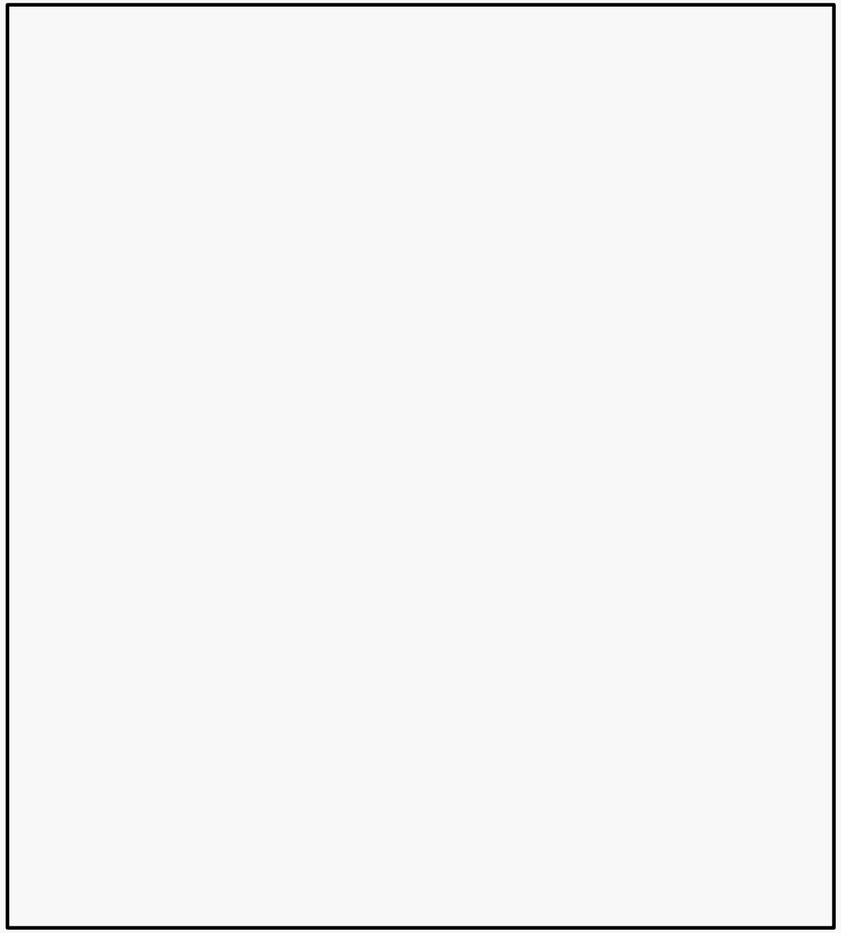
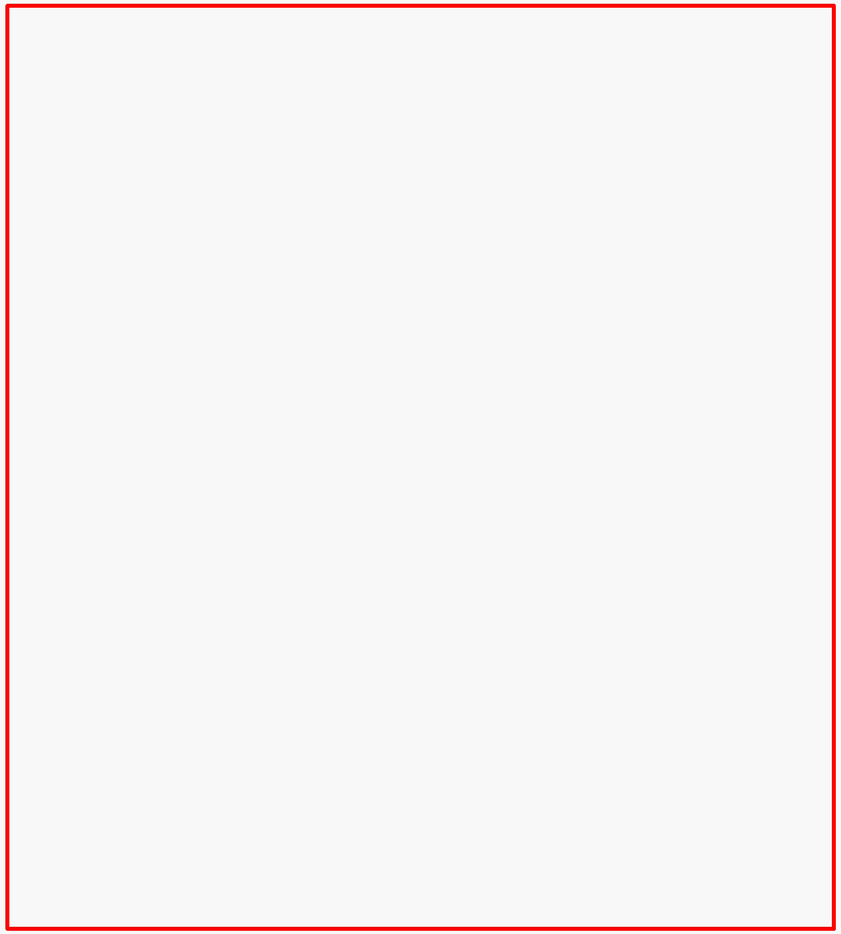
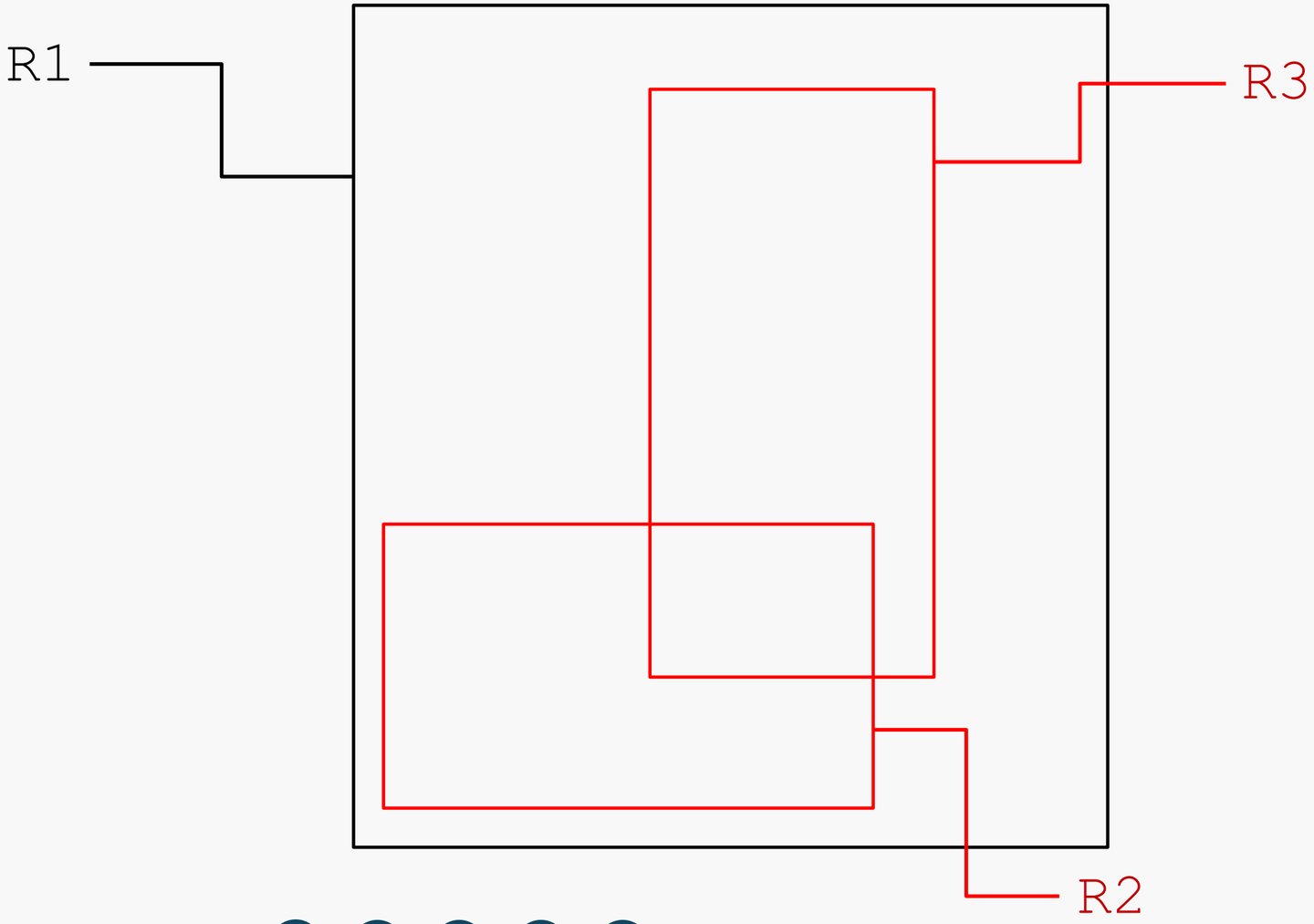
R-TREE



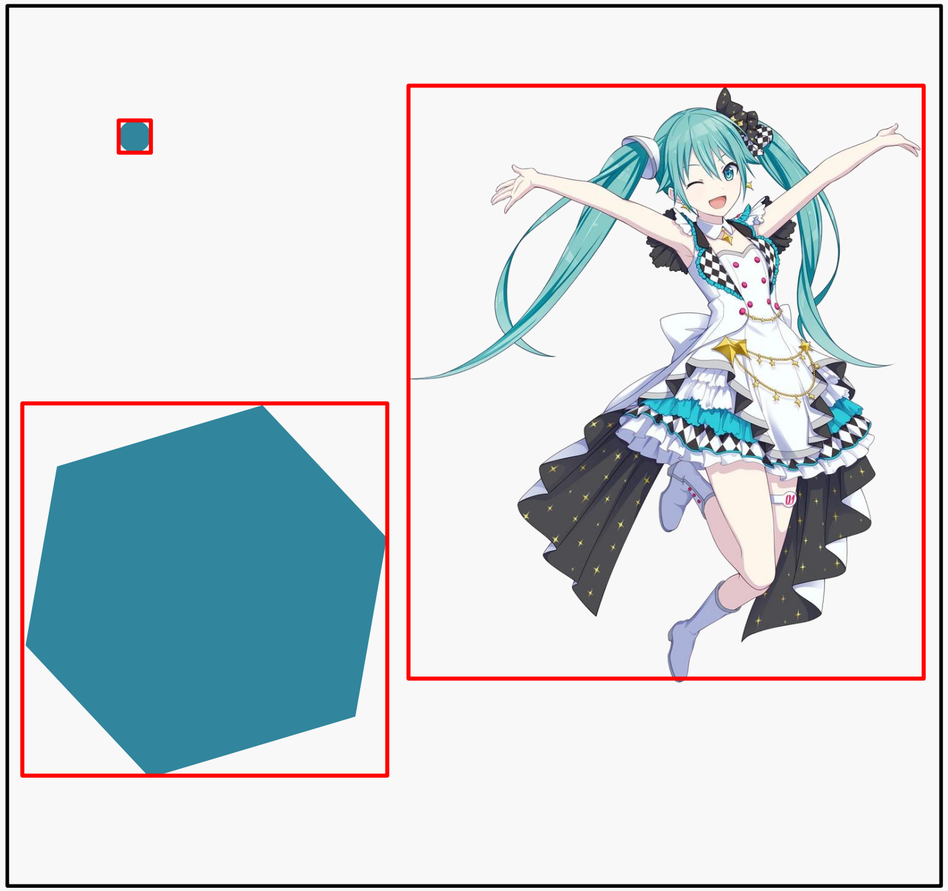
R-TREE



R-TREE



I. Definitions



R-Tree is a tree-like spatial data structures catering to spatial indexing.

- Coordinates
- Polygons

Assume we maintain a database of aircrafts…

- Which aircrafts crossed the 3D region R at time instance t_x?
- Which aircrafts crossed the 3D region R between the time interval $[t_{start}, t_{end}]$?
- Which were the k nearest neighbors of aircraft a at time instance t_x?
- Given the current motion characteristics of the aircrafts, predict the nearest neighbor of aircraft a at some time instance t_x in the near future.
- Which aircrafts will cross the 3D region R between $[t_{start}, t_{end}]$?
- Determine the pairs of aircrafts (a_i, a_j) such that their distance at time instance t_x was less than 5km.

History

Assume we maintain a database of aircrafts…

l the 3D region R at time instance t_x?

l the 3D region R between the time

est neighbors of aircraft a at time

Computer Scientist Antonin Guttman

on characteristics of the aircrafts,

ghbor of aircraft a at some time

future.

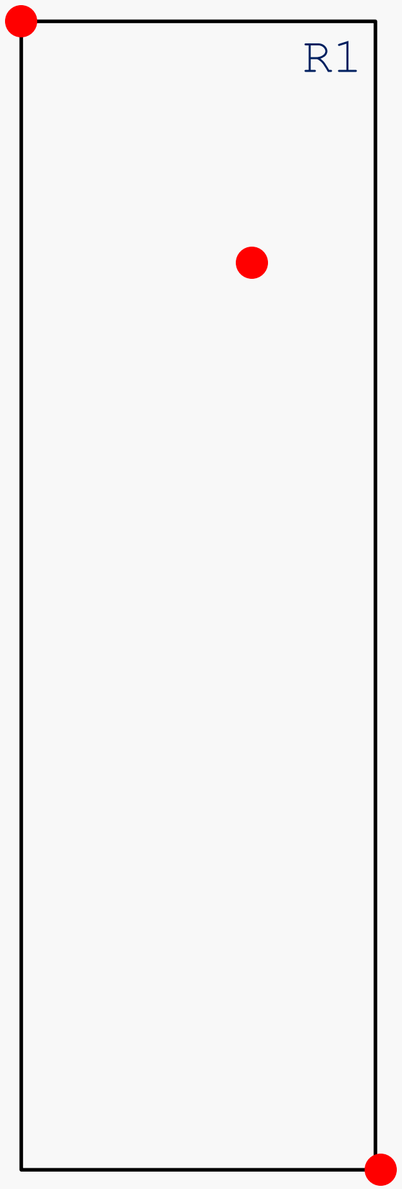
R-Trees: A dynamic index structure for spatial searching (1984)

oss the 3D region R between

aircrafts (a_i, a_j) such that their

ce t_x was less than 5km.

II. Structure and algorithms



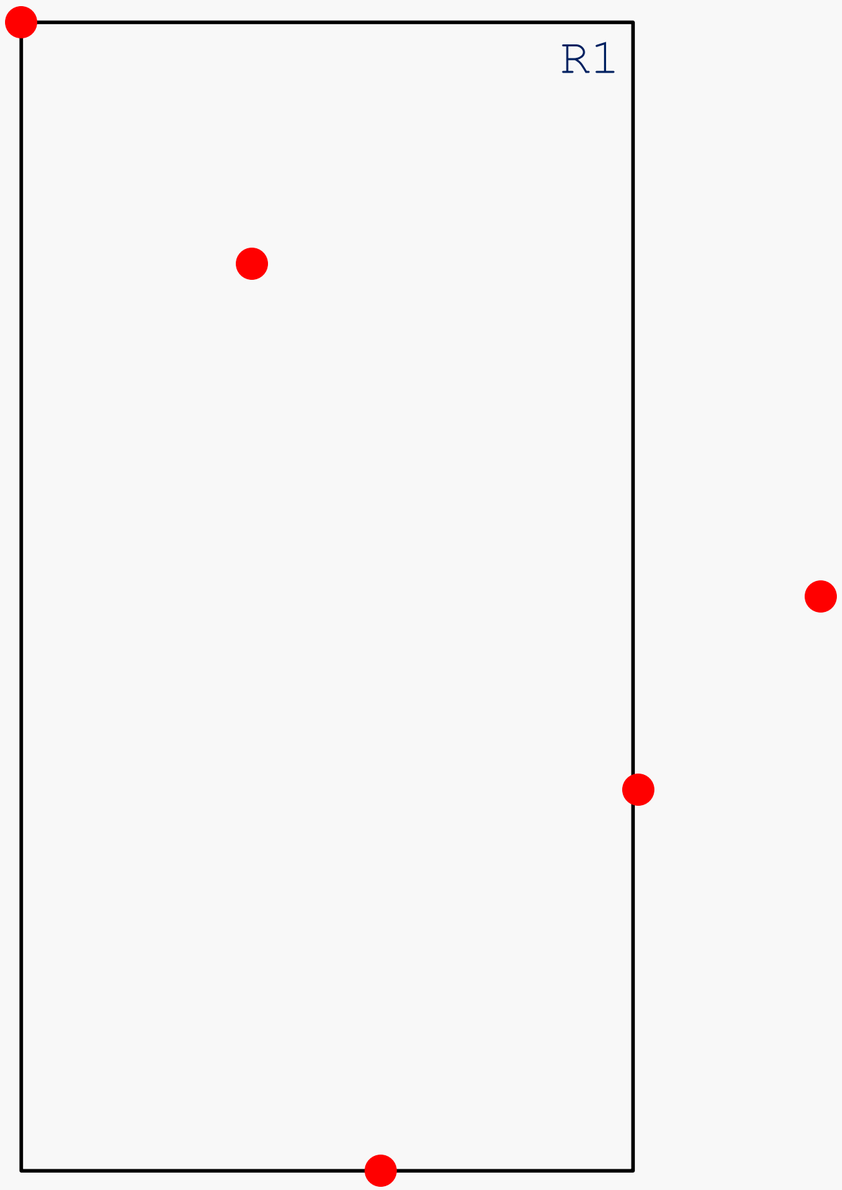
- Height-balanced.
- Nodes contain entries.
- Entries contain :

+ MBR containing its children if internal node, MBR containing object if leaf node.

+ Pointer to its children if internal node, pointer to object if leaf node.

