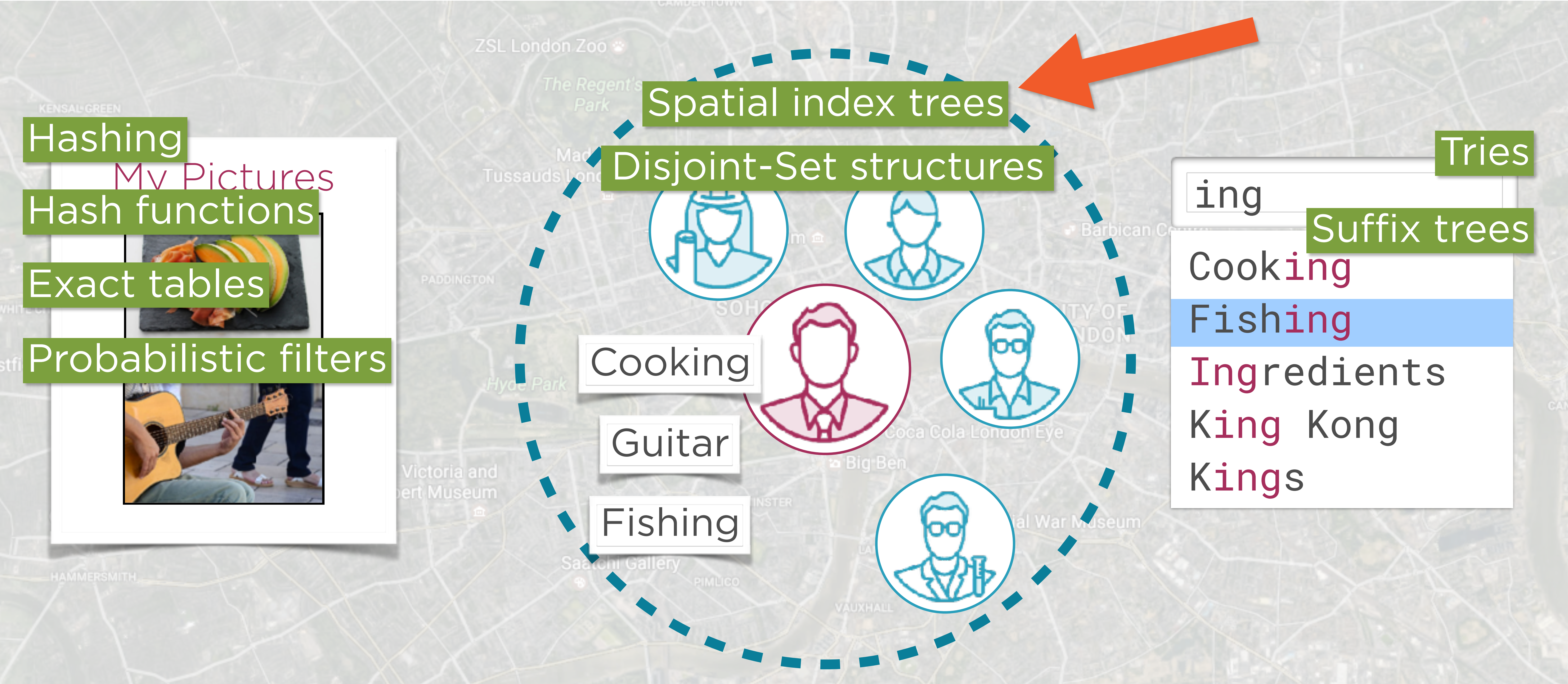
Querying Space and Time with Tree Structures

Rasmus Resen Amossen
SOLUTION ARCHITECT

rasmus.resen.org

The Match Finder App



Hashing

My Pictures

Hash functions

Exact tables

Probabilistic filters

Spatial index trees

Disjoint-Set structures

Cooking

Guitar

Fishing

Tries

ing

Suffix trees

Cooking

Fishing

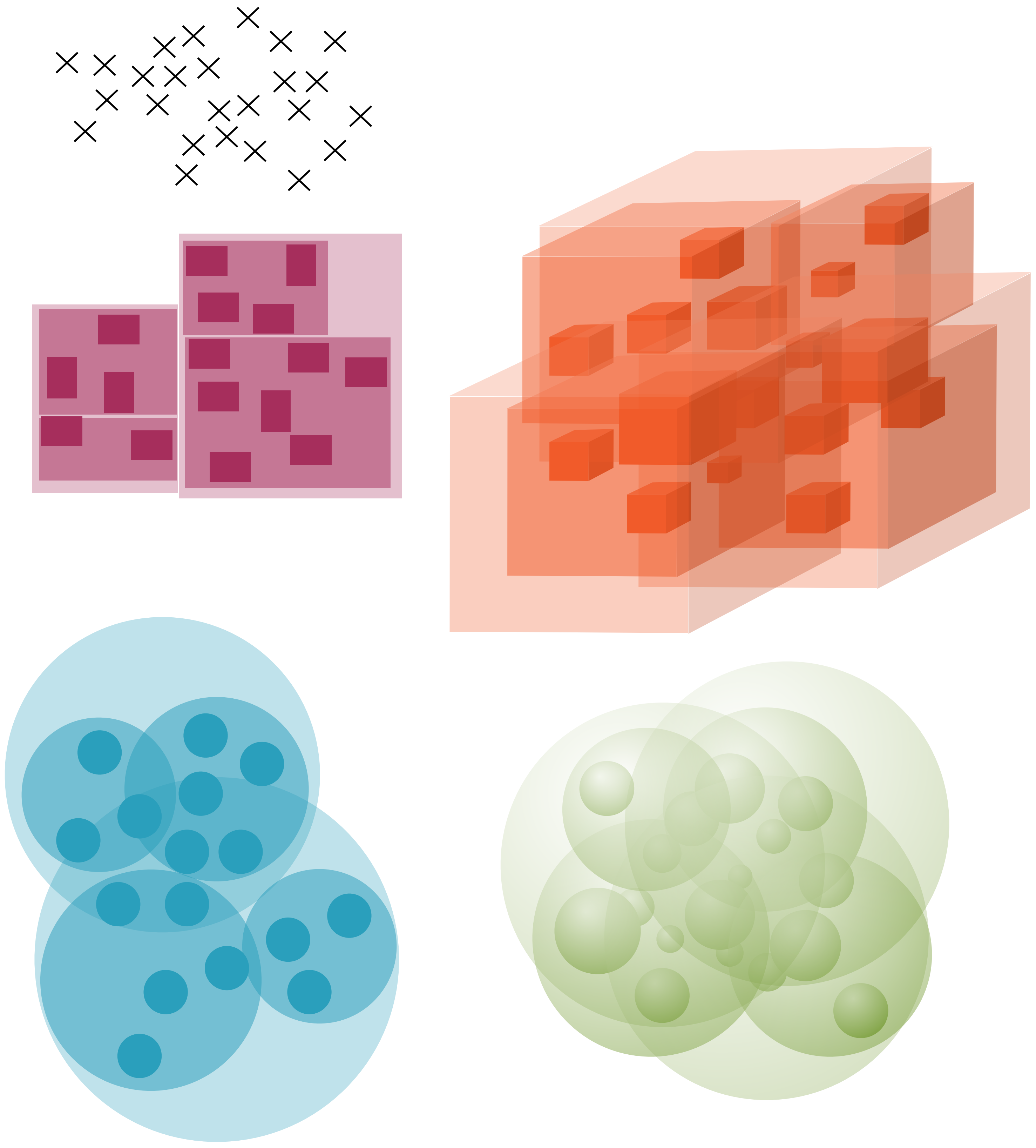
Ingredients

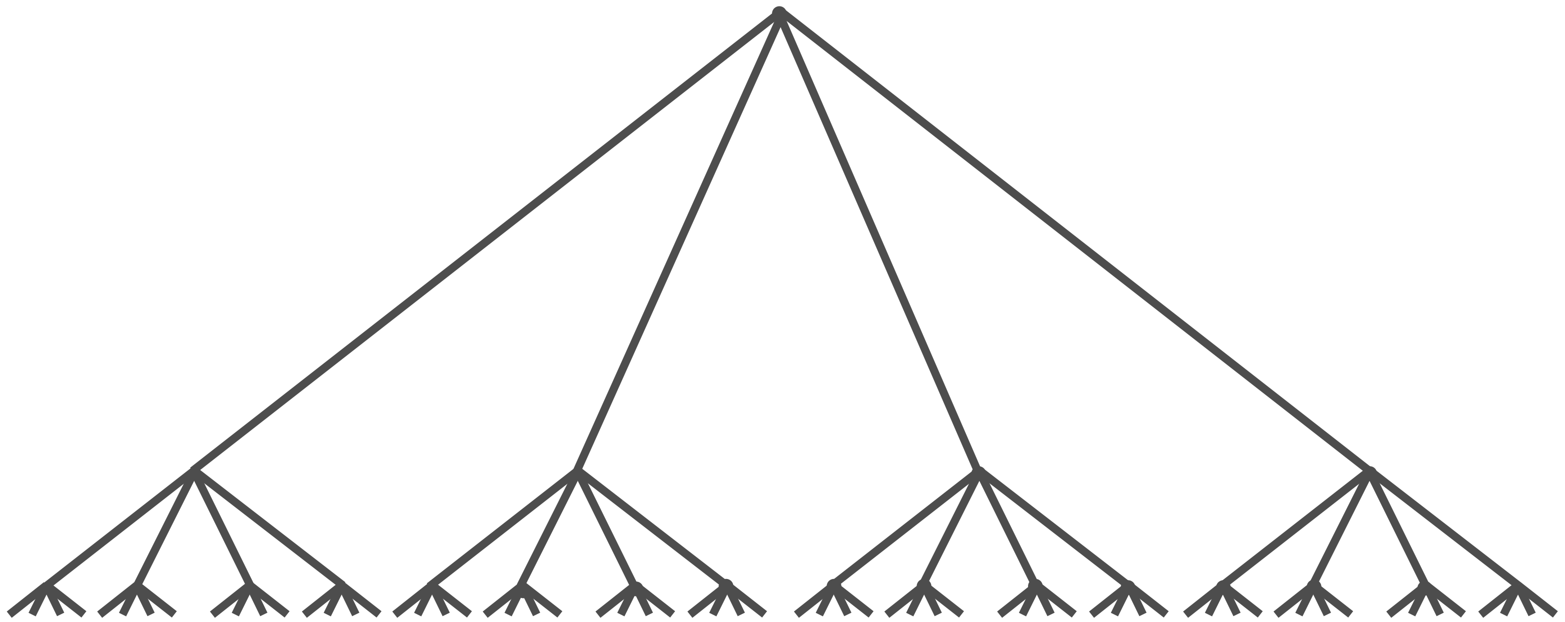
King Kong

Kings

Demo

Spatial search - initializing the solution



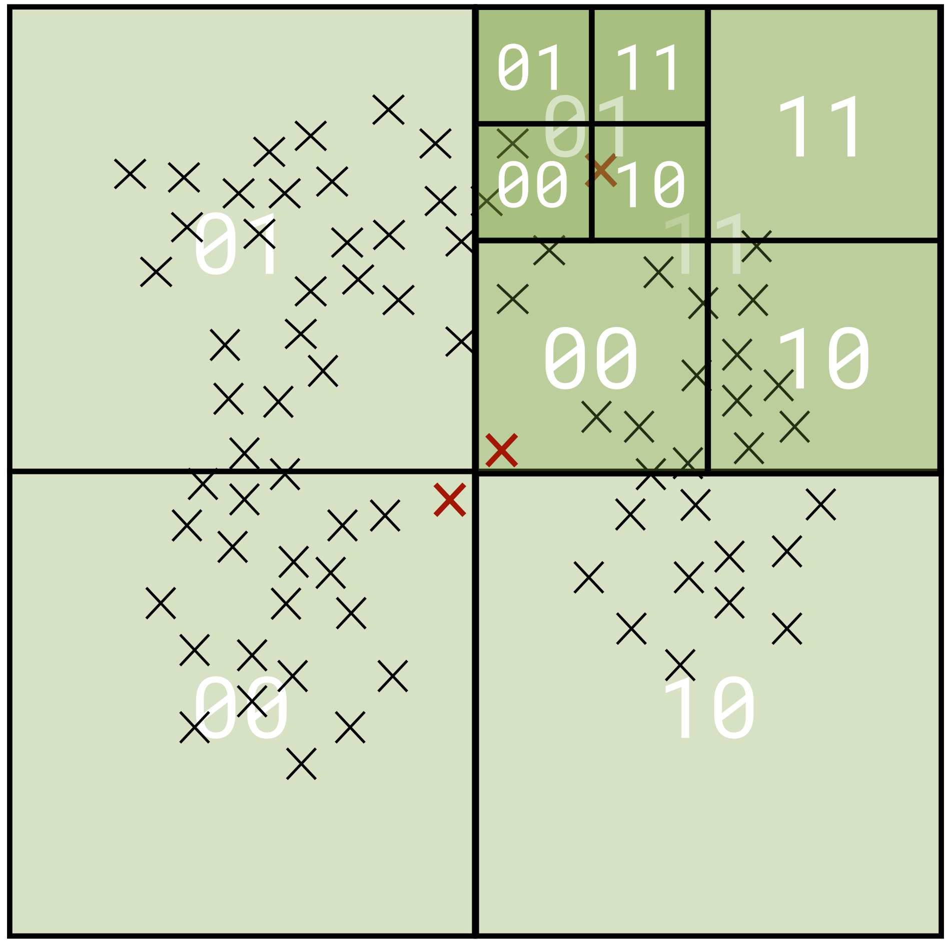


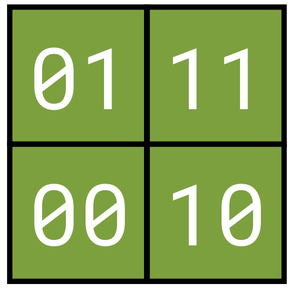
1. Geohash: mostly for points
2. Basic tree: **B-tree**
3. With rectangles: **R-tree**
4. With spheres: **M-tree**

