CSE 594: Spatial Data Science & Engineering

Lecture 6

Spatial Data Storing and Indexing

Part 2

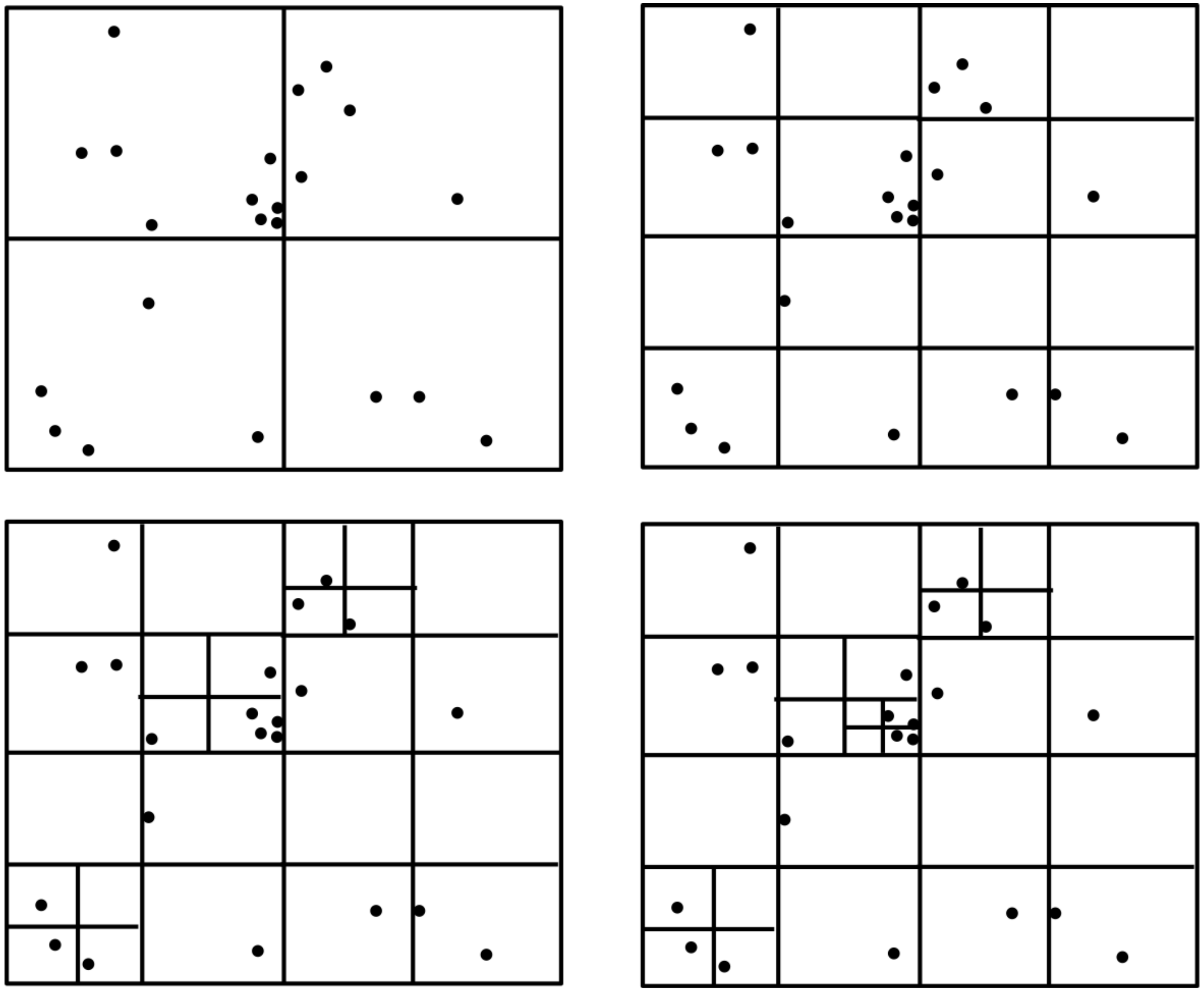
Quad Tree

- Based on the scheme of dividing a space into grids
- Recursively divide space into quadrants until each quadrant contains a maximum number of elements
- Maximum number of elements is a parameter and depends on disk page size

Constructing Quad Tree

1. Start with the entire space and check if there are more than m elements
2. If more than m elements, divide the space into 4 equal quadrants
3. A root node is created and one leaf node per quadrant is attached
4. Points in each quadrant are recorded in the corresponding leaf node
5. Repeat steps 1-4 recursively for each leaf node

Constructing Quad Tree



Quad Tree

Advantages

- Makes the grid data structure adaptive to the distribution of data, much like a k-d tree
- Unlike the basic Kd tree, a node can hold multiple data elements
- Does not need to be reconstructed

Disadvantages

- Contain many empty leaves
- Parameter fine tuning is complex. Performance depends on the appropriate parameter value.
- Rarely used as a disk-based index structure, because internal nodes are much larger than a disk page size. Some variants exist to pack internal nodes into a disk page

Quad Tree

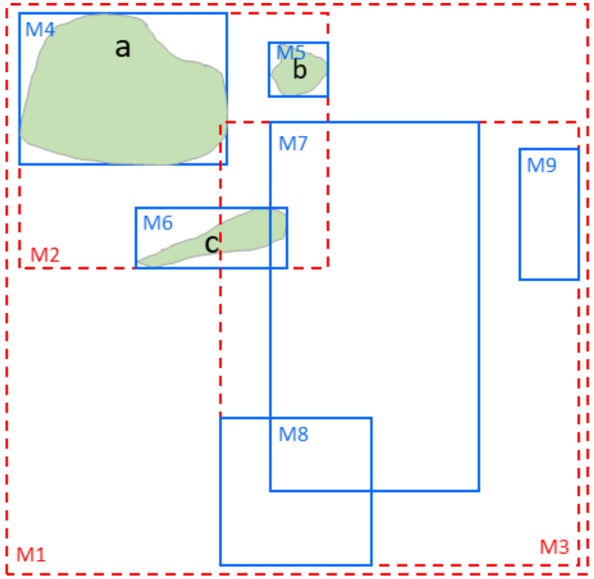
- How to insert a new node into a Quad tree?
- How to delete a node from a quad tree?
- How to perform a range search with quad tree?
- How to perform a KNN search with quad tree?