R1

II. Structure and algorithms



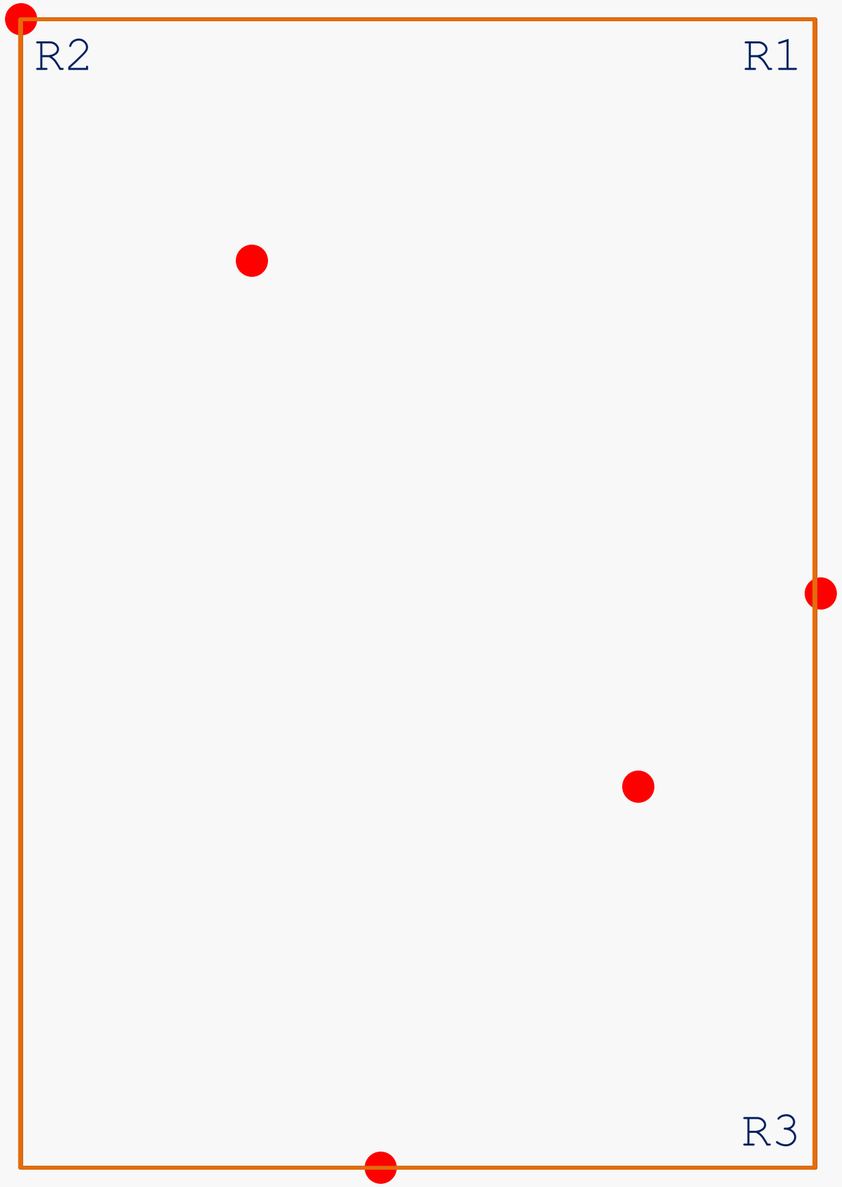
- Height-balanced.
- Nodes contain entries.
- Entries contain :

  + MBR containing its children if internal node, MBR containing object if leaf node.

  + Pointer to its children if internal node, pointer to object if leaf node.

R1

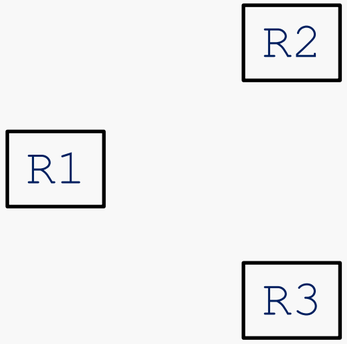
II. Structure and algorithms



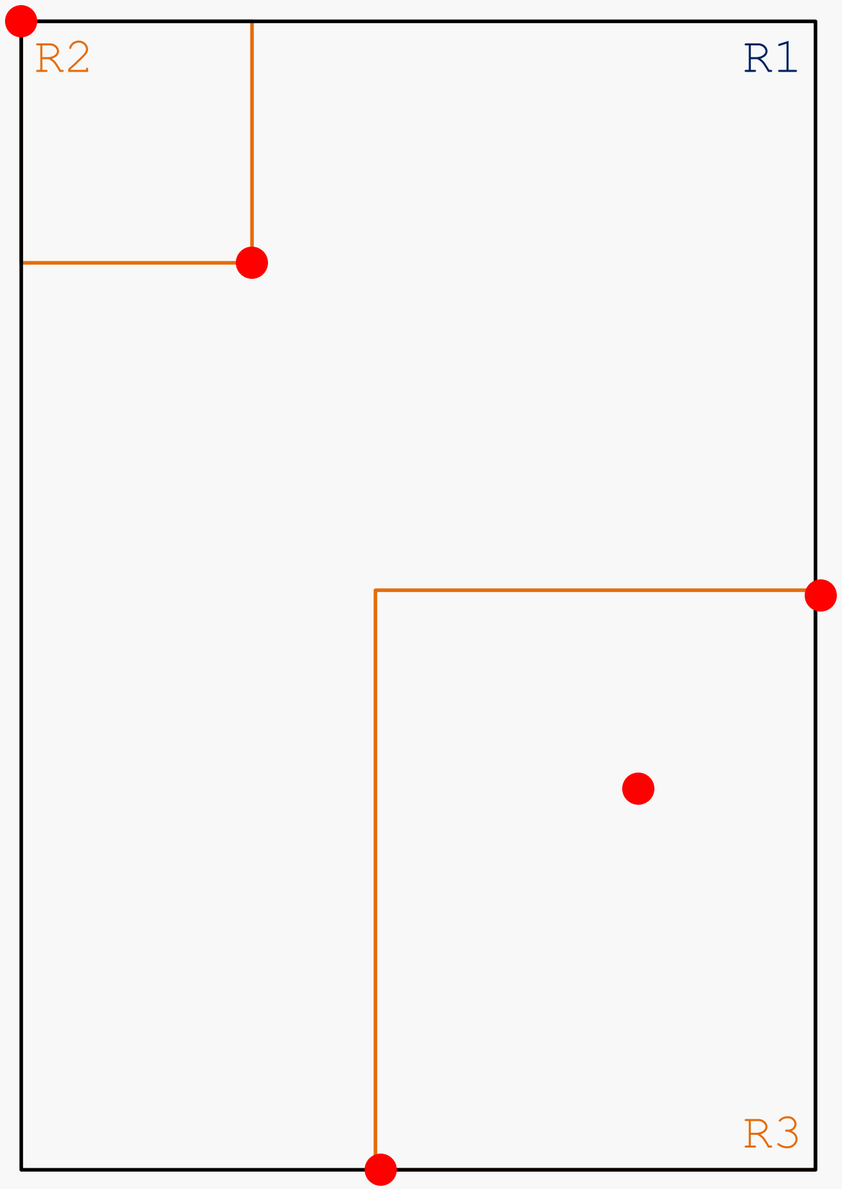
- Height-balanced.
- Nodes contain entries.
- Entries contain :

+ MBR containing its children if internal node, MBR containing object if leaf node.

+ Pointer to its children if internal node, pointer to object if leaf node.



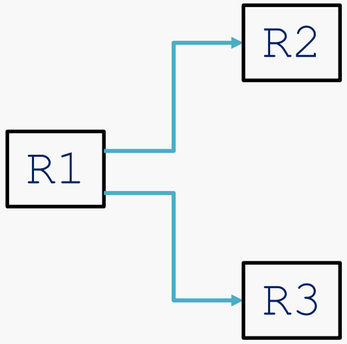
II. Structure and algorithms



- Height-balanced.
- Nodes contain entries.
- Entries contain :

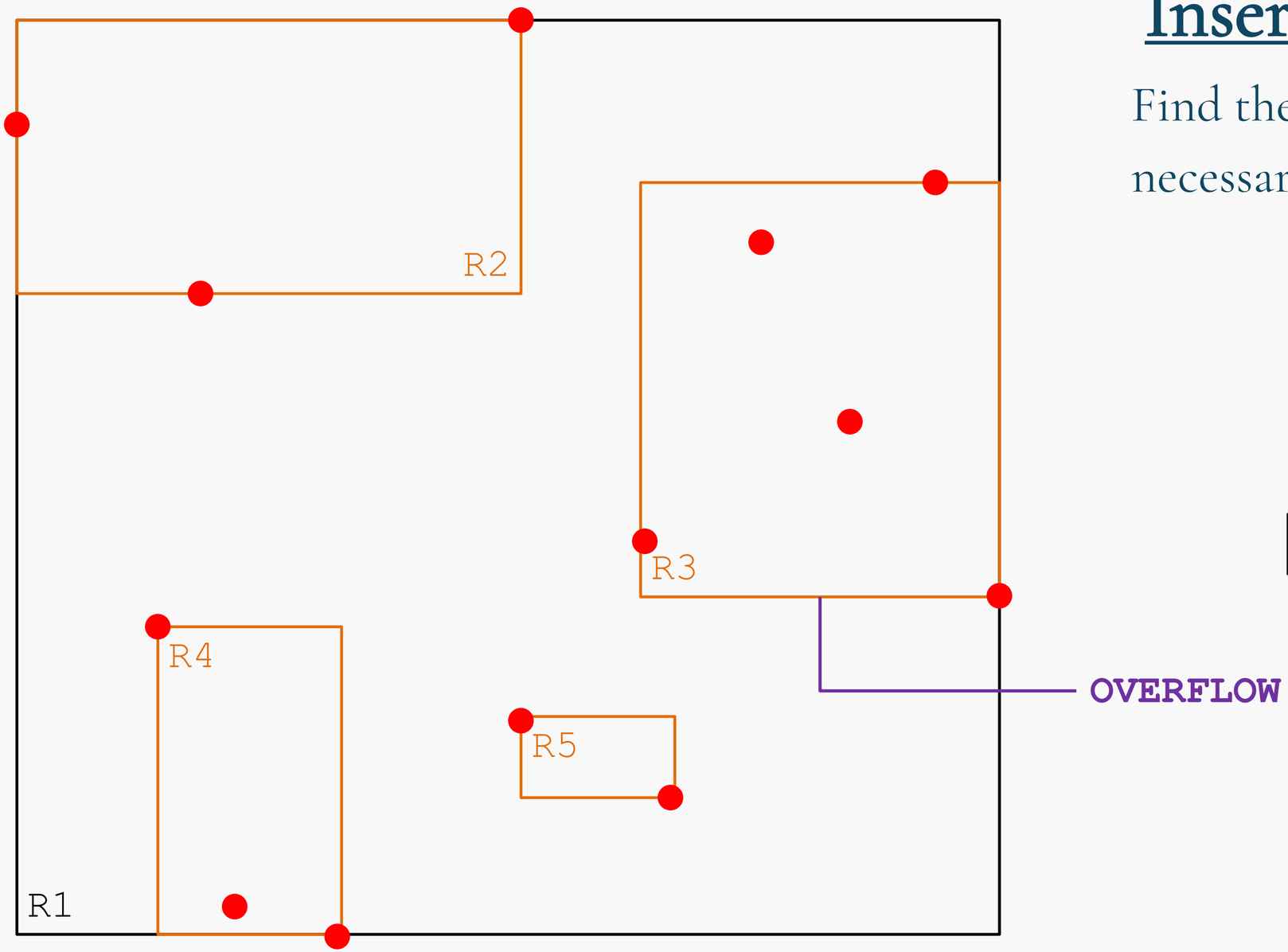
+ MBR containing its children if internal node, MBR containing object if leaf node.

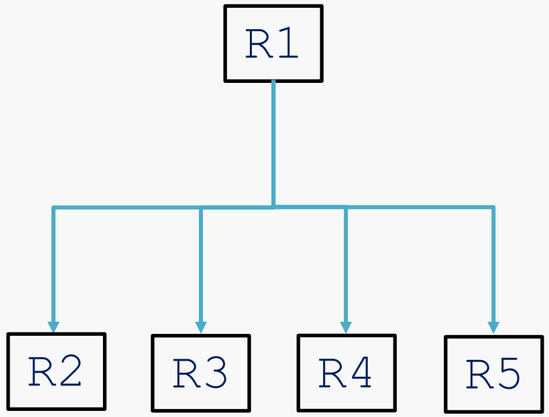
+ Pointer to its children if internal node, pointer to object if leaf node.



Insertion

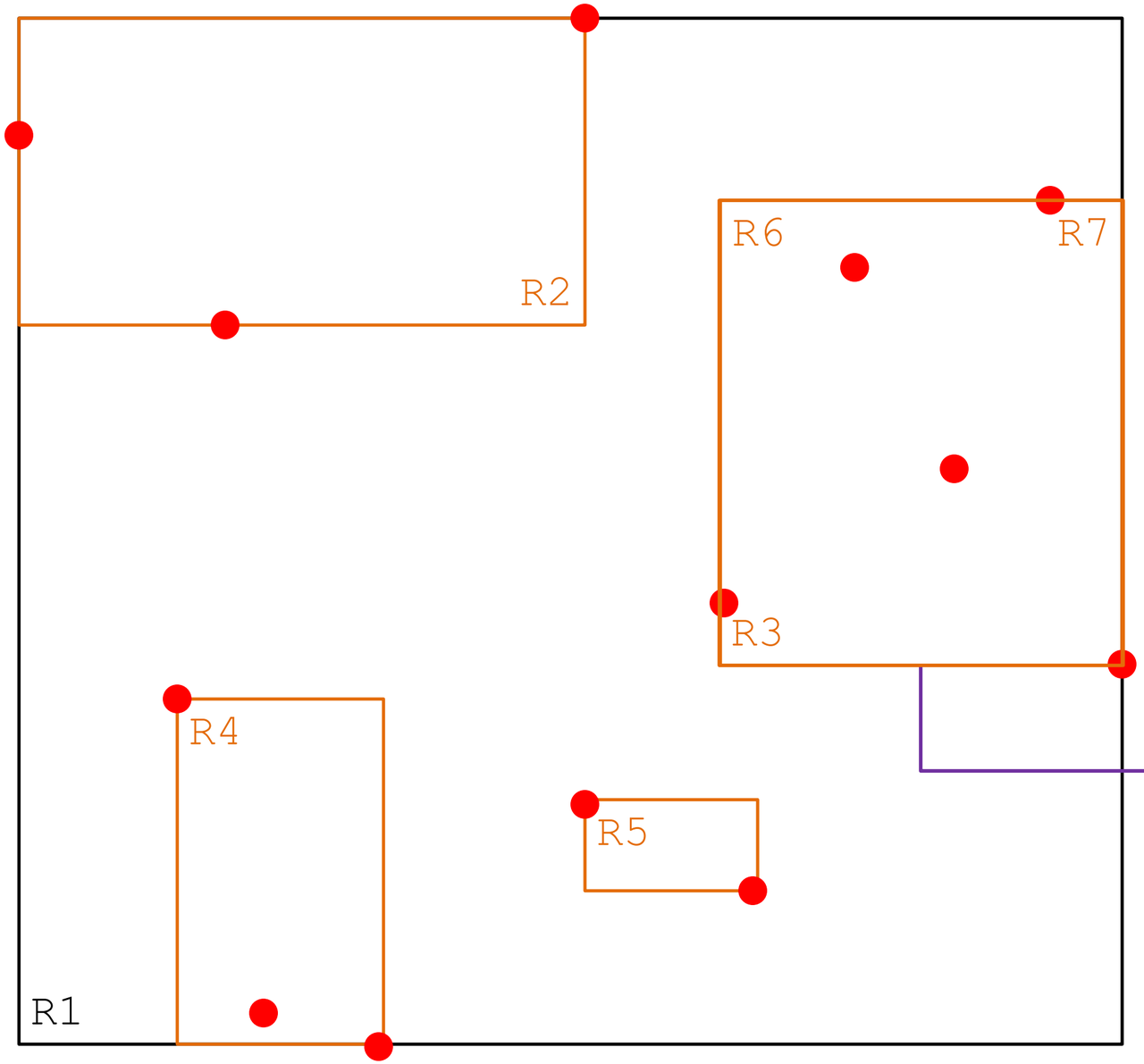
Find the best leaf node to insert, split if necessary.

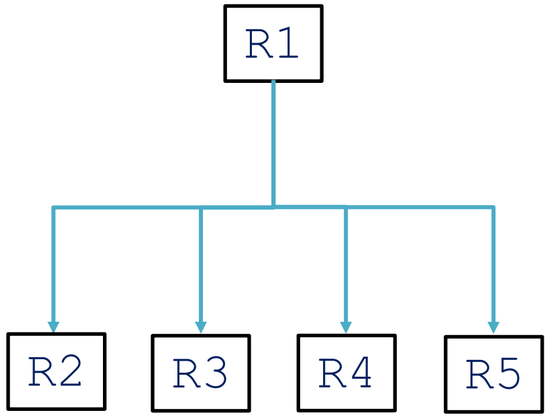


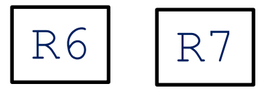


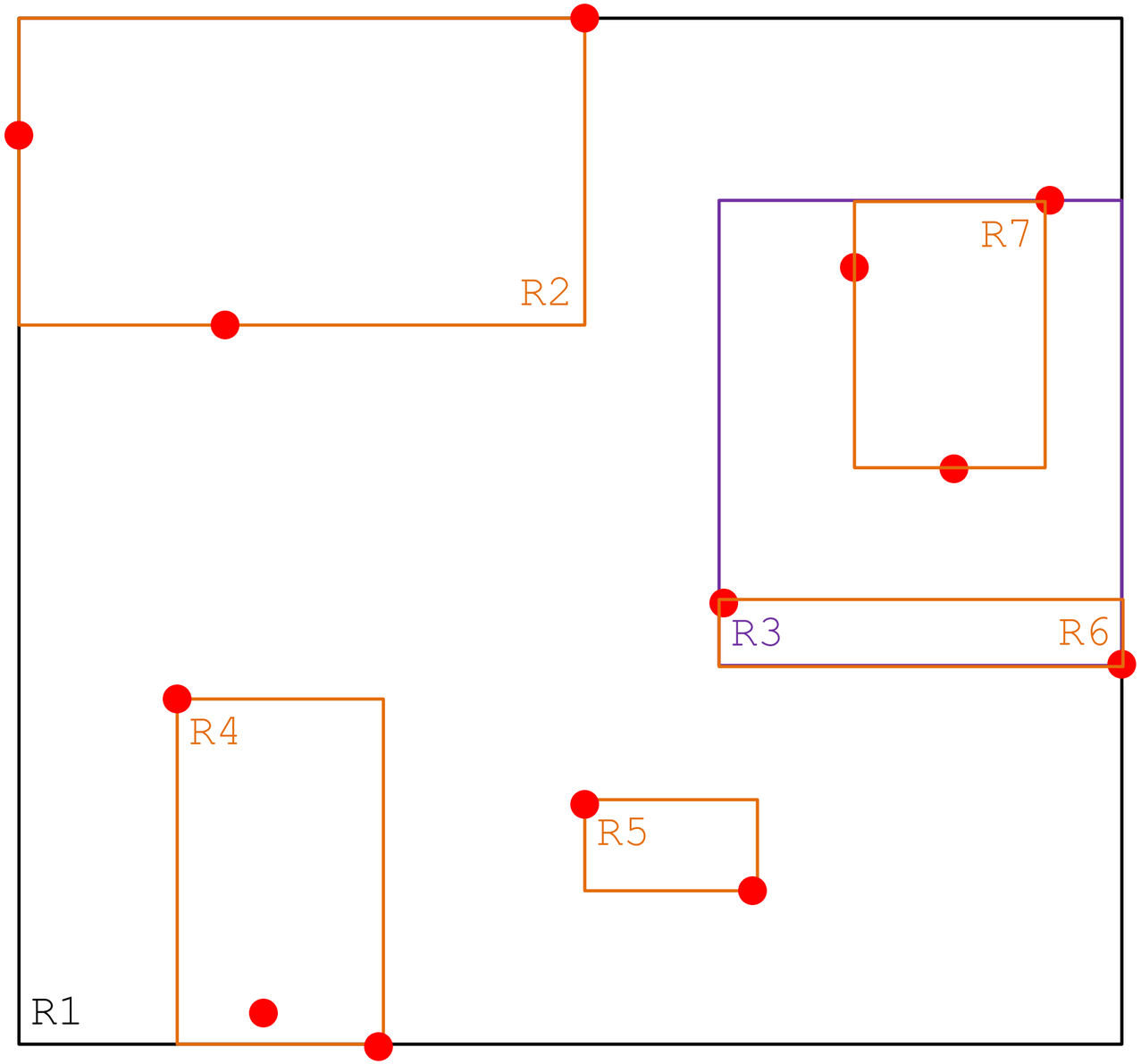
Insertion

Find the best leaf node to insert, split if necessary.