Geohashing

Geohash for ×: 110110

```
110110101001101101
110110101001100011
```





26 bits to represent whole earth with around 60 cm precision

B-tree to index geohashes

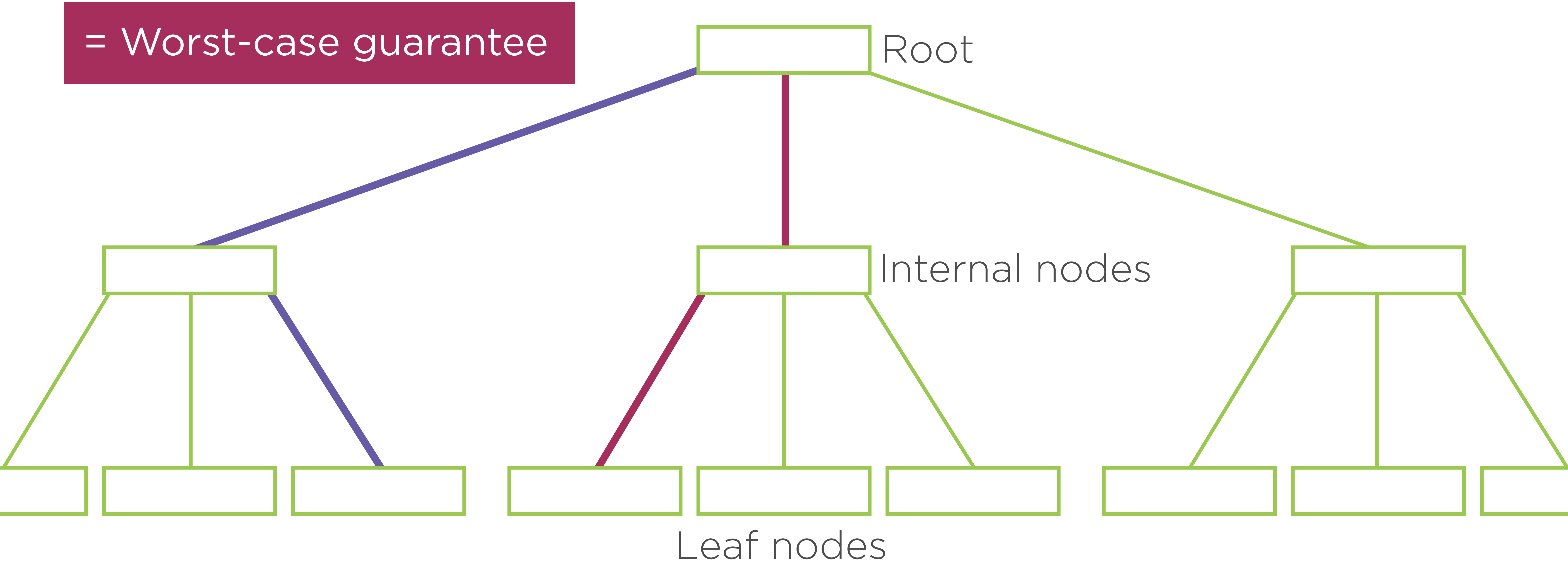
Wikipedia: “Geohash”

MongoDB: tinyurl.com/mongogeohash
SQL Server: tinyurl.com/sqlSpatial

B-trees

Balanced

= Worst-case guarantee



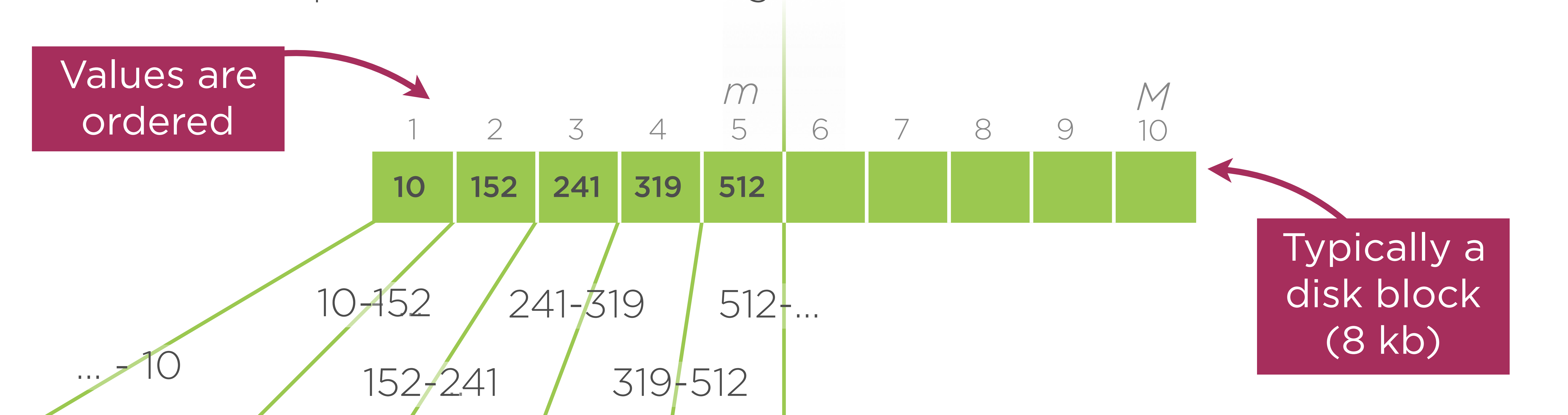
Nodes

All nodes have the same capacity, M

Inner and leaf nodes require minimum m elements ($m \geq M/2$)

Height, h, gives $(M+1)^h - 1$ values in total

Example: $M = 100$ and $h = 5$ gives 10 billion values in the tree



Values are ordered

Typically a disk block (8 kb)

Lookup

Start at root node

Find matching sub-tree

Traverse recursively…

…until found or not found

Find 164

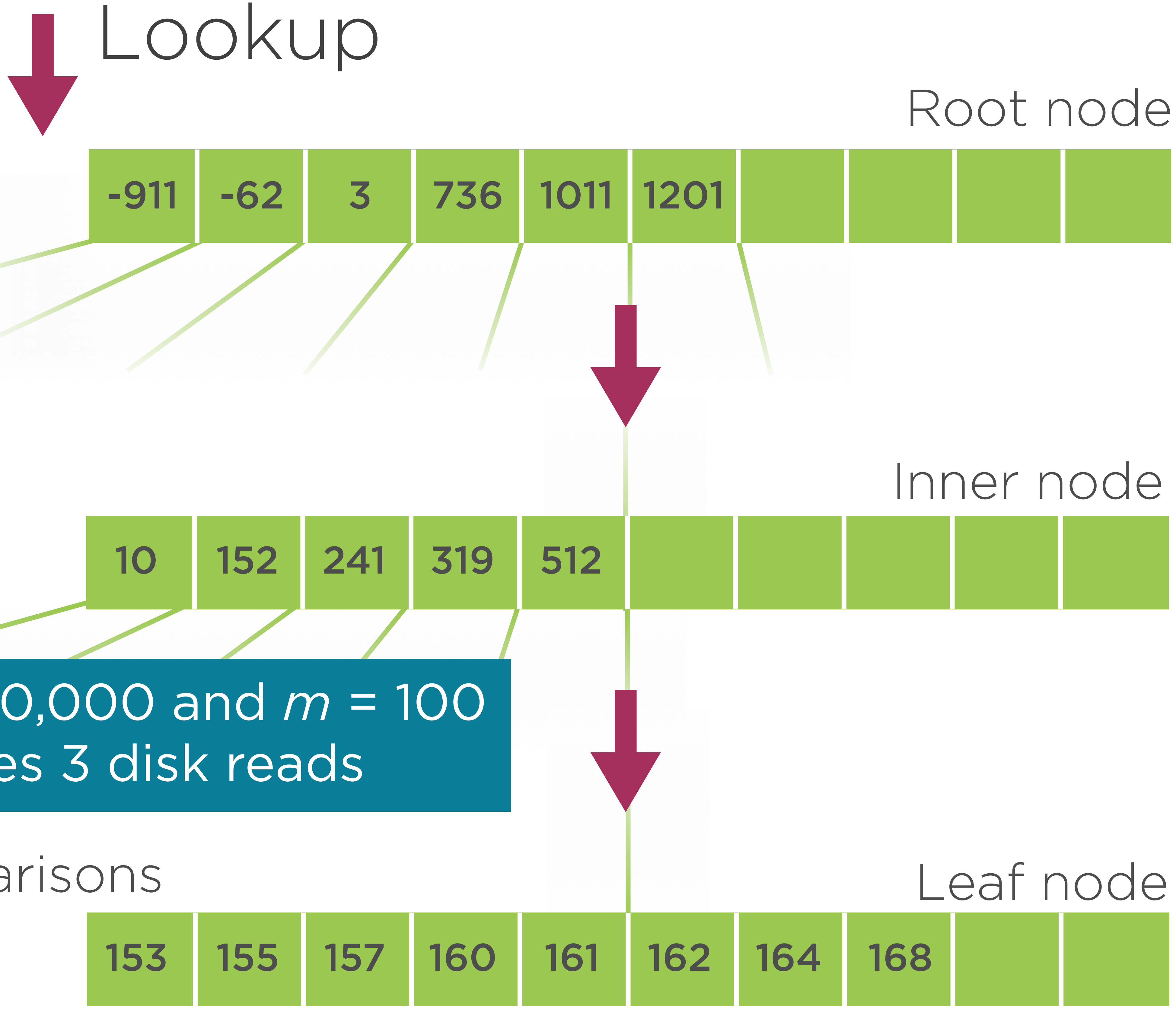
For N values…

$\lceil \log_{m+1}(N) \rceil$ levels

$O(\log_2(m) \cdot \log_{m+1}(N))$ comparisons

$= O(\log(N))$

N = 1,000,000 and m = 100 gives 3 disk reads



Insertion

Build tree bottom-up

$M = 4$, $m = 2$

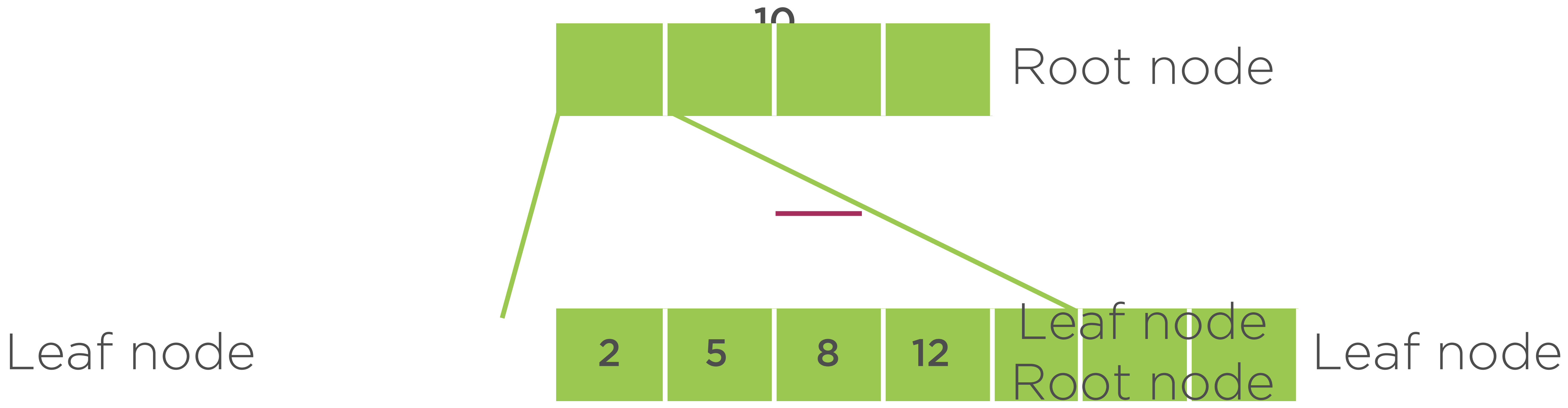
[illegible]