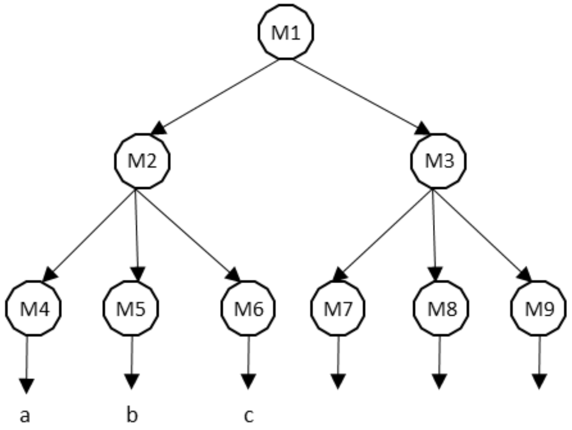
R-Tree

- R-tree is a generalization of B-tree to handle two or multi-dimensional data
- B-trees rely on a total ordering or integers that represent the physical nearness. Two points that are adjacent in ordering may not physically near to each other
- Physically near objects are grouped together in nodes
- Like a B-tree, objects are stored inly at leaf. Internal nodes provide a search path to the leaves.
- Unlike b-trees, can handle multi-dimensional data

R-Tree

- R-trees are balanced, have a fixed node size
- Maximum number of objects allowed in a node is predefined
- Internal nodes contain minimum bounding rectangles (MBR). An MBR minimally encloses all spatial objects in a node
- Try to minimize the overlap between MBRs
- When a node is full, it is split, objects are assigned to new nodes, and MBRs of each node are adjusted