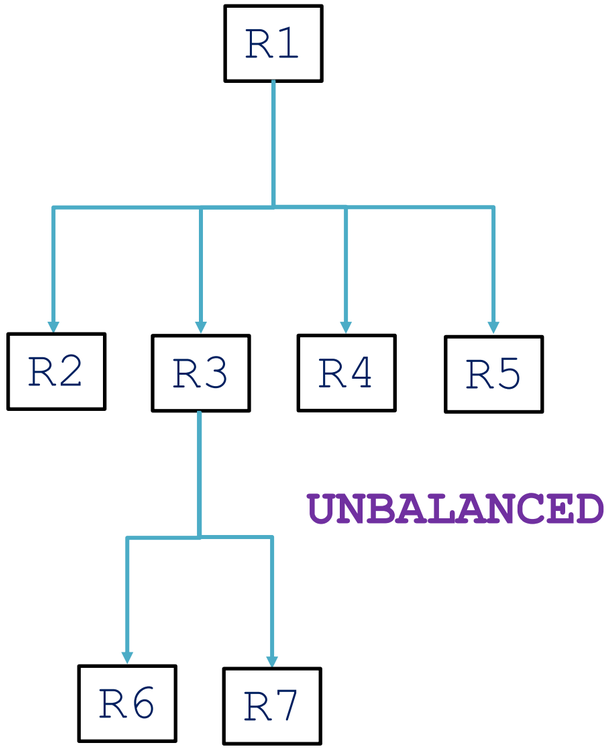


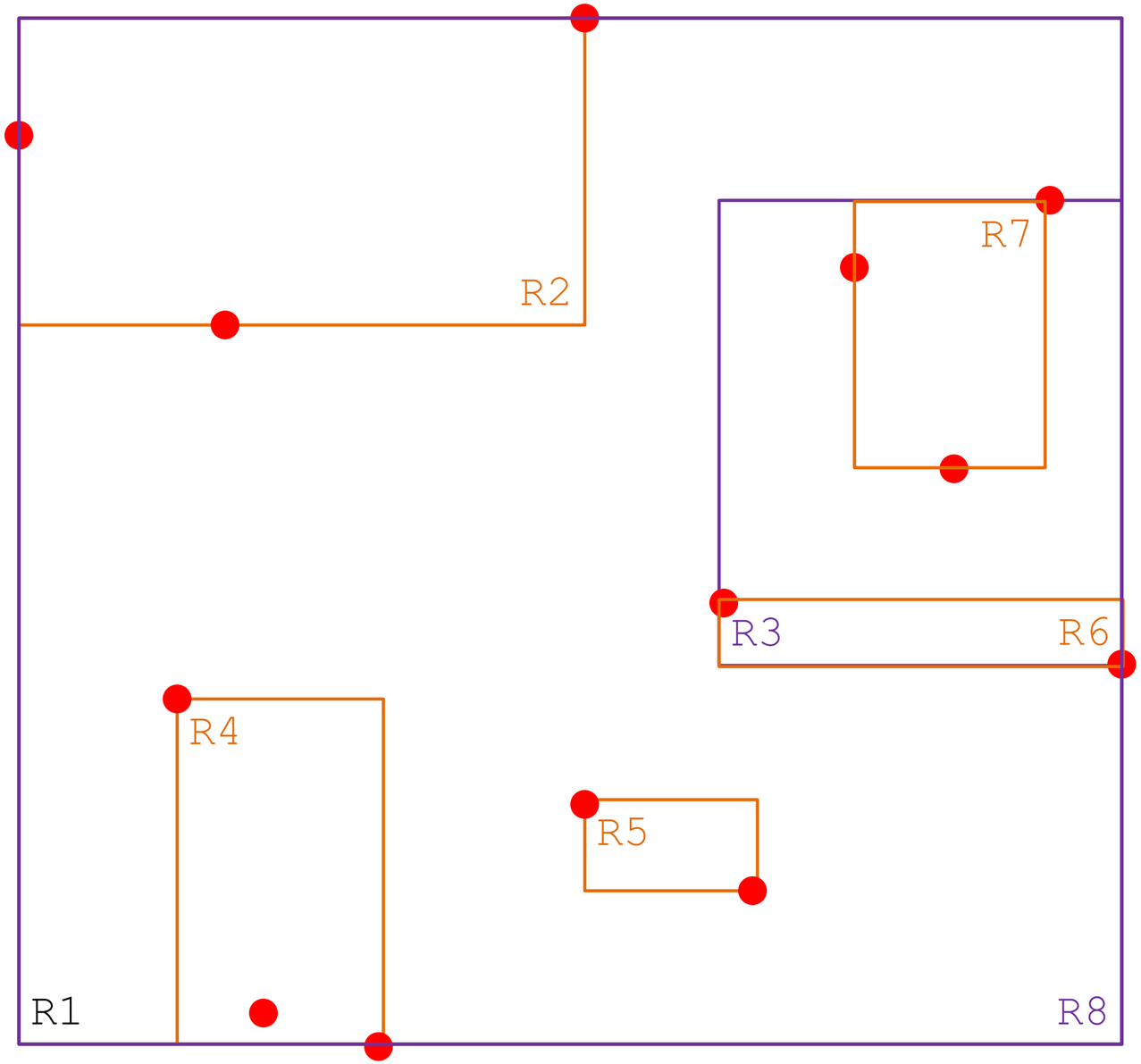




Insertion

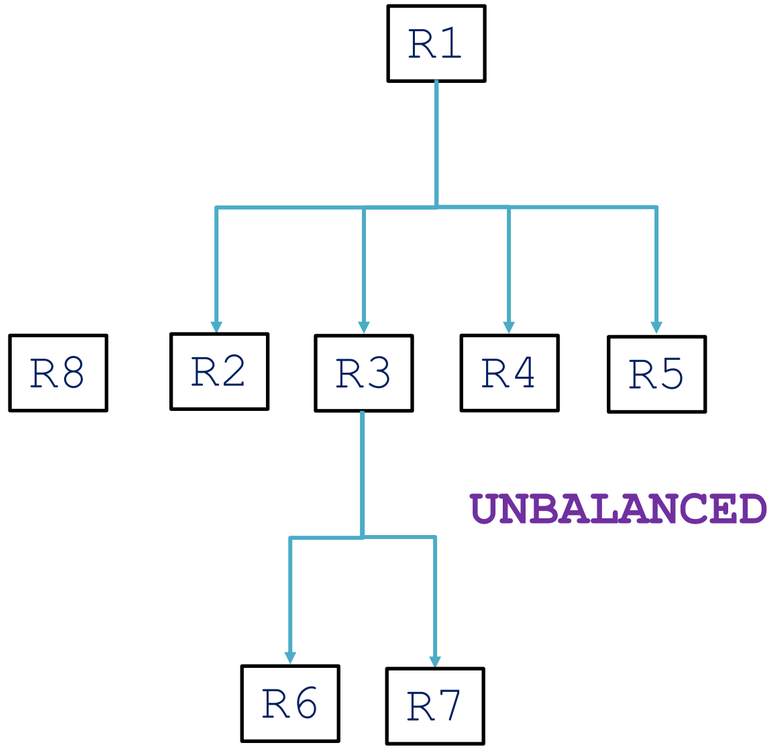
Find the best leaf node to insert, split if necessary.

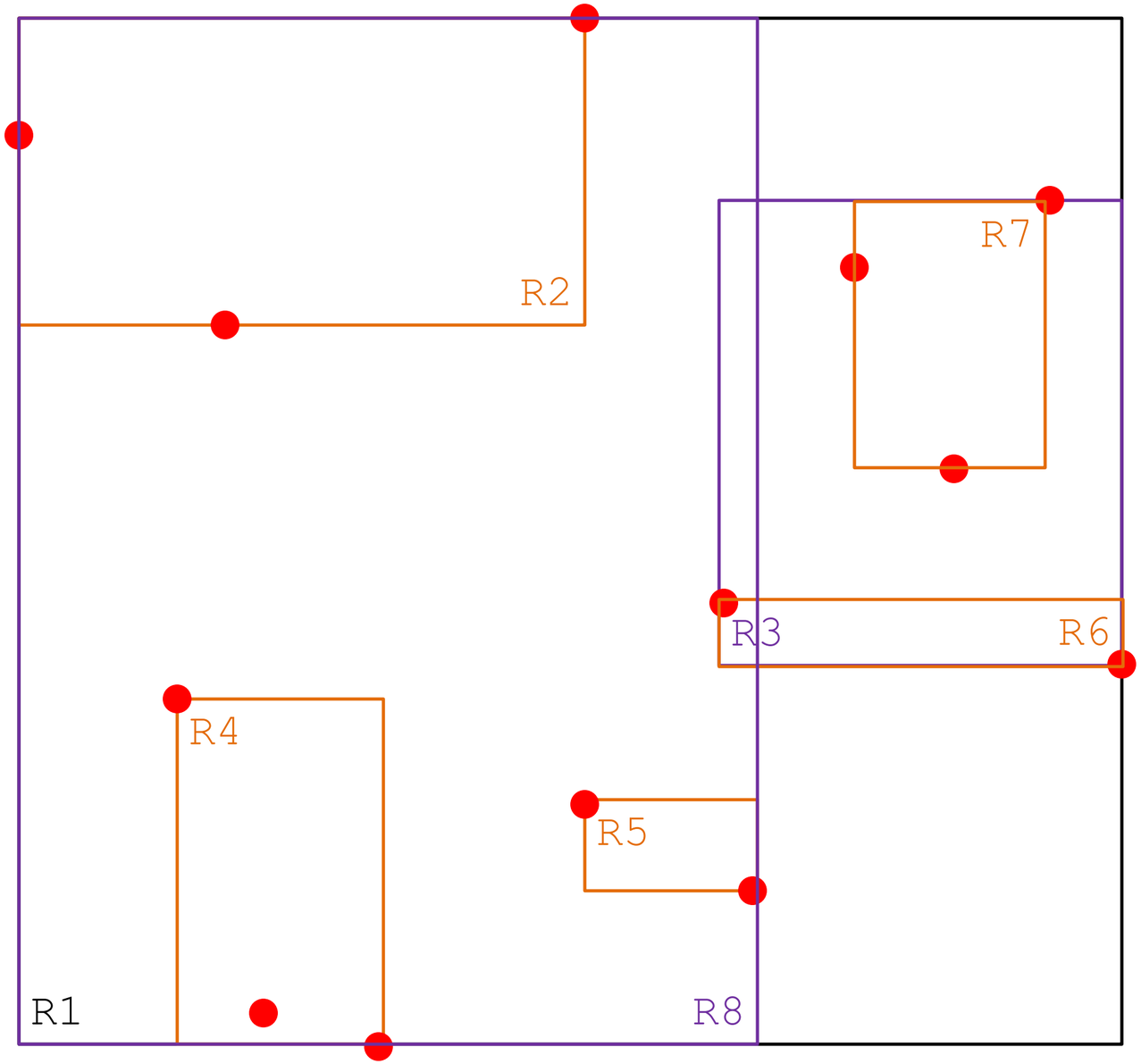




Insertion

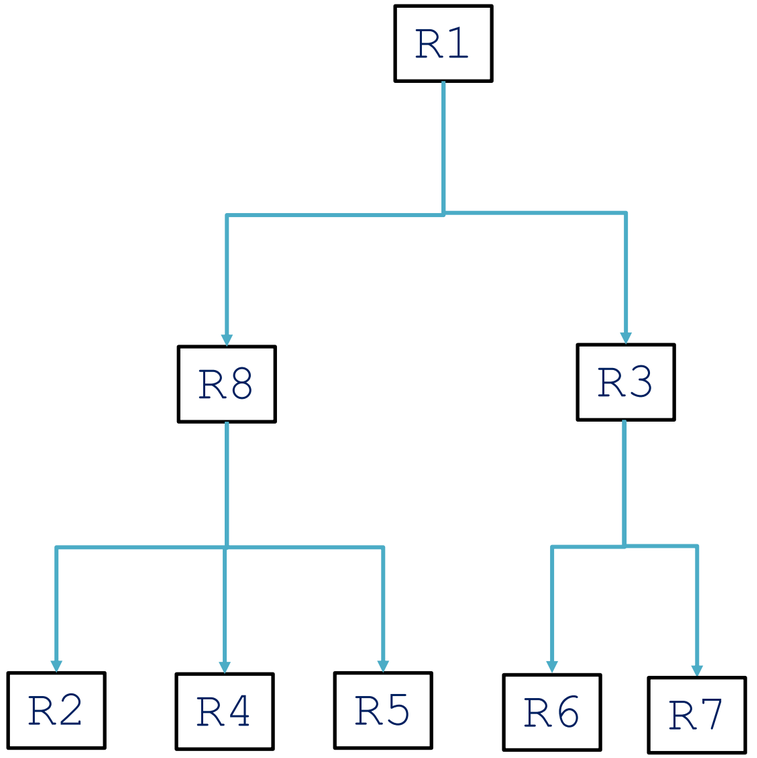
Find the best leaf node to insert, split if necessary.



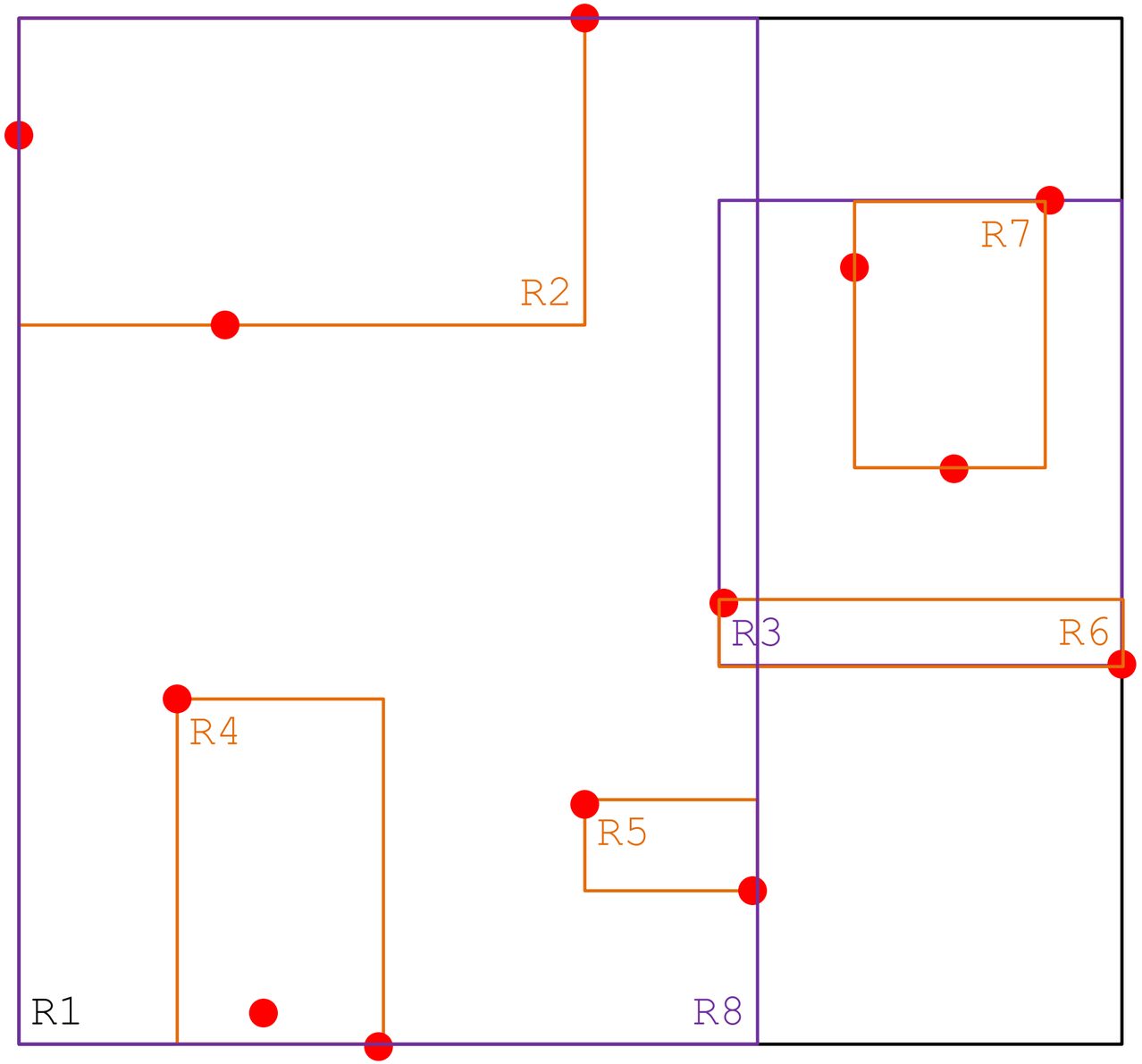


Insertion

Find the best leaf node to insert, split if necessary.