Leaf node
Root node

Insertion

Use median as split point

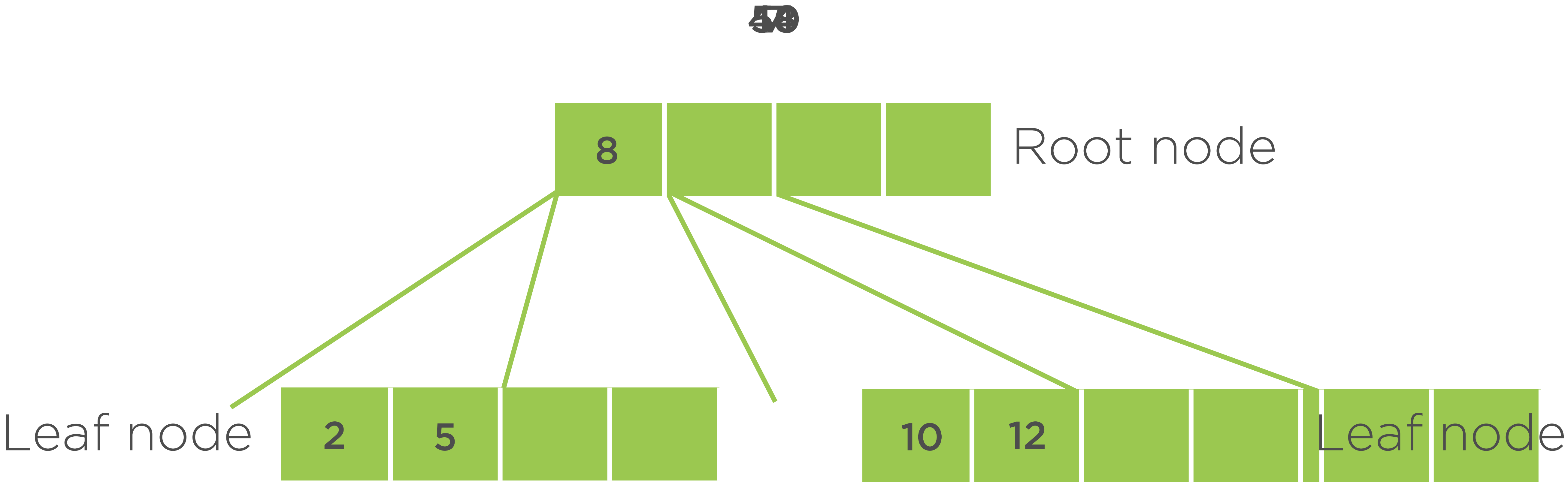
Put median at parent node



Insertion

Use median as split point

Put median at parent node



Root node

Leaf node

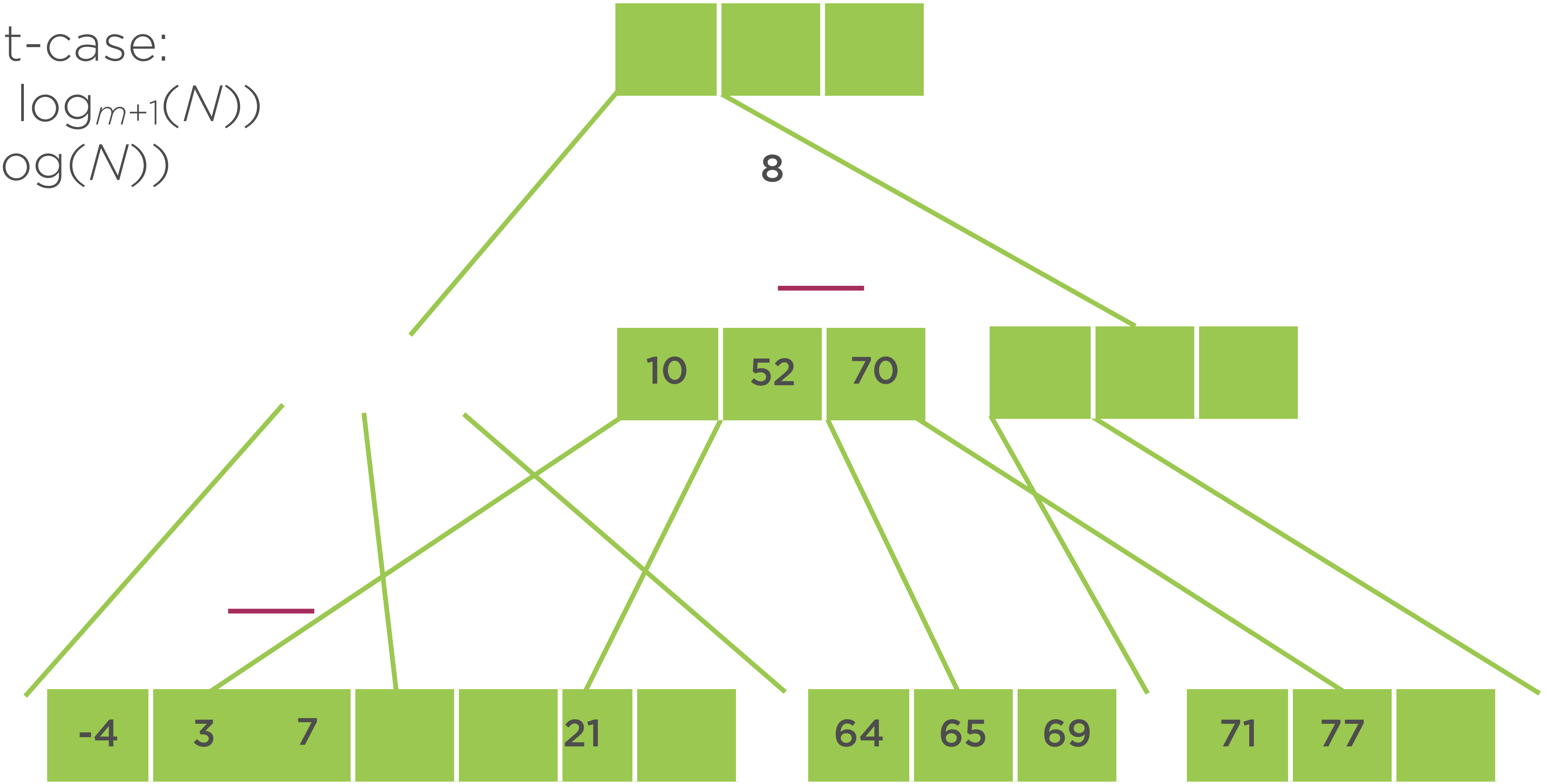
Leaf node

Insertion

$M = 3$

$m = 1$

Worst-case:
$O(M \cdot \log_{m+1}(N))$
$= O(\log(N))$



Deletion

Delete value from node

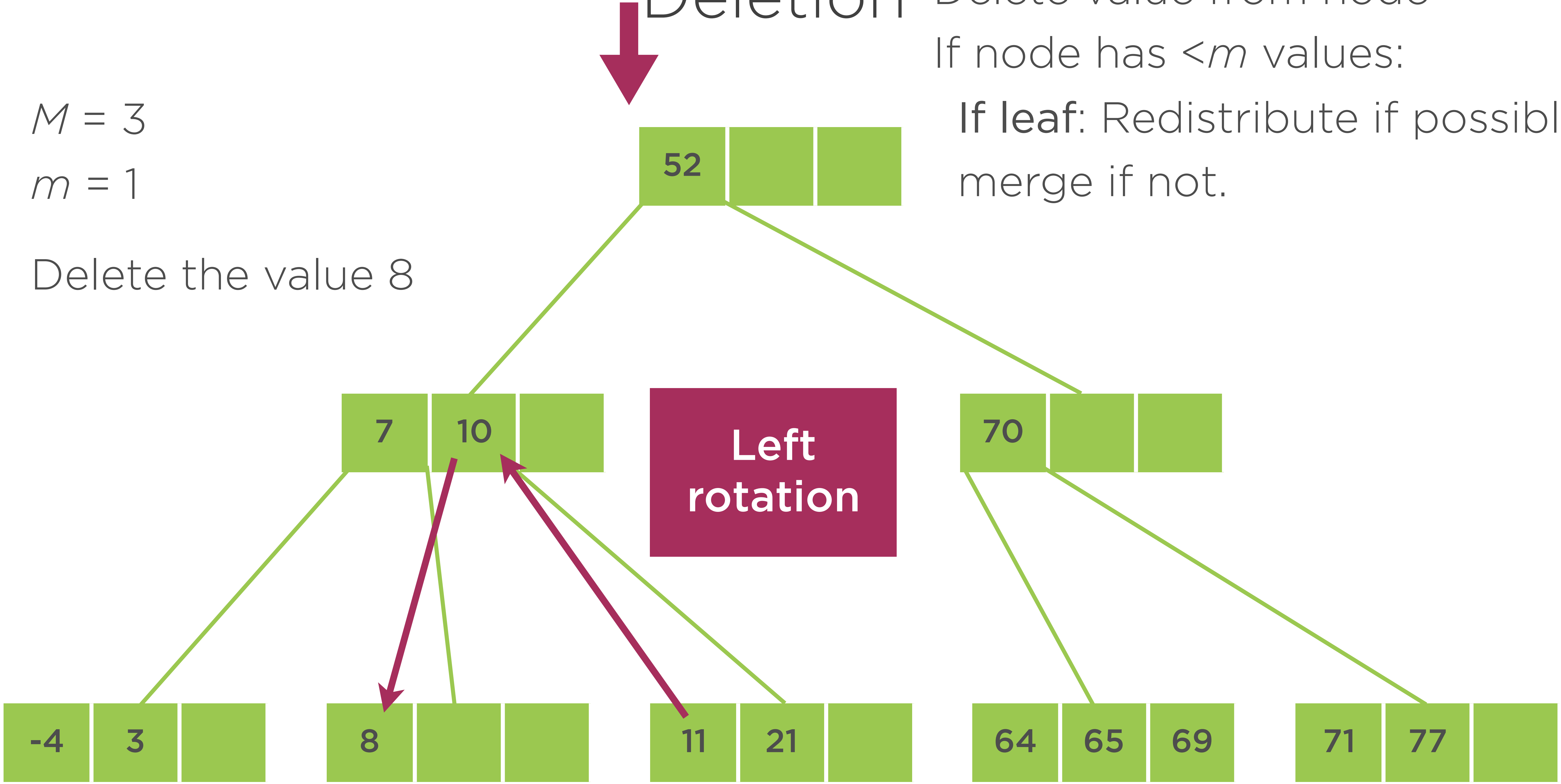
If node has <*m* values:

- **If leaf**: Redistribute if possible, merge if not.

M = 3

m = 1

Delete the value 8



Deletion

Delete value from node

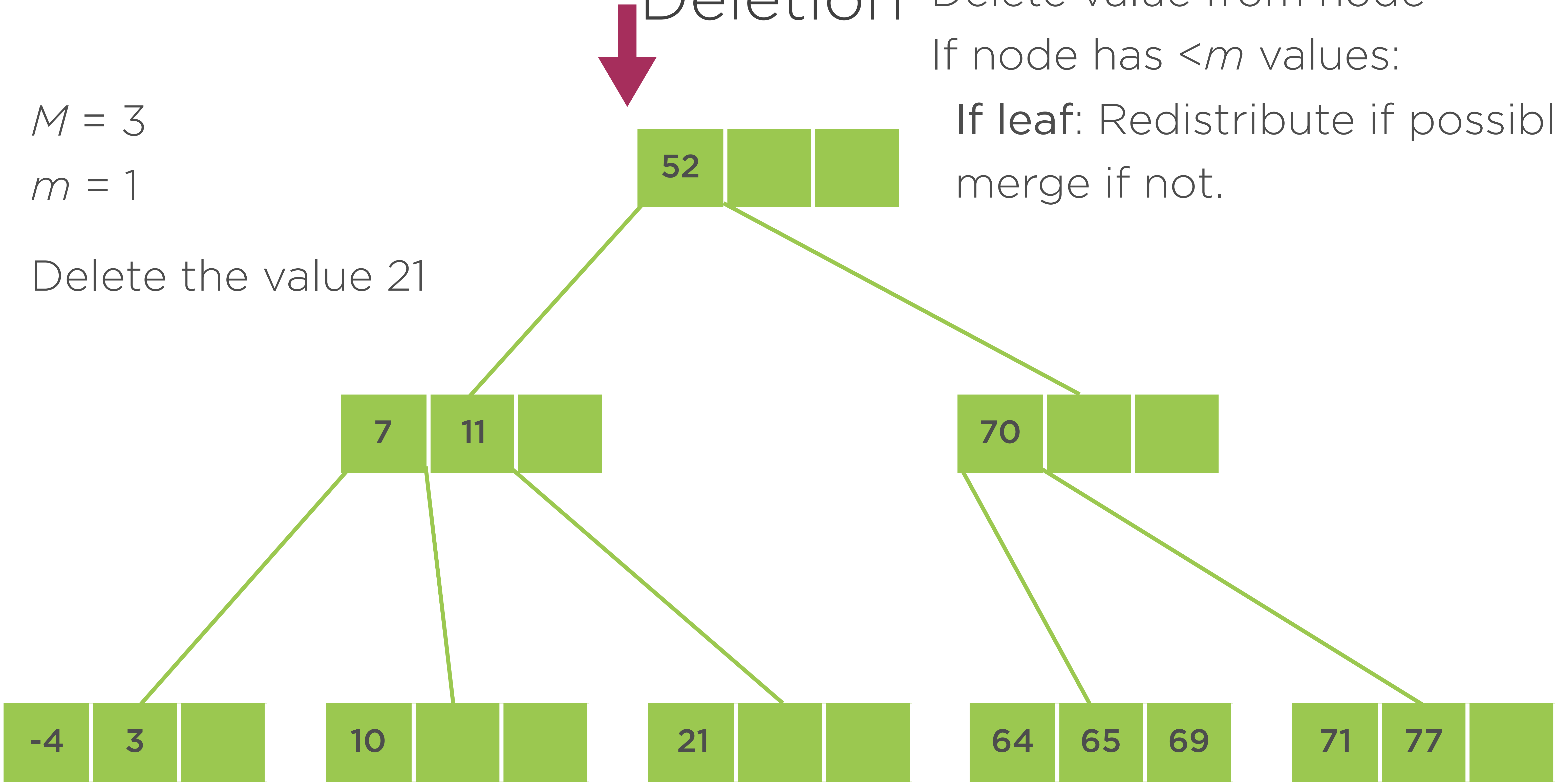
If node has <*m* values:

If leaf: Redistribute if possible, merge if not.

M = 3

m = 1

Delete the value 21



Deletion

$M = 3$

$m = 1$

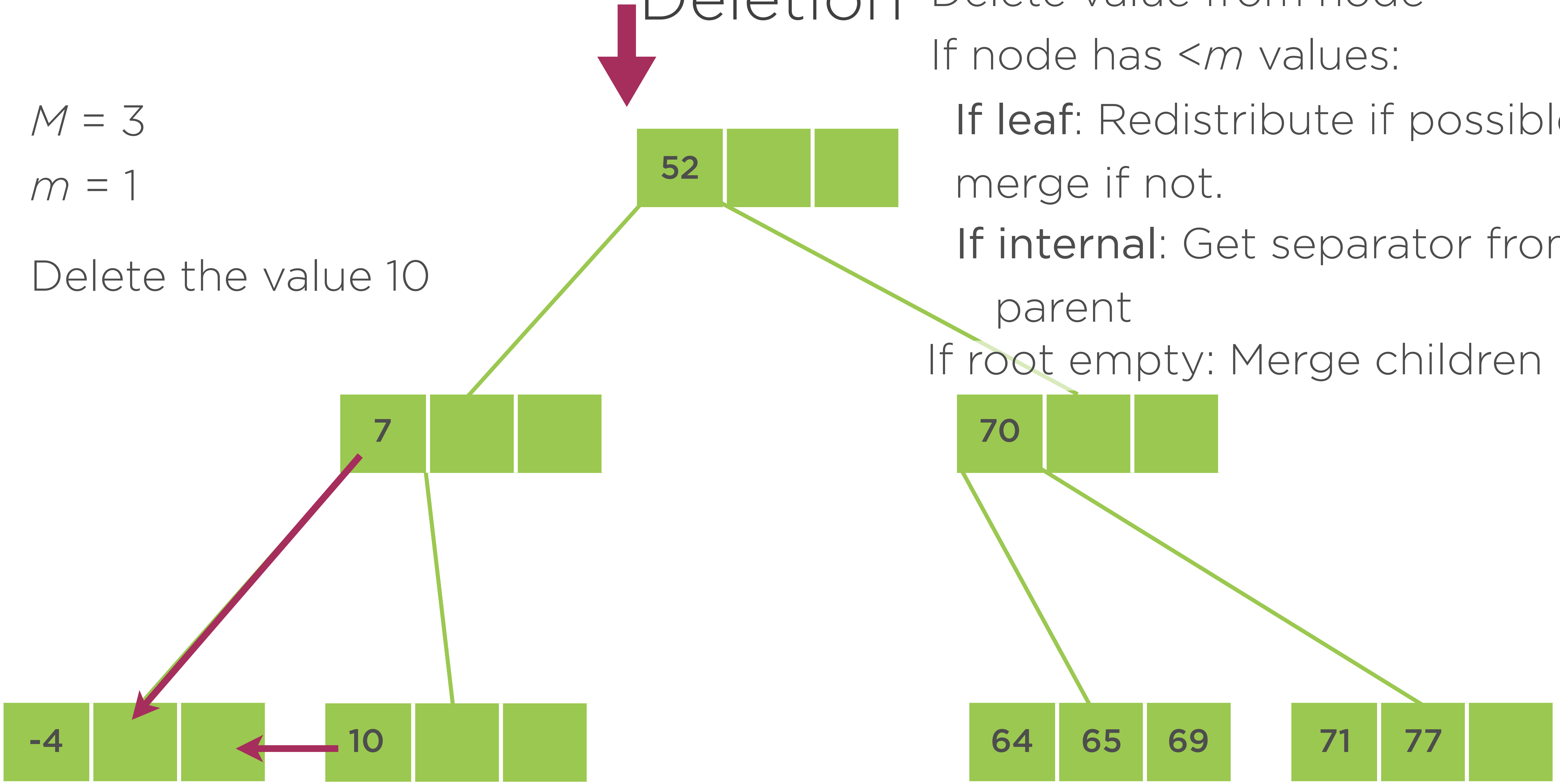
Delete the value 10

Delete value from node

If node has <m values:

- **If leaf**: Redistribute if possible, merge if not.
- **If internal**: Get separator from parent

If root empty: Merge children

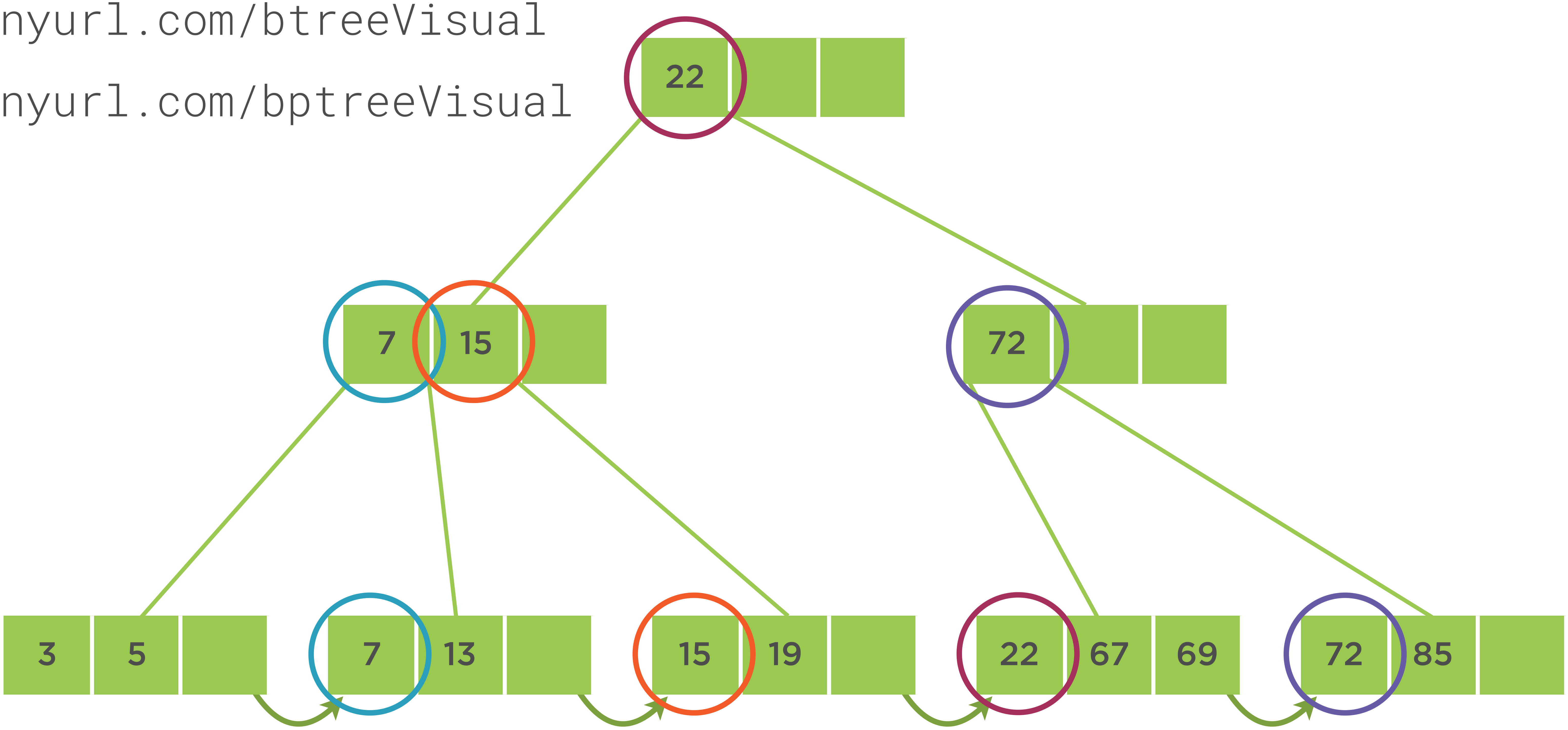


B+-trees

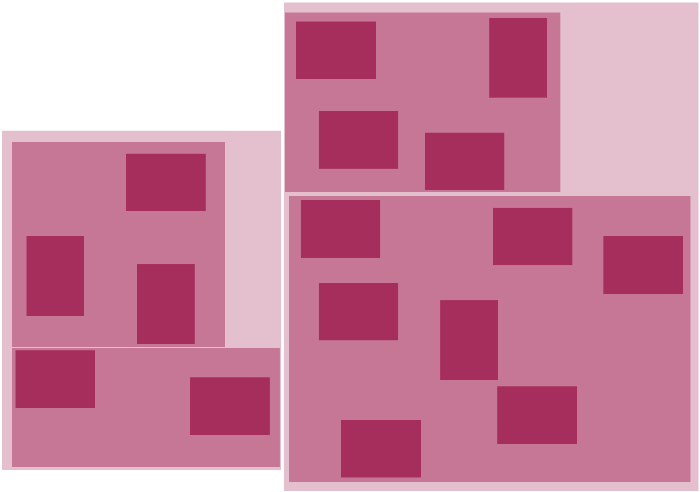
Wikipedia: “B-tree”

tinyurl.com/btreeVisual

tinyurl.com/bptreeVisual

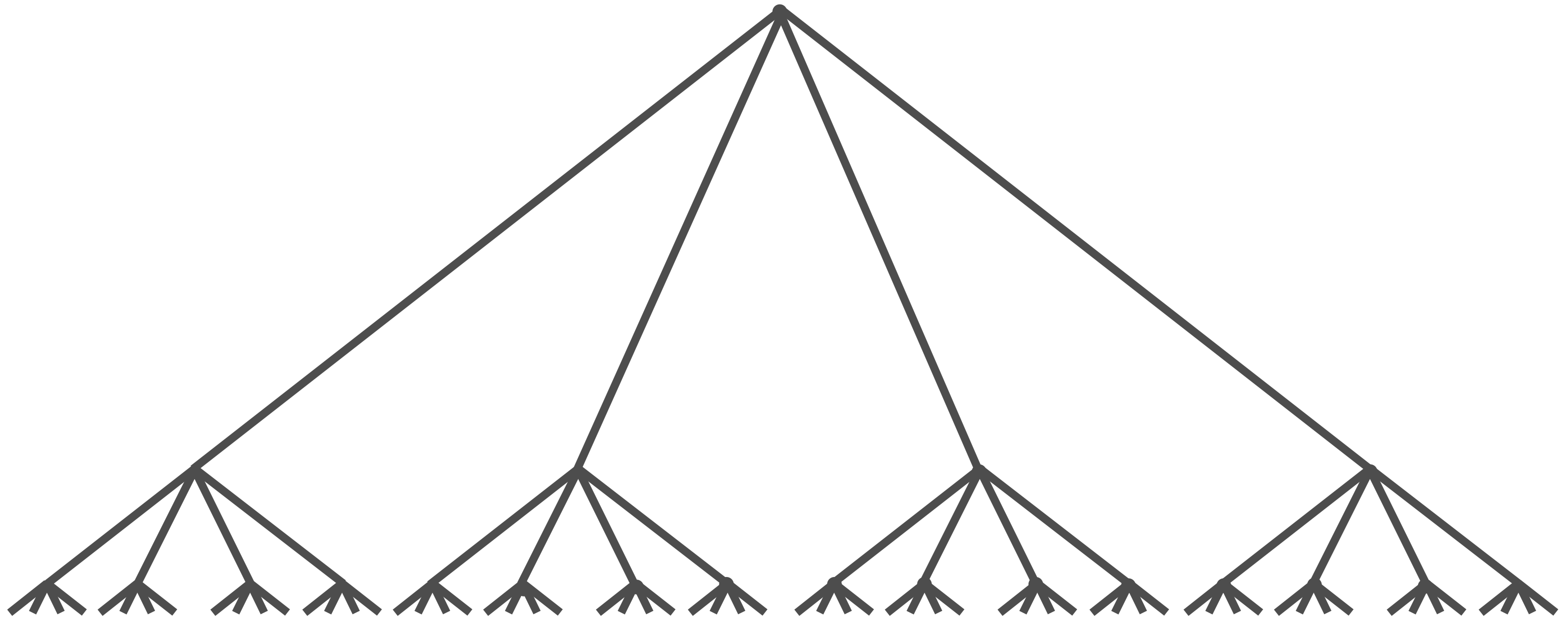


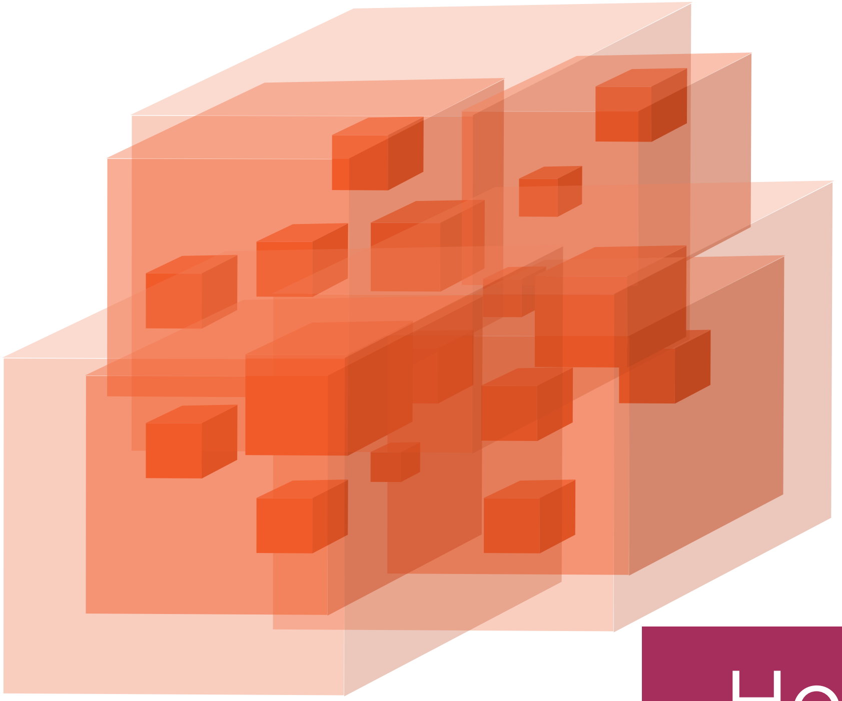
R-trees



Get all rectangles within query box

Get nearest rectangle(s) relative to query box





Create hierarchy of nested rectangles

Organize nested rectangles in B-tree

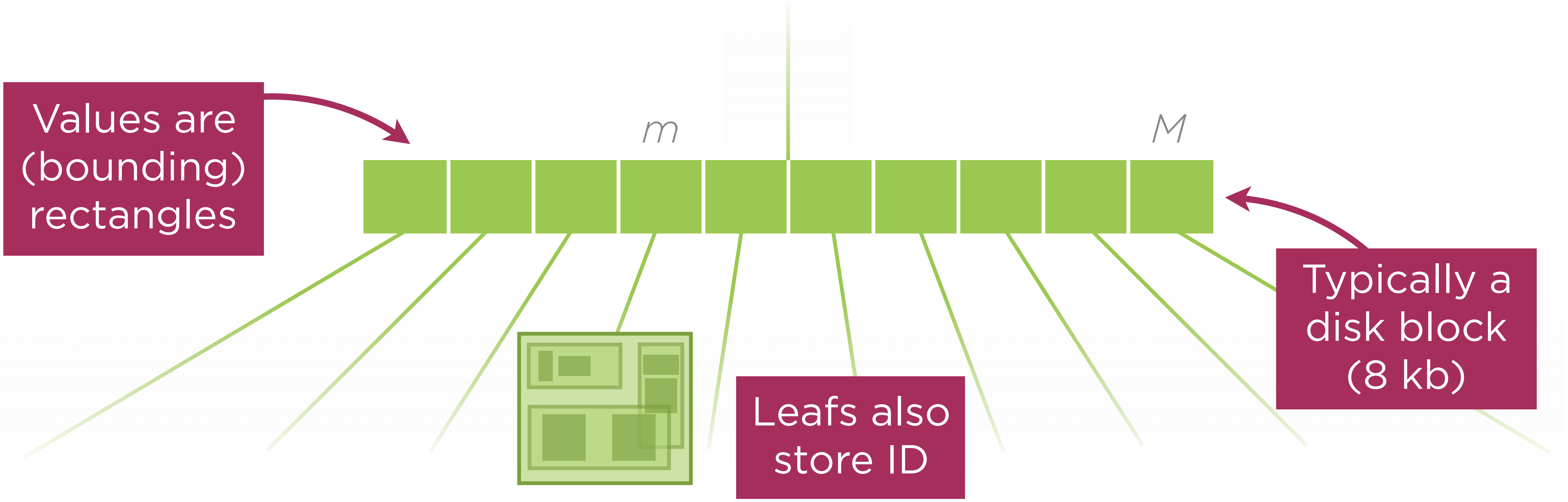
How to manage covering rectangles?

Nodes

All nodes have the same capacity, *M*

Inner and leaf nodes require minimum *m* elements

m between 30% and 40% of *M*

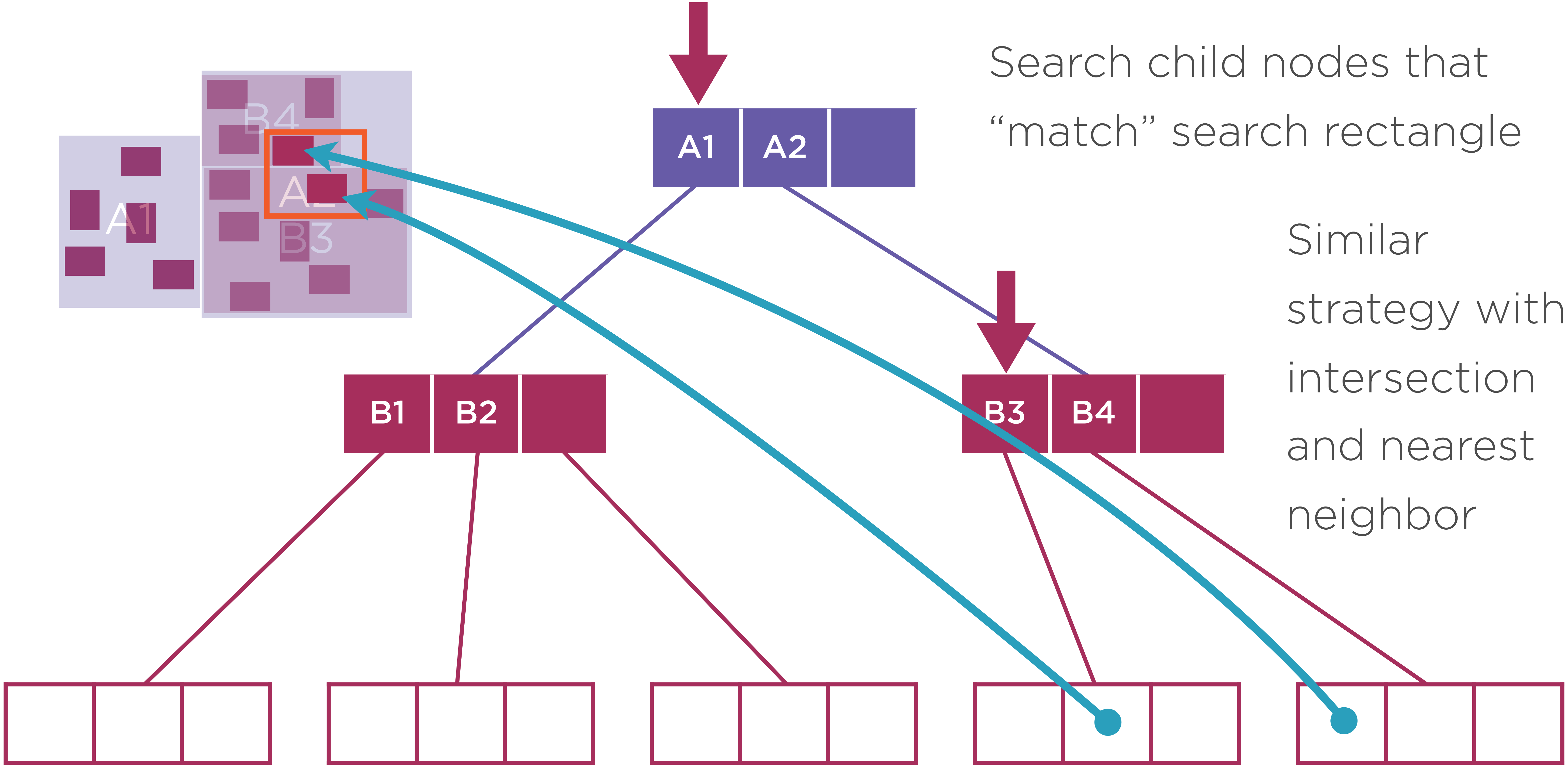


Values are (bounding) rectangles

Typically a disk block (8 kb)

Leafs also store ID

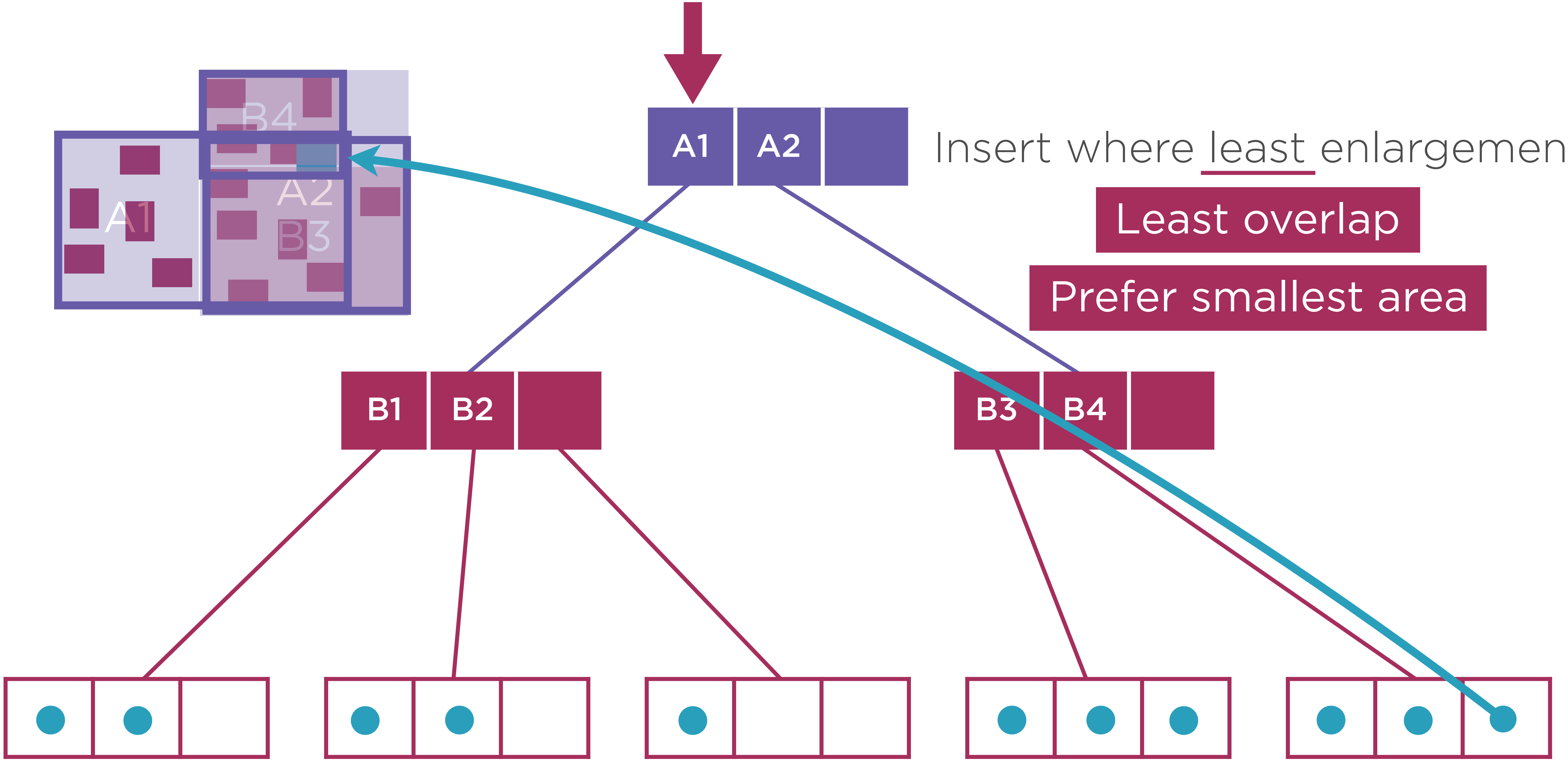
Lookup



Search child nodes that “match” search rectangle

Similar strategy with intersection and nearest neighbor

Insertion



Insert where least enlargement

Least overlap

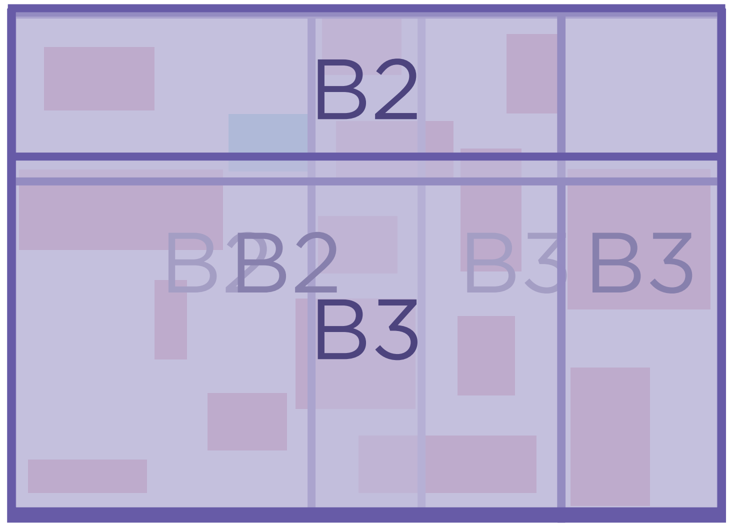
Prefer smallest area

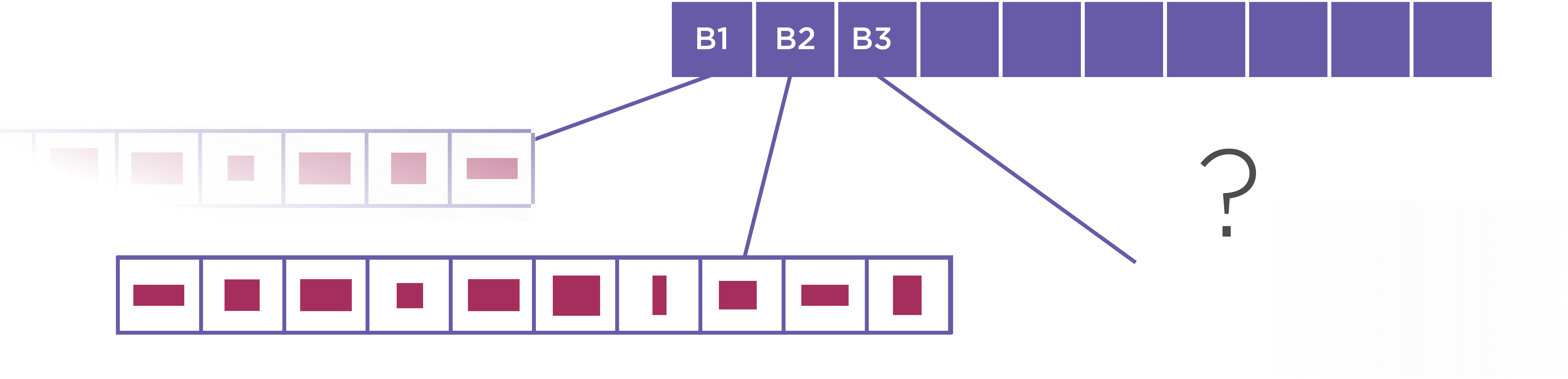
Insertion - Node Split

2^N possible splits of N rectangles

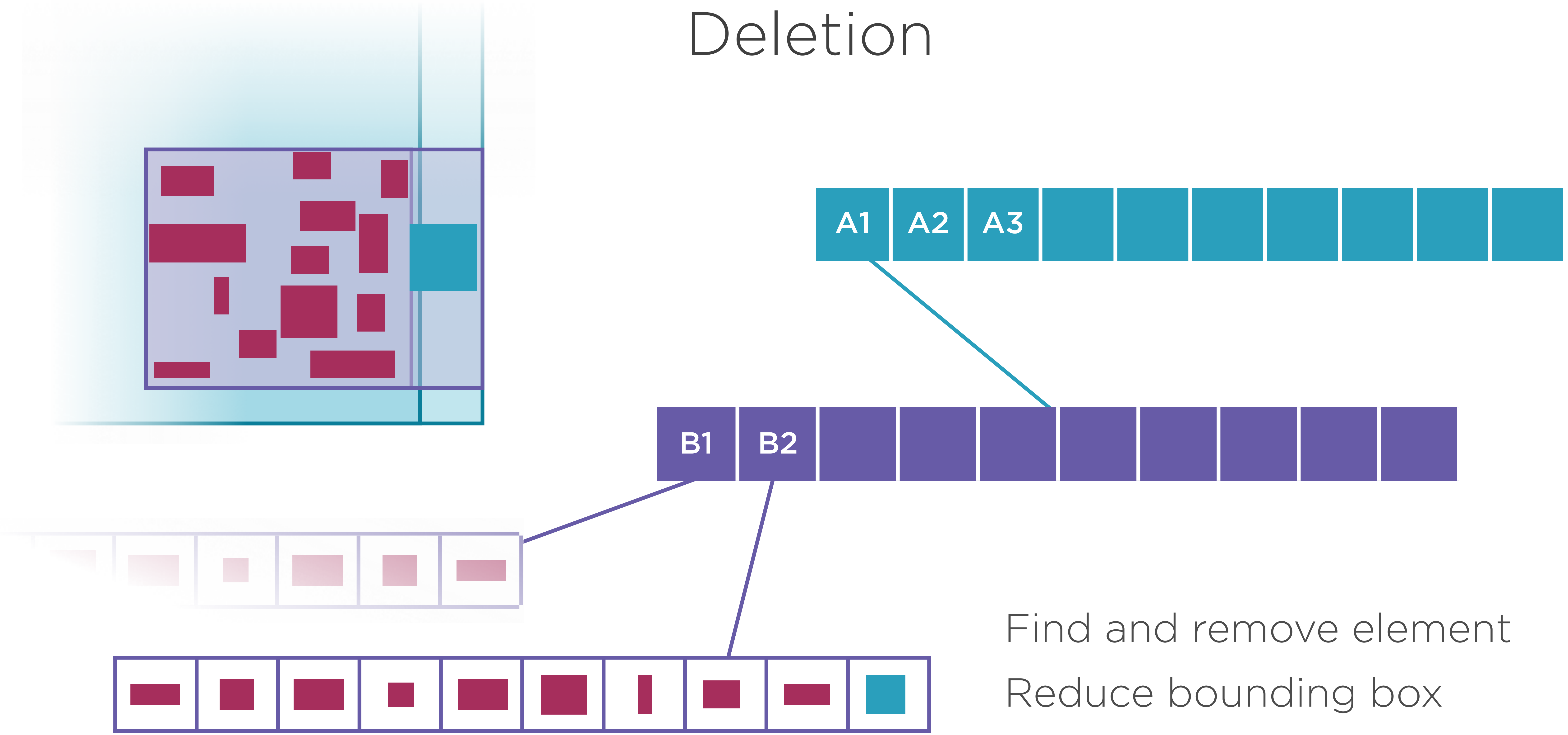
Split strategy highly affects performance

Greene's split
Linear split
Quadratic split
R* tree split





Deletion



Find and remove element

Reduce bounding box

Recurse upwards

Deletion - Underflow

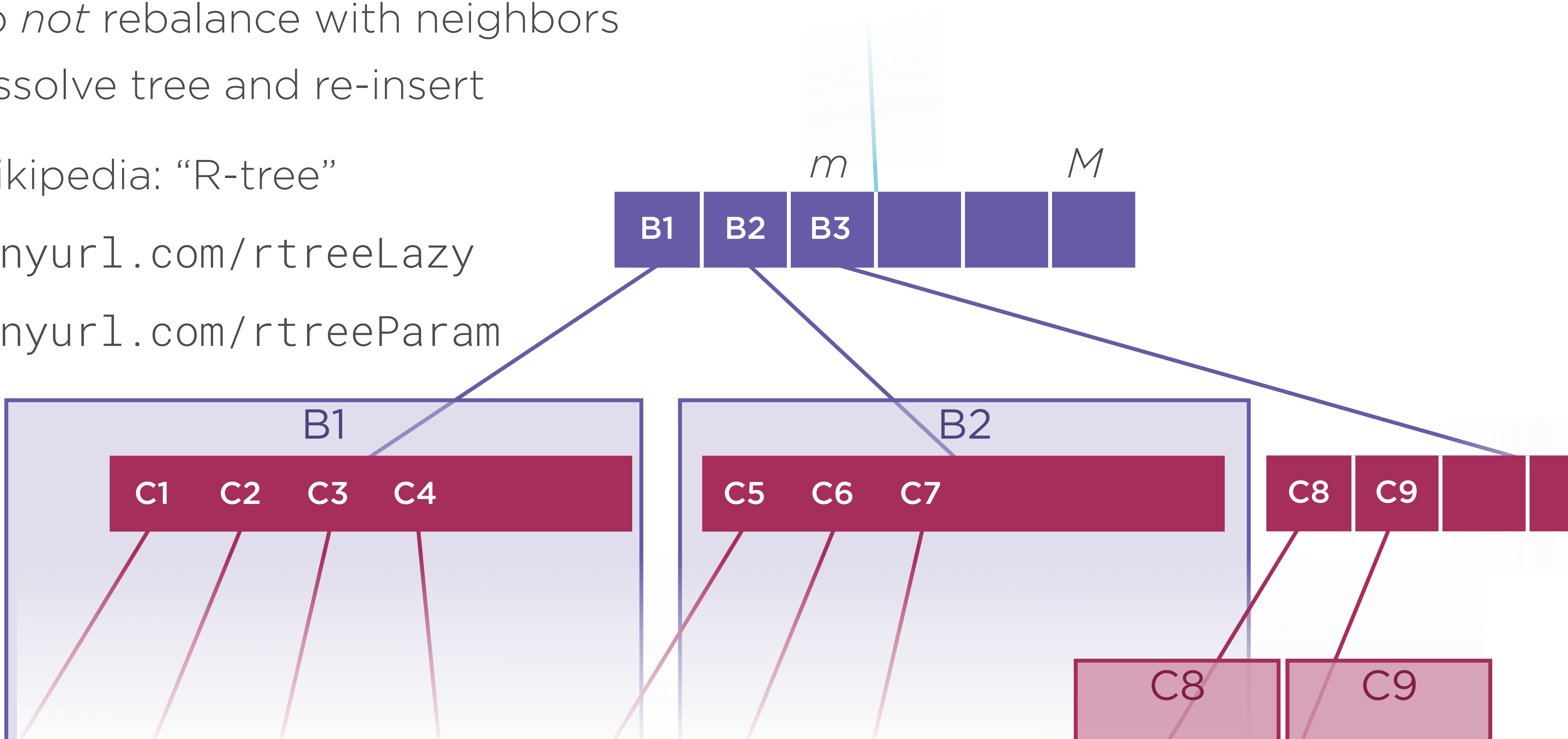
Do *not* rebalance with neighbors

Dissolve tree and re-insert

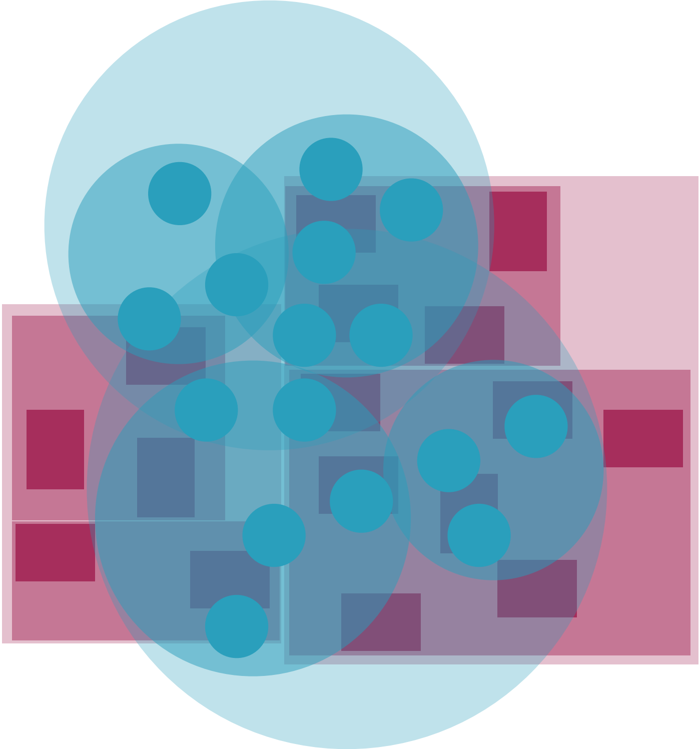
Wikipedia: “R-tree”

`tinyurl.com/rtreeLazy`

`tinyurl.com/rtreeParam`

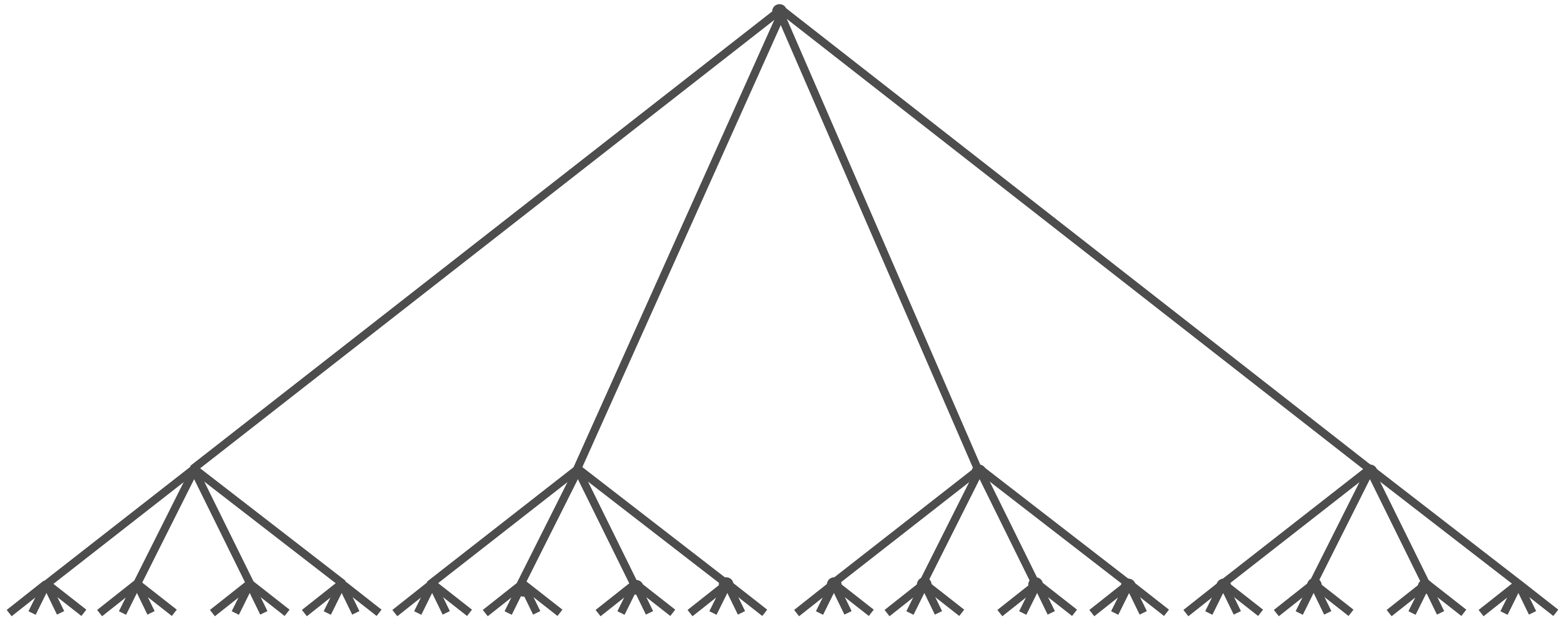


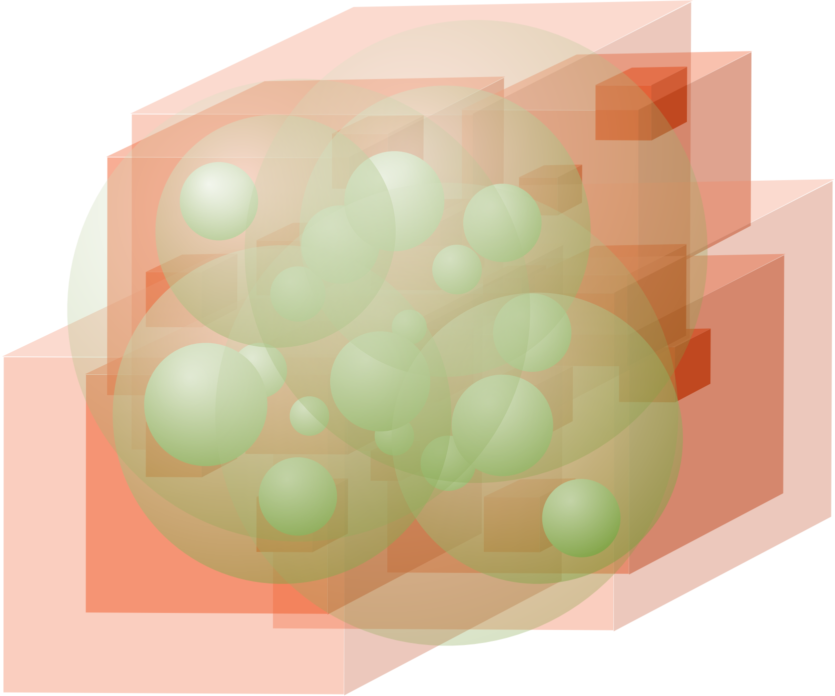
M-trees



Get all ~~rectangles~~ points within query ~~box~~ sphere

Get nearest ~~rectangle(s)~~ point(s) relative to query ~~box~~ sphere

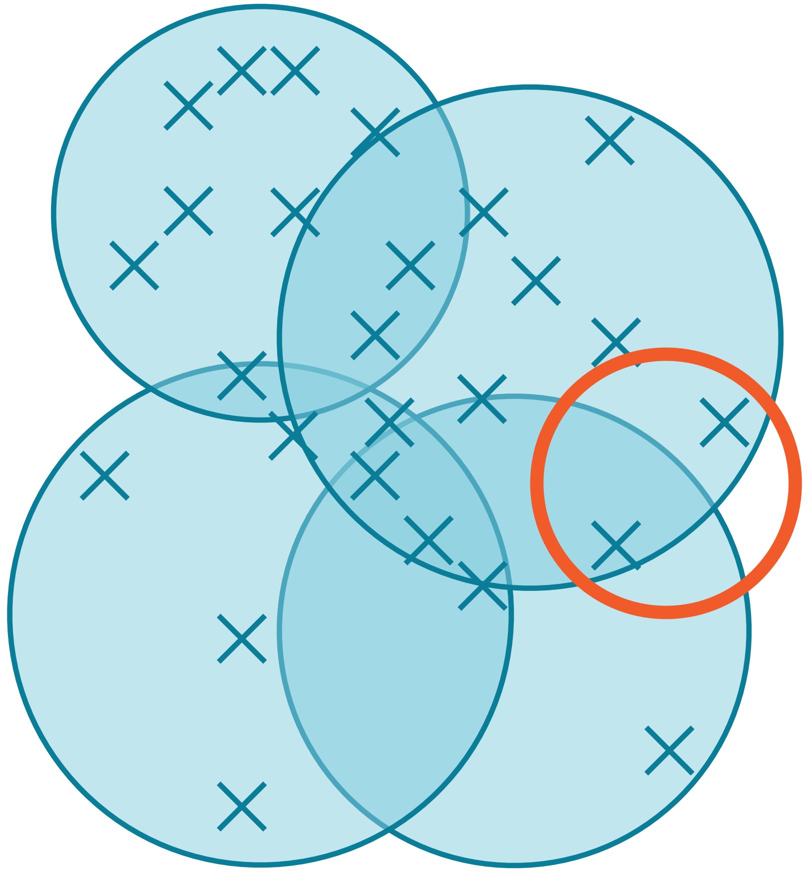




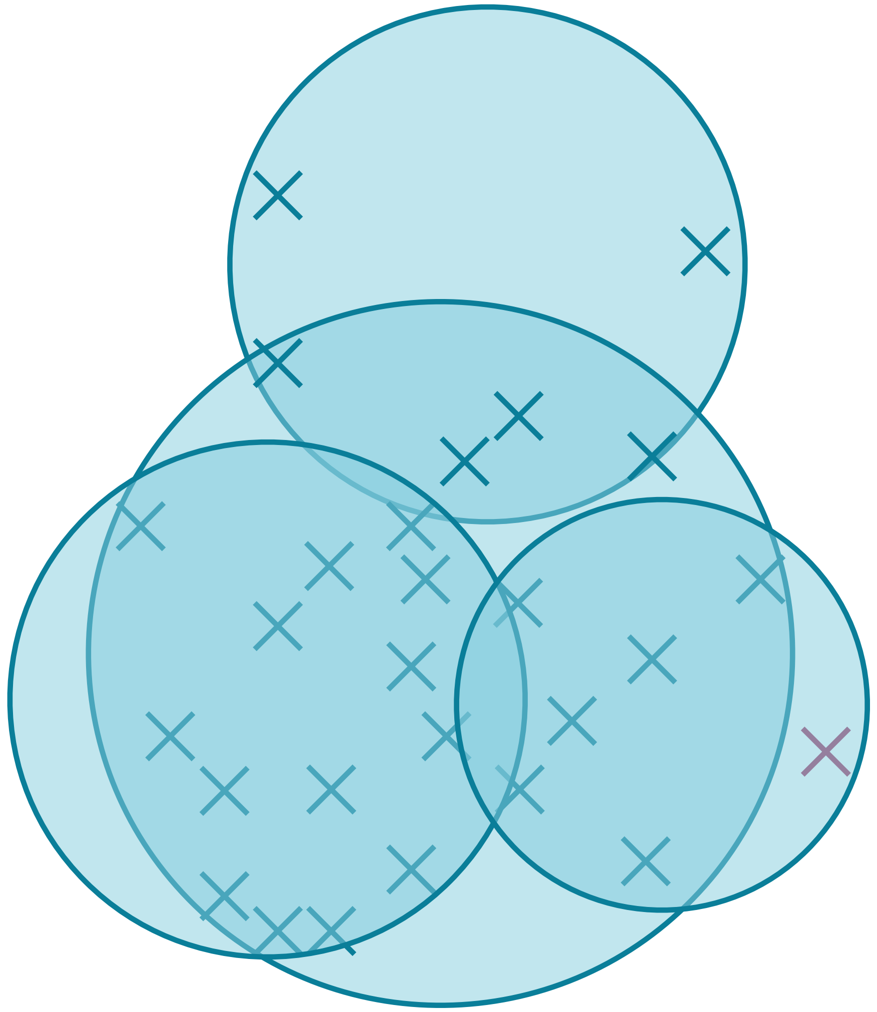
Create hierarchy of nested ~~rectangles~~ spheres

Organize nested ~~rectangles~~ spheres in B-tree

Lookup



Insertion



Short description: `tinyurl.com/mtreeShort`

Longer description: `tinyurl.com/mtreeLong`

Symmetric M-tree with deletion: `tinyurl.com/mtreeSym`

R-tree vs M-tree

What’s actually available?

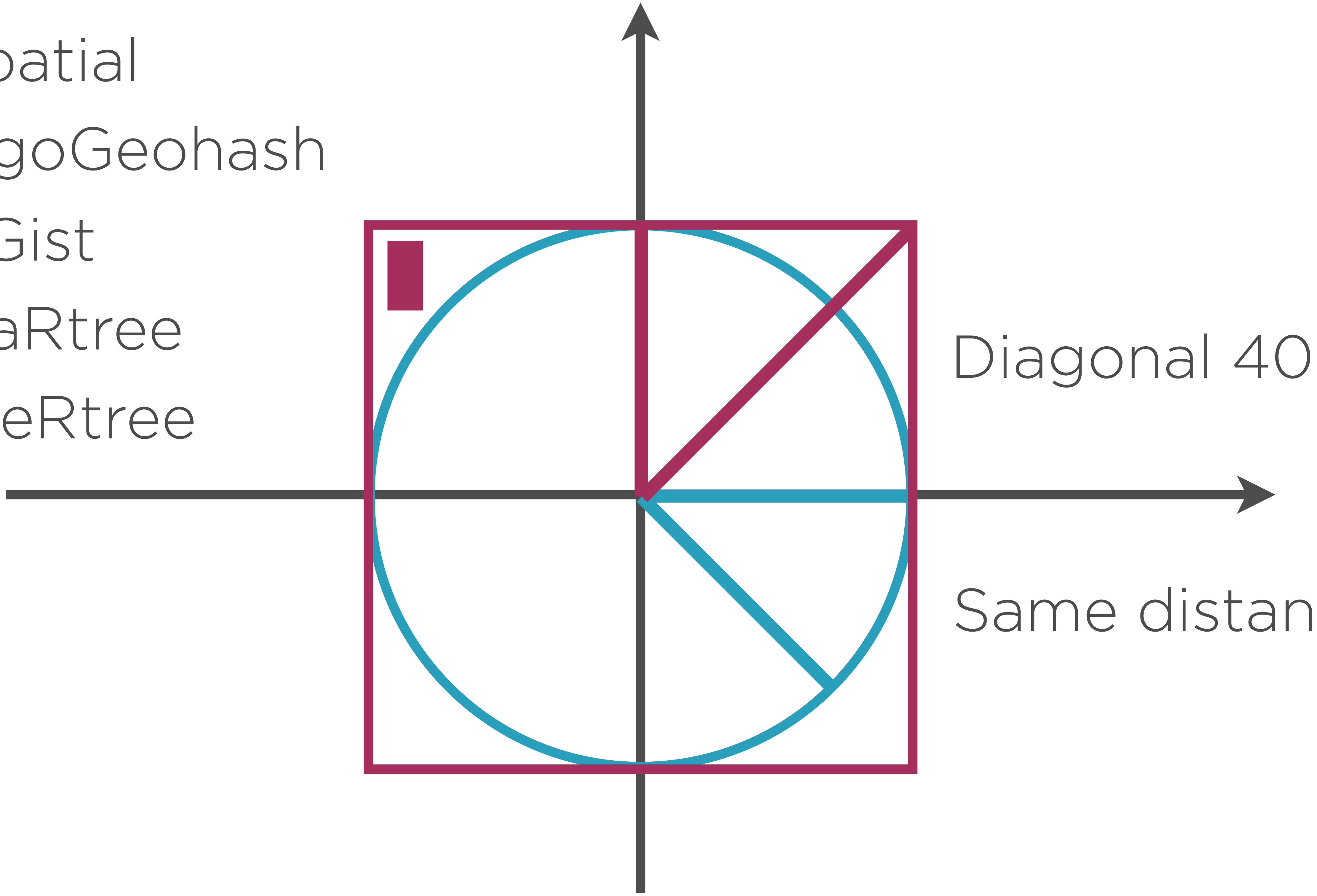
tinyurl.com/sqlSpatial

tinyurl.com/mongoGeohash

tinyurl.com/psqlGist

tinyurl.com/mariaRtree

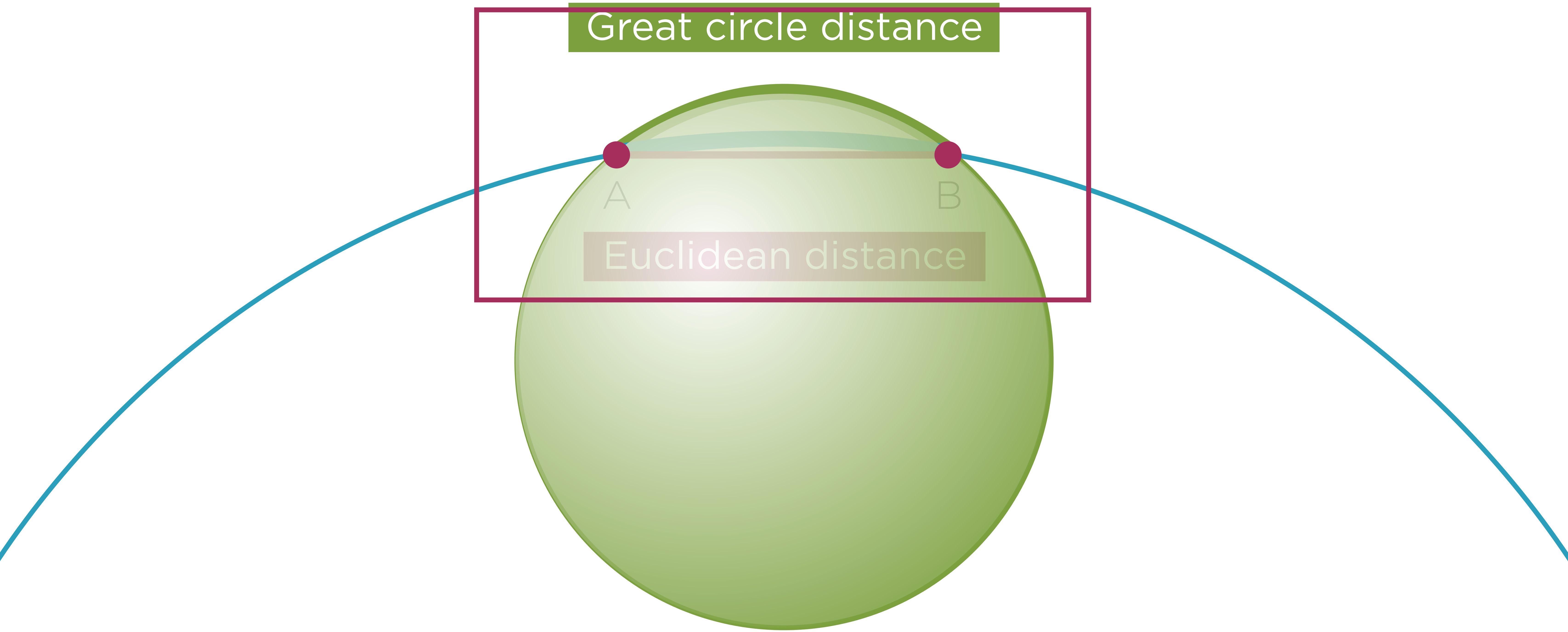
tinyurl.com/oracleRtree



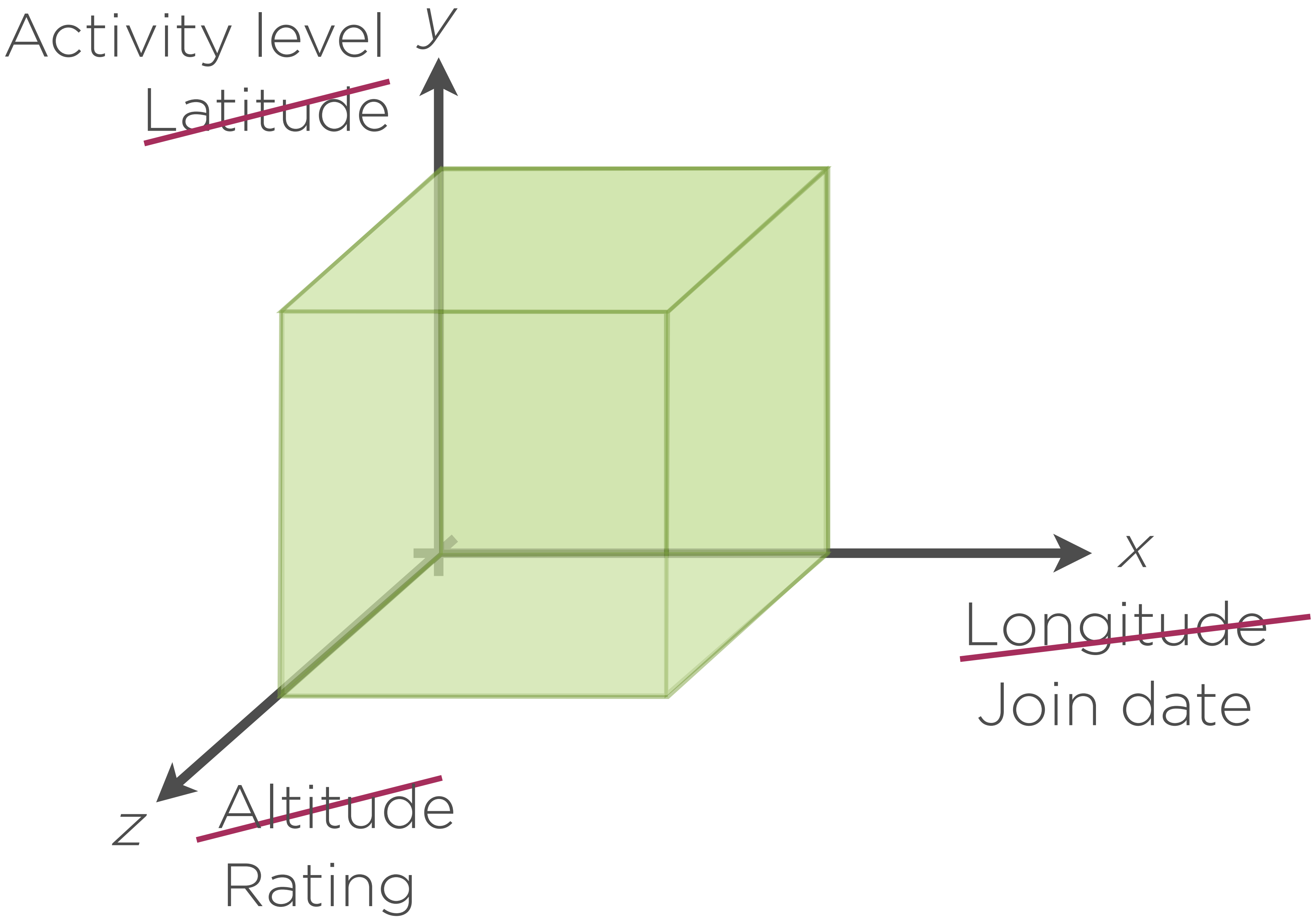
Diagonal 40% longer

Same distance

What is Distance?



What is Space?



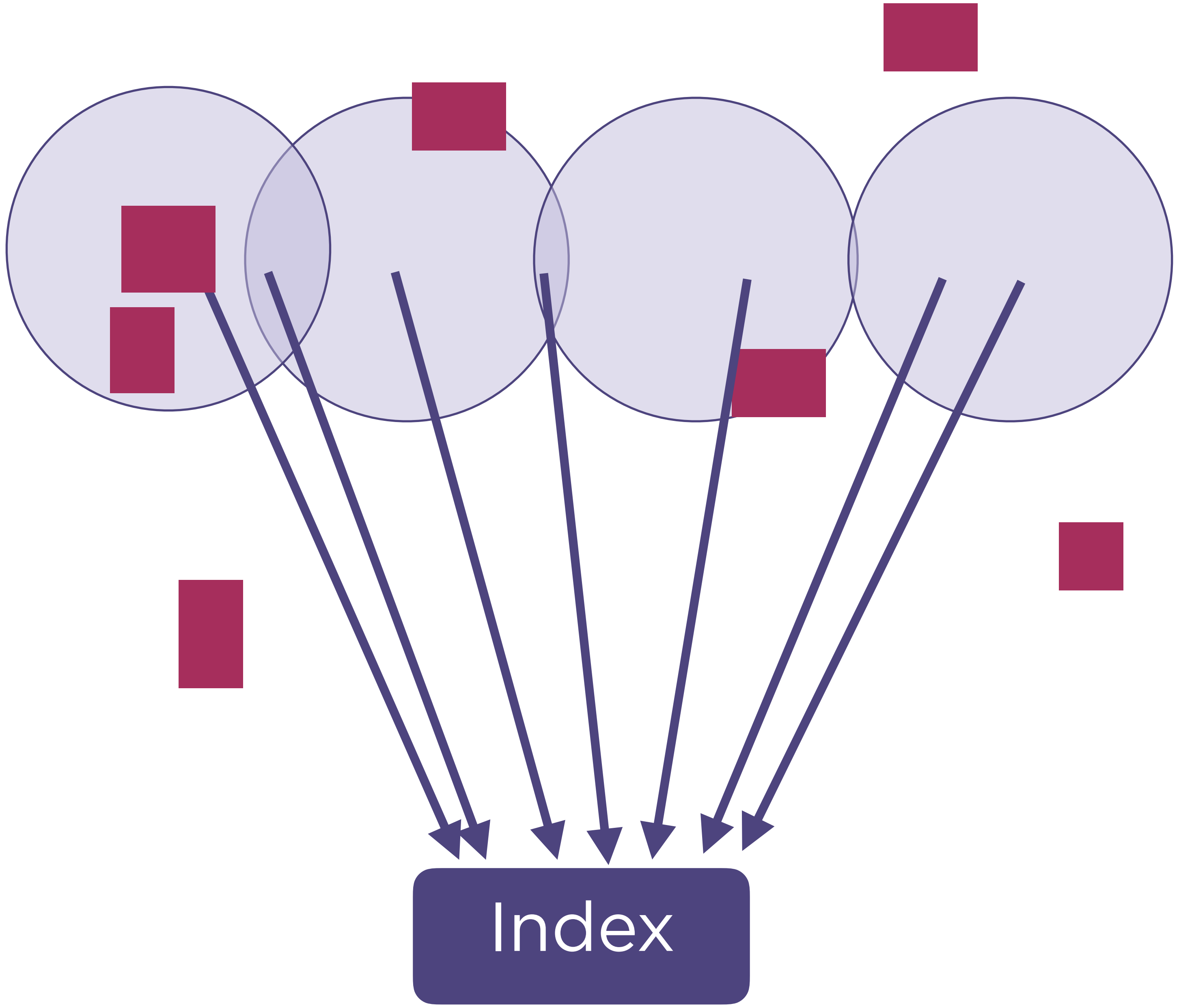
(x, y)

(x, y, z)

(x, y, z, w)

$(x, y, z, w, \ldots)$

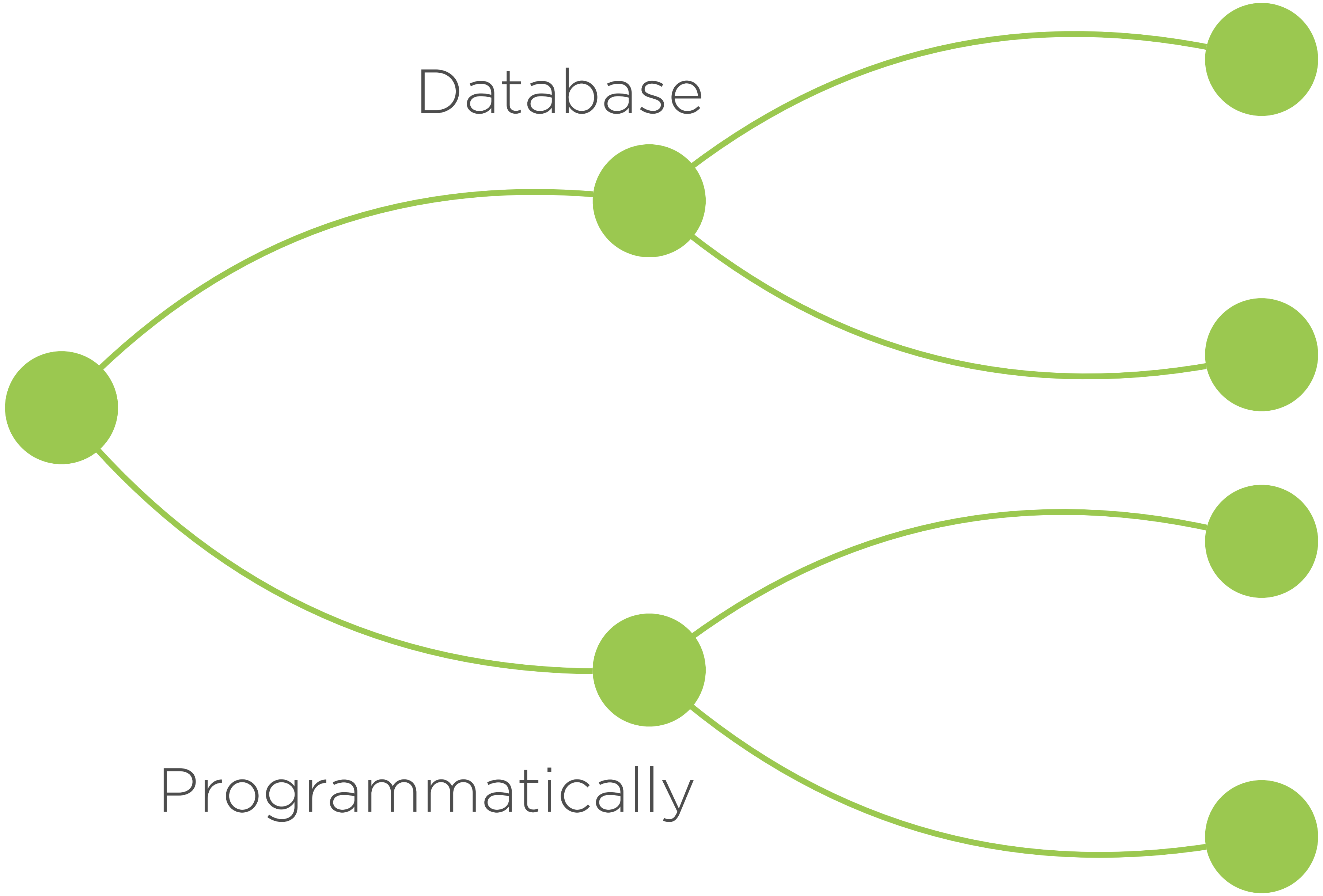
Moving Objects



Demo

Spatial search - Using an R-tree

Options



Database

Geohash or similar
E.g. MongoDB or SQL Server

Other, e.g. R-tree

Programmatically

M-tree
No NuGet package

R-tree
Only two NuGet packages
Decides on this approach for demo purposes

Lessons Learned

Geohashing
Recursive division of area
(mostly for points)

Basic structure: B-tree

Balanced

Disk friendly

R-tree
Hierarchy of rectangles

M-tree
Hierarchy of spheres

Splits bounding
box/sphere when full

Moving objects difficult

Position, time,
anything numerical