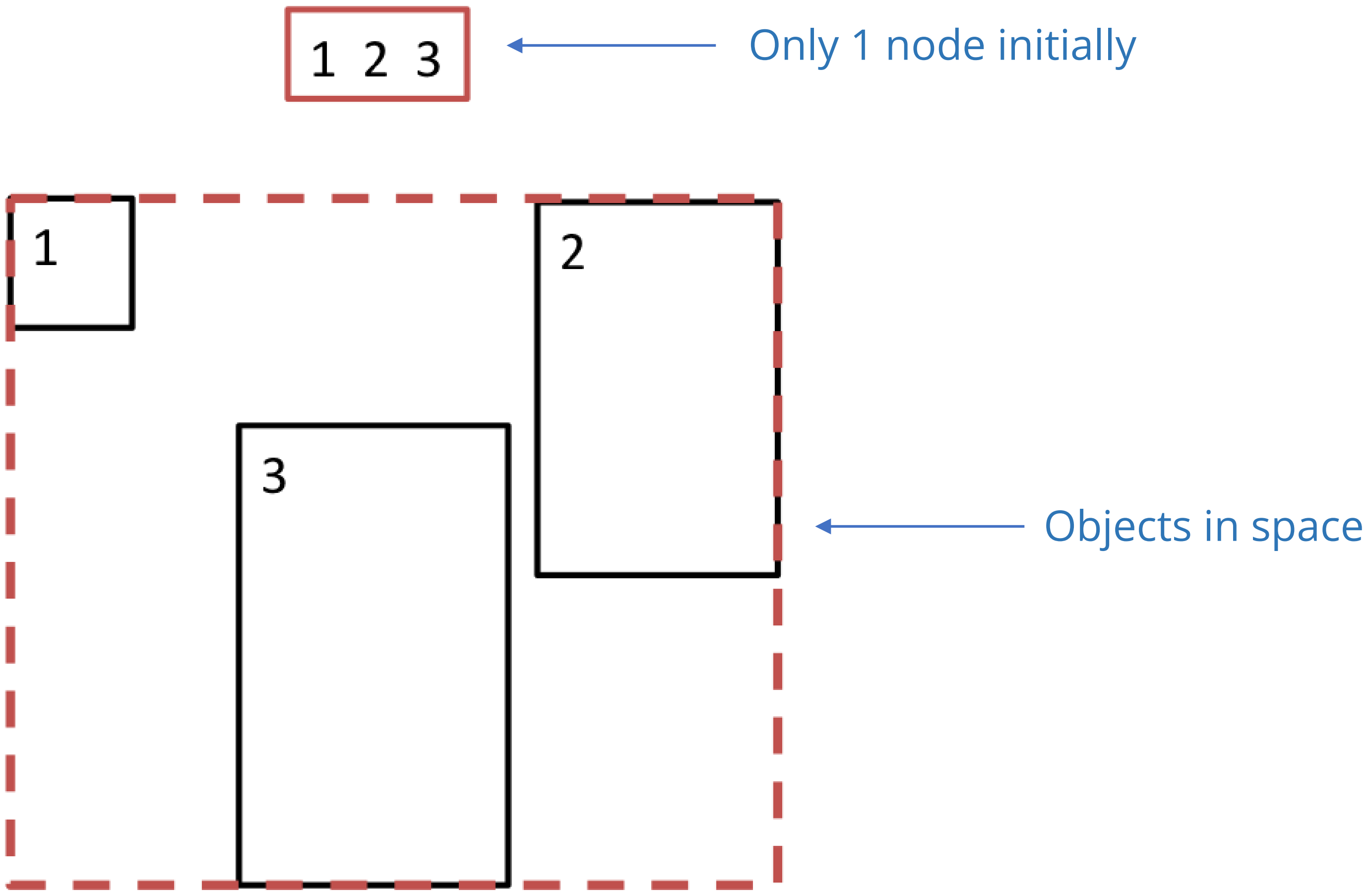




Images Source: https://gistbok.ucgis.org/bok-topics/spatial-indexing

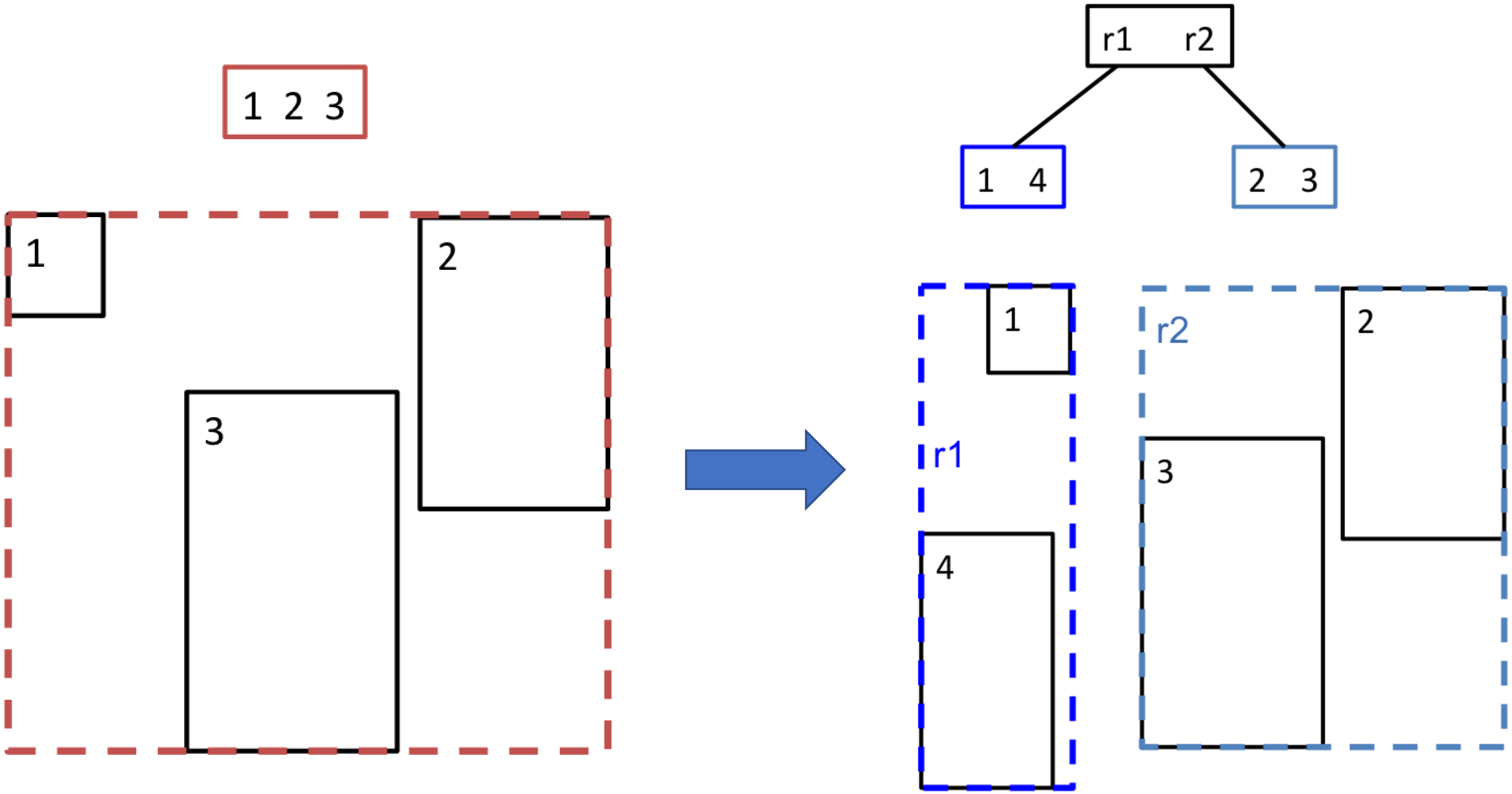
Construction of R-Tree

Construct an R-tree with the given objects in the space. Assume a node can hold at most 3 objects