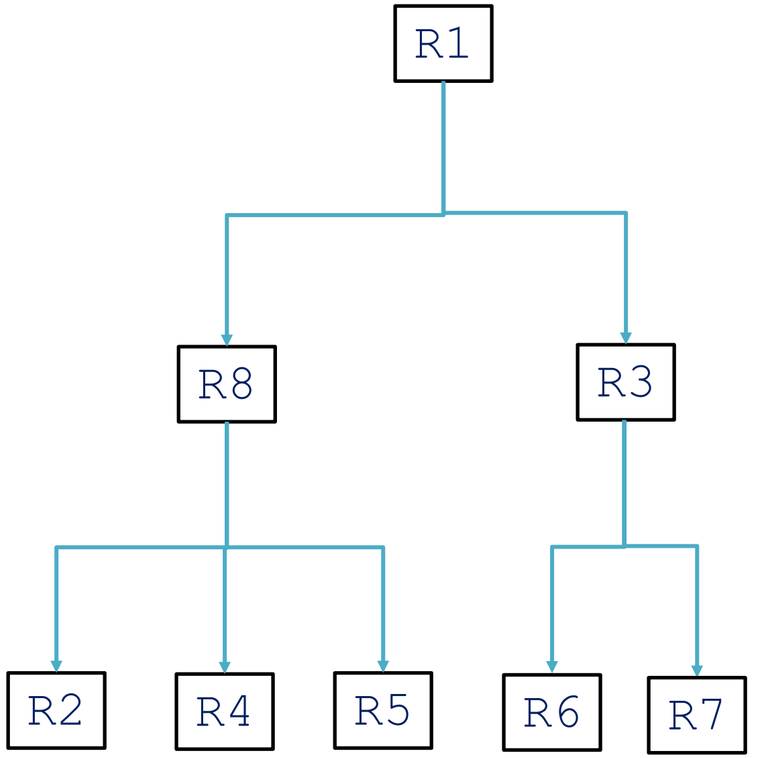


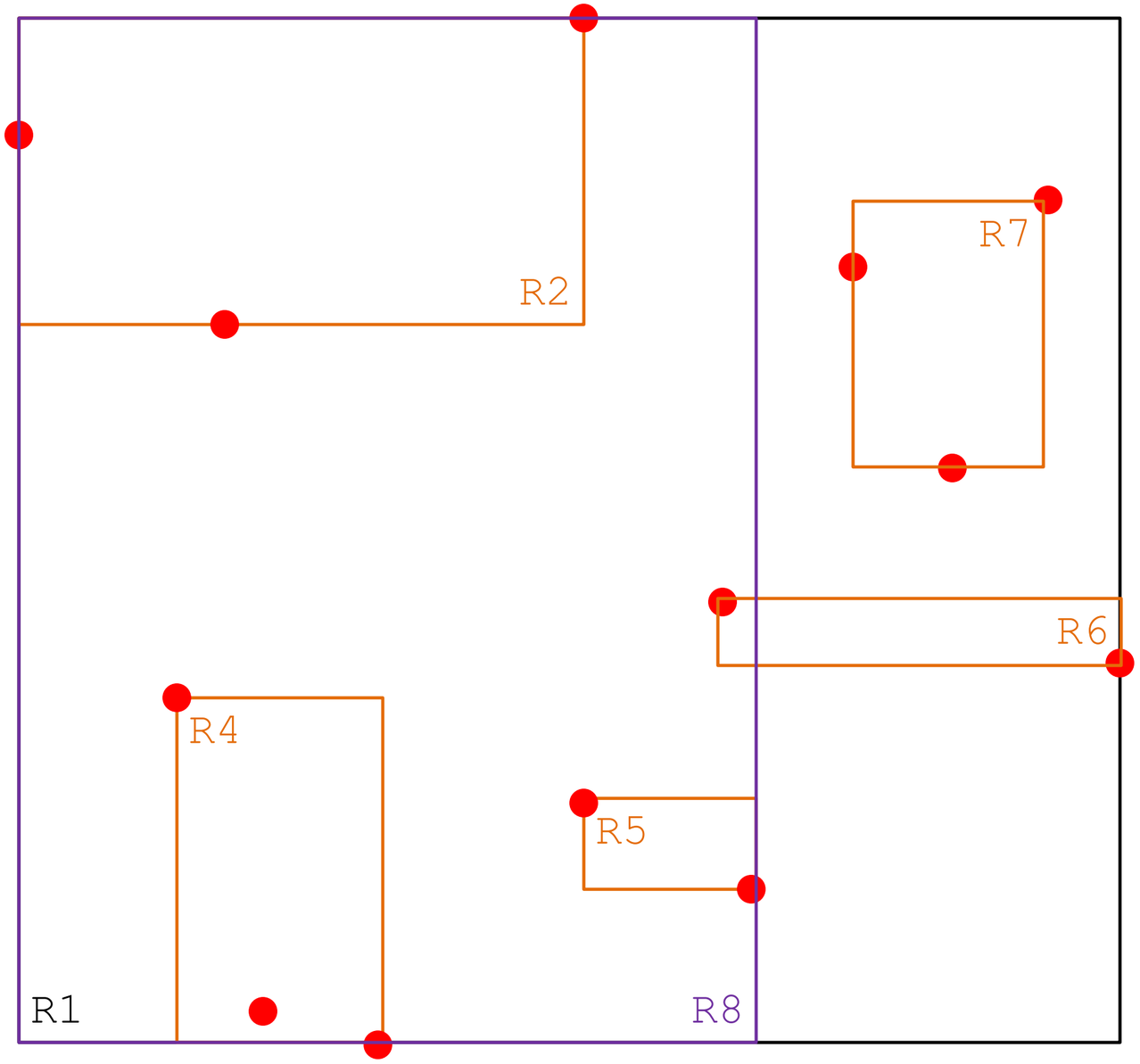
O(nlogn)



Deletion

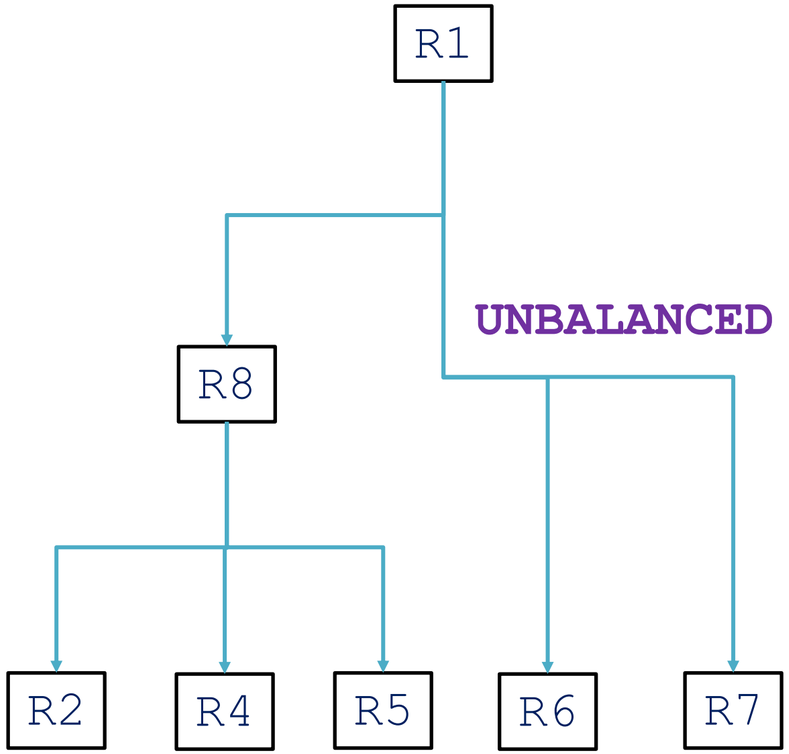
Remove an object, adjust parent nodes, and reintegrate underfilled nodes.

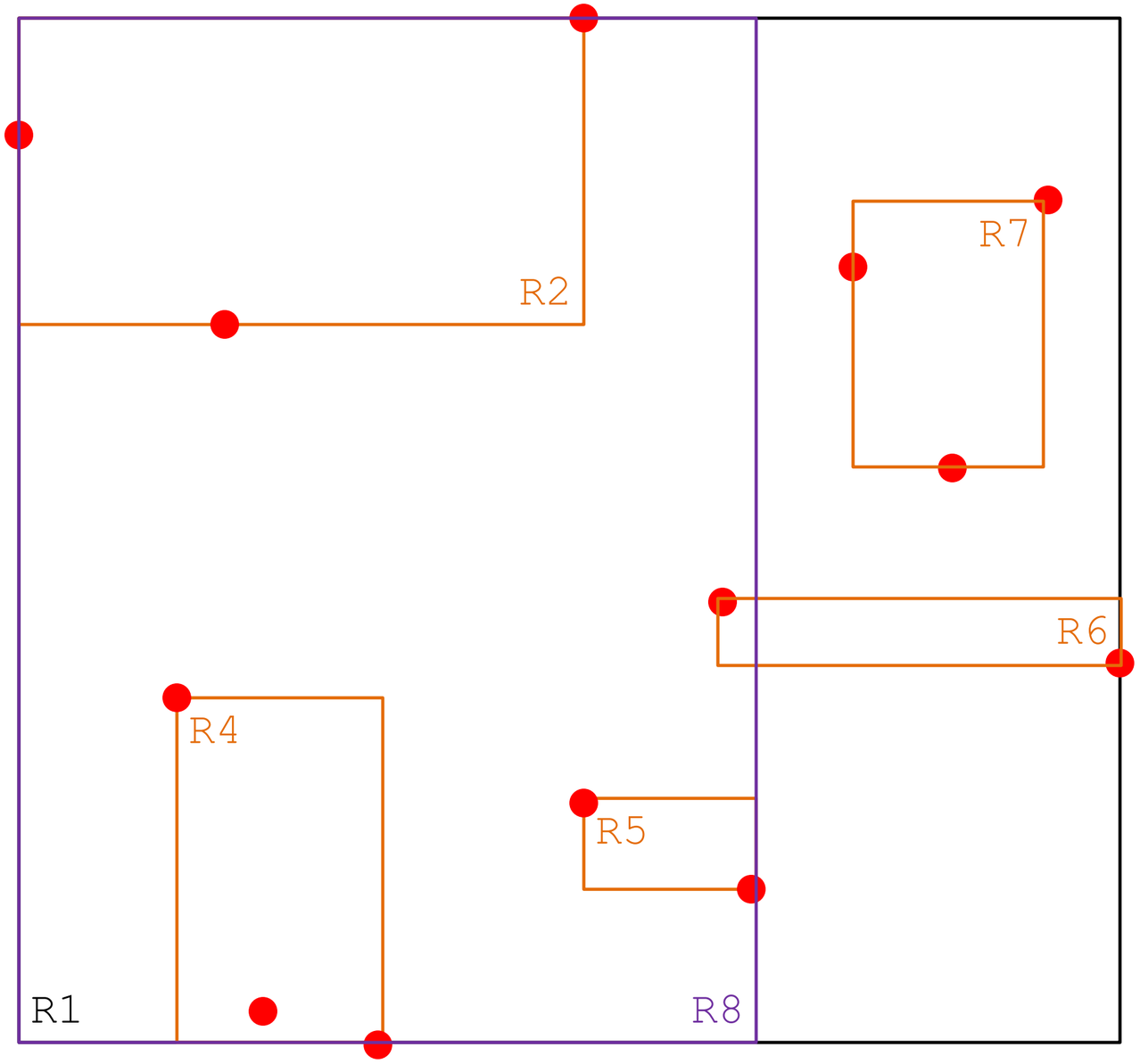




Deletion

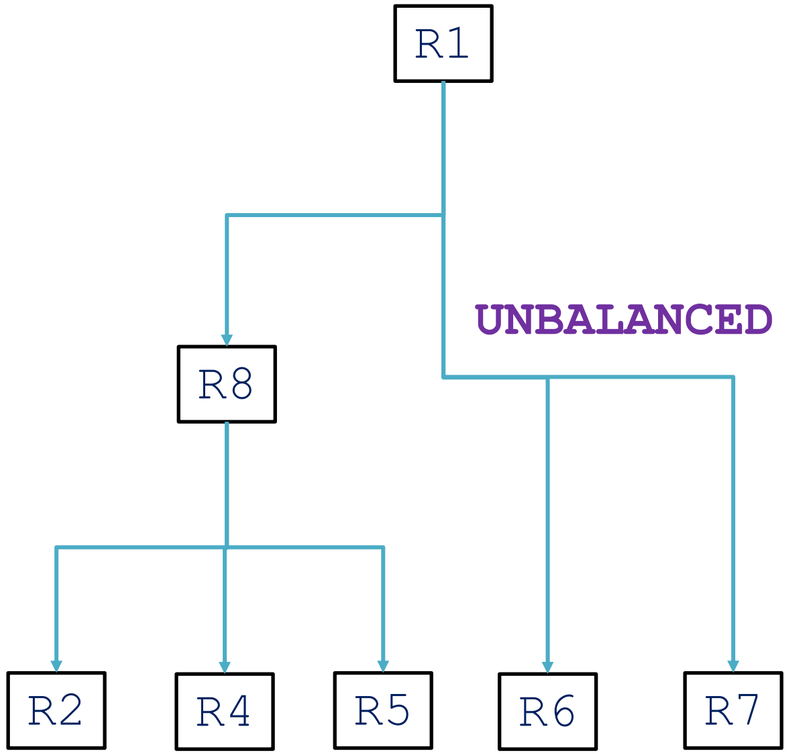
Remove an object, adjust parent nodes, and reintegrate underfilled nodes.

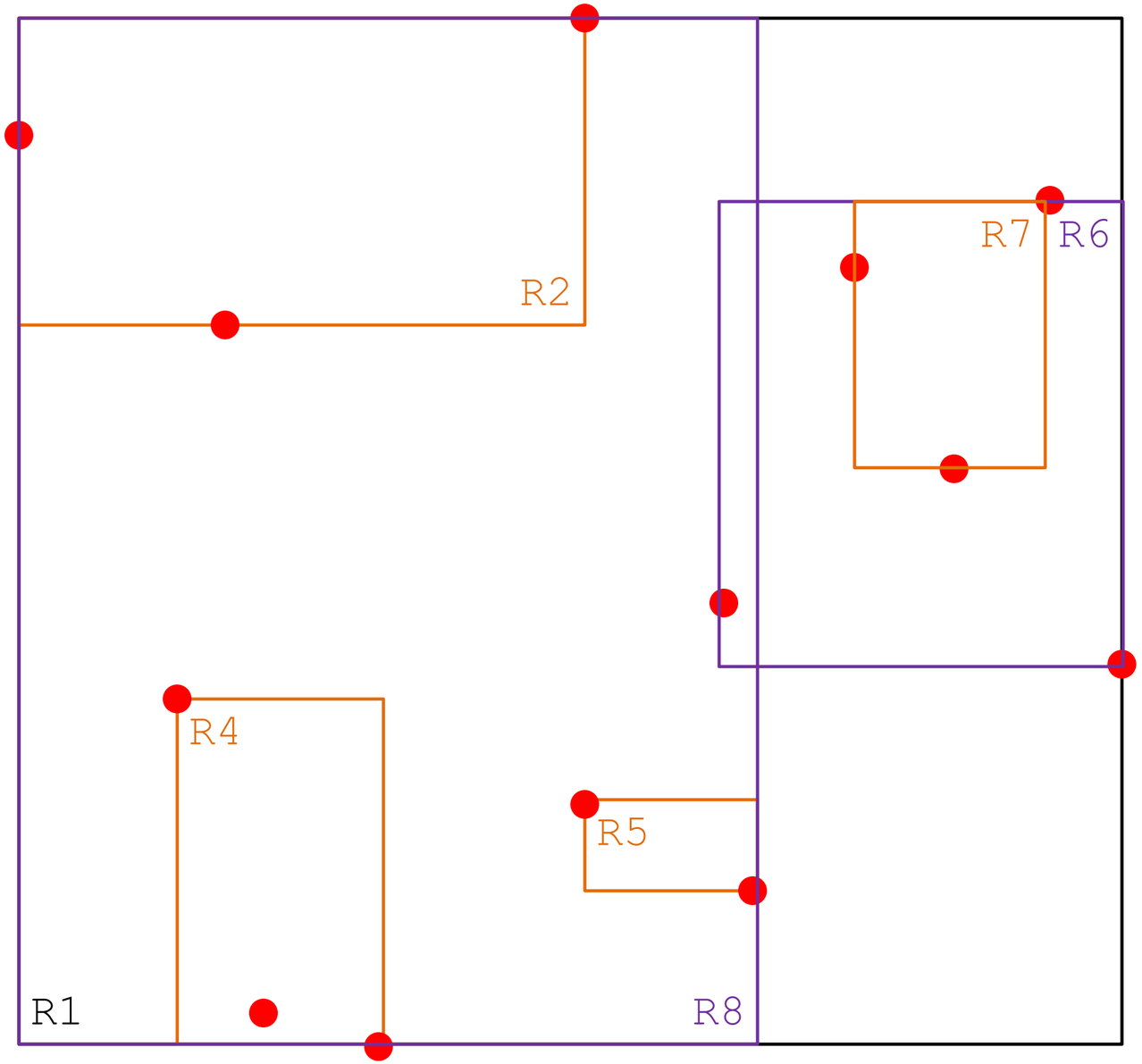




Deletion

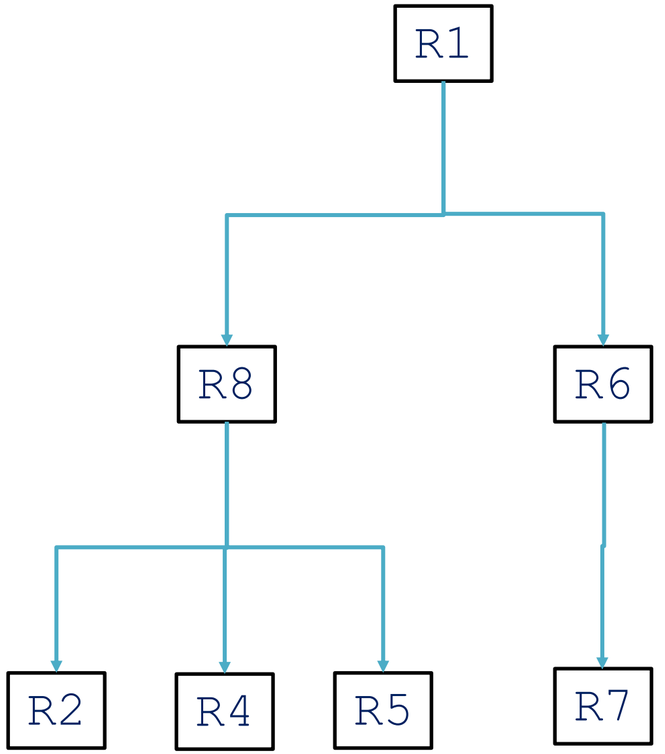
Remove an object, adjust parent nodes, and reintegrate underfilled nodes.



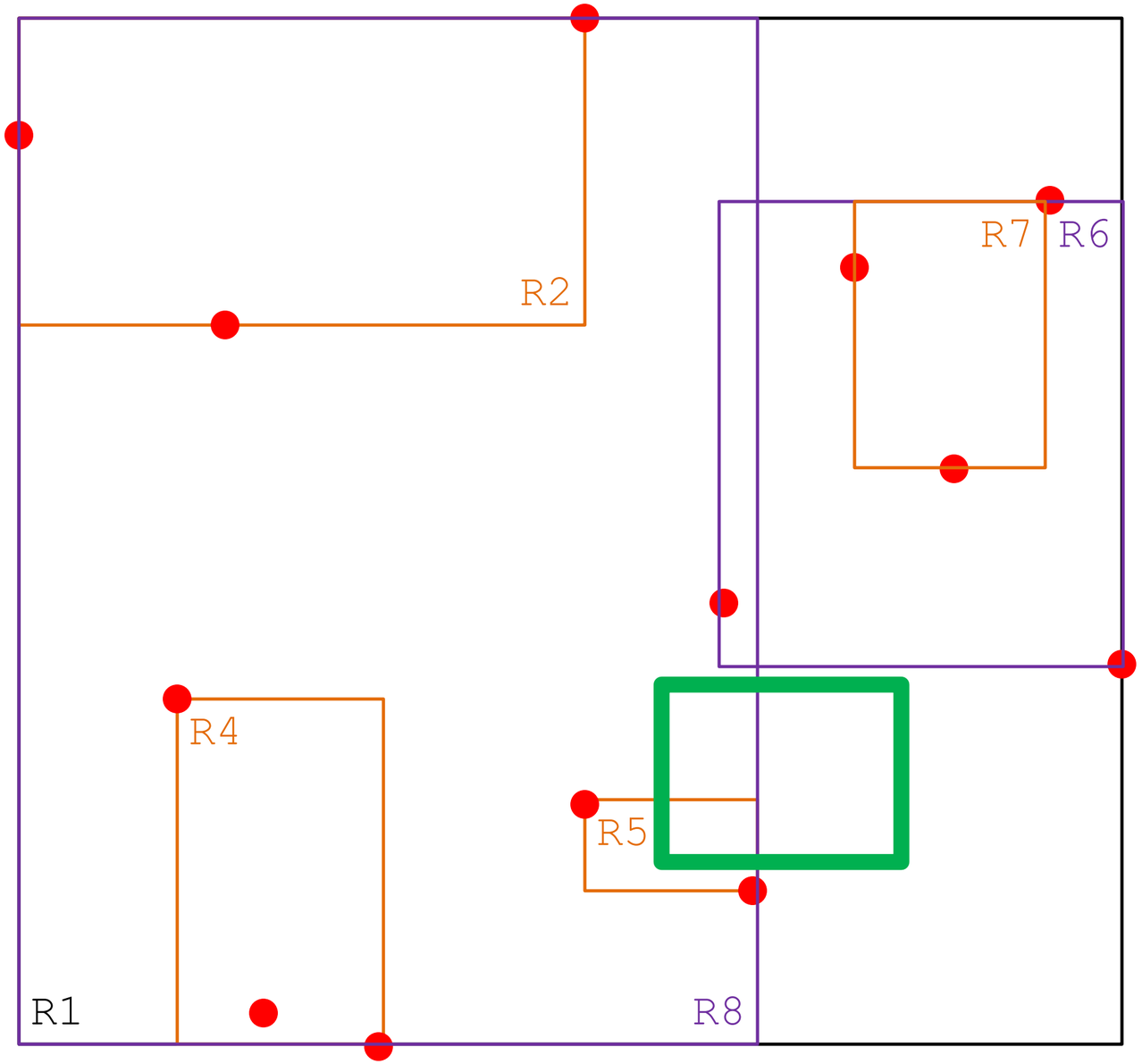


Deletion

Remove an object, adjust parent nodes, and reintegrate underfilled nodes.

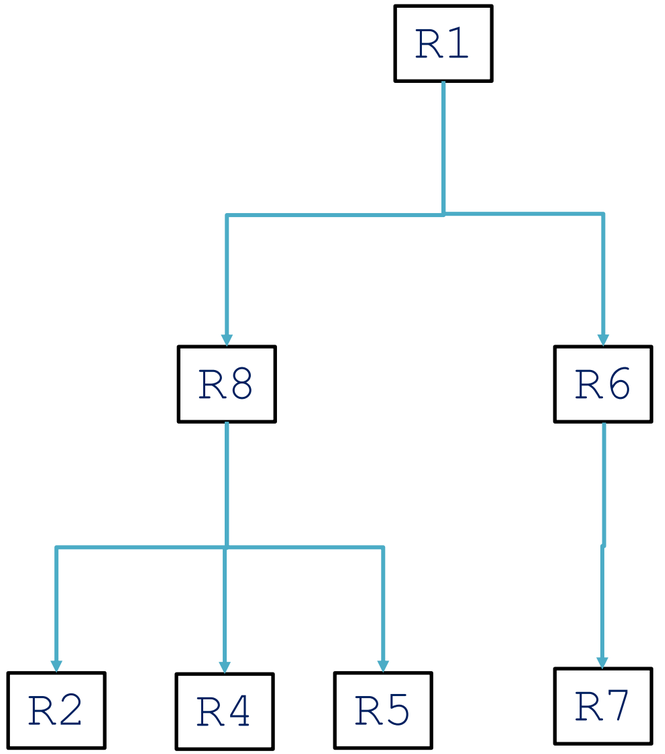


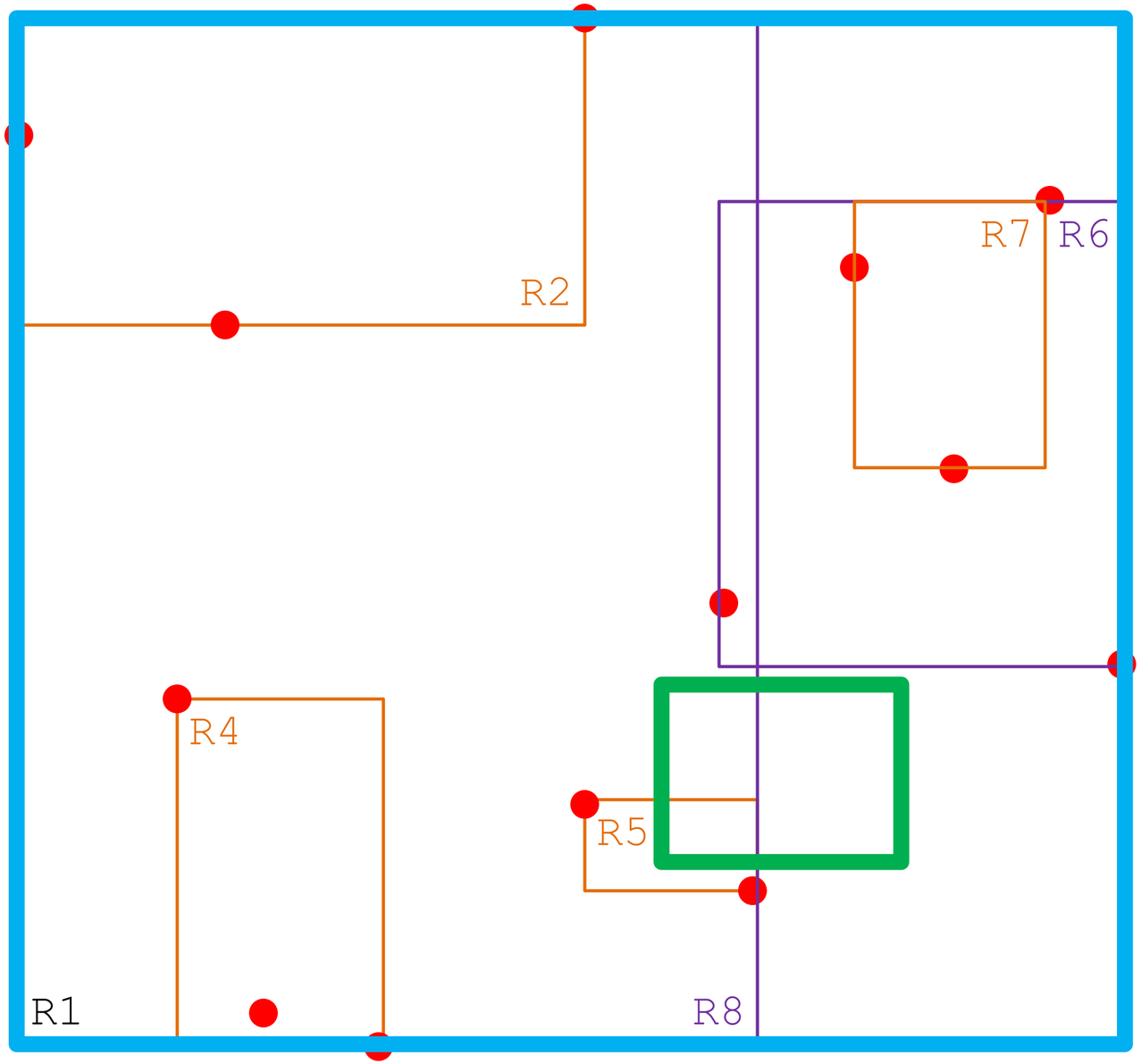
O(nlogn)



Search

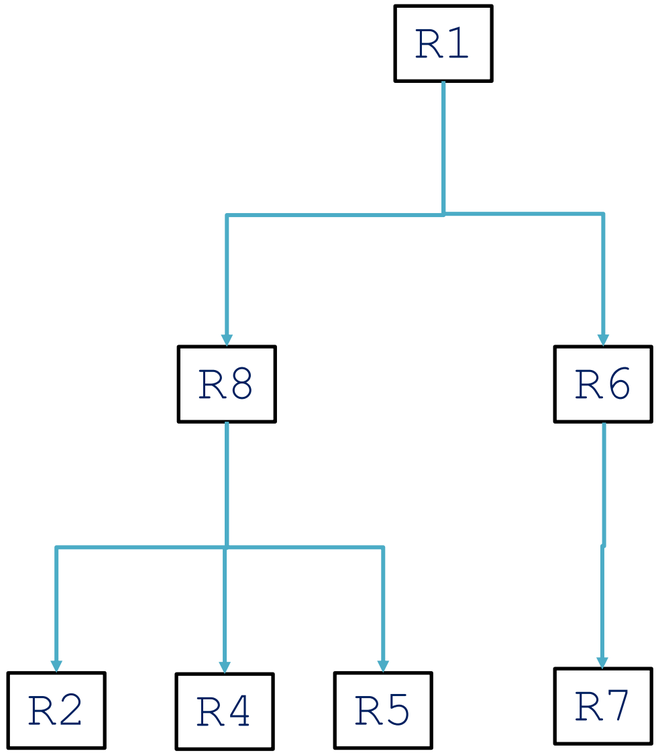
Recursively search bounding boxes matching query criteria.

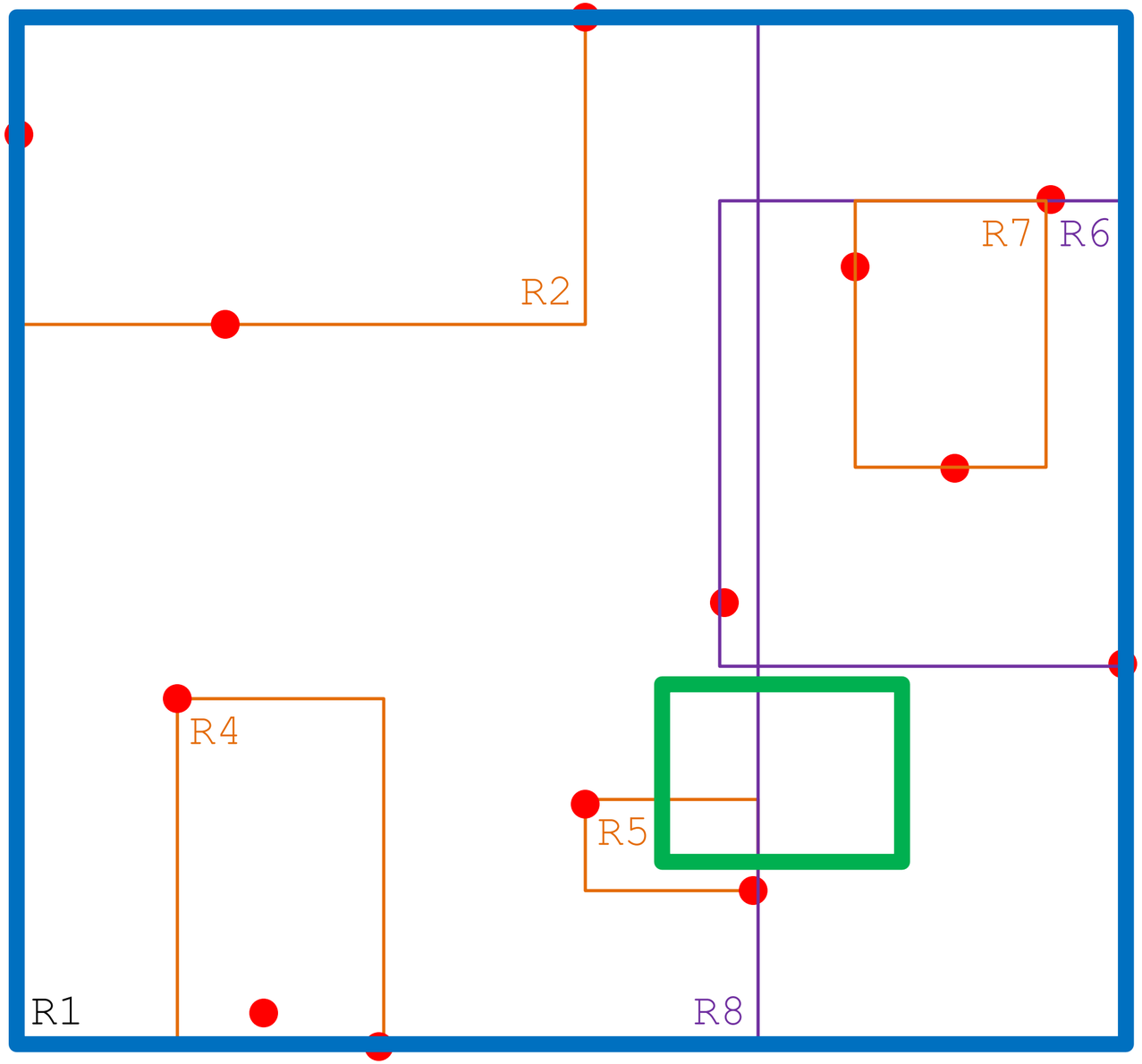




Search

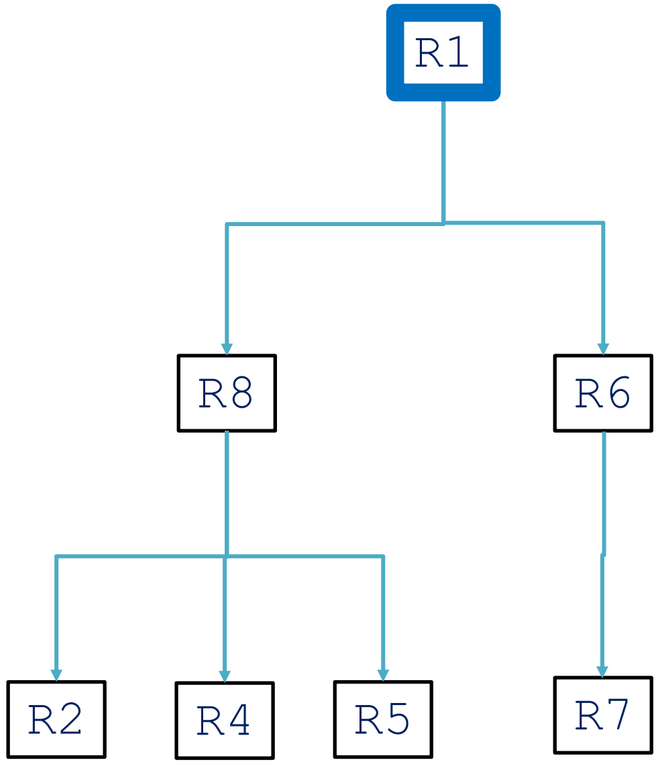
Recursively search bounding boxes matching query criteria.

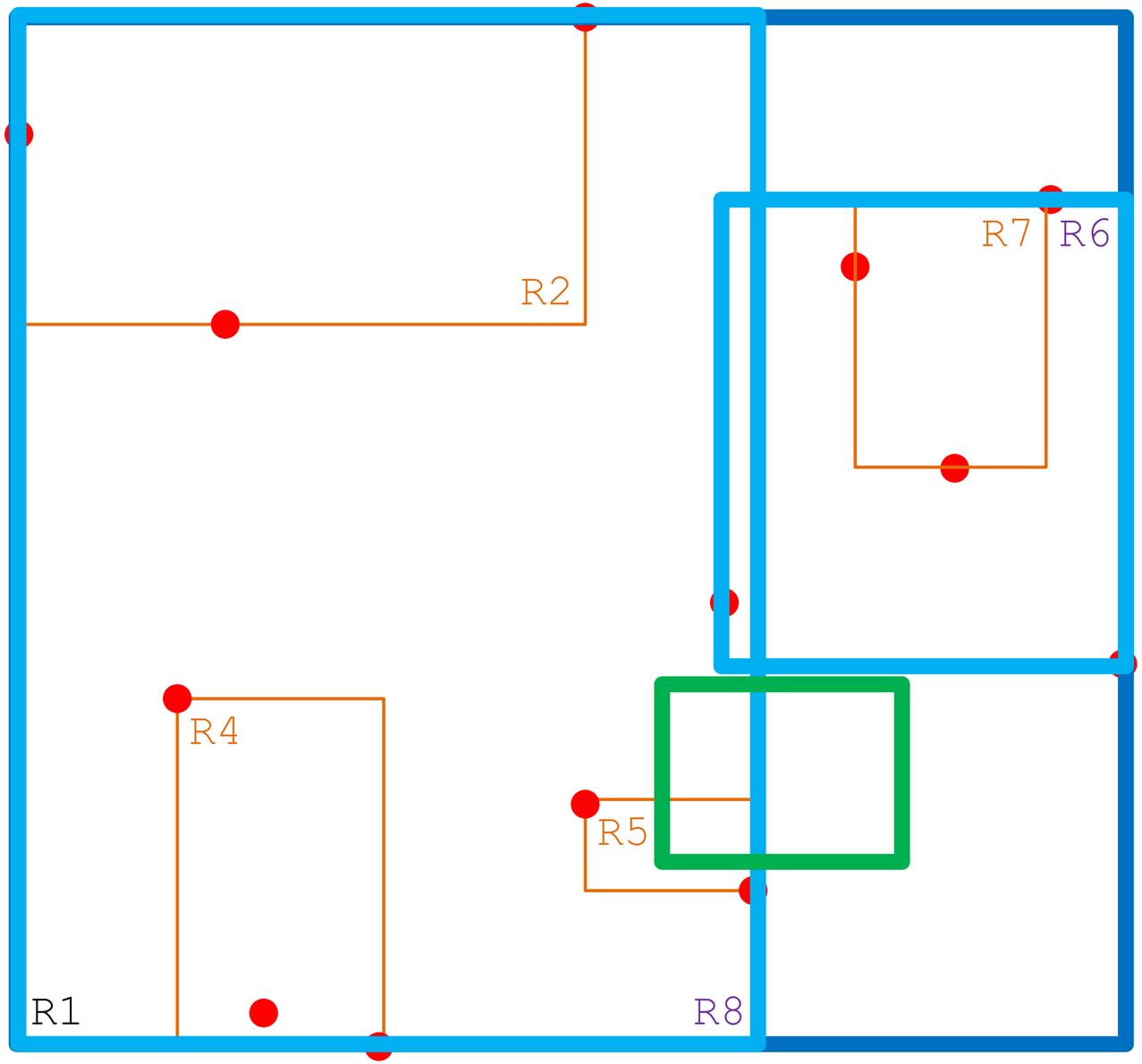




Search

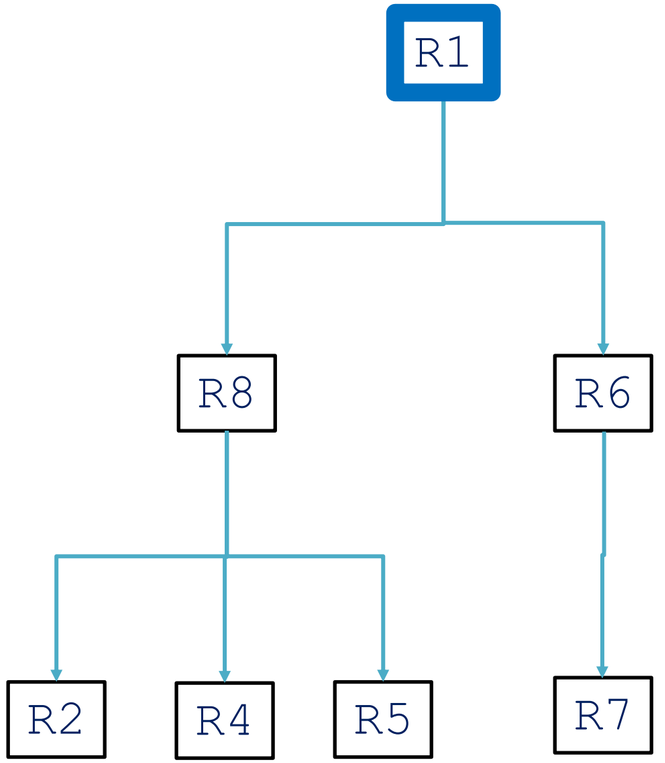
Recursively search bounding boxes matching query criteria.

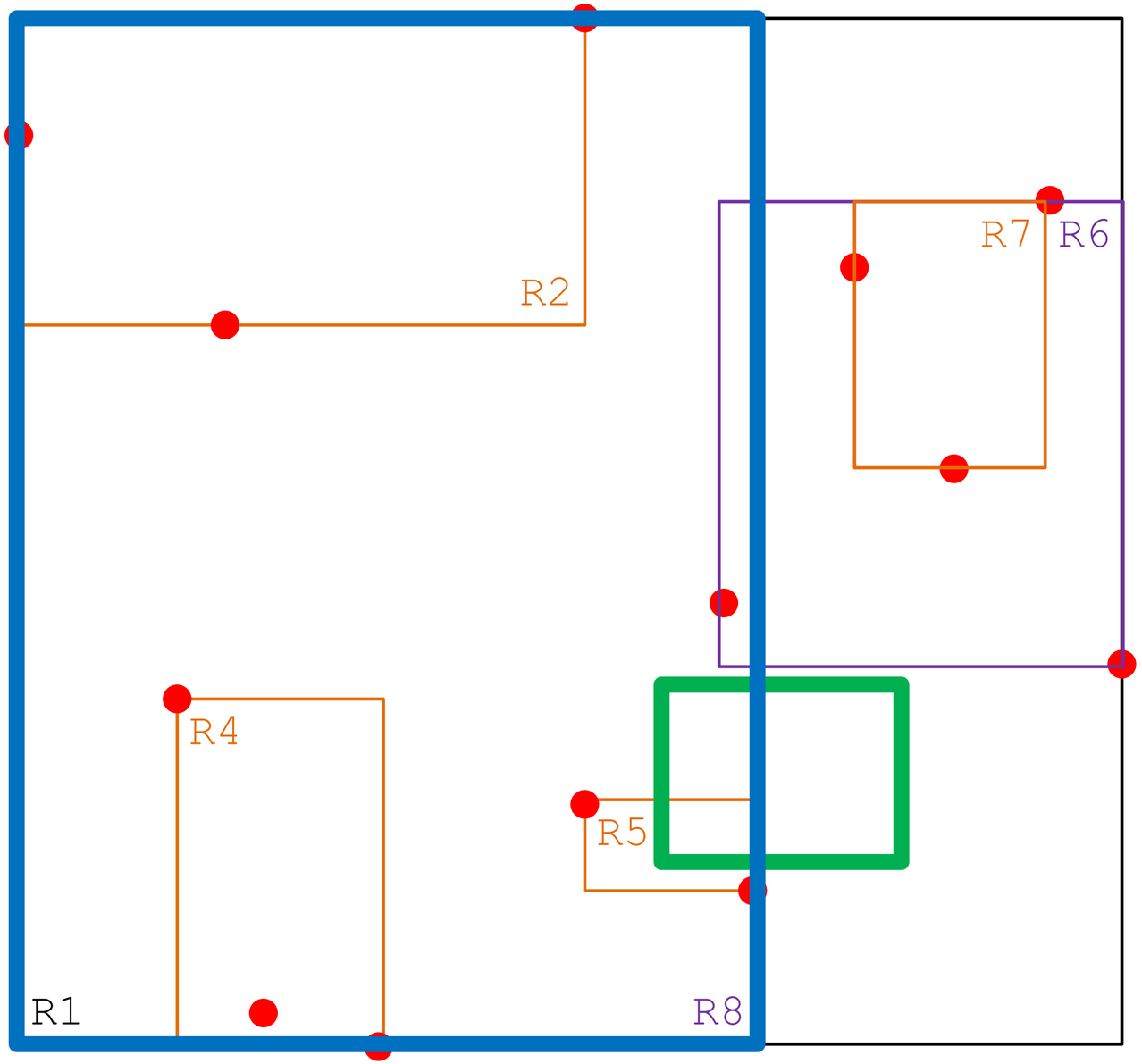




Search

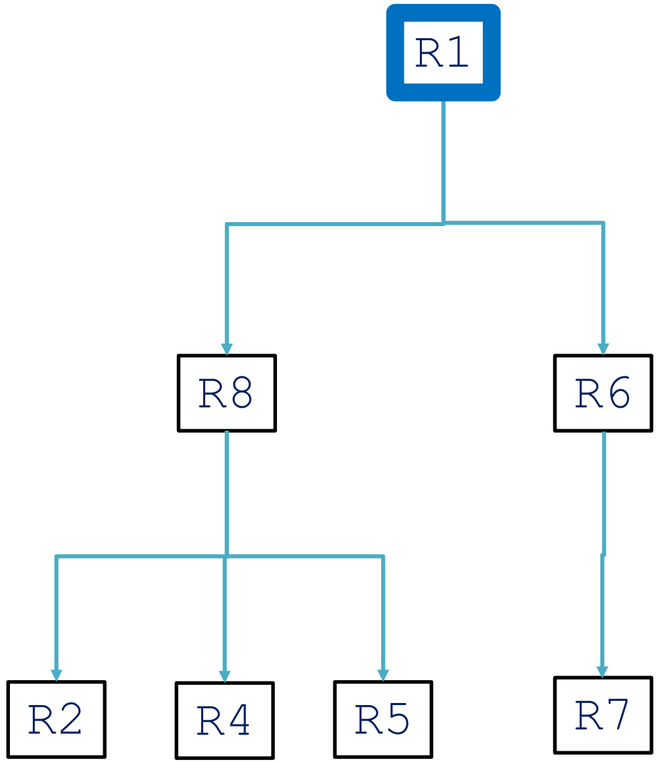
Recursively search bounding boxes matching query criteria.

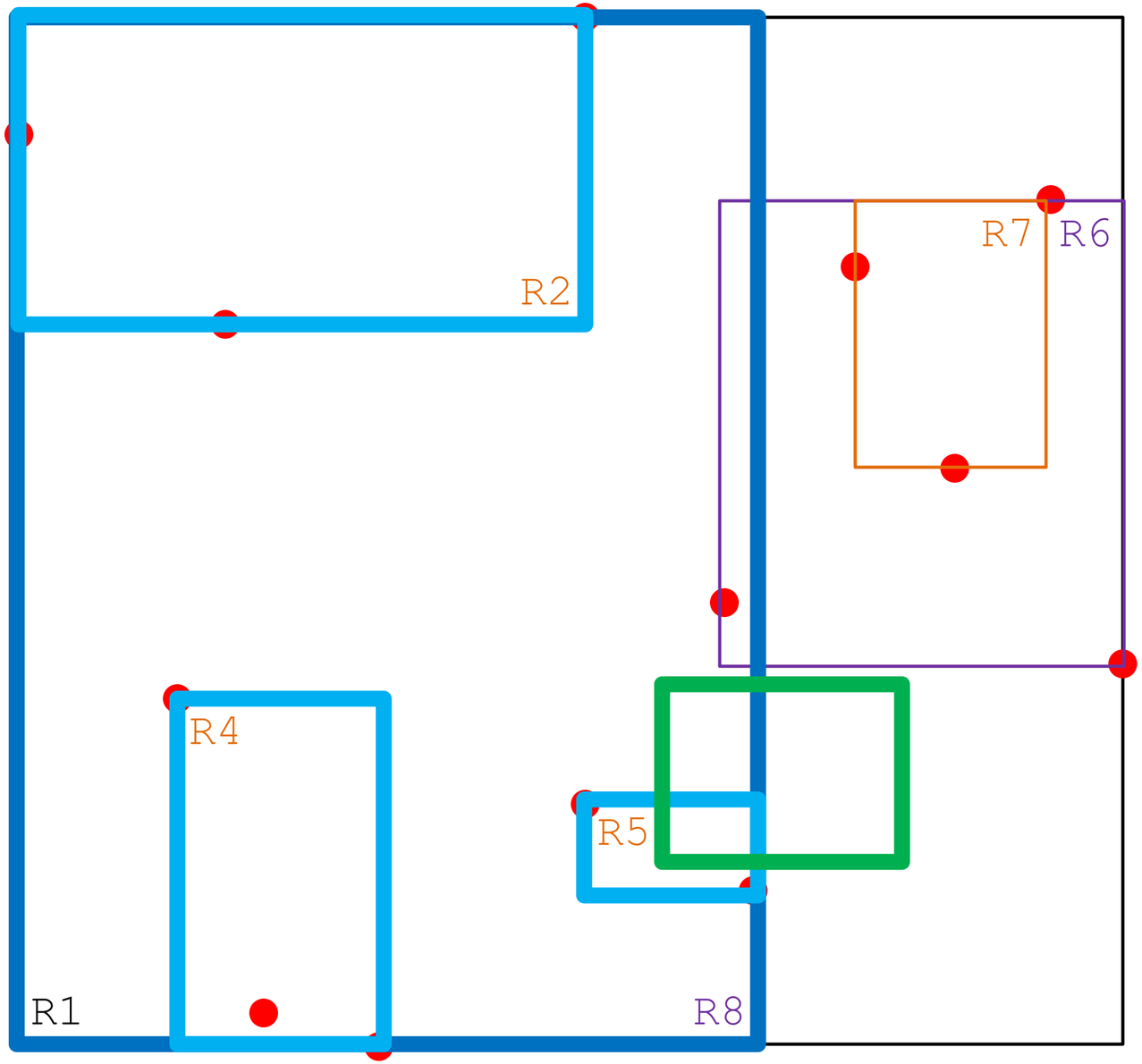




Search

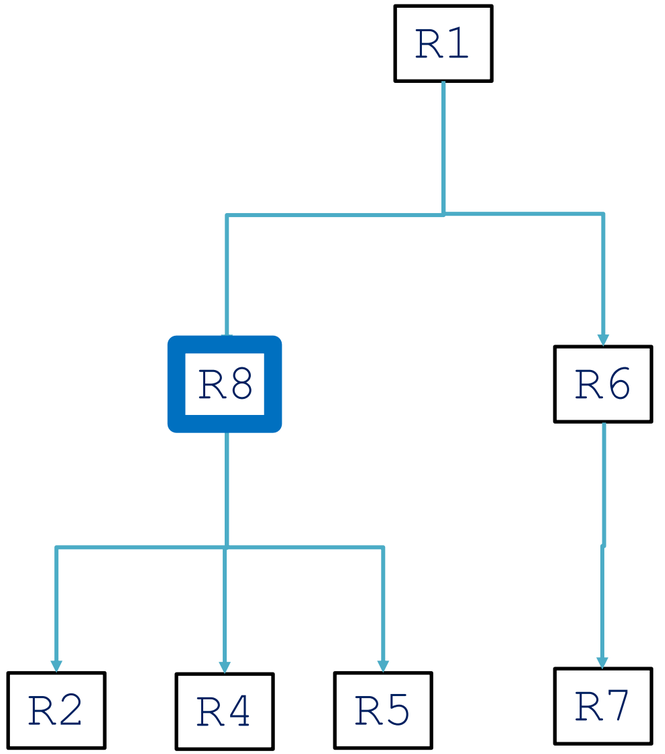
Recursively search bounding boxes matching query criteria.

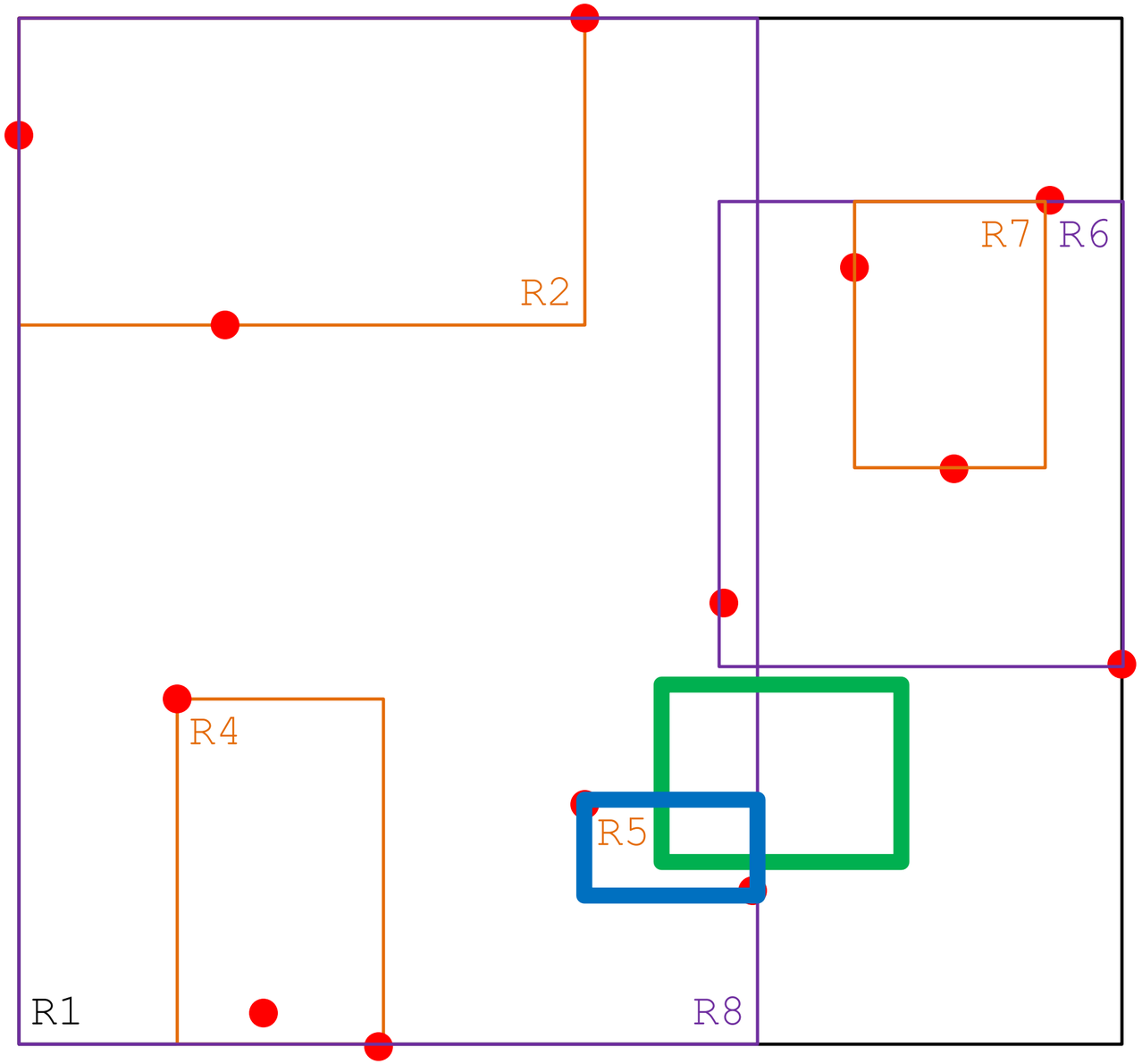




Search

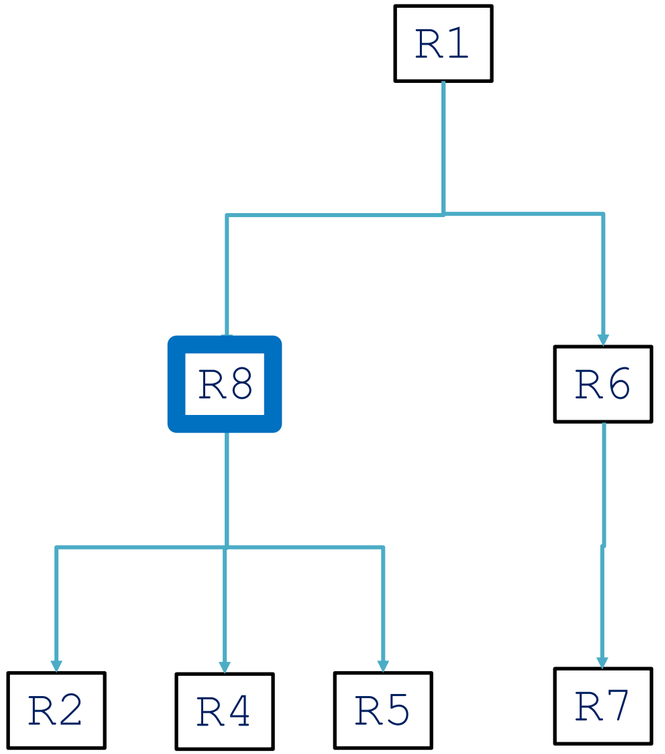
Recursively search bounding boxes matching query criteria.



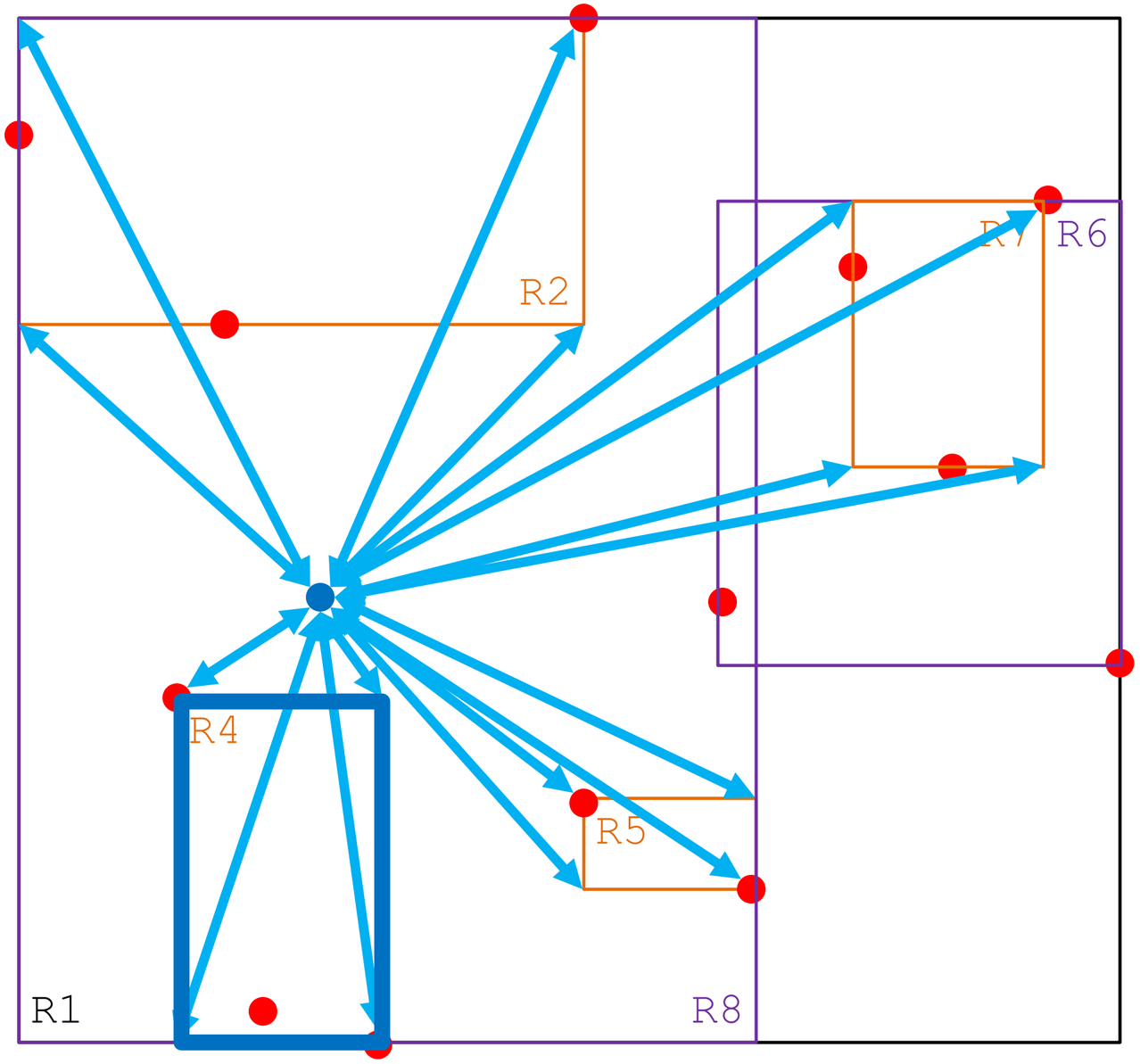


Search

Recursively search bounding boxes matching query criteria.

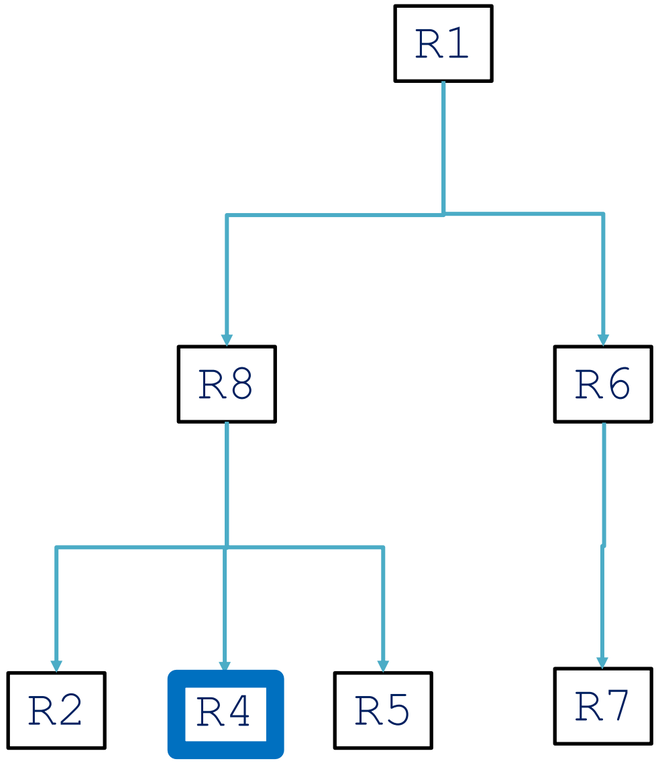


O(n)



Nearest Neighbor Search

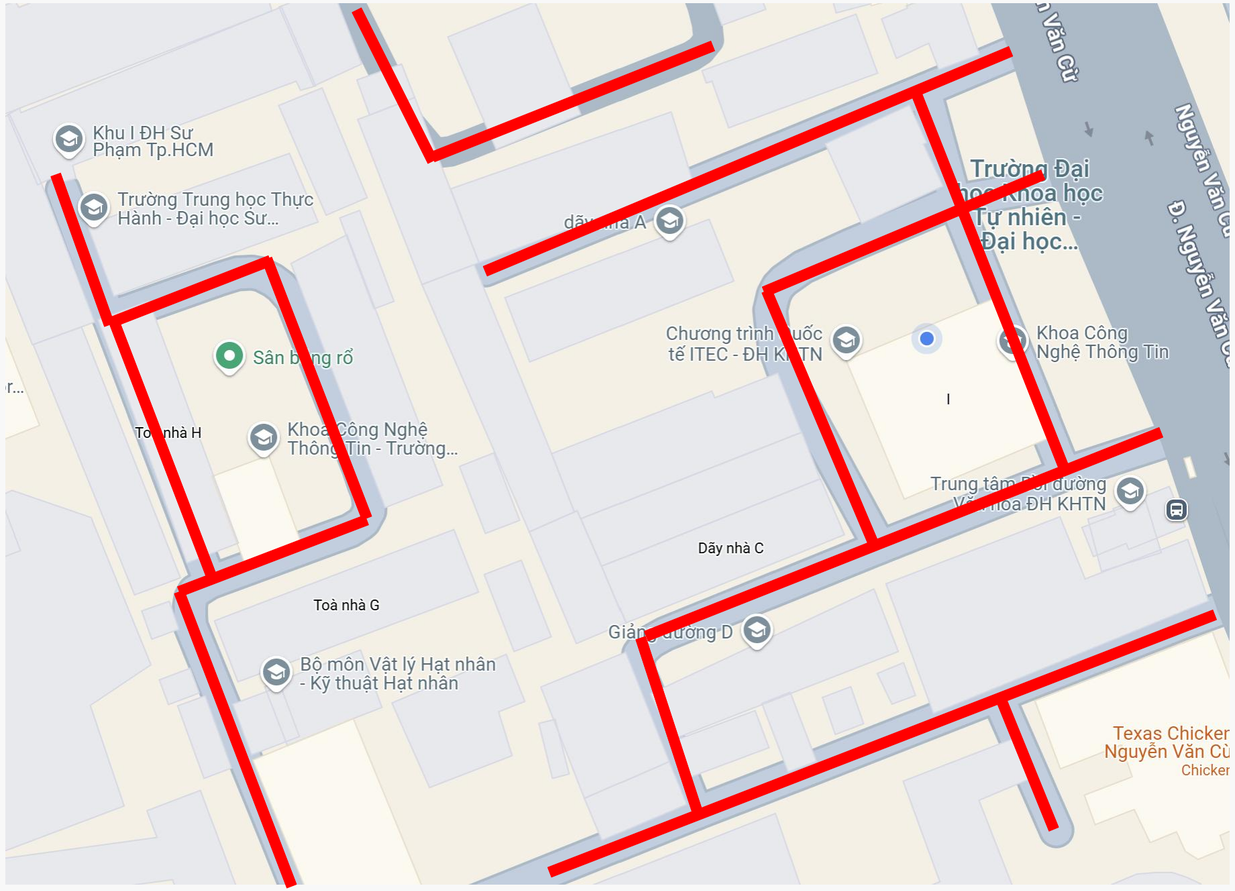
Traverse the tree to find the closest object to a given query point.



O(n)

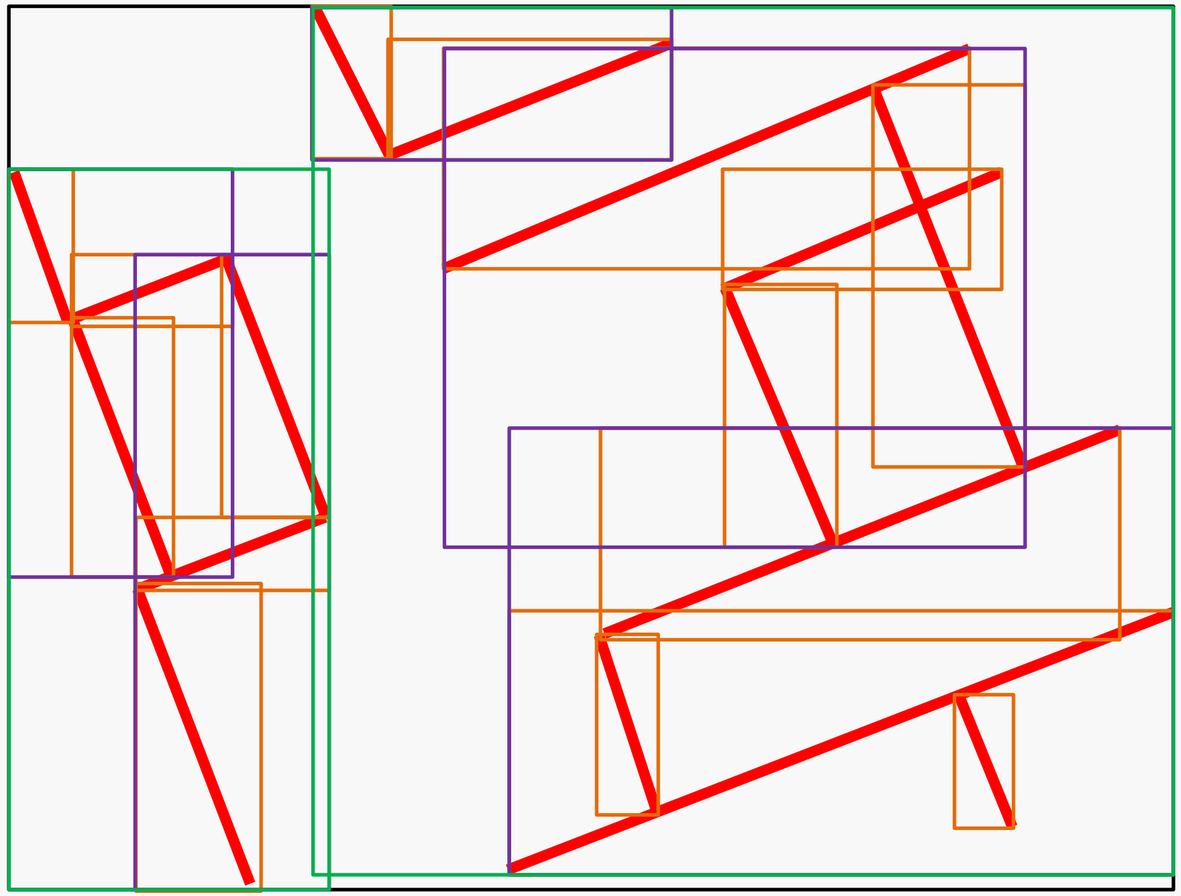
Geographic Information Systems

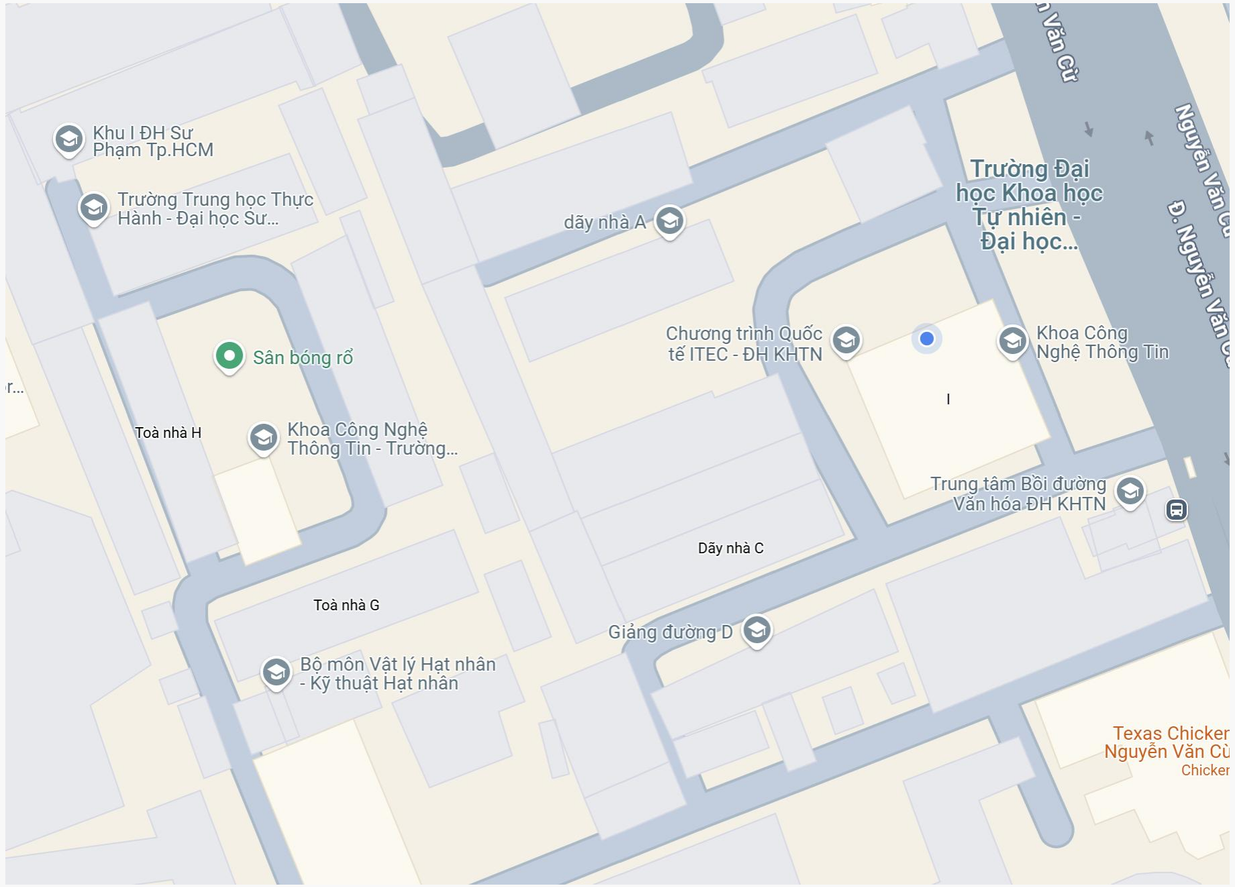
Maps and navigation systems.



Geographic Information Systems

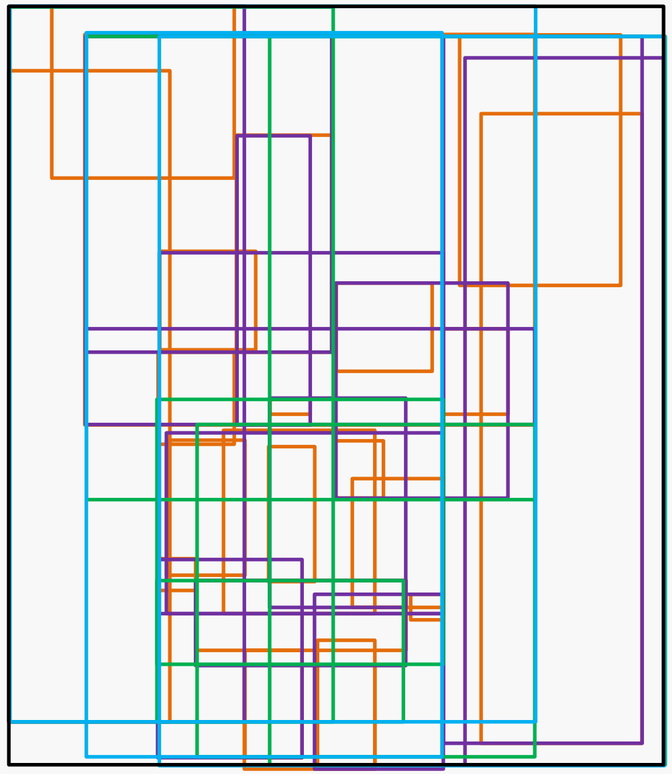
Maps and navigation systems.





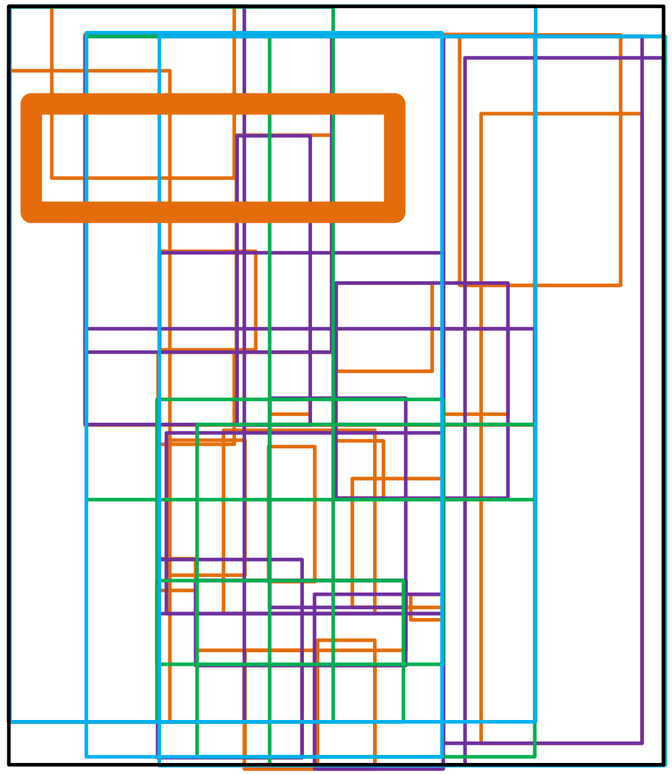
Computer Graphics

Efficient rendering and collision detection.



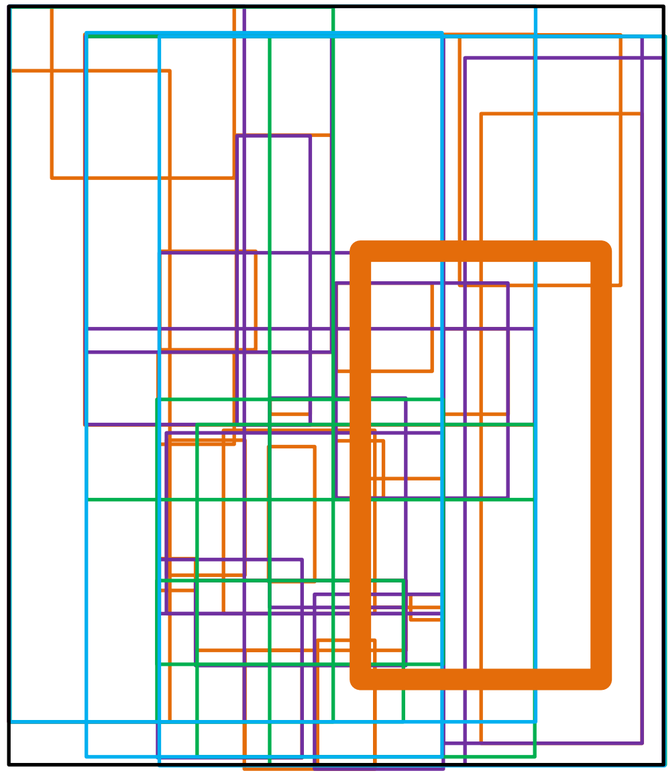
Computer Graphics

Efficient rendering and collision detection.



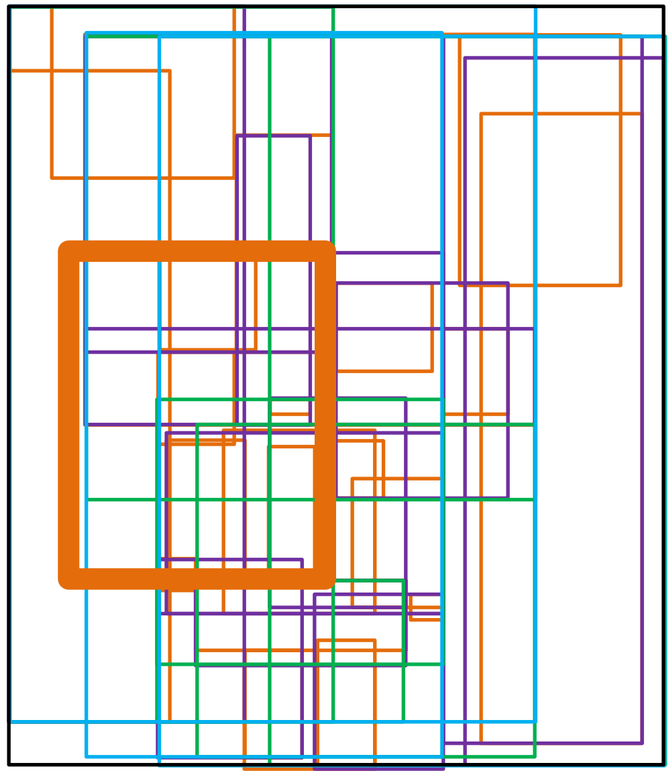
Computer Graphics

Efficient rendering and collision detection.



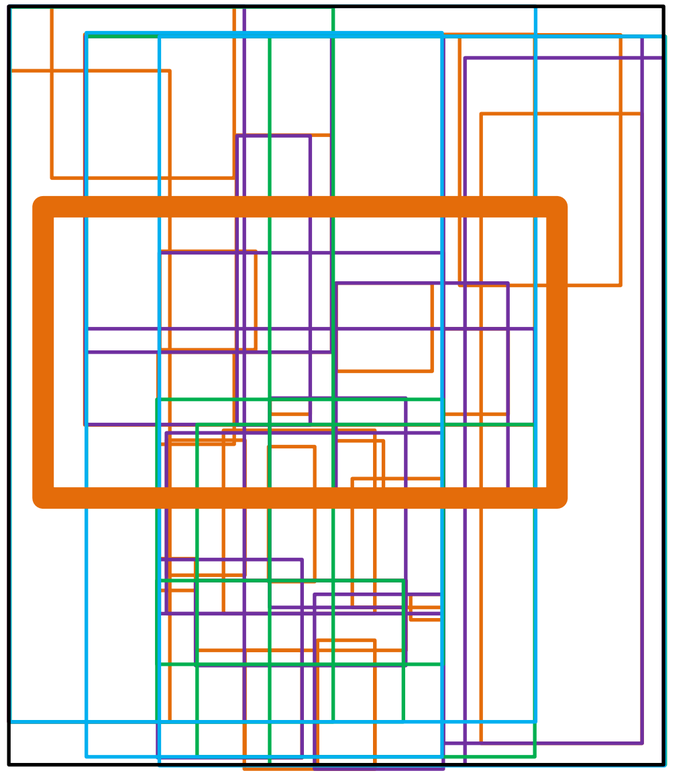
Computer Graphics

Efficient rendering and collision detection.



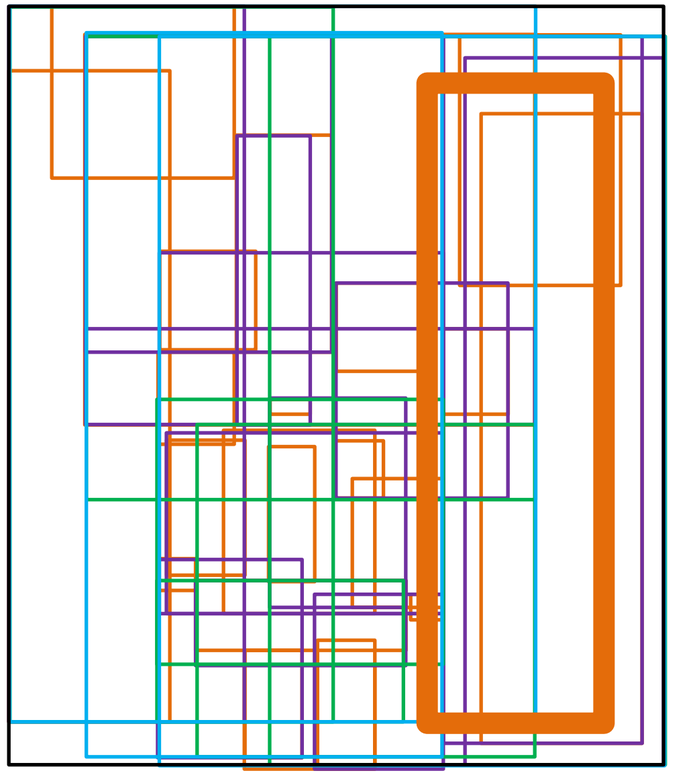
Computer Graphics

Efficient rendering and collision detection.



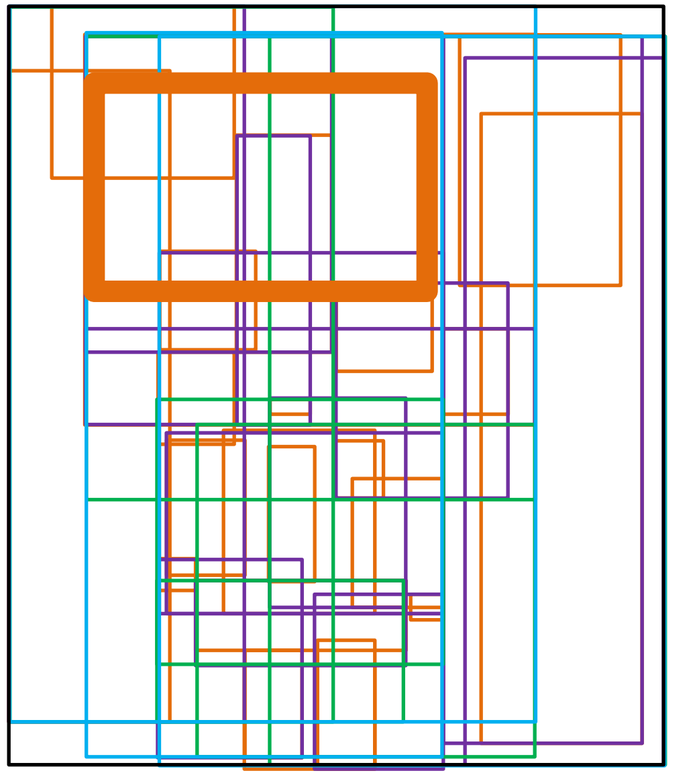
Computer Graphics

Efficient rendering and collision detection.



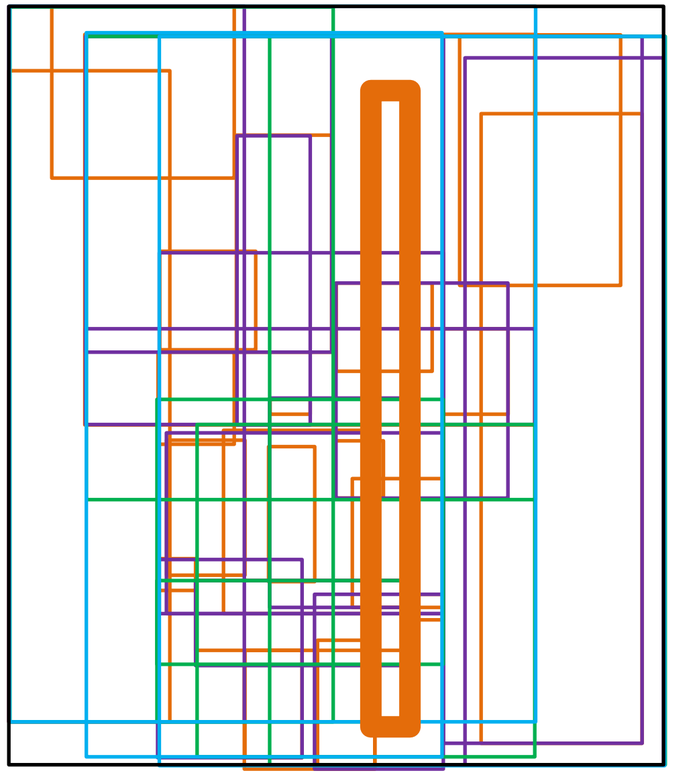
Computer Graphics

Efficient rendering and collision detection.



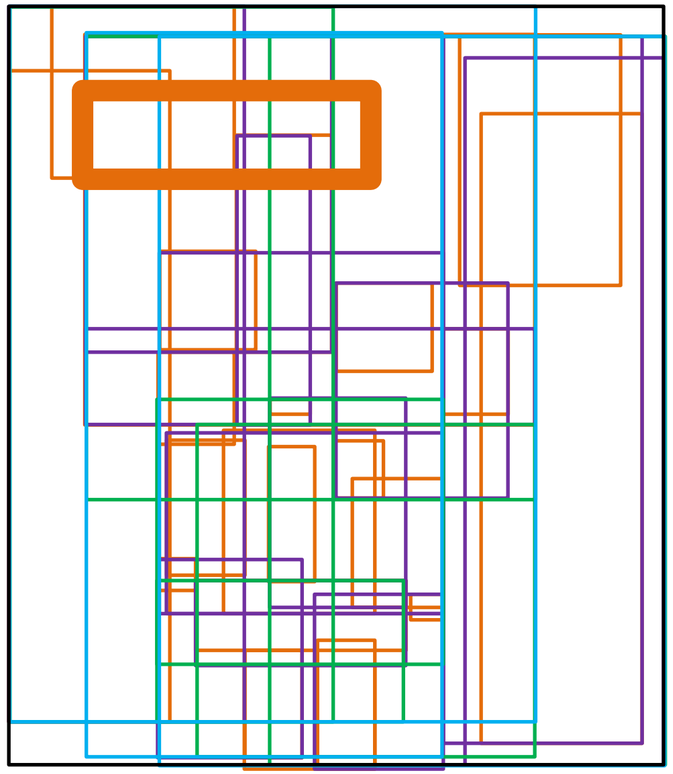
Computer Graphics

Efficient rendering and collision detection.



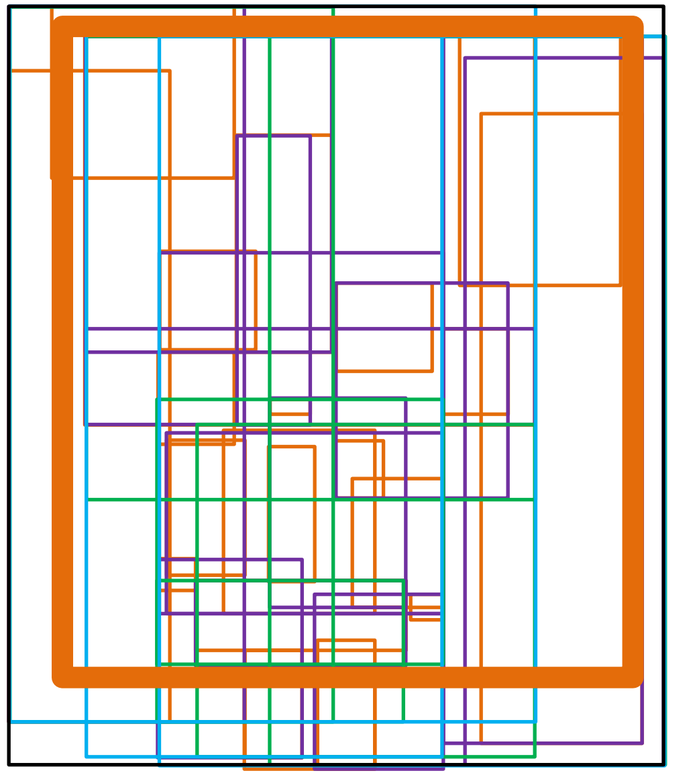
Computer Graphics

Efficient rendering and collision detection.



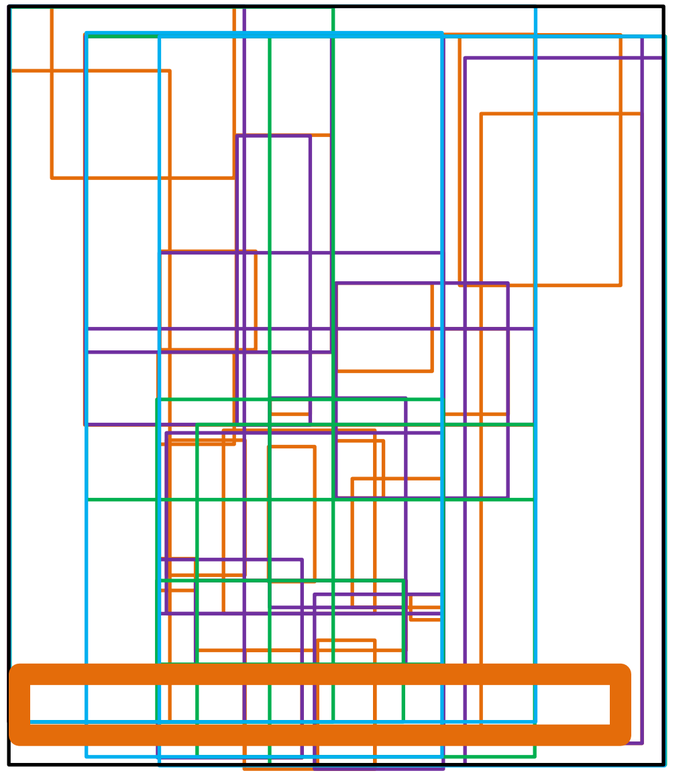
Computer Graphics

Efficient rendering and collision detection.



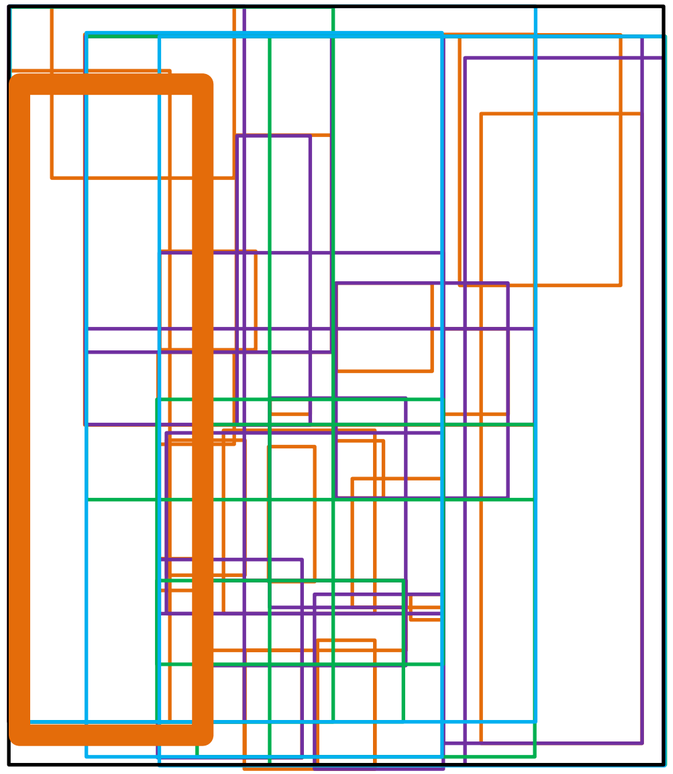
Computer Graