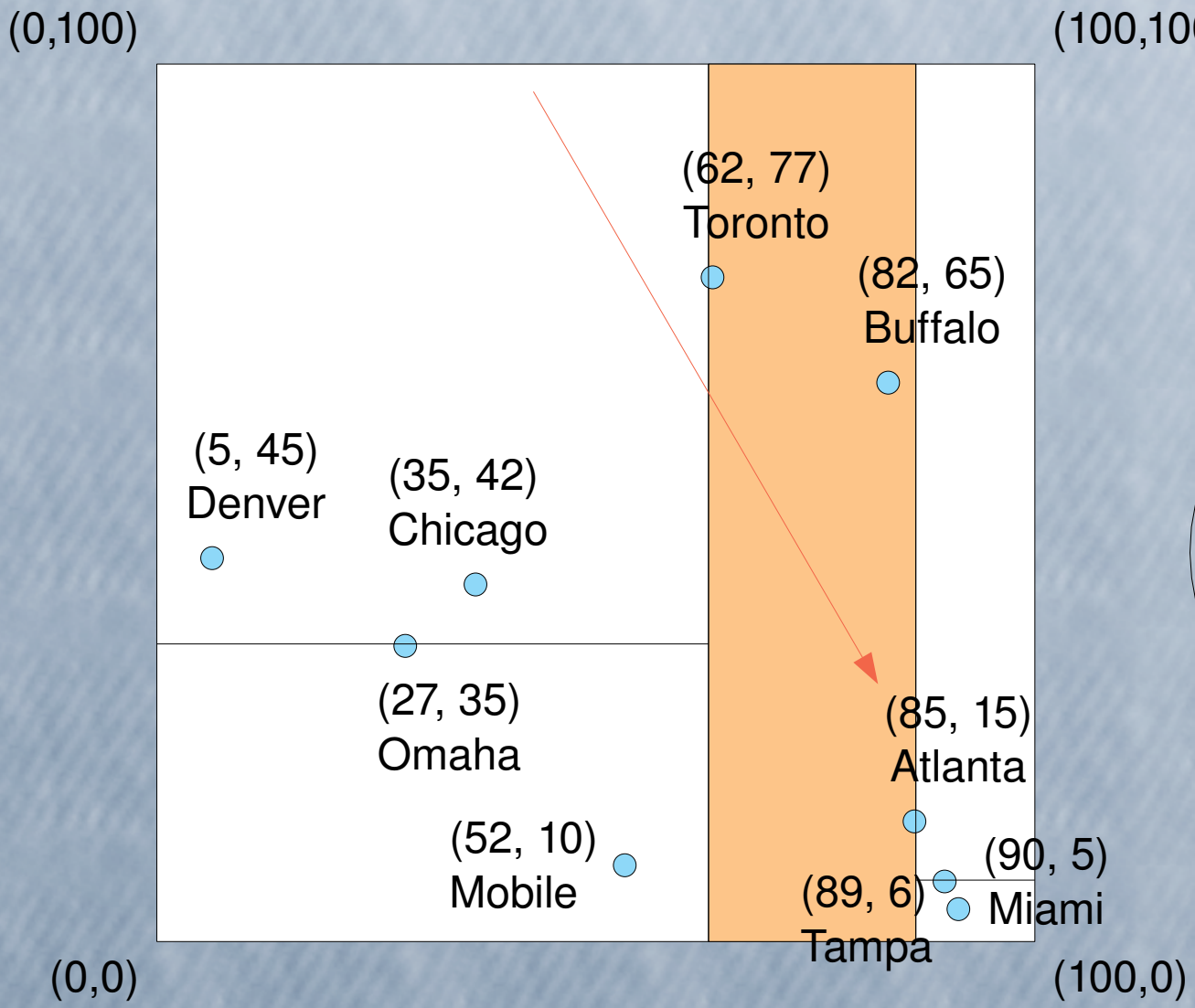
Multidimensional Range Trees



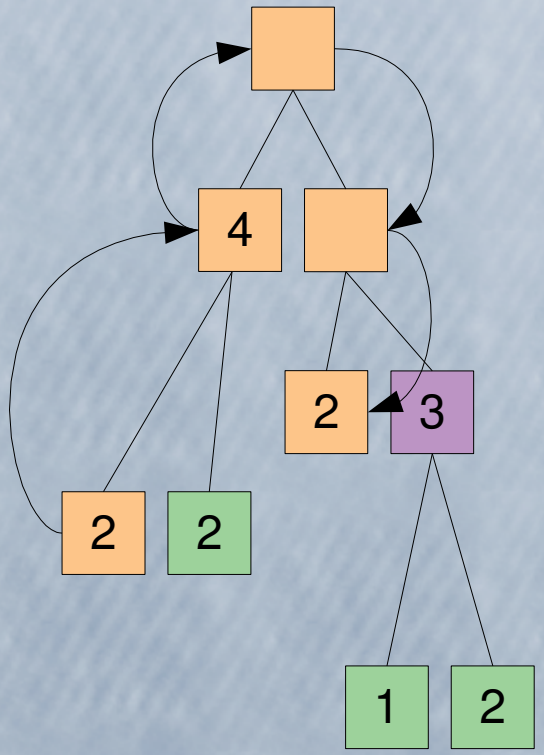
Multidimensional
Tree Data Structure



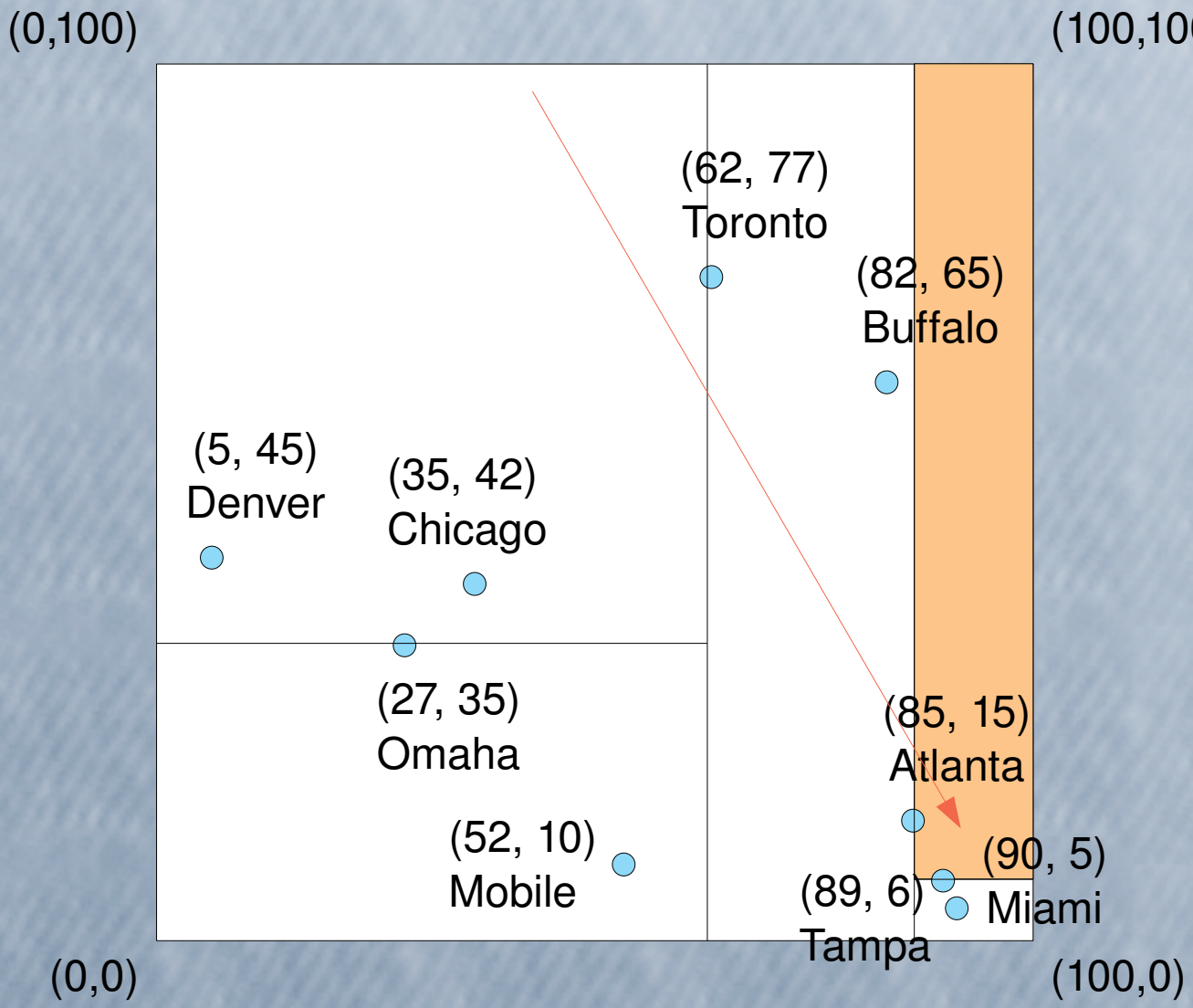
Multidimensional Range Trees



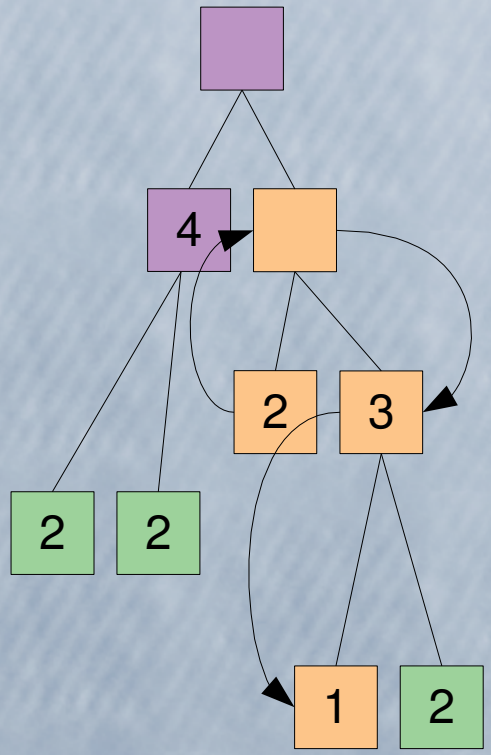
Multidimensional
Tree Data Structure



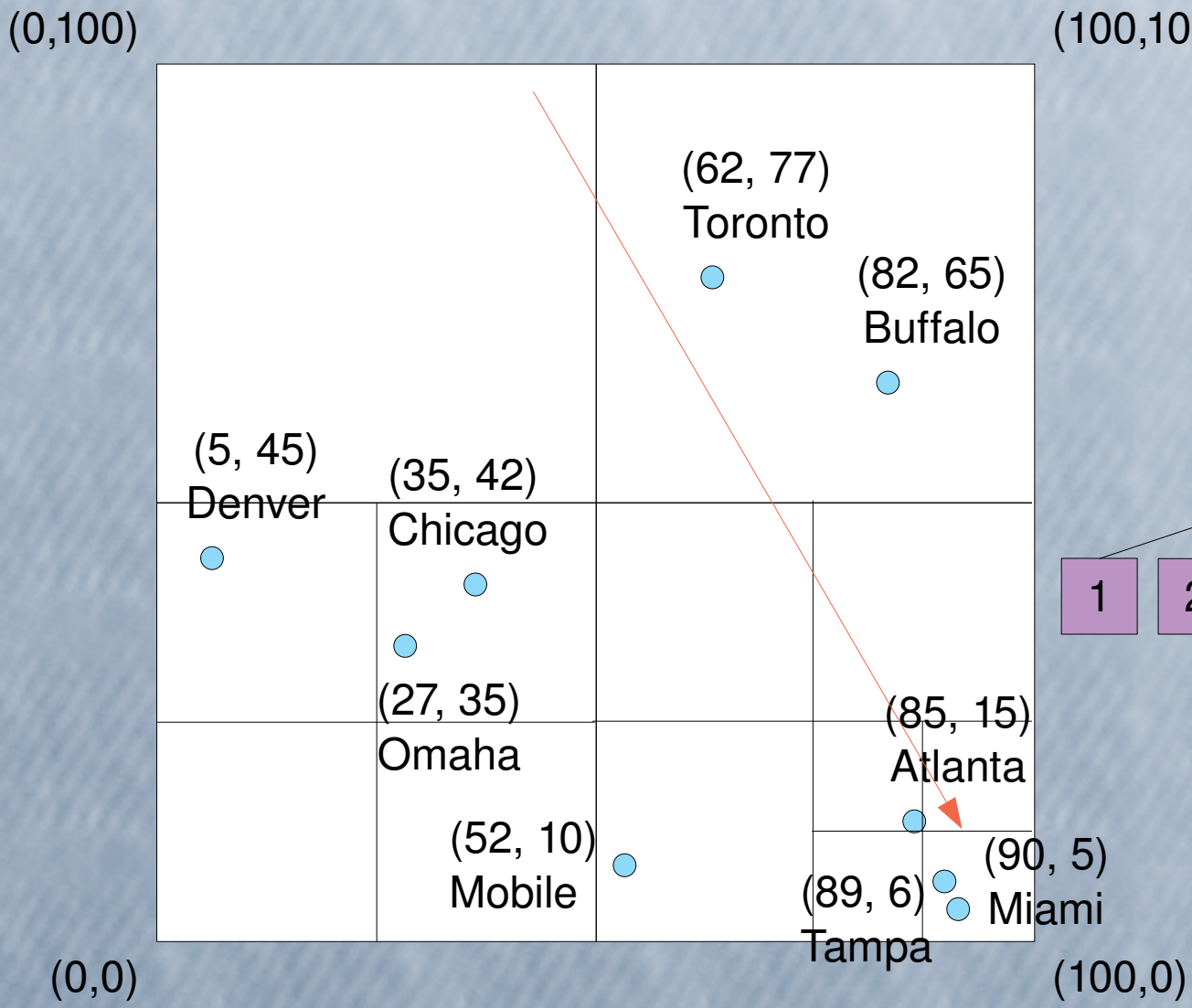
Multidimensional Range Trees

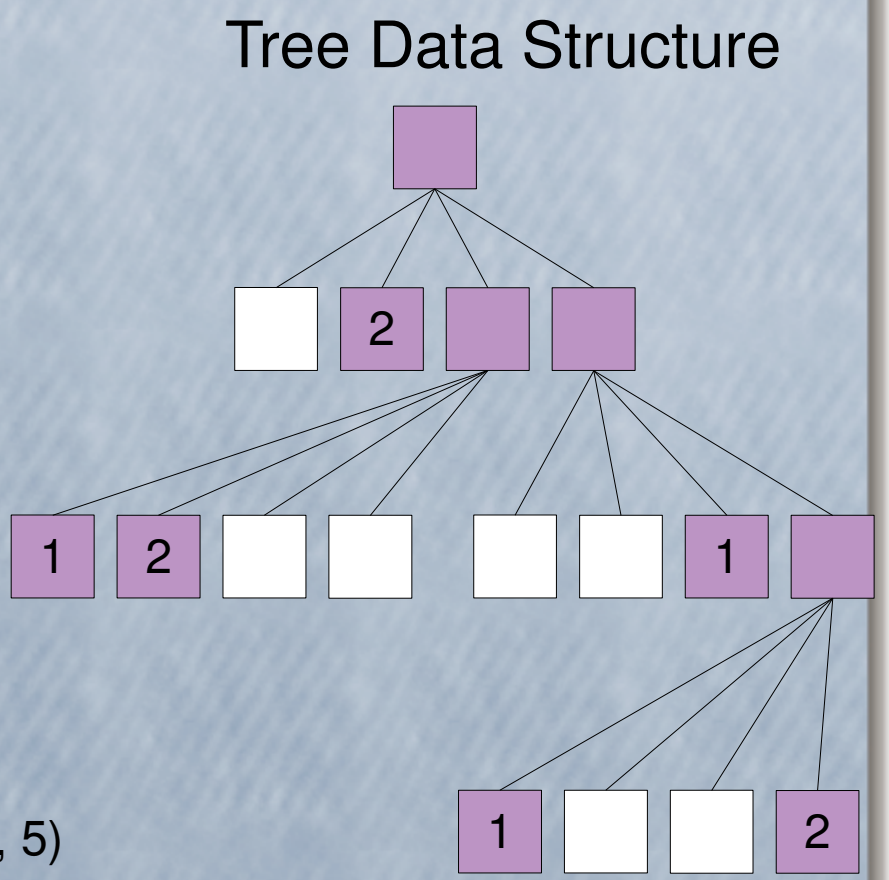


Multidimensional
Tree Data Structure

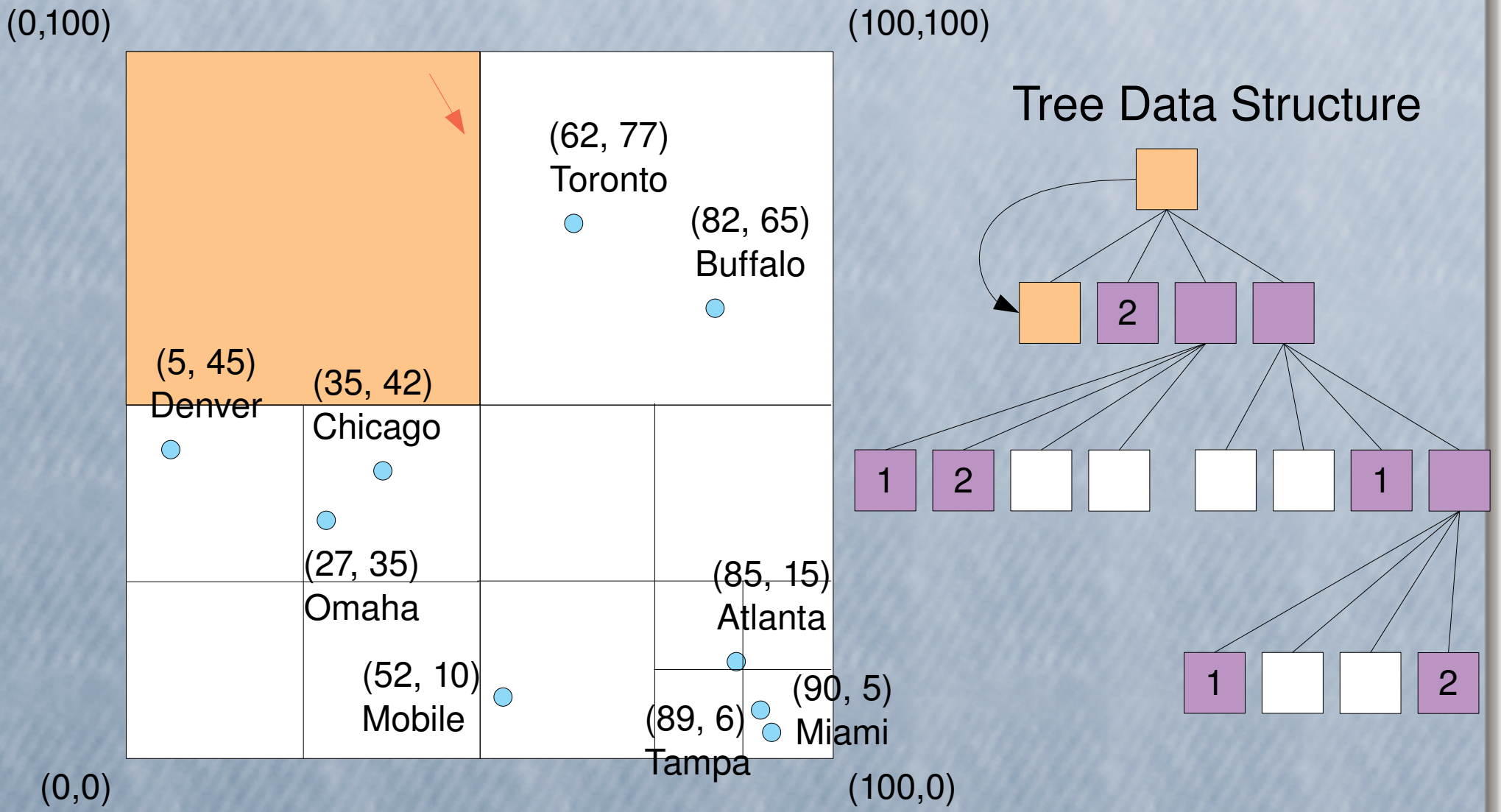


Quadtrees

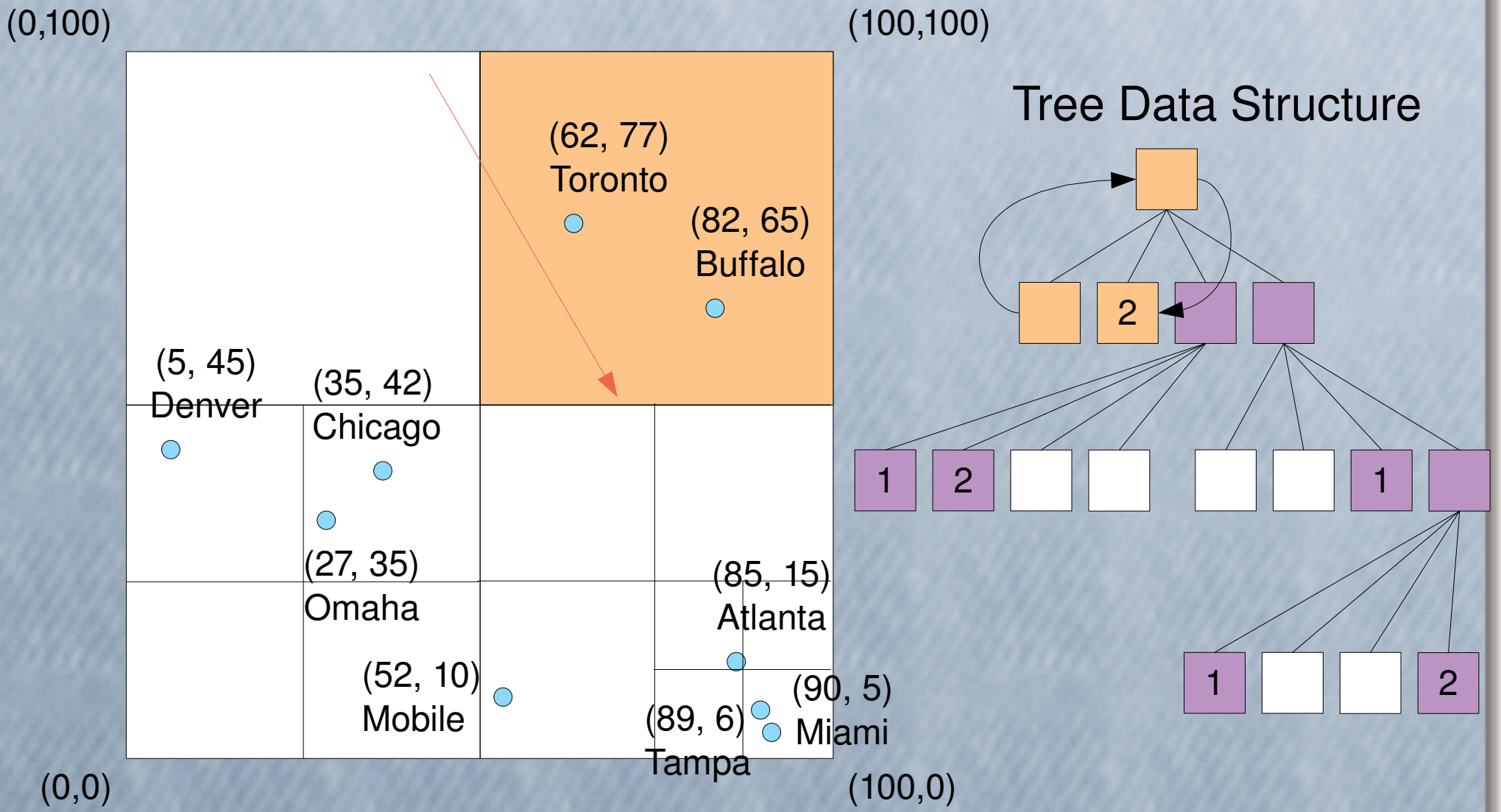

(0,100)
(100,100)
(62, 77)
Toronto
(82, 65)
Buffalo
(5, 45)
Denver
(35, 42)
Chicago
(27, 35)
Omaha
(85, 15)
Atlanta
(52, 10)
Mobile
(90, 5)
Miami
(89, 6)
Tampa
(0,0)
(100,0)


Tree Data Structure
2
1
2
1
1
2

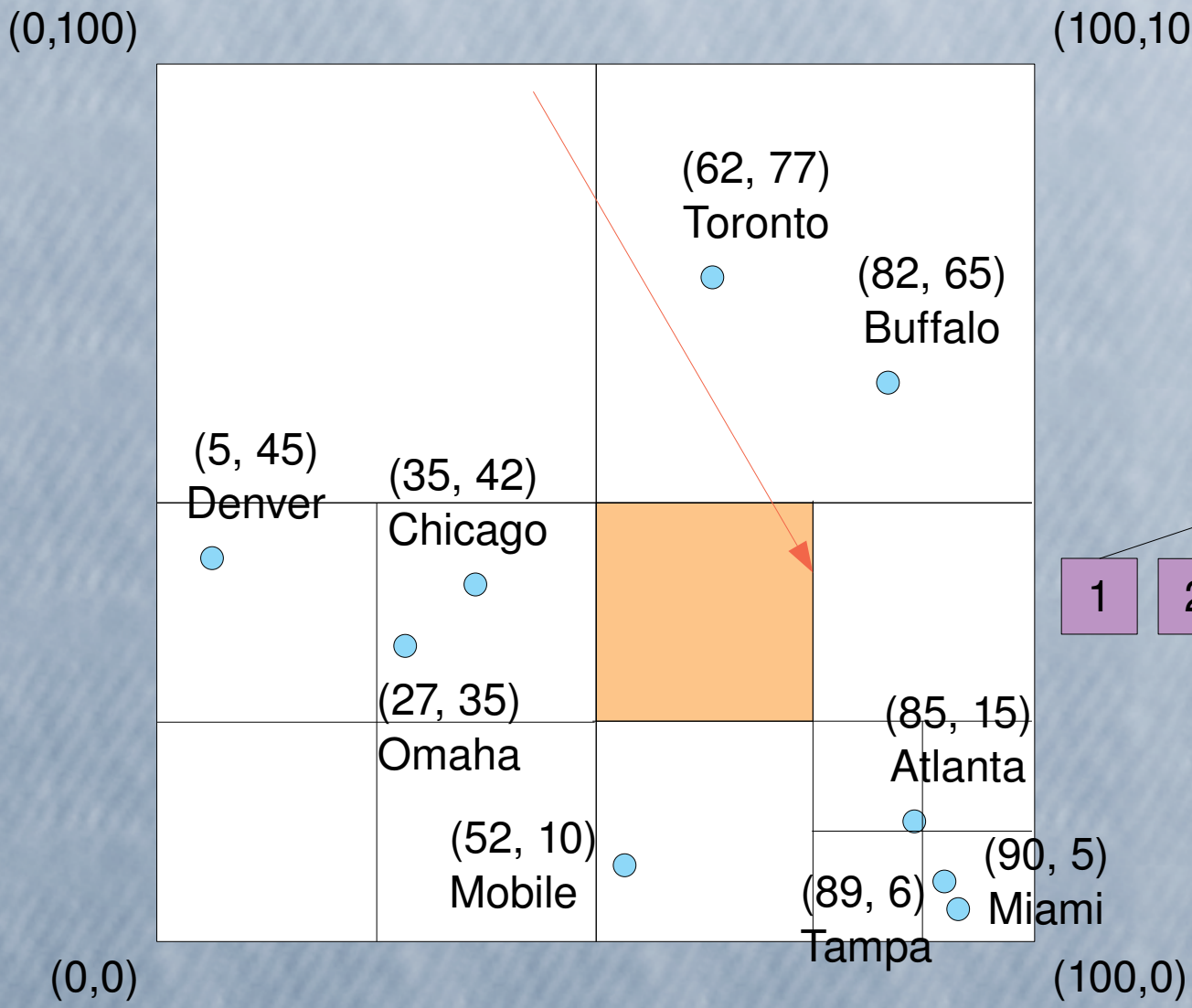
Quadtrees


(0,100)
(100,100)
(62, 77)
Toronto
(82, 65)
Buffalo
(5, 45)
Denver
(35, 42)
Chicago
(27, 35)
Omaha
(52, 10)
Mobile
(85, 15)
Atlanta
(90, 5)
Miami
(89, 6)
Tampa
(0,0)
(100,0)
Tree Data Structure
2
1
2
1
1
2

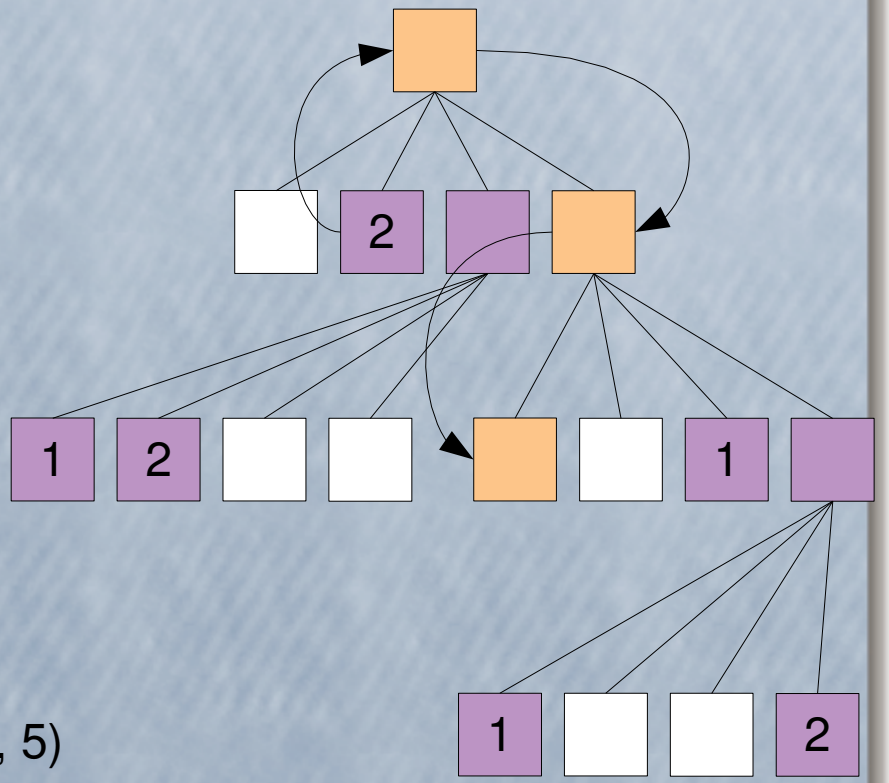
Quadtrees


(0,100)
(100,100)
(62, 77)
Toronto
(82, 65)
Buffalo
(5, 45)
Denver
(35, 42)
Chicago
(27, 35)
Omaha
(52, 10)
Mobile
(85, 15)
Atlanta
(89, 6)
Tampa
(90, 5)
Miami
(0,0)
(100,0)
Tree Data Structure
2
1
2
1
1
2

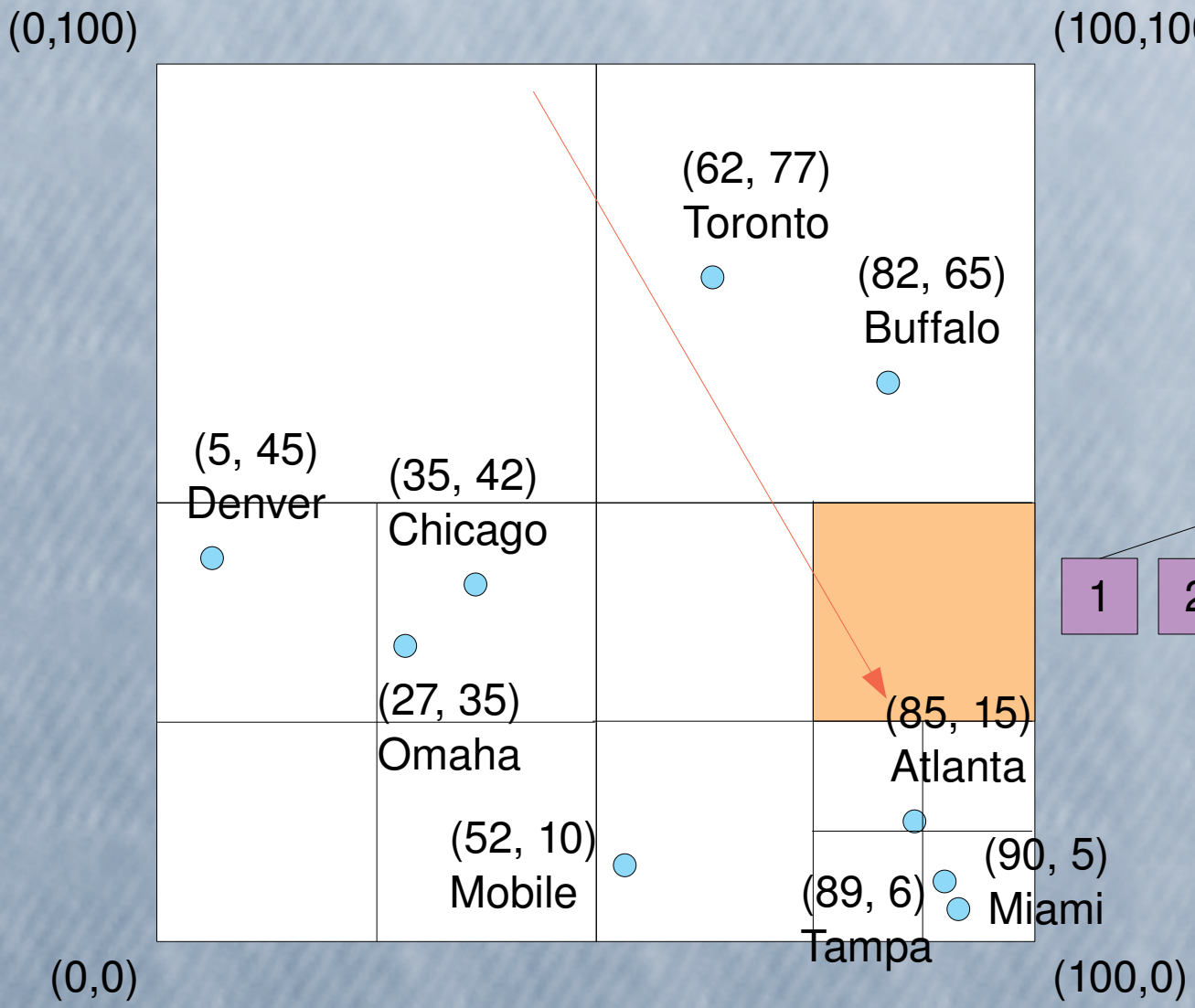
Quadtrees

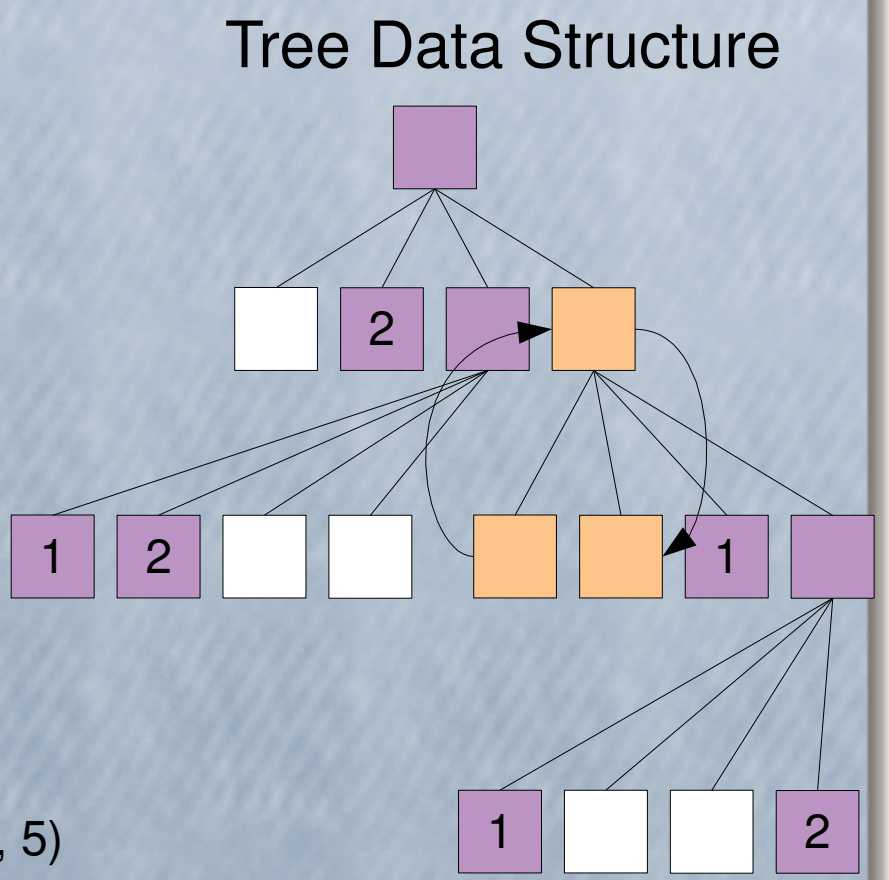


Tree Data Structure

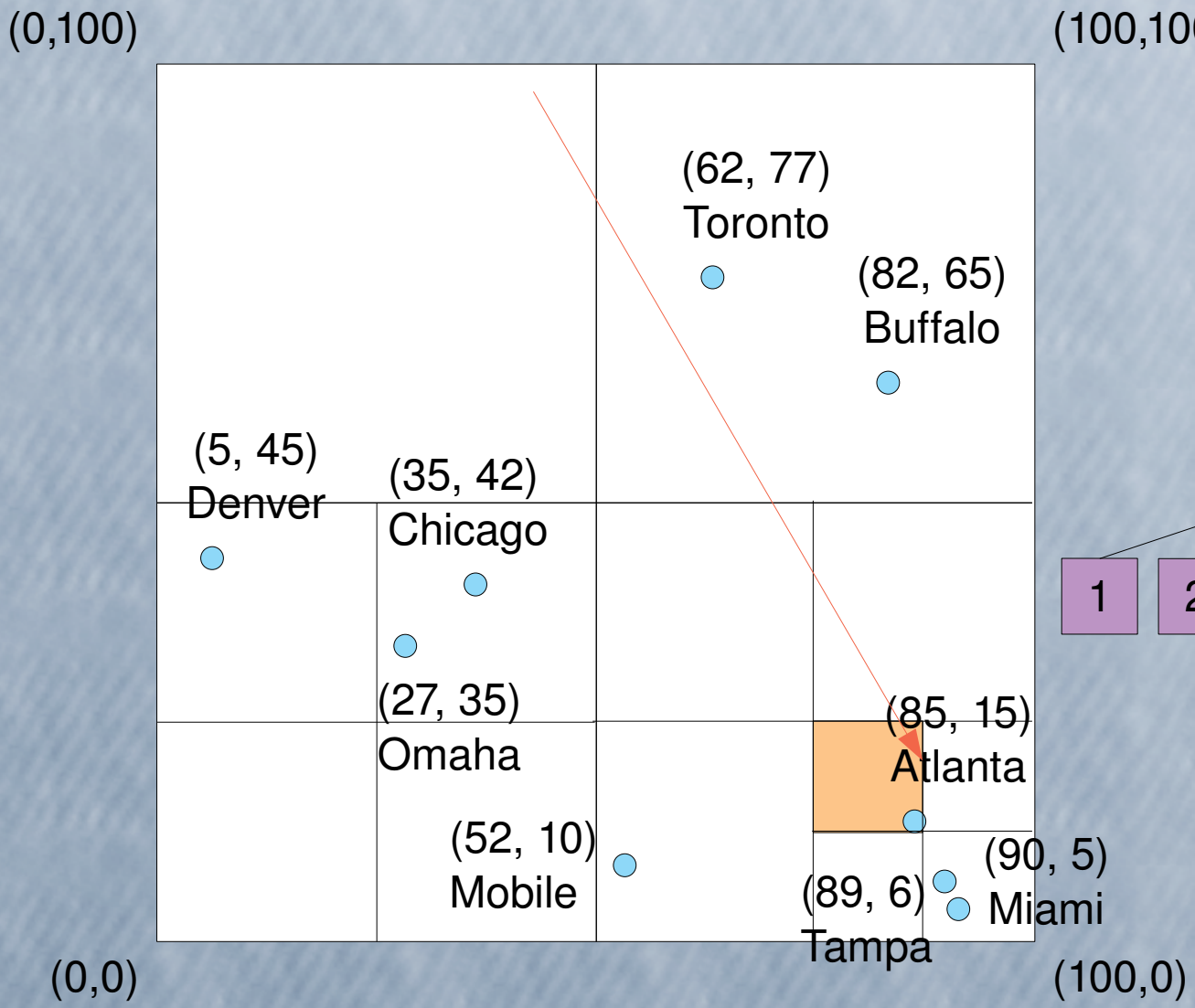


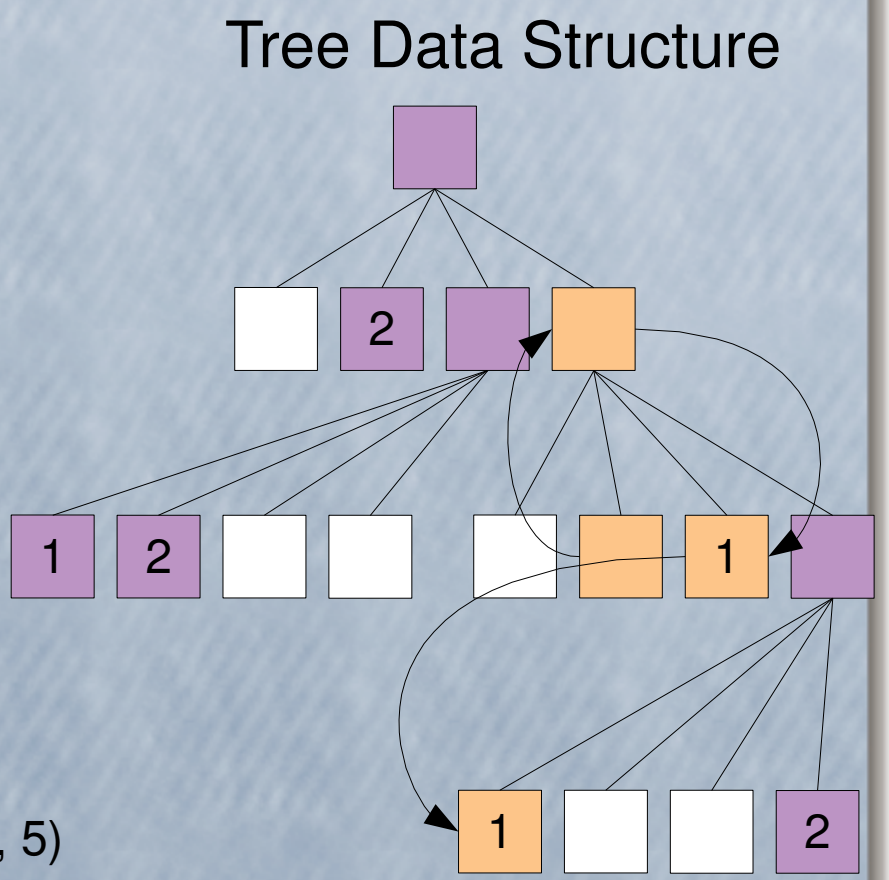
Quadtrees


(0,100)
(100,100)
(62, 77)
Toronto
(82, 65)
Buffalo
(5, 45)
Denver
(35, 42)
Chicago
(27, 35)
Omaha
(85, 15)
Atlanta
(52, 10)
Mobile
(90, 5)
Miami
(89, 6)
Tampa
(0,0)
(100,0)

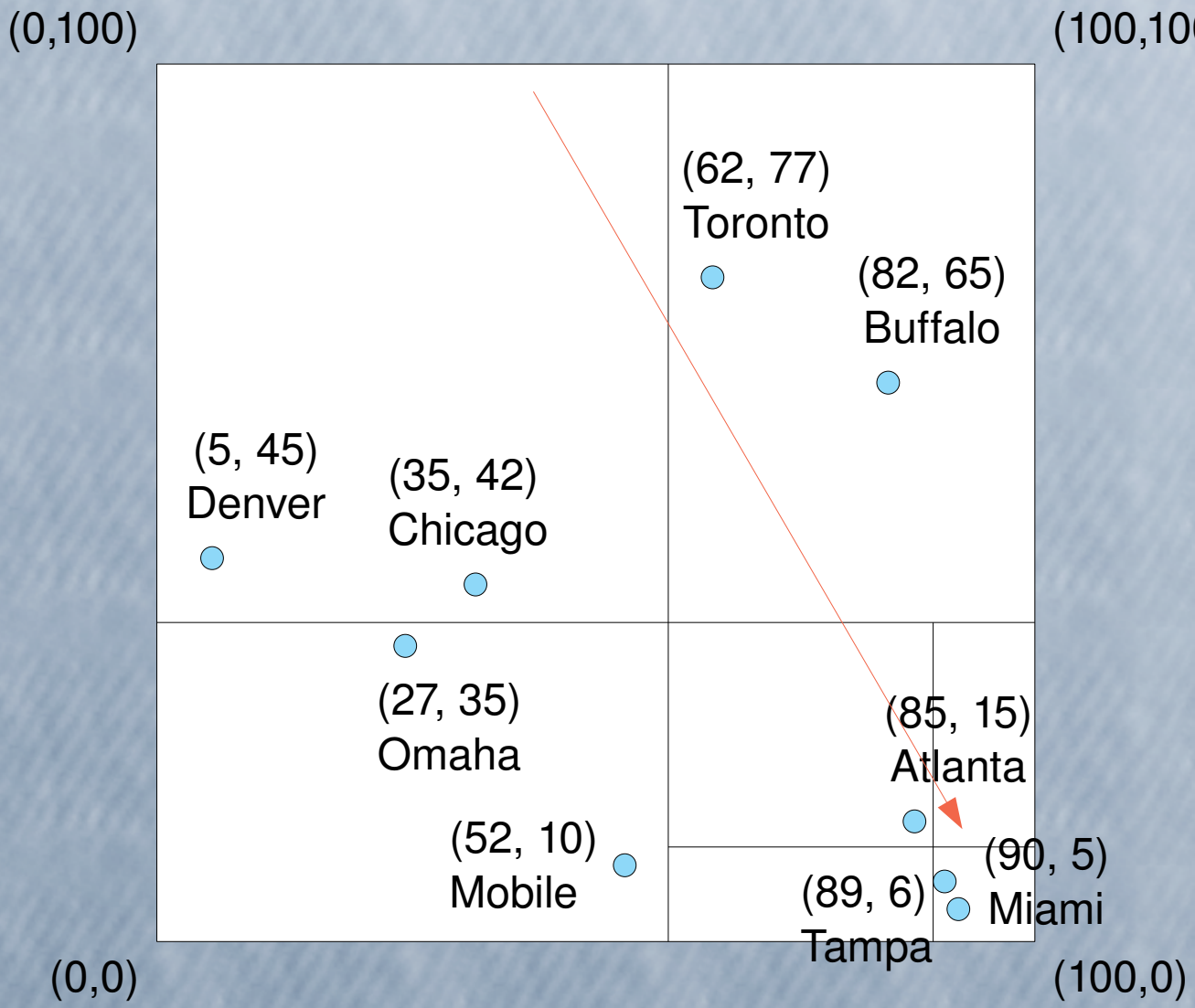

Tree Data Structure
2
1
2
1
1
2

Quadtrees

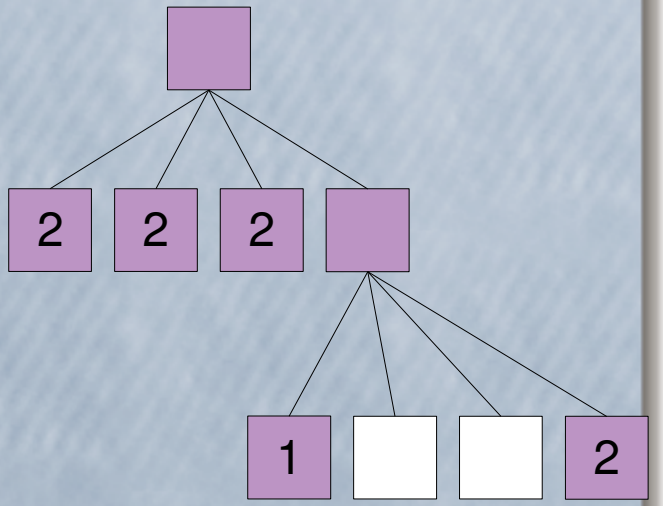

(0,100)
(100,100)
(62, 77)
Toronto
(82, 65)
Buffalo
(5, 45)
Denver
(35, 42)
Chicago
(27, 35)
Omaha
(85, 15)
Atlanta
(52, 10)
Mobile
(90, 5)
Miami
(89, 6)
Tampa
(0,0)
(100,0)


Tree Data Structure
2
1
2
1
1
2

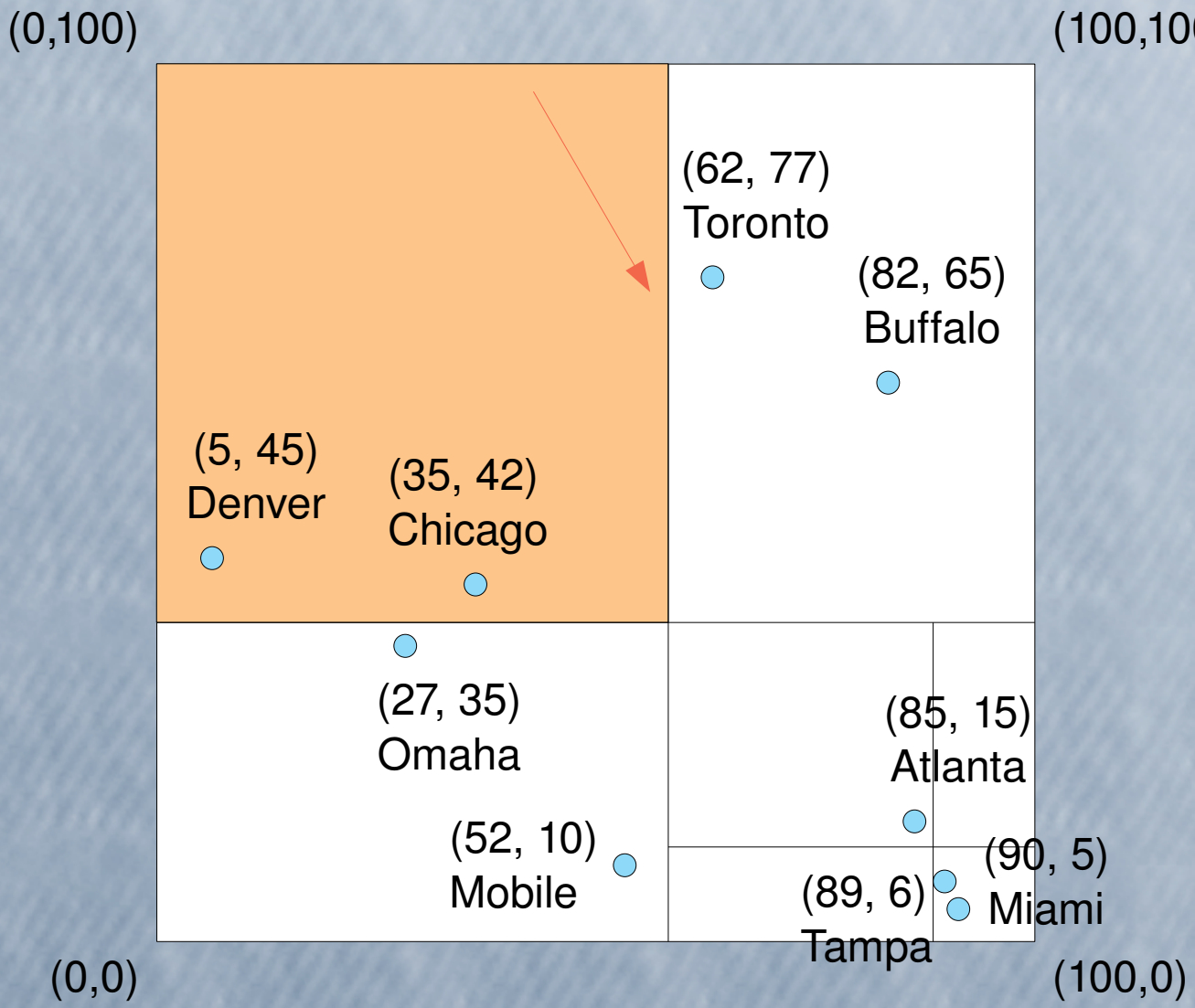
Quadtrees

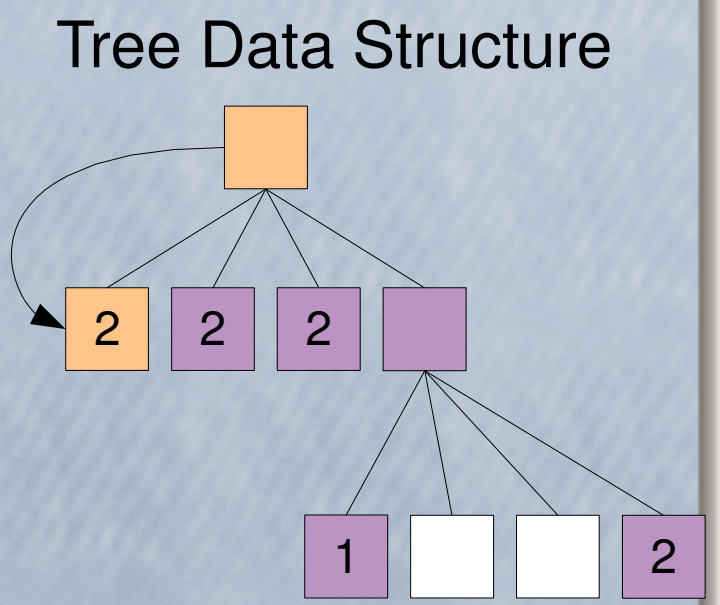


Tree Data Structure

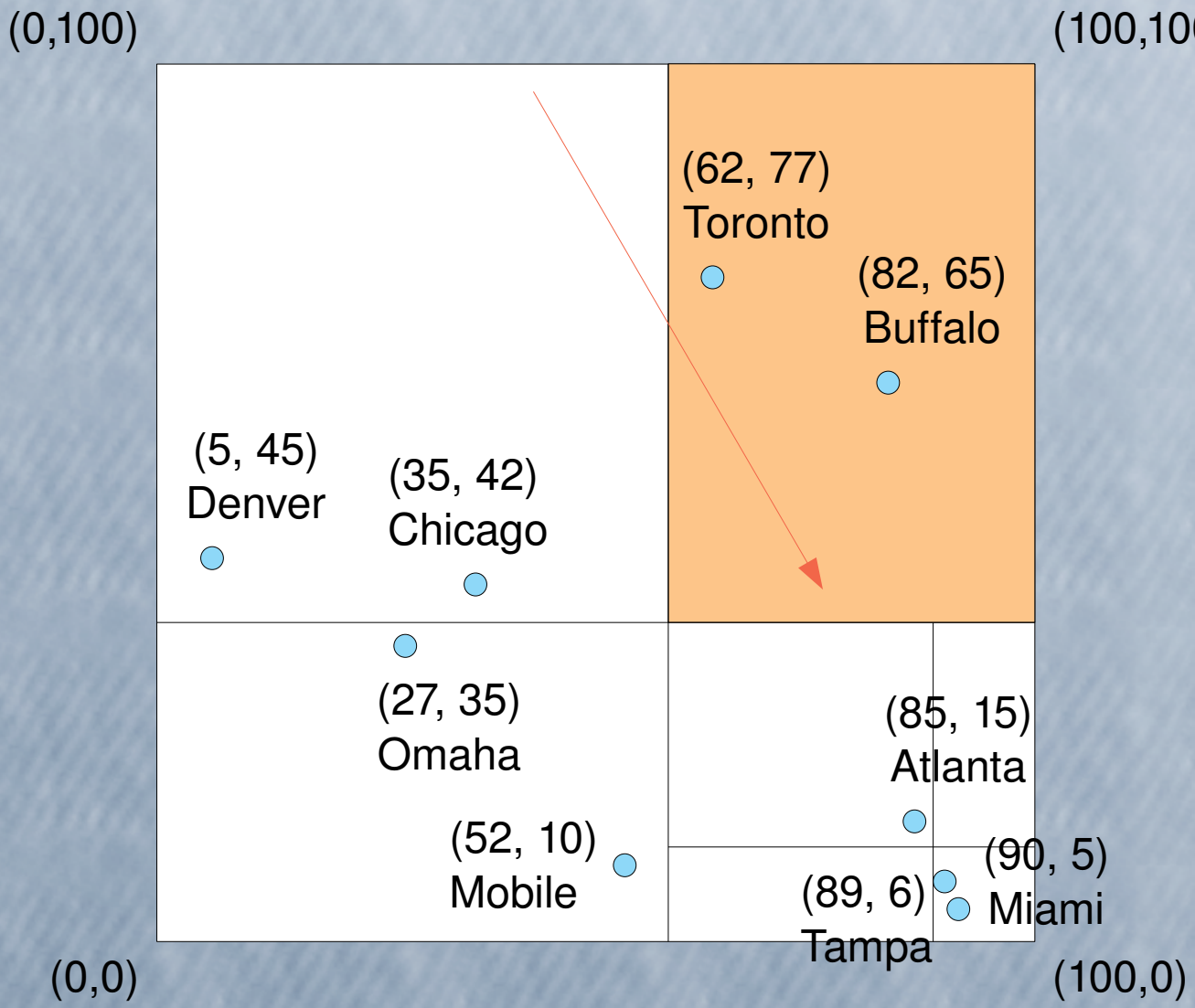


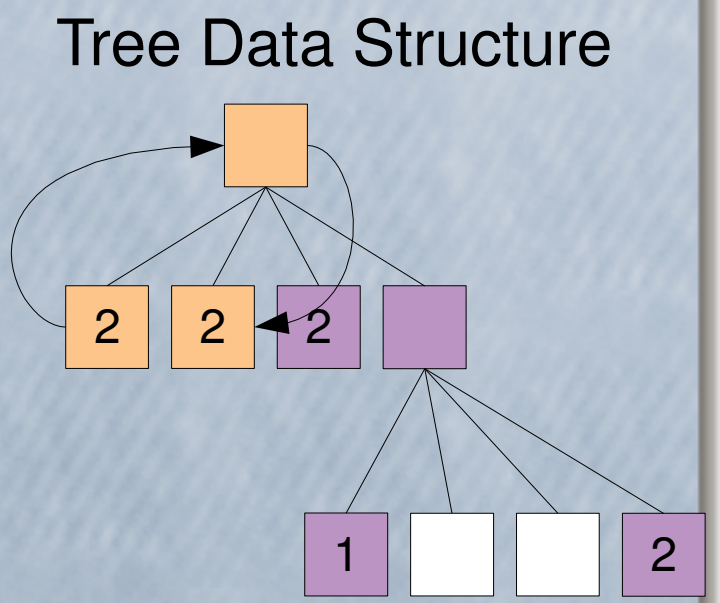
Quadtrees


(0,100)
(100,100)
(62, 77)
Toronto
(82, 65)
Buffalo
(5, 45)
Denver
(35, 42)
Chicago
(27, 35)
Omaha
(85, 15)
Atlanta
(52, 10)
Mobile
(90, 5)
Miami
(89, 6)
Tampa
(0,0)
(100,0)

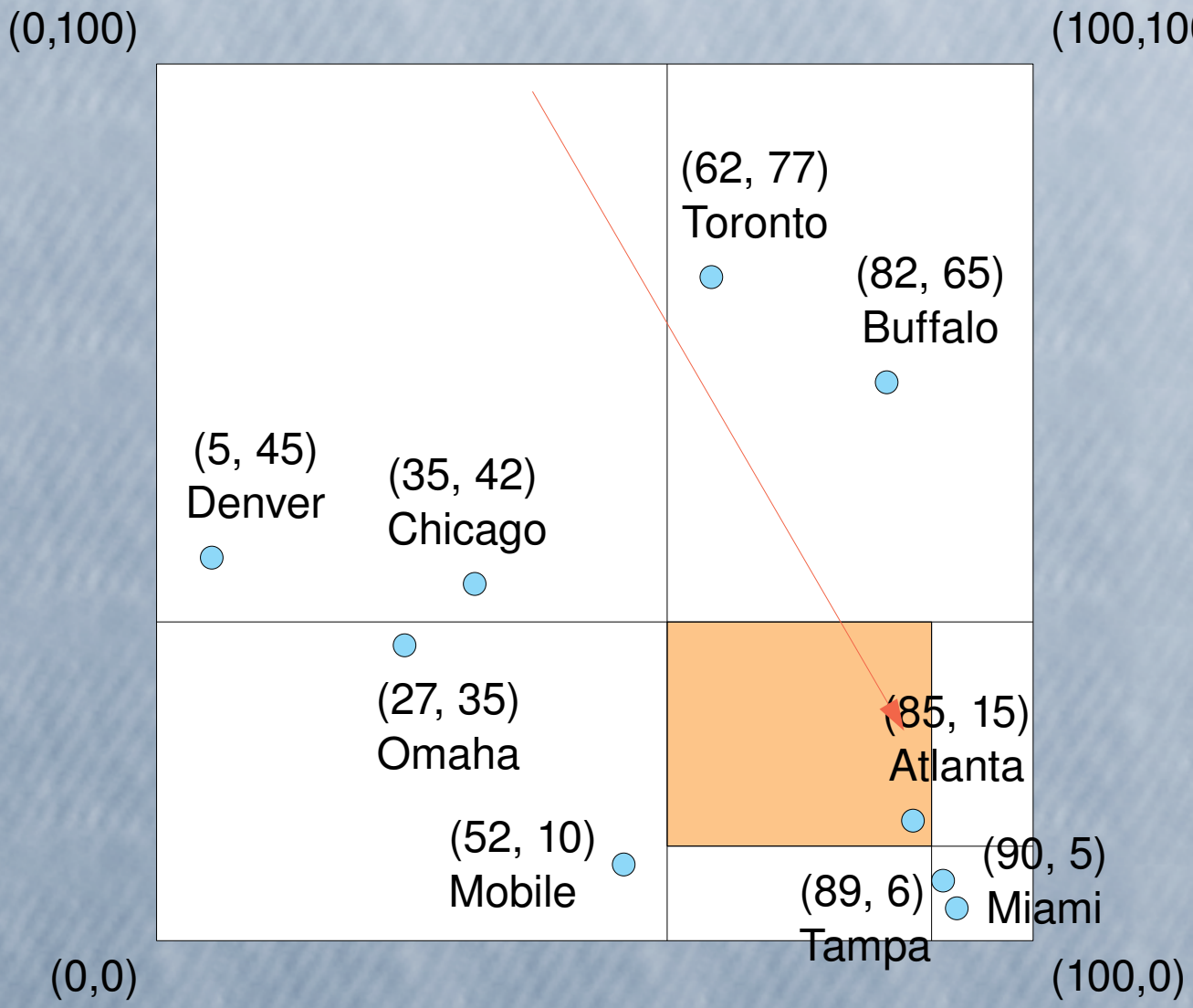

Tree Data Structure
2
2
2
1
2

Quadtrees

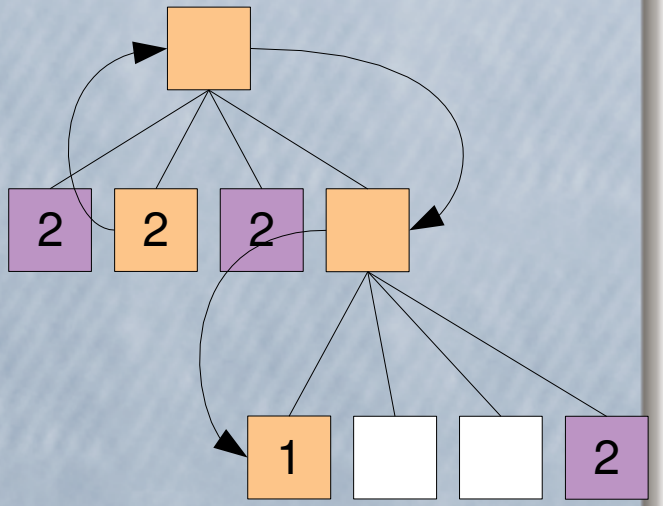

(0,100)
(100,100)
(62, 77)
Toronto
(82, 65)
Buffalo
(5, 45)
Denver
(35, 42)
Chicago
(27, 35)
Omaha
(85, 15)
Atlanta
(52, 10)
Mobile
(90, 5)
Miami
(89, 6)
Tampa
(0,0)
(100,0)


Tree Data Structure
2
2
2
1
2

Quadtrees

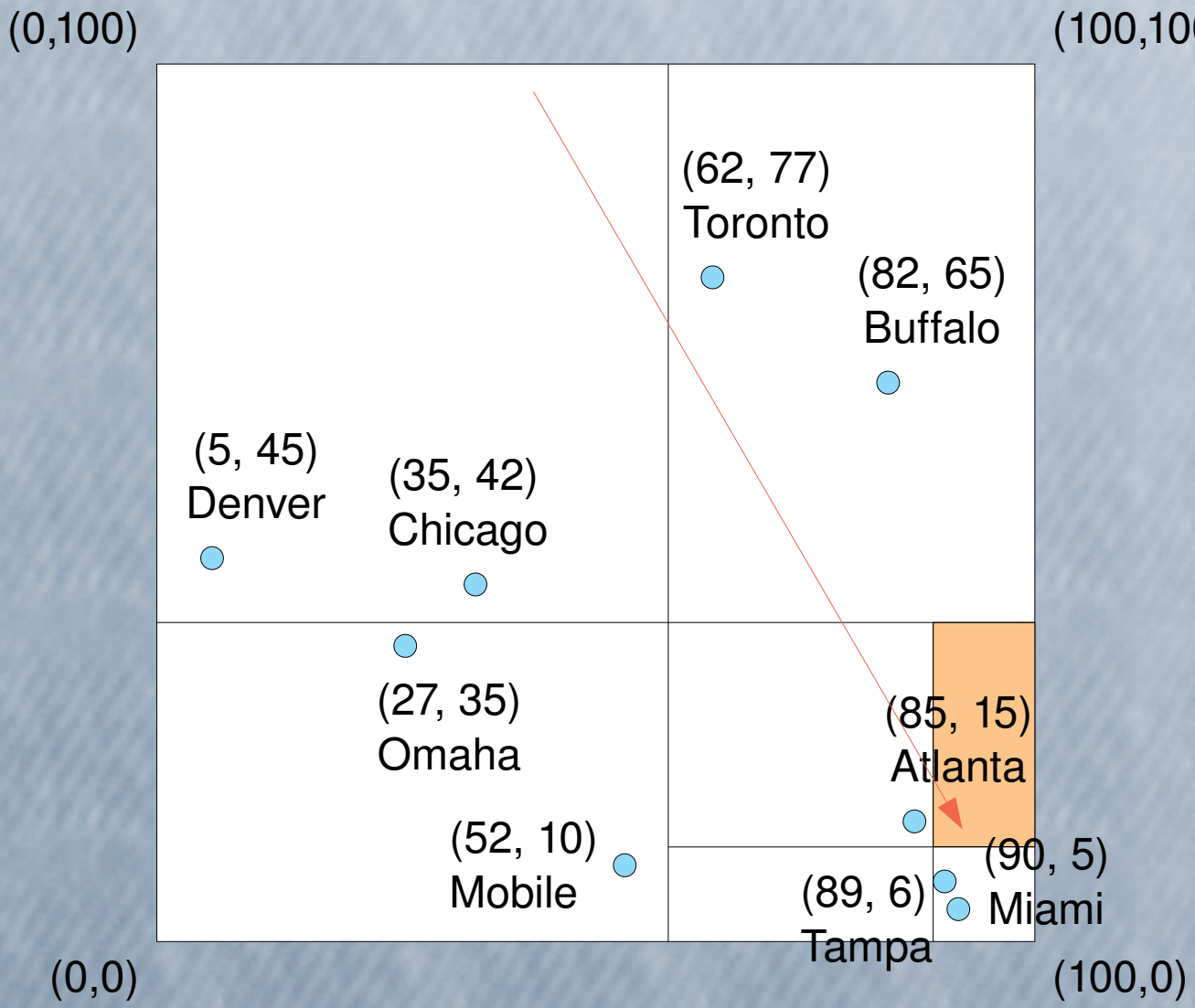


Tree Data Structure

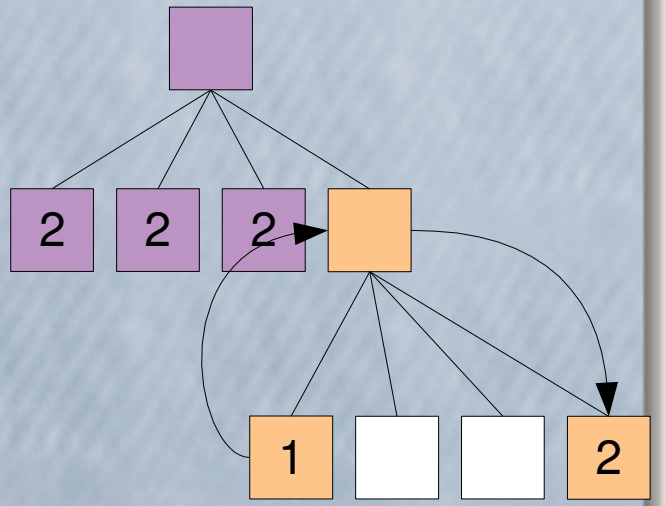


Worst case:
Leaf -> Root -> Leaf

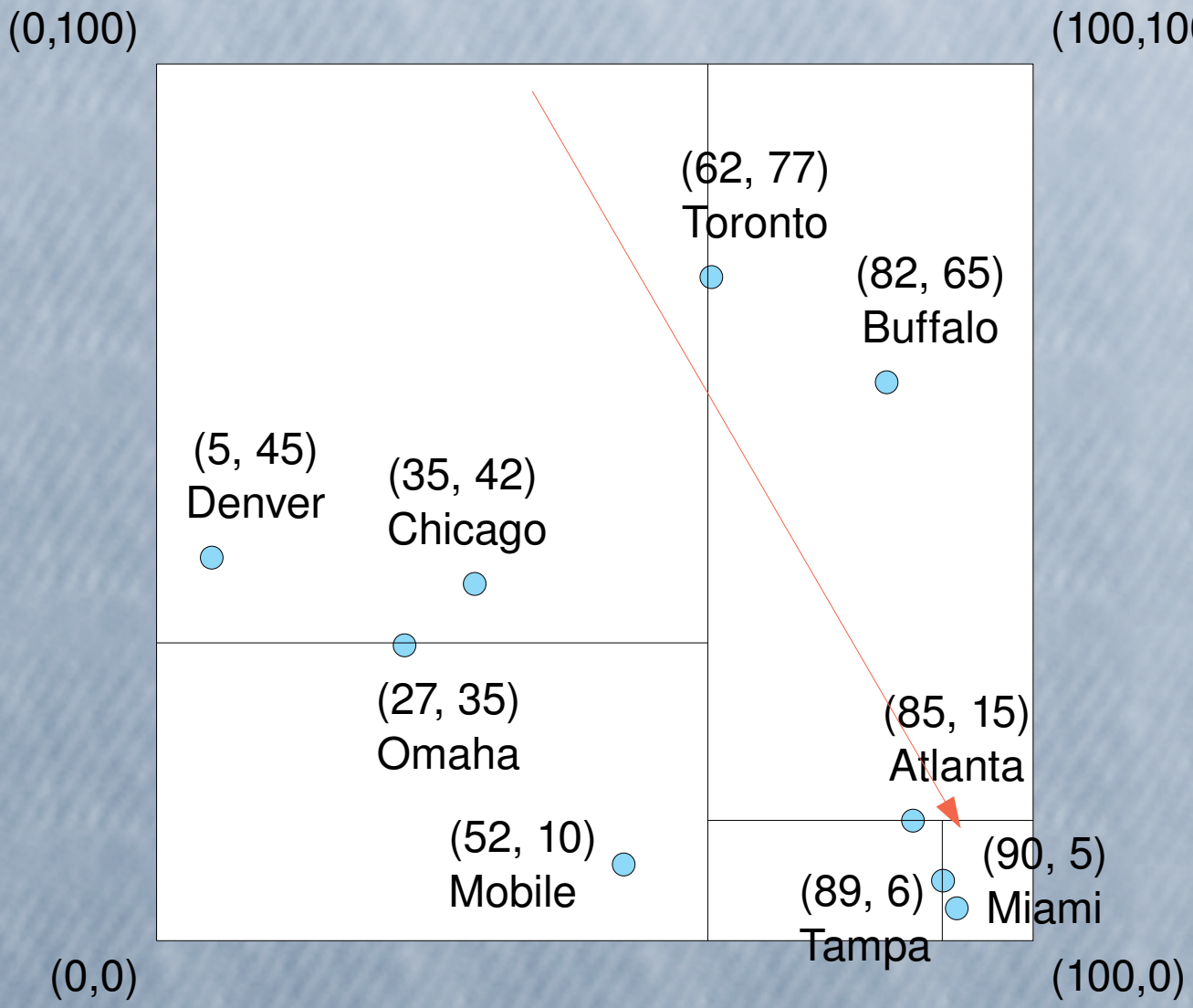
Quadtrees



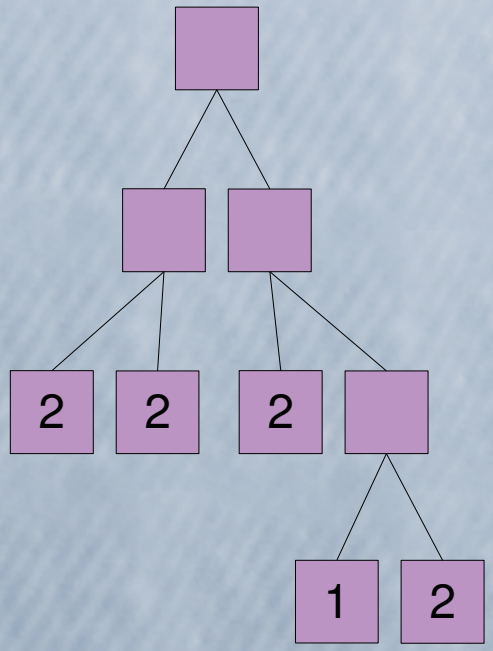
Tree Data Structure



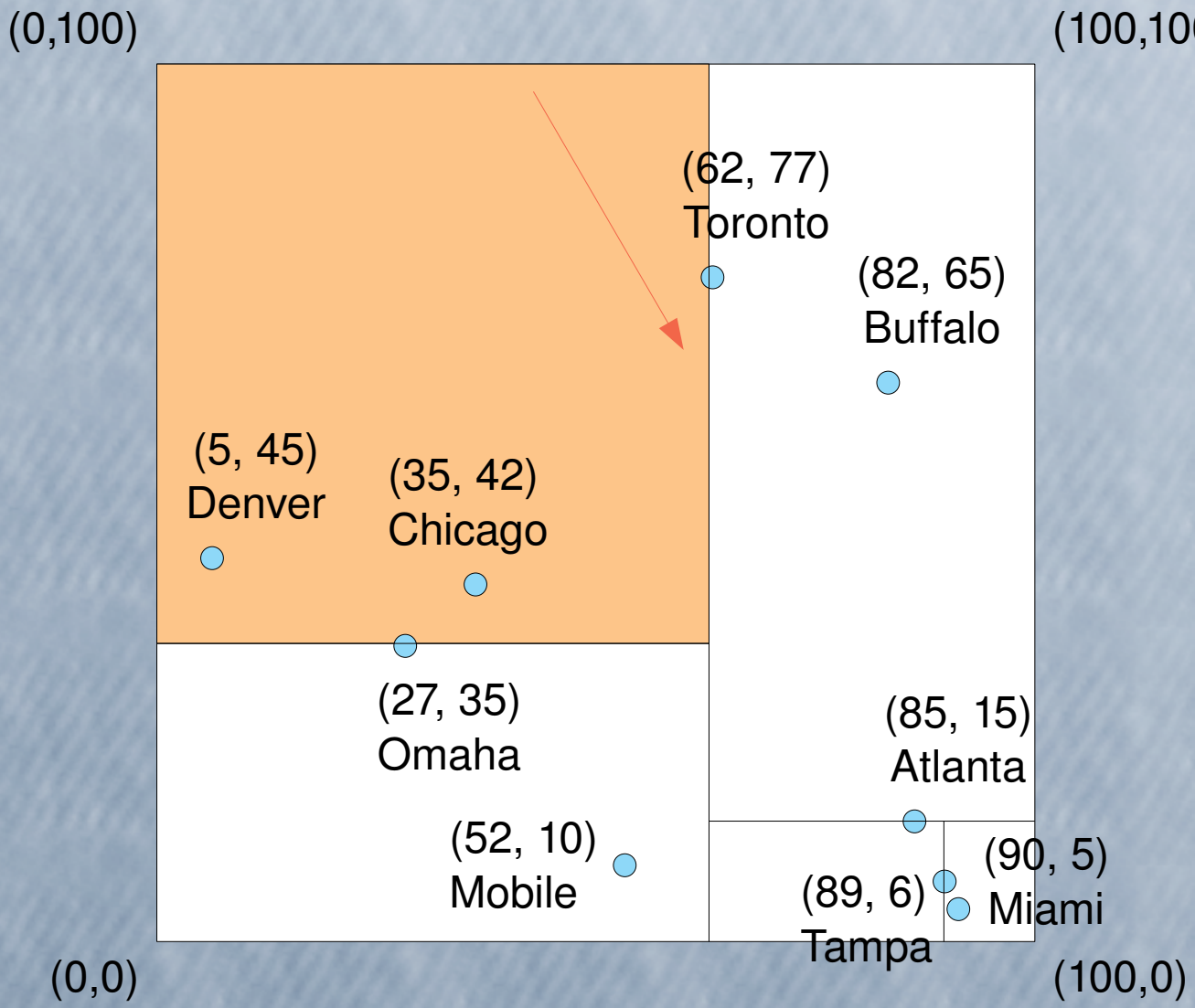
kd-Trees



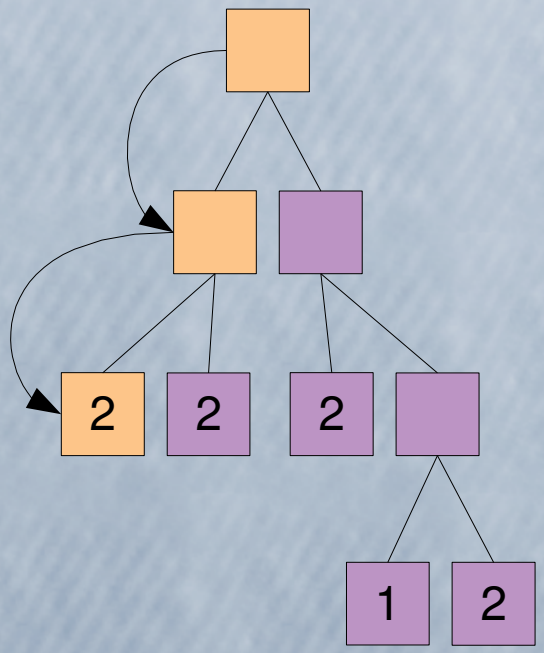
Tree Data Structure



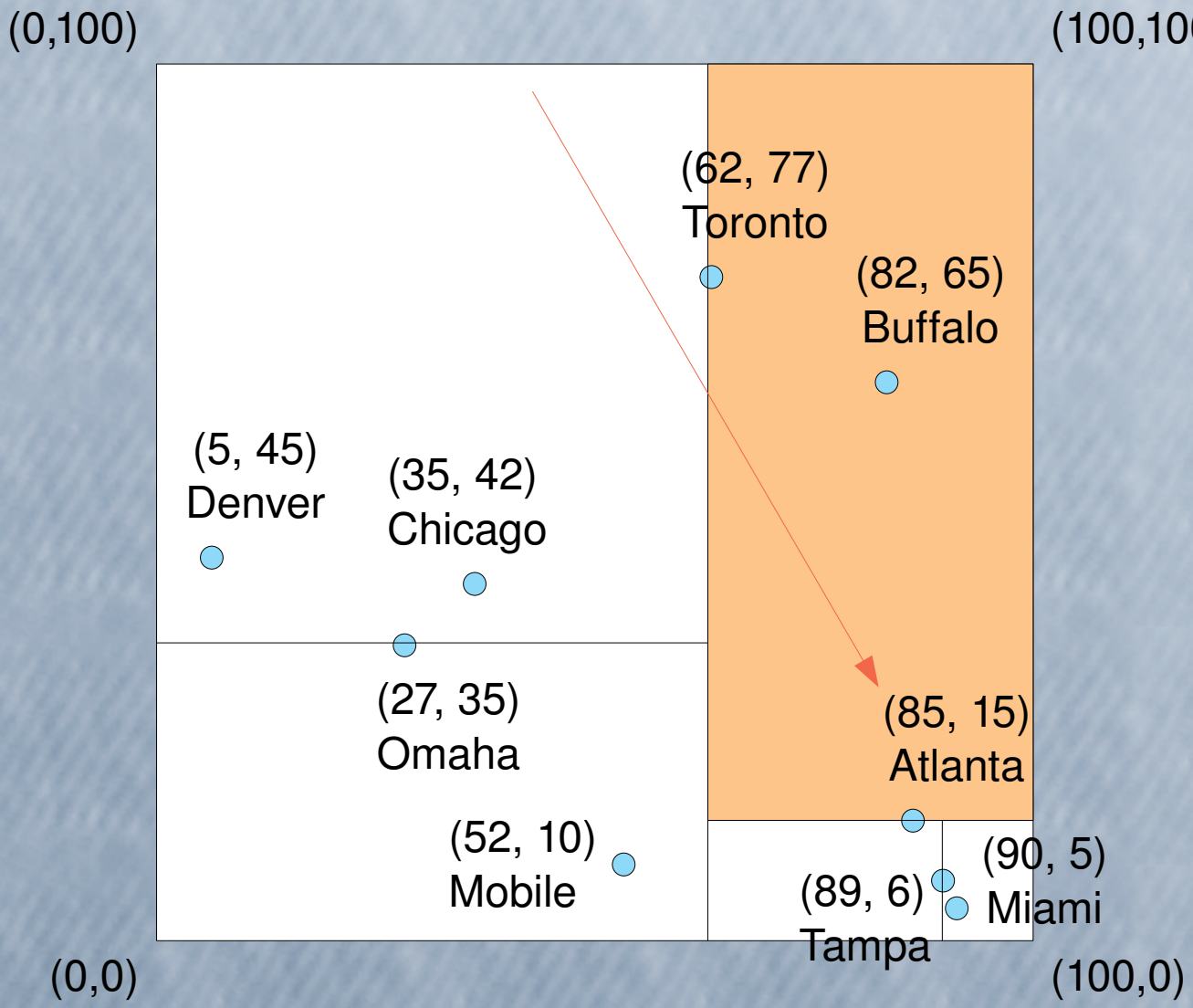
kd-Trees



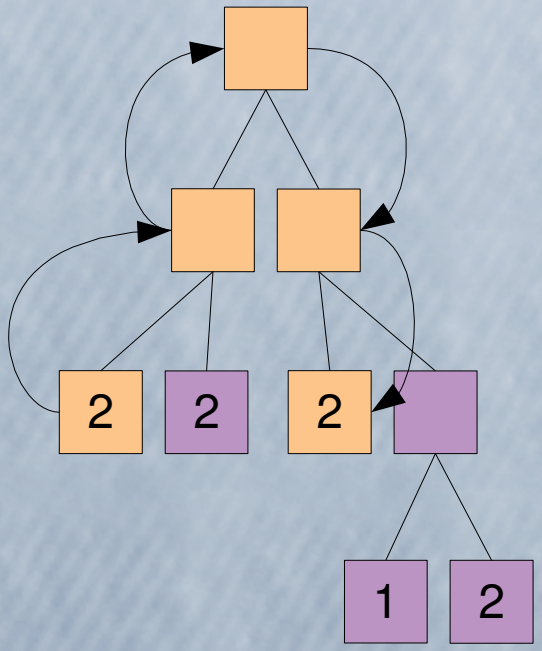
Tree Data Structure



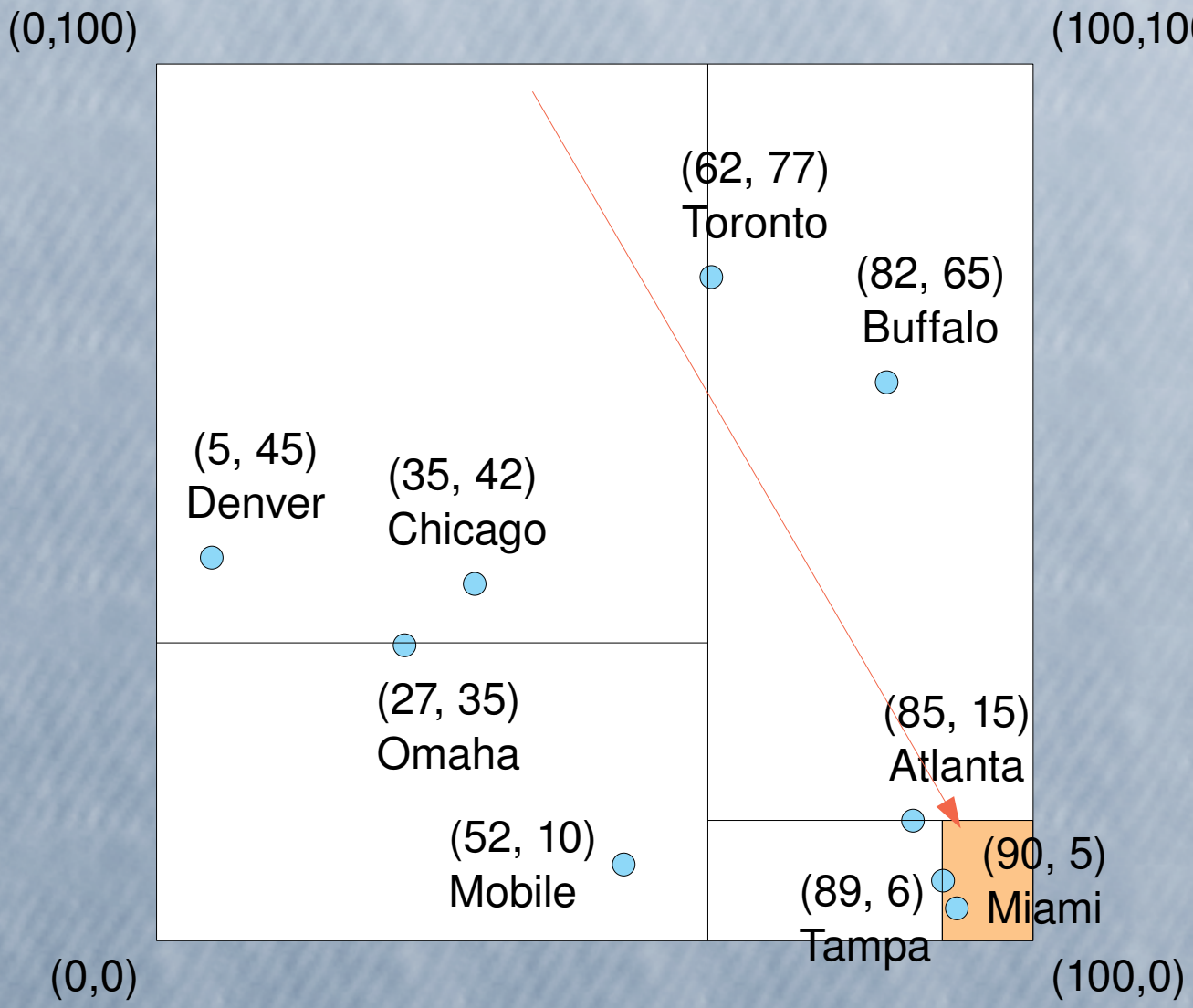
kd-Trees



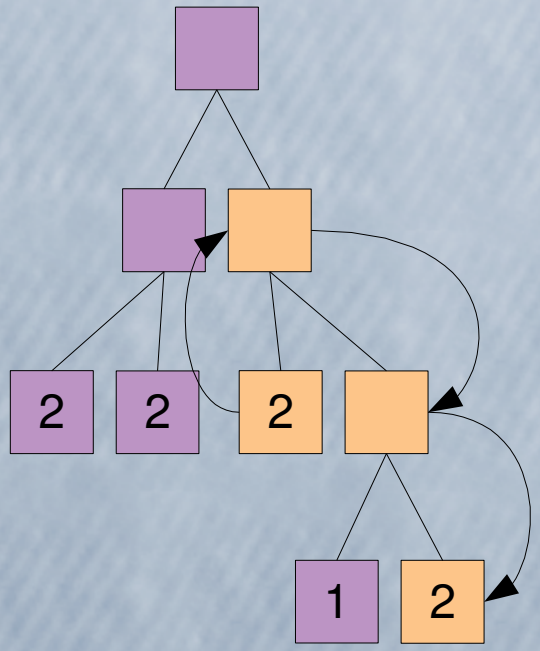
Tree Data Structure



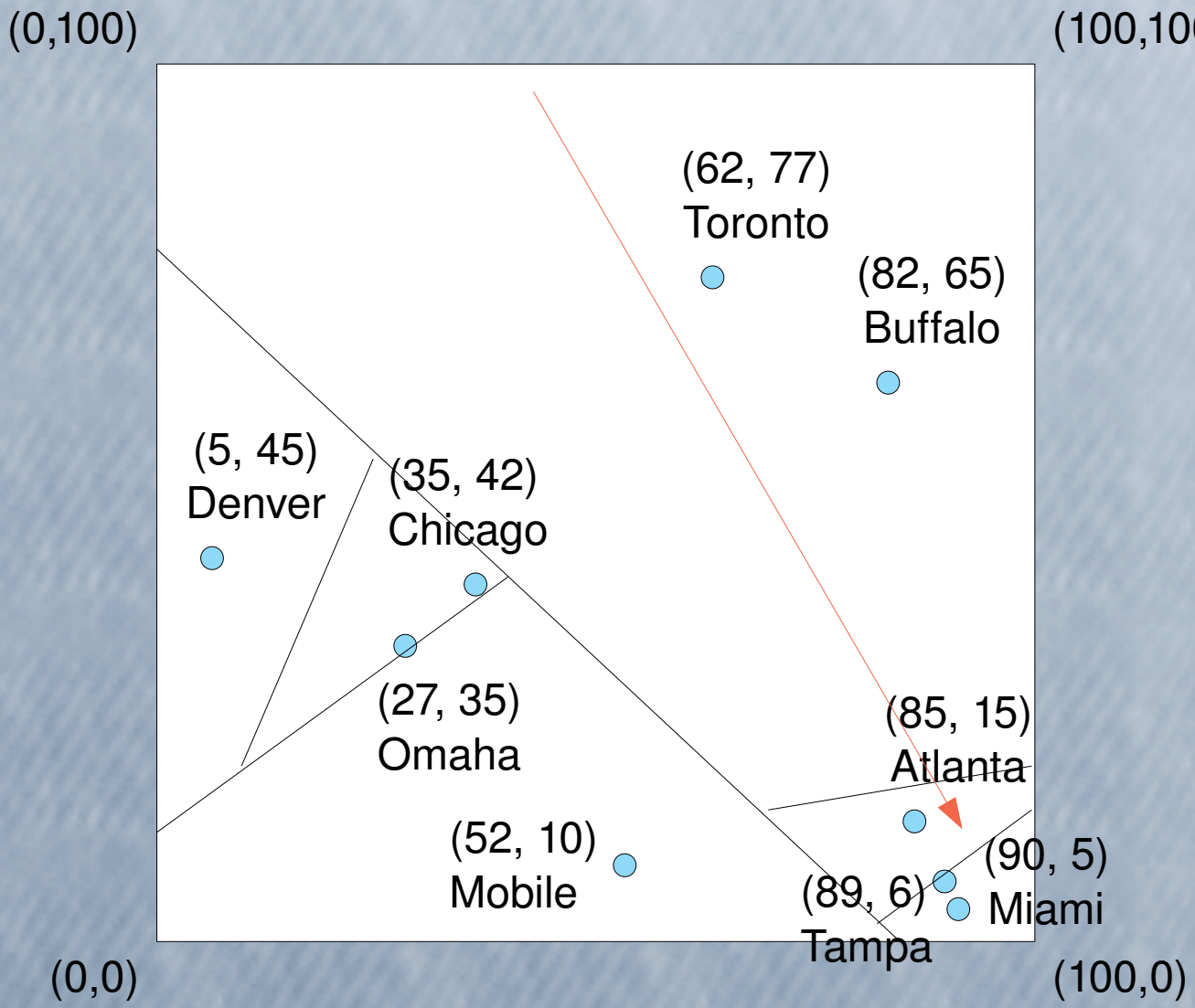
kd-Trees



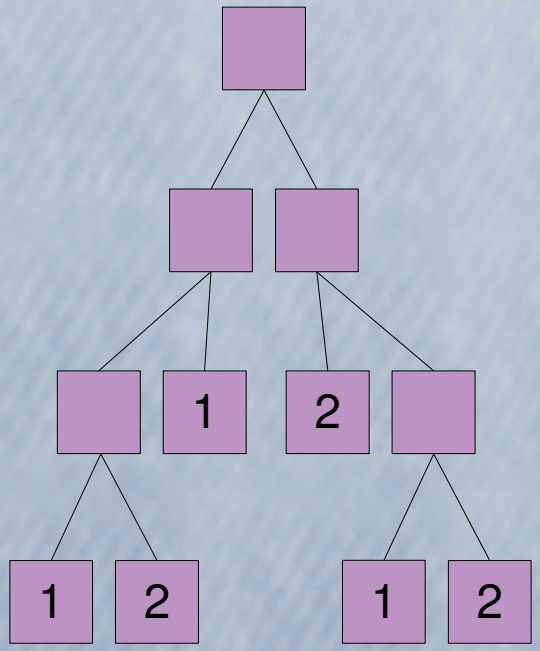
Tree Data Structure



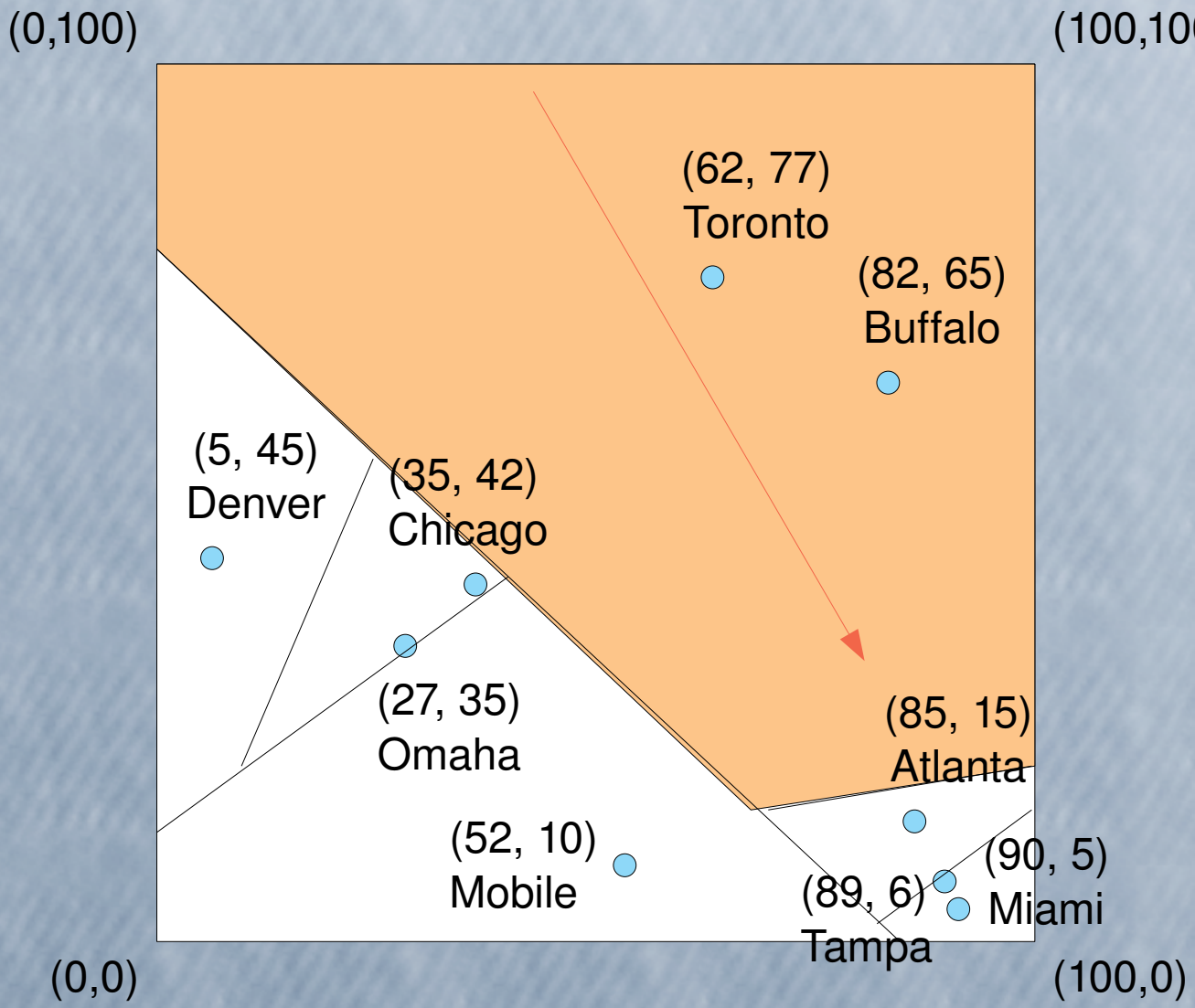
BSP Trees



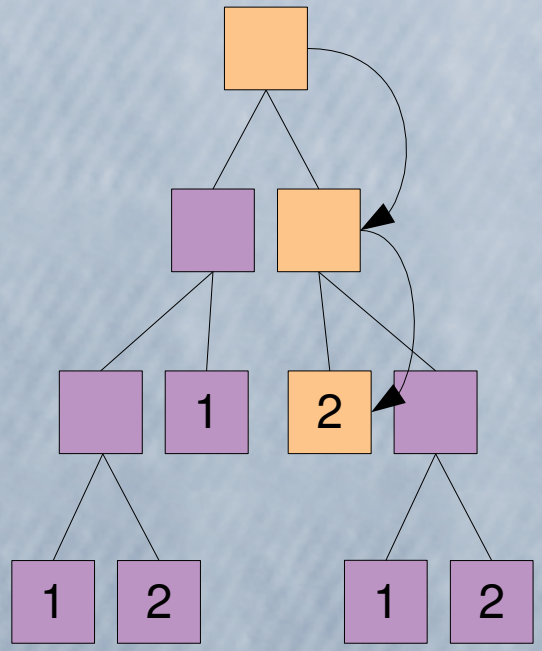
Tree Data Structure



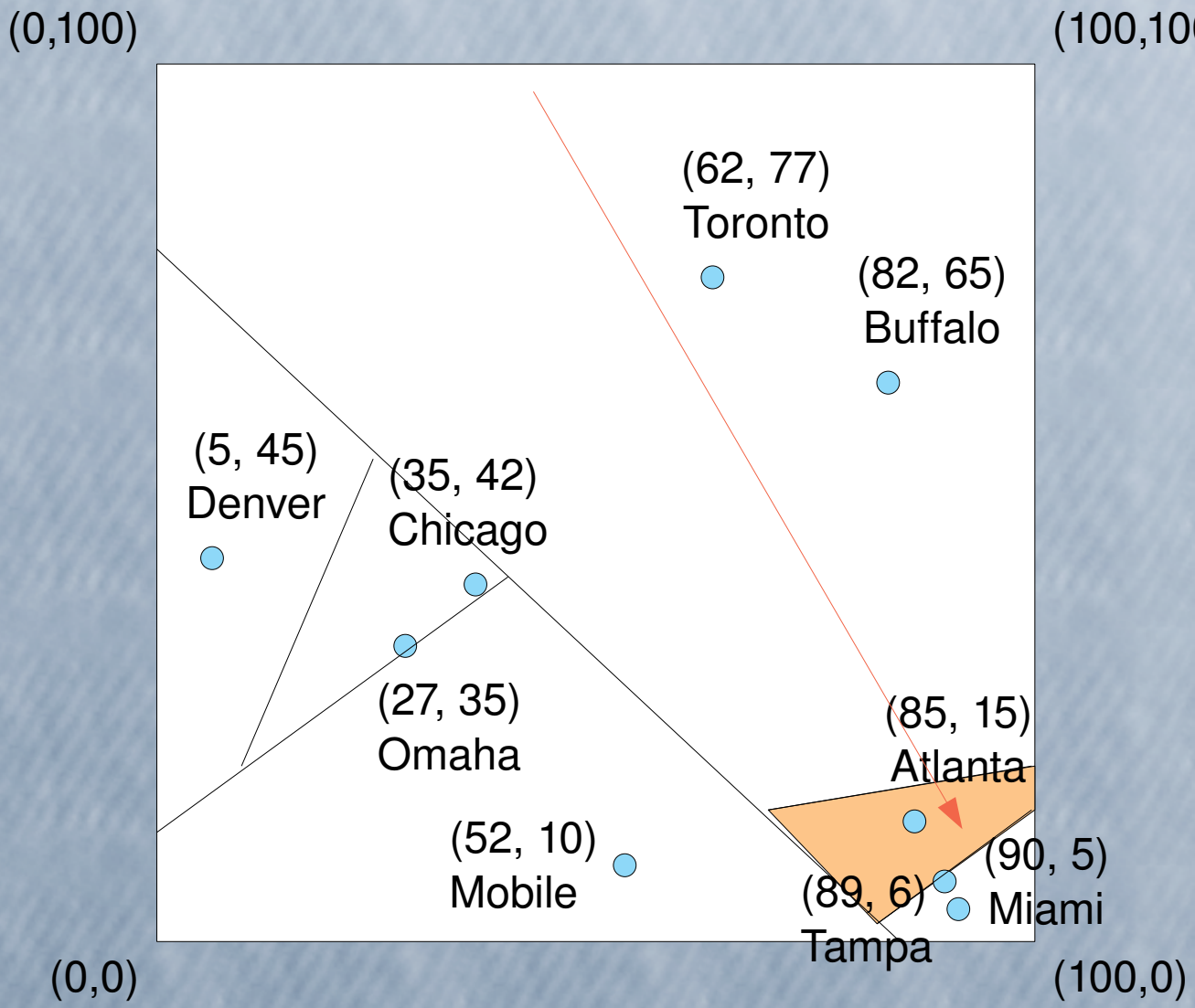
BSP Trees



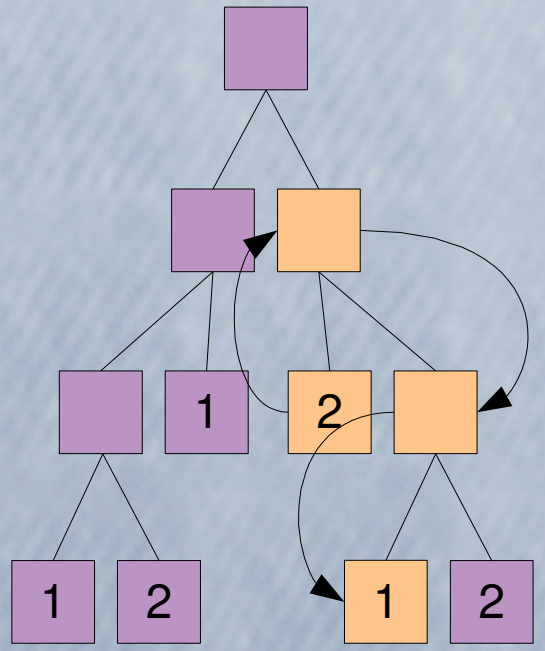
Tree Data Structure



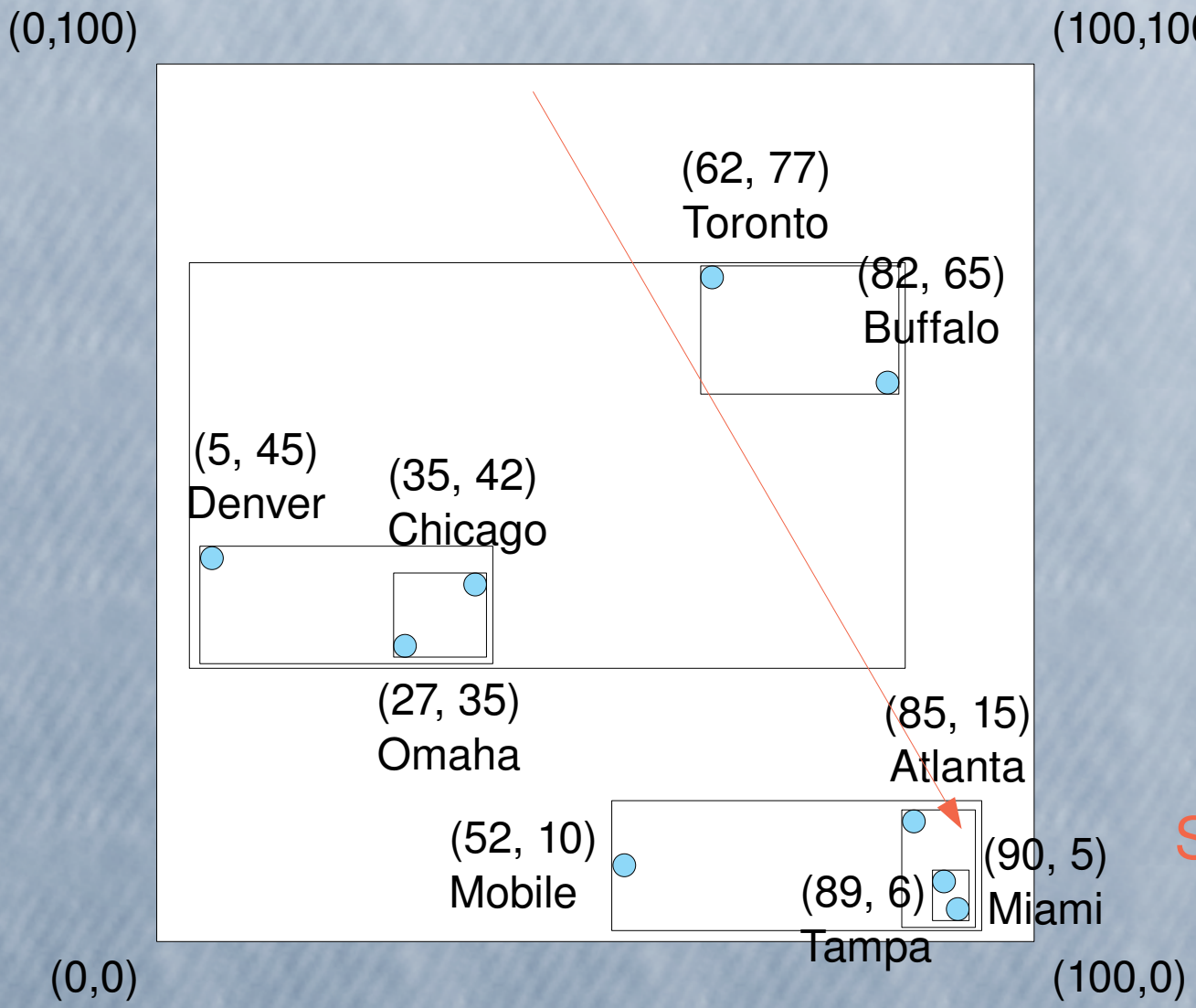
BSP Trees



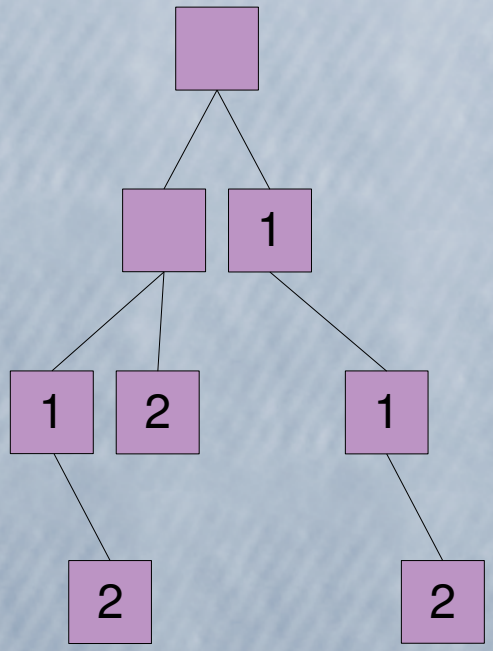
Tree Data Structure



Bounding Volume Hierarchies

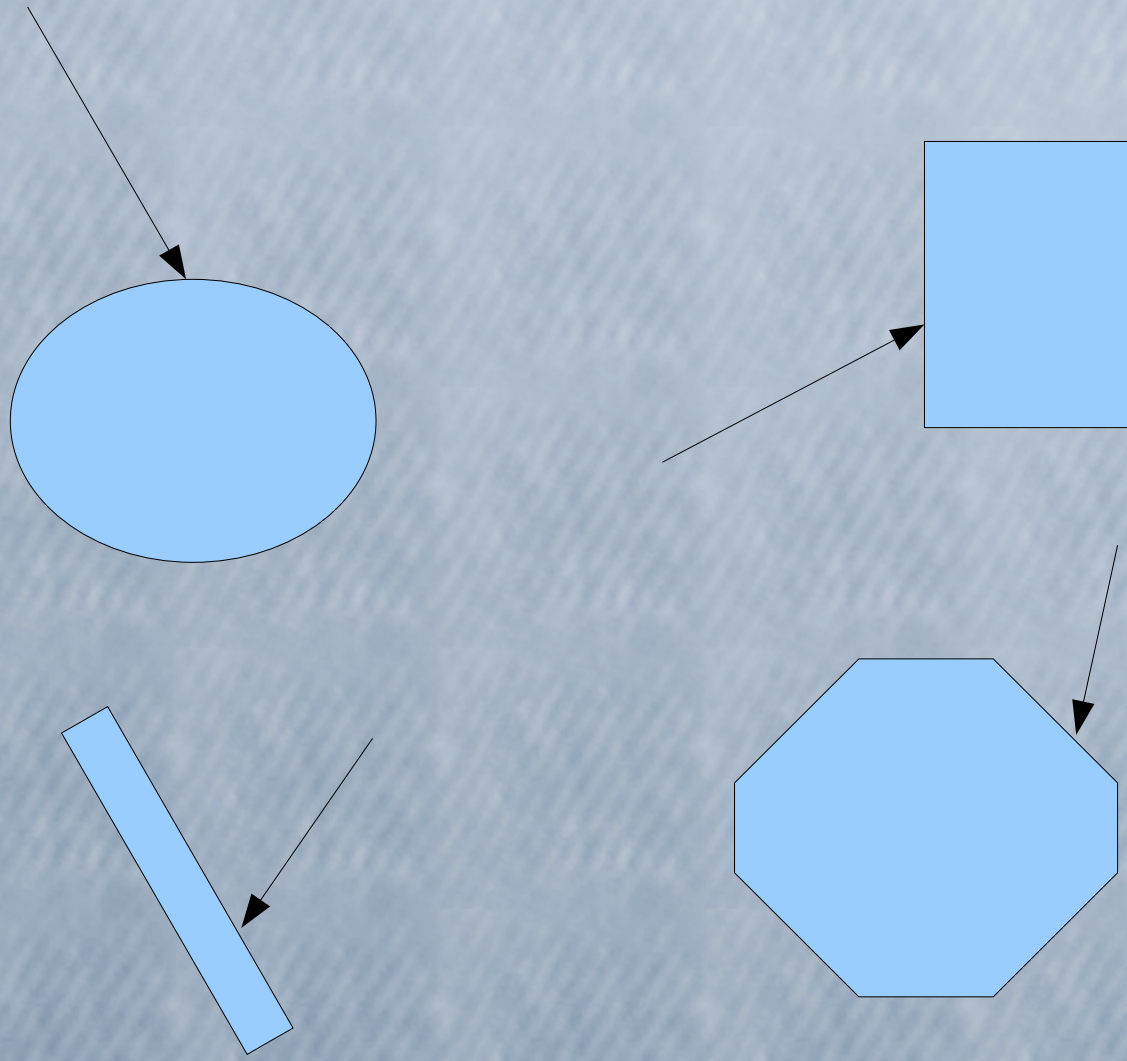


Tree Data Structure

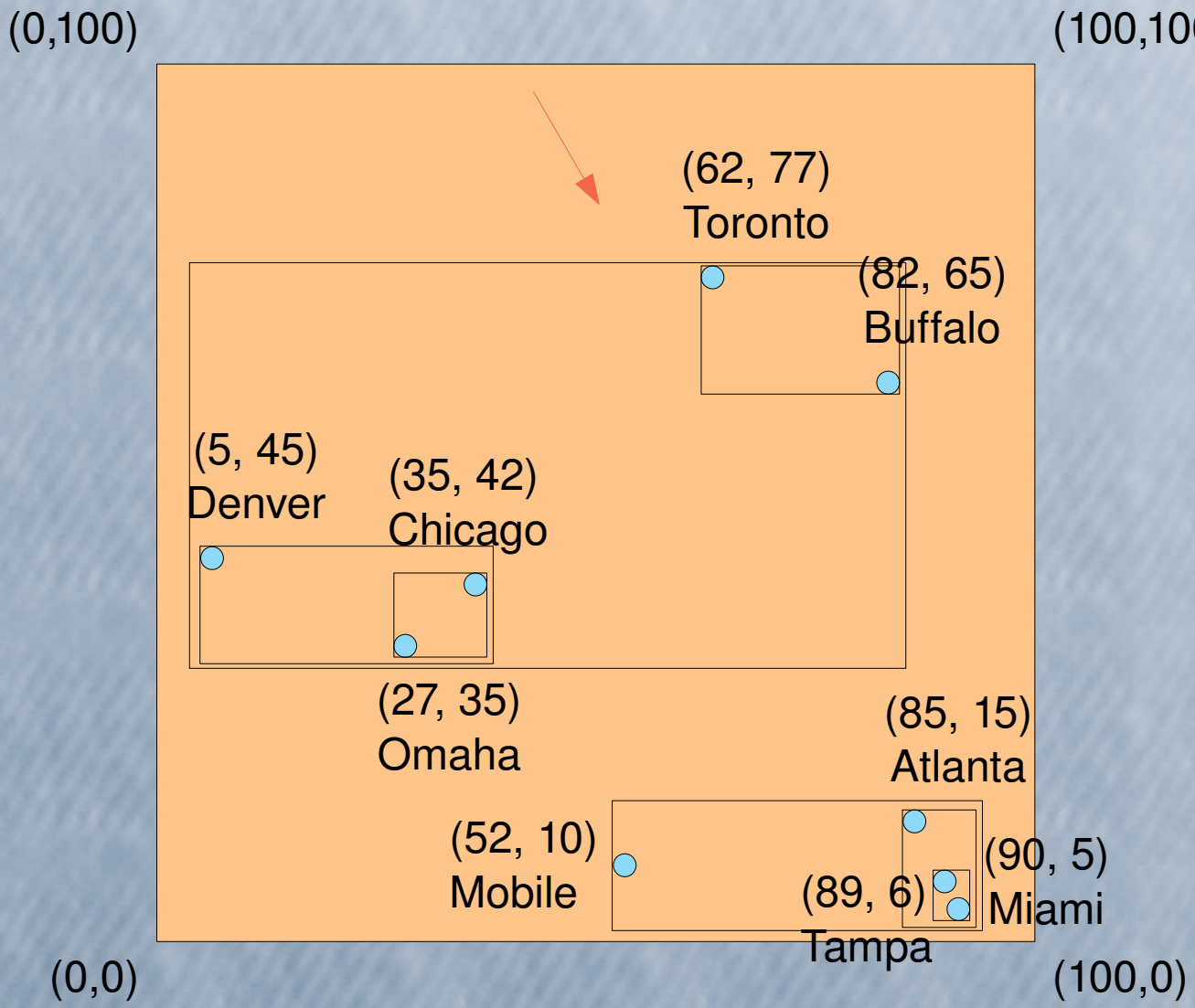


Some data not at leaves

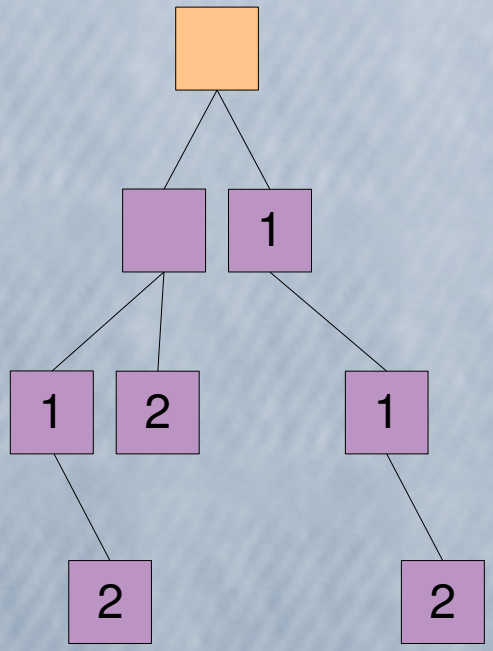
Intersection Against Bounding Volume Instead of Planes



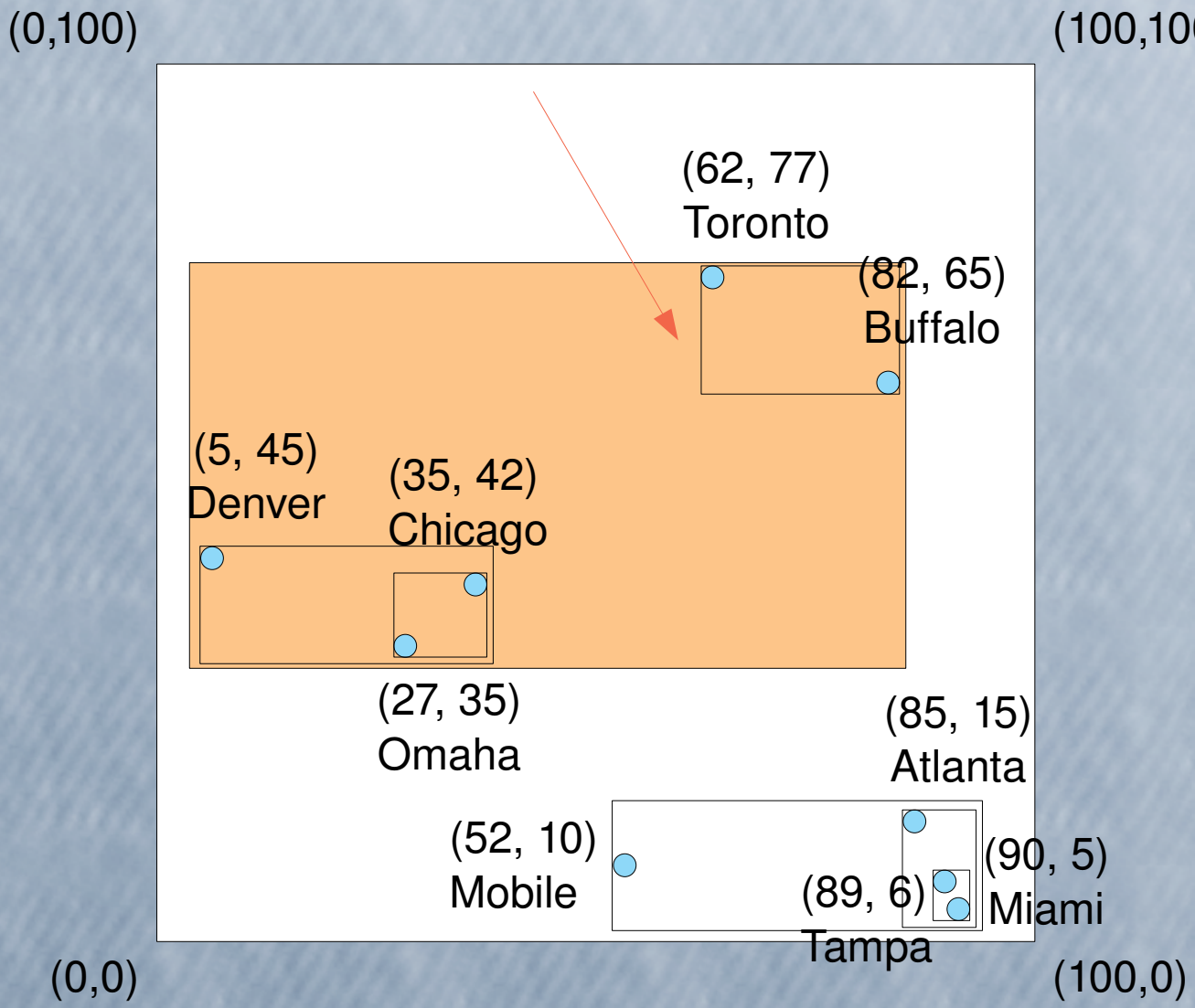
Bounding Volume Hierarchies



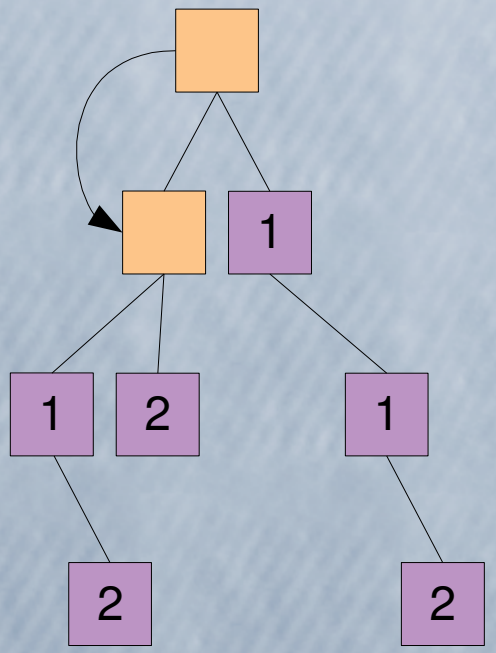
Tree Data Structure



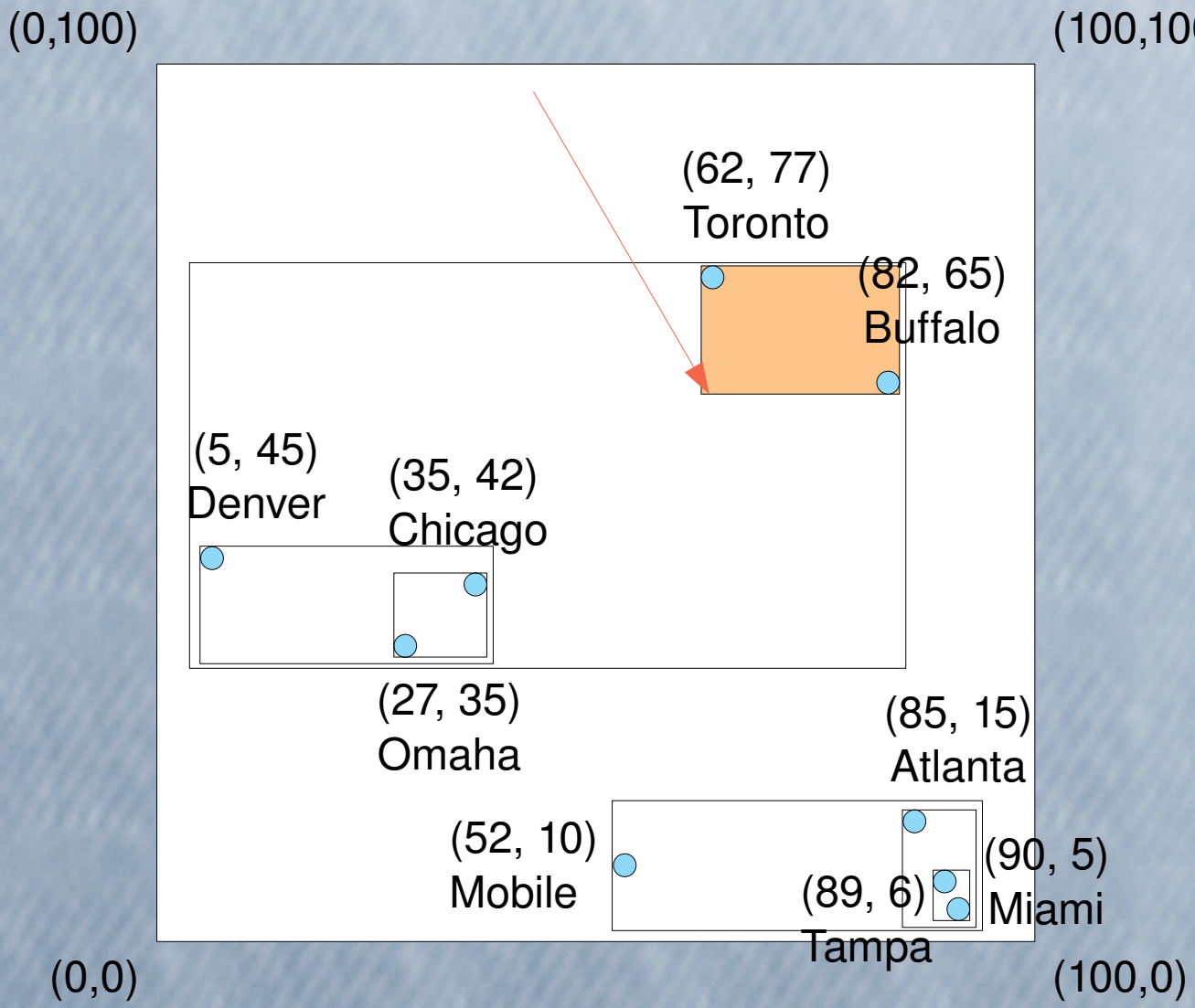
Bounding Volume Hierarchies



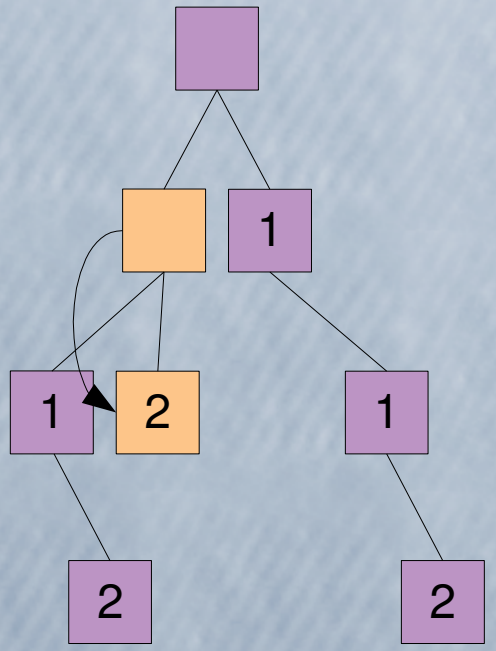
Tree Data Structure



Bounding Volume Hierarchies

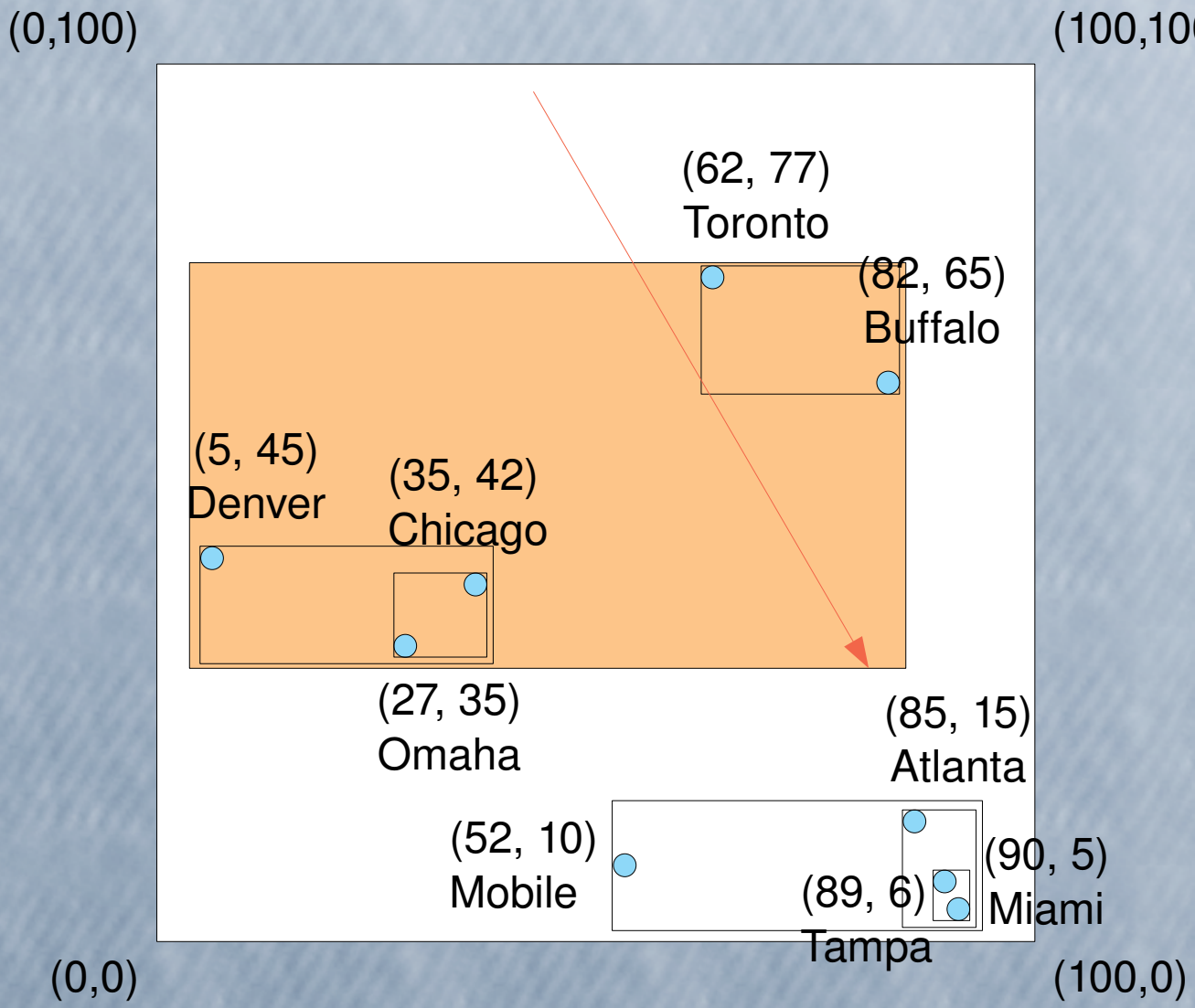


Tree Data Structure

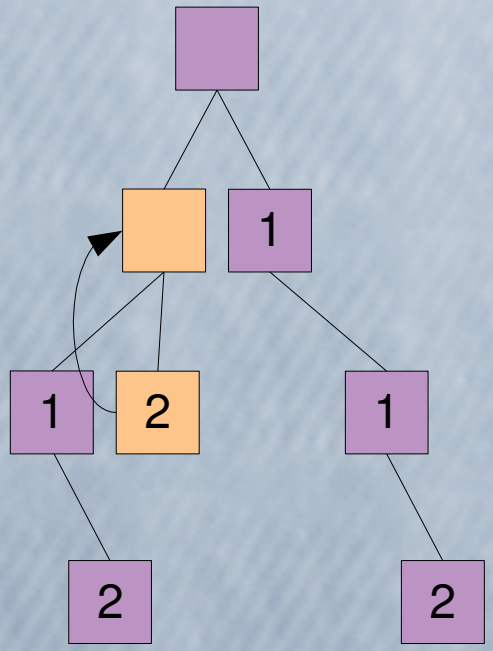


Bounding Volume Hierarchies

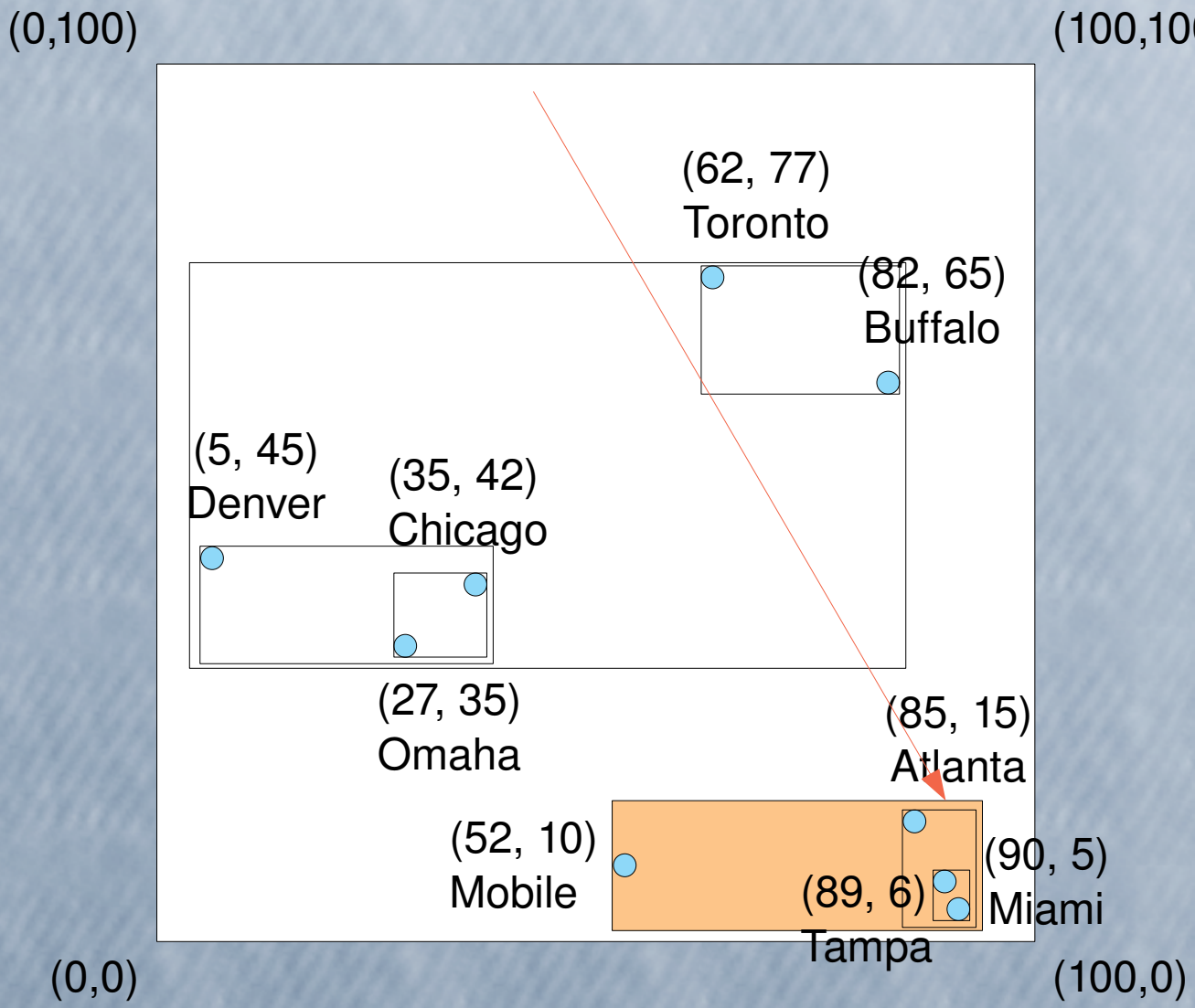
Test both enter and exit times from bounding volumes, why?



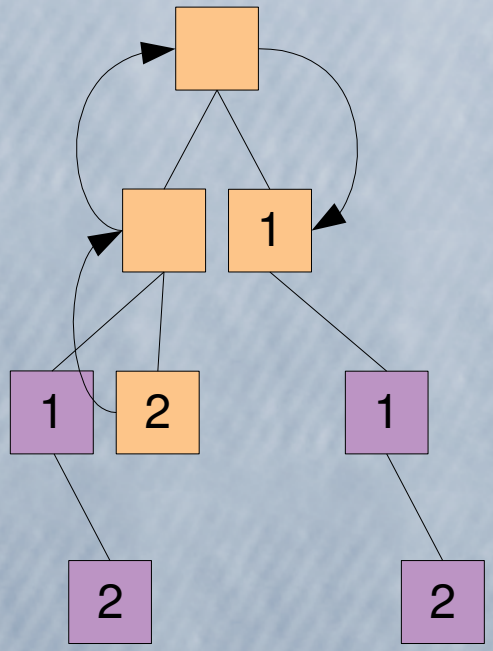
Tree Data Structure



Bounding Volume Hierarchies



Tree Data Structure



Next Time

- Paper Selections Due
- More operations on spatial data structures
- Read: Chapter 1, Section 6 and 7 (pages 90-163)