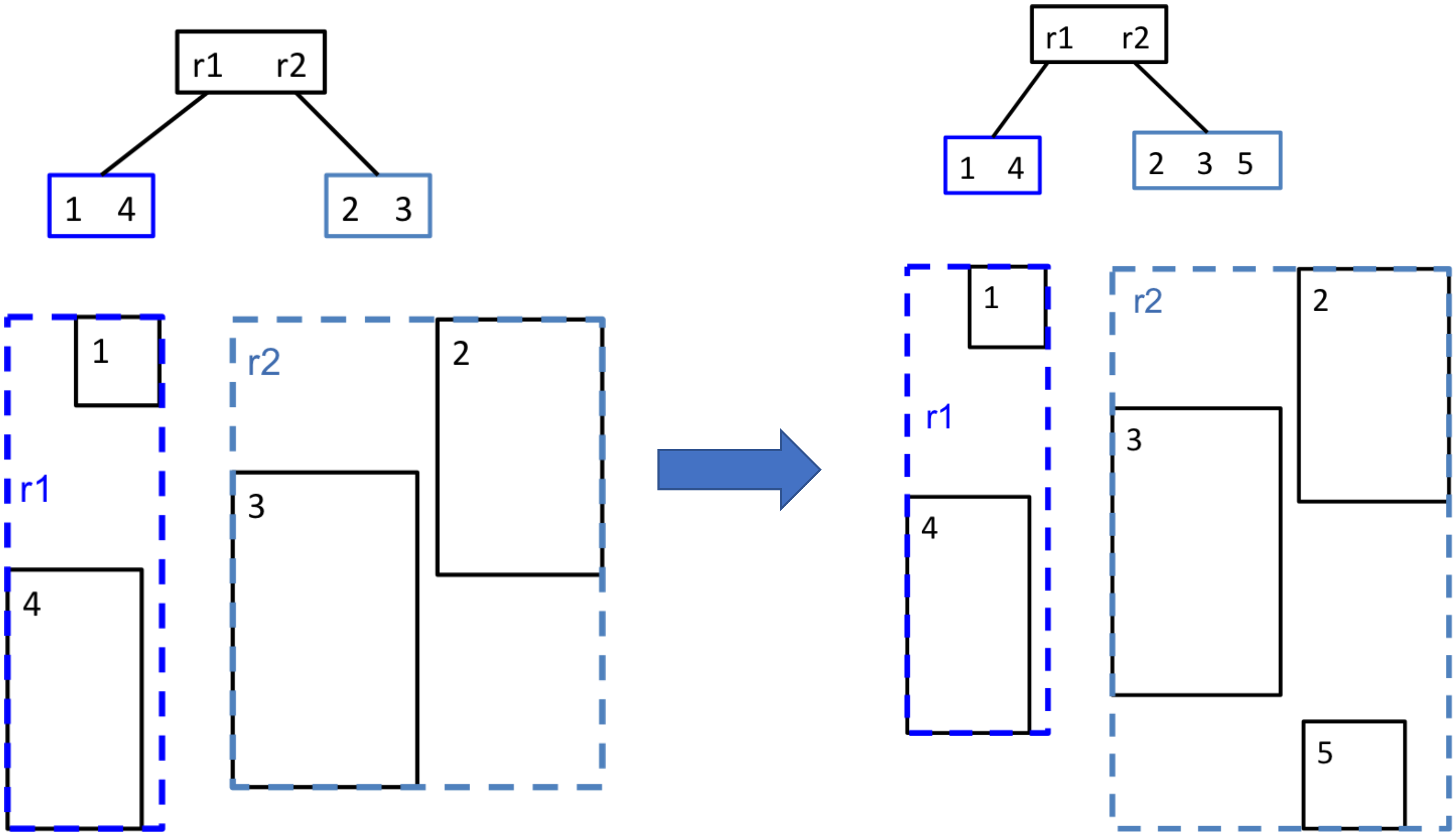
Construction of R-Tree

Add the fourth region. Requires a split



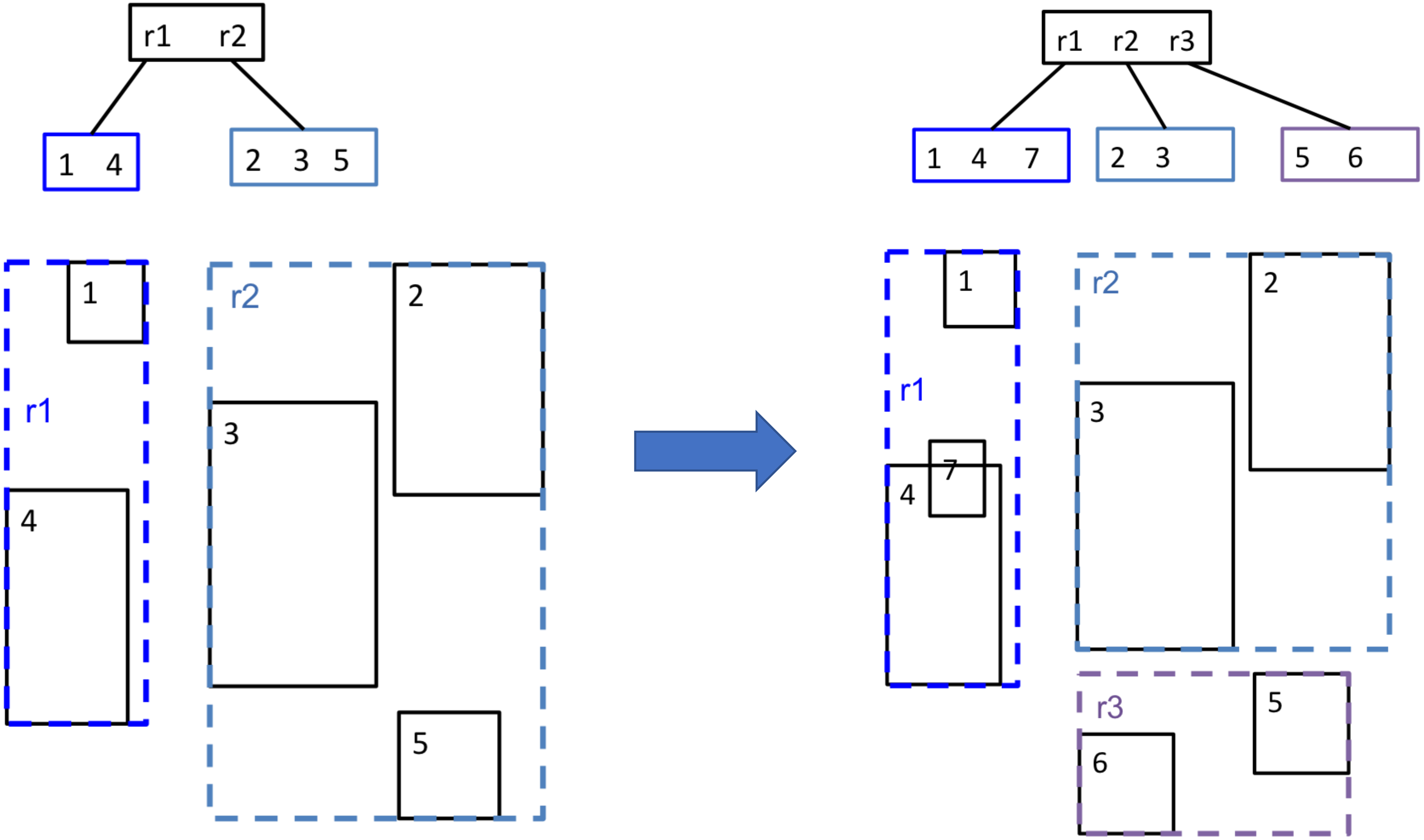
Construction of R-Tree

Add the fifth region. When extending a bounding rectangle, try for minimum expansion



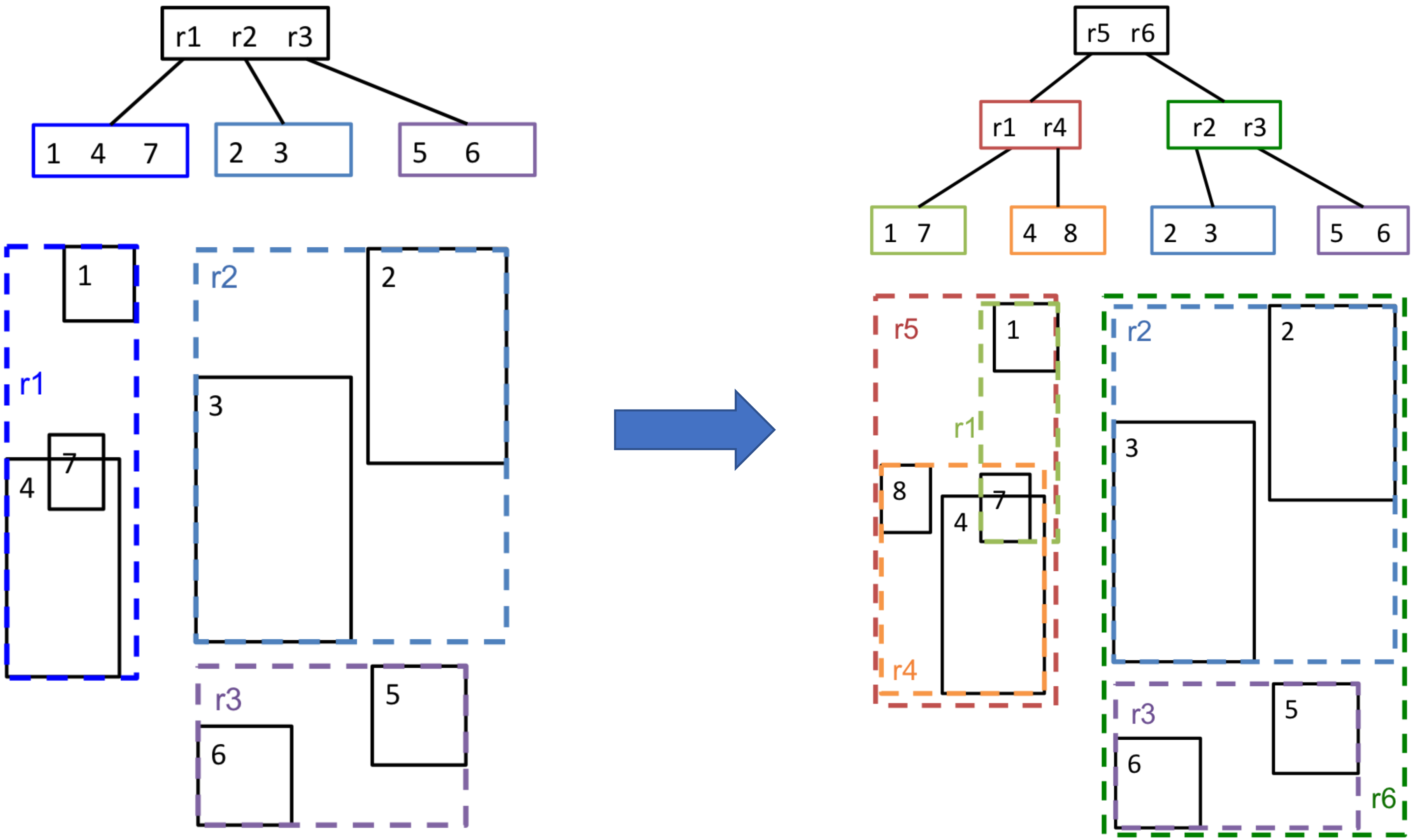
Construction of R-Tree

Adding the sixth region needs a split. Adding 7th region does not require split.



Construction of R-Tree

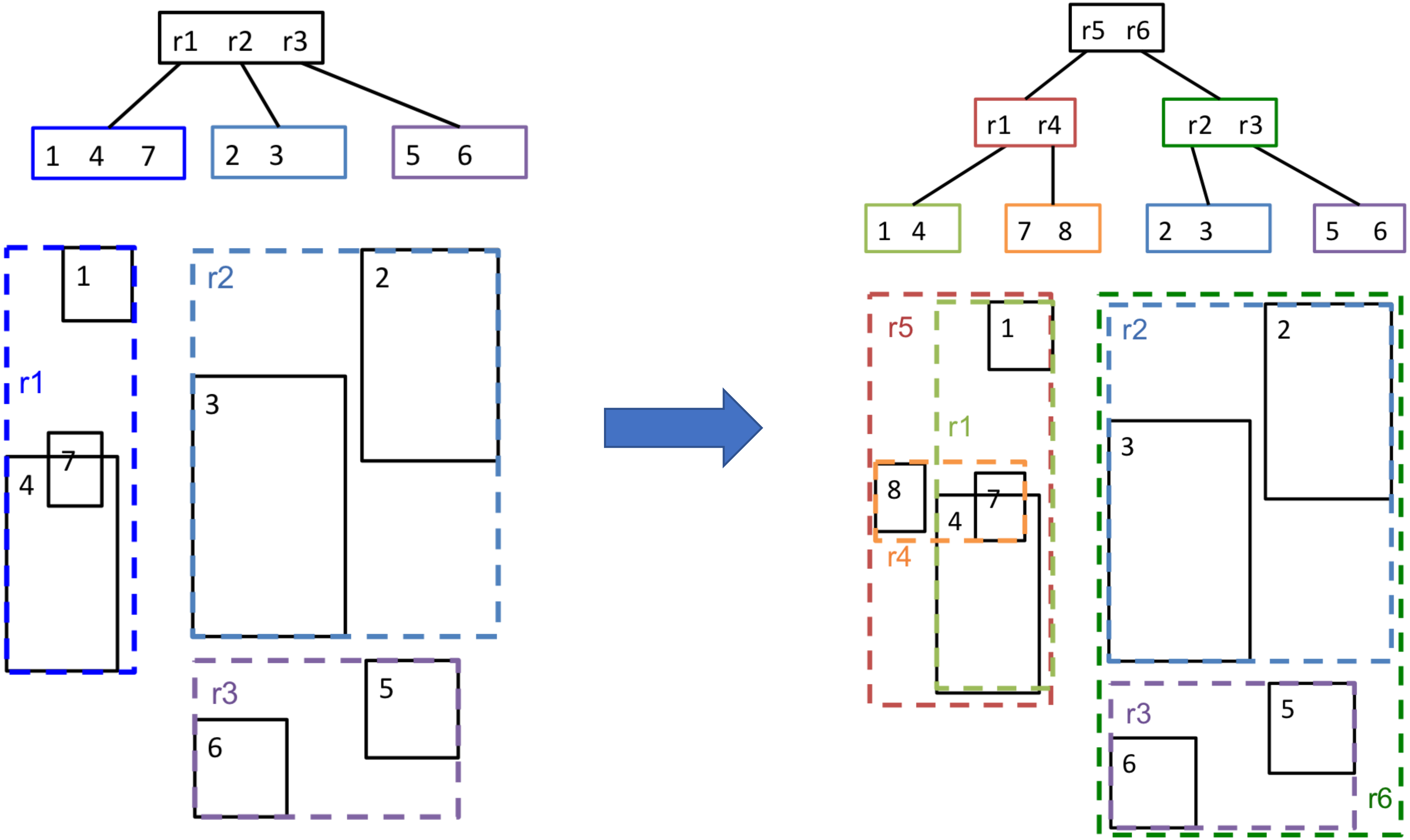
Adding 8th object forces a split, all the way up to the root.



Is there an alternate better choice?

Construction of R-Tree

Bounding rectangles can be made better.



R-Tree

Advantages

- Adapts to distribution of spatial objects
- Aligns well with disk architecture
- Handles all types of spatial objects well, and generalizes to higher dimensions
- Bulk loading is better than inserting objects one by one

Disadvantages

- Performance can degrade if frequent inserts or deletes are made
- Bounding rectangles overlap, causing the search operation to explore more nodes
- Bounding rectangles may contain too much empty space

R-Tree

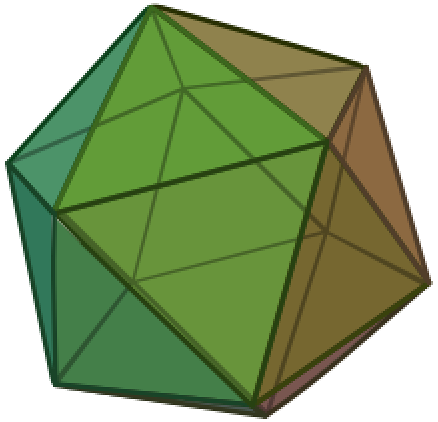
- How to delete a node from a R-tree?
- How to perform a range search with R-tree?
- How to perform a KNN search with R-tree?

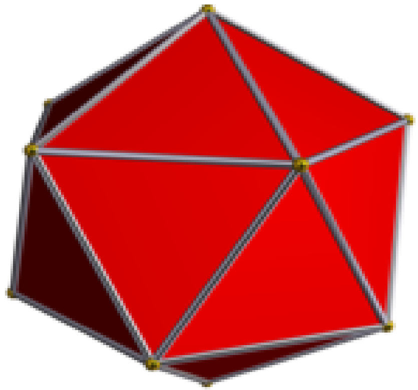
H3 Index

- An open-source geospatial analysis tool that provides a hexagonal, hierarchical spatial index
- The entire space is divided into a grid of hexagons
- Combines the benefits of a hexagonal global grid system with a hierarchical indexing system
- Uses gnomonic map projections centered on icosahedron faces
- An icosahedron-based map projection results in twenty separate two dimensional planes instead of a single plane

Icosahedron

- A polyhedron with twenty faces
- A regular icosahedron has 30 edges and 20 equilateral triangle faces
- A total of 12 vertices, each vertex is a meeting point of 5 faces
- The properties of an isohedron define some properties of the H3 index

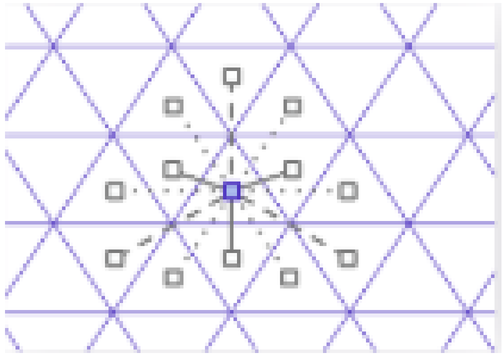




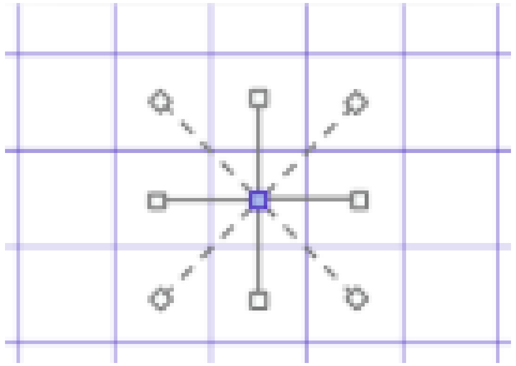
Images Source: Wikipedia

Why Hexagonal Cells?

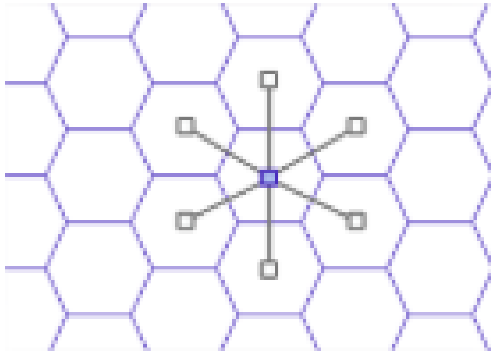
Triangular Cells



Square Cells



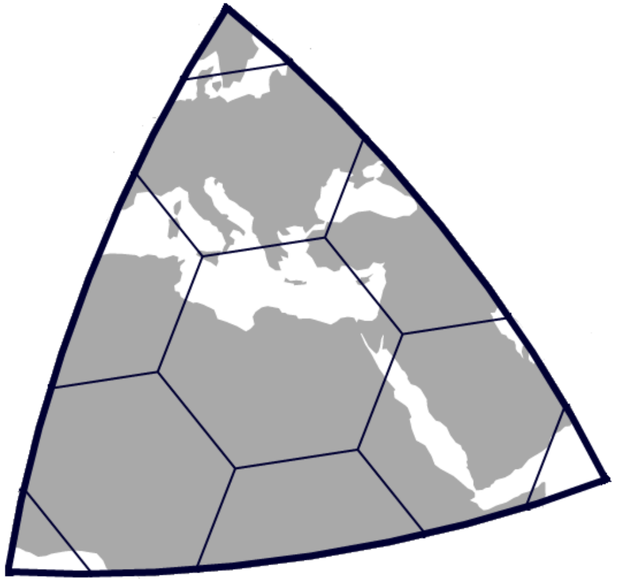
Hexagonal Cells



- Neighbors of a square cells have two types of center-to-center distance
- Triangles have three different distances between neighbors
- Hexagons have only one distance between centers of neighboring cells, simplifying performing analysis and reducing number of parameters

H3 Indexing

- The H3 index divides the grid into hexagons and pentagons in 16 different hierarchical resolutions
- The most coarse-grained base resolution consists of 122 cells, some cells are contained by multiple faces
- Not possible to cover only with hexagons, there are exactly 12 pentagons at each resolution
- Pentagons are located at icosahedron vertices
- Each finer resolution has cells with one seventh the area of the coarser resolution



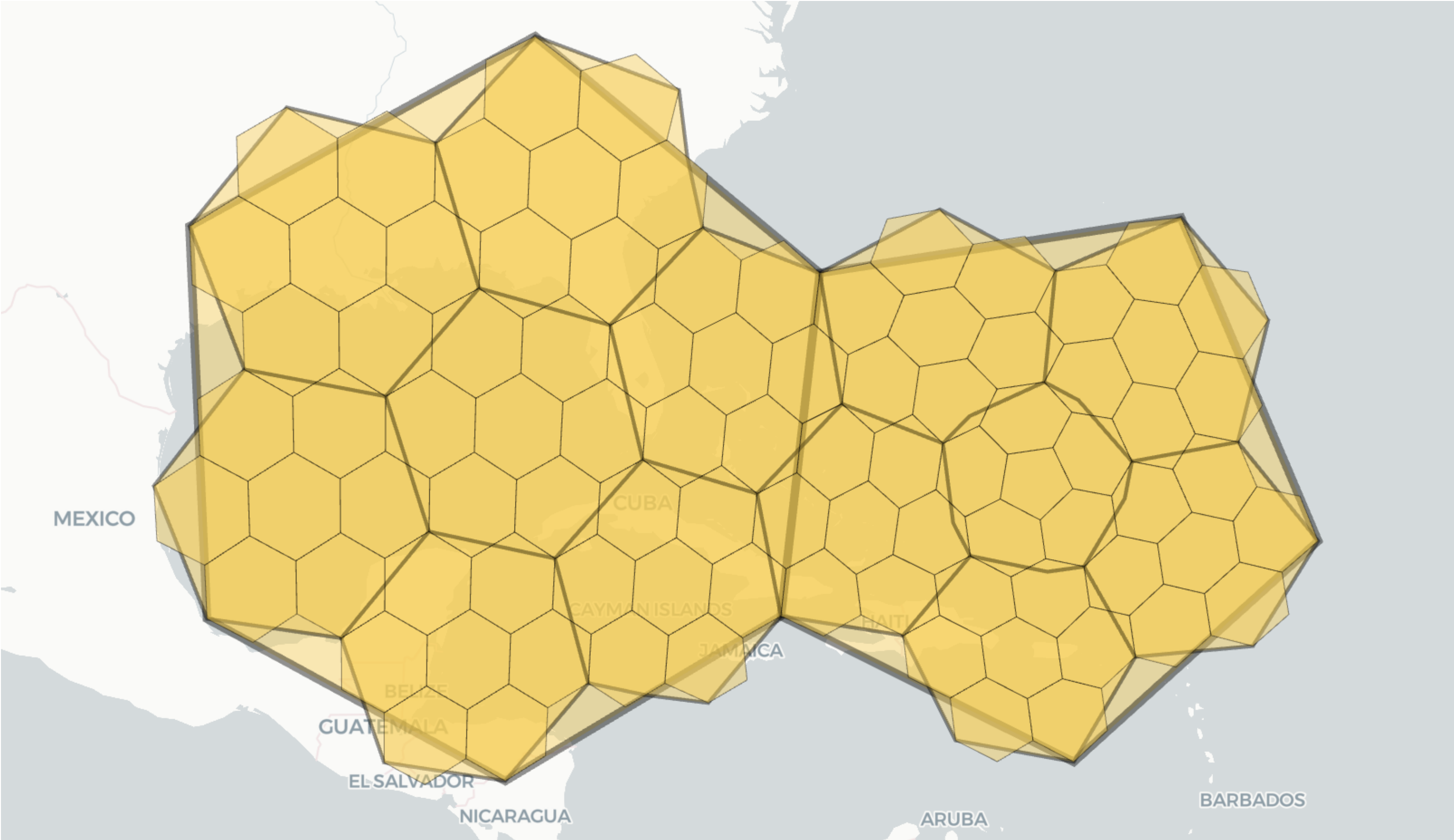
H3 Index Cell Statistics

Res	Total number of cells	Number of hexagons	Number of pentagons
0	122	110	12
1	842	830	12
2	5,882	5,870	12
3	41,162	41,150	12
4	288,122	288,110	12
5	2,016,842	2,016,830	12
6	14,117,882	14,117,870	12
7	98,825,162	98,825,150	12
8	691,776,122	691,776,110	12
9	4,842,432,842	4,842,432,830	12
10	33,897,029,882	33,897,029,870	12
11	237,279,209,162	237,279,209,150	12
12	1,660,954,464,122	1,660,954,464,110	12
13	11,626,681,248,842	11,626,681,248,830	12
14	81,386,768,741,882	81,386,768,741,870	12
15	569,707,381,193,162	569,707,381,193,150	12

$$Count(r) = 2 + 120*7^{r}$$

Children of Hexagon and Pentagon Cells

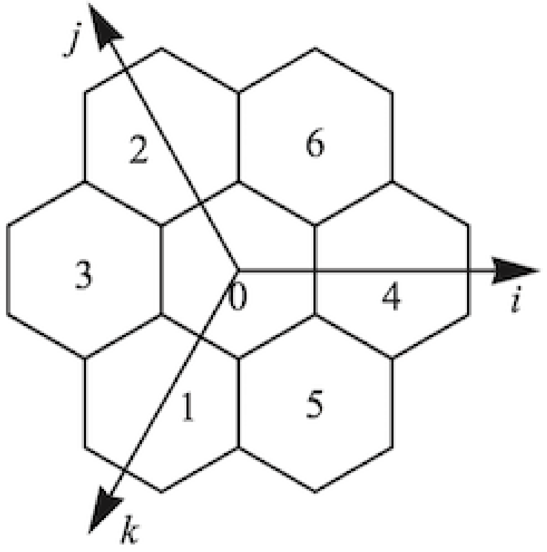
- Hexagons have 7 hexagon children
- Pentagons have 6 children: 5 hexagons and 1 pentagon



Images Source: https://h3geo.org/docs/core-library/restable/

H3 Cell Index

- The H3 index of a resolution r cell begins with a cell number for base resolution (resolution 0)
- Base cell number is followed by a sequence of r digits 0-6, each indicating a child in the coarser resolution
- For pentagon cells with 6 children, index hierarchy produced by sub-digit 1 is removed at all resolutions



octal	0	1	2	3	4	5	6
ijk+	(0, 0, 0)	(0, 0, 1)	(0, 1, 0)	(0, 1, 1)	(1, 0, 0)	(1, 0, 1)	(1, 1, 0)
binary	000	001	010	011	100	101	110

H3 Cell Index Representation

- An H3 cell index represents a cell in the H3 grid system at a particular resolution
- Consists of 64-bit integer in order, highest bit first, as follows:
  - ➢ 1 bit reserved and set to 0
  - ➢ 4 bits to indicate the index mode (invalid index, cell index, directed edge index, vertex index). Cell index is indicated by mode 1
  - ➢ 3 bits reserved and set to 0
  - ➢ 4 bits to indicate the cell resolution 0-15
  - ➢ 7 bits to indicate the base cell 0-121
  - ➢ 3 bits to indicate each subsequent digit 0-6 from resolution 1 to resolution r (45 bits reserved for resolutions 1-15)
  - ➢ The tree bits for each unused digit are set to 7

H3 Index

Advantages

- Convenient for modeling because neighbors are equidistant
- Can handle a variety of spatial properties and types, such as raster and vector data, as well as GPS points

Disadvantages

- Not convenient when equal area property is desired
- The gnomonic projection used in H3 sacrifices accuracy in calculations
- Hexagonal hierarchy produces spatial error when divided into smaller units

H3 Index

- How to find the index of a query point?
- How to perform a range search with H3 index?
- How to perform a KNN search with H3 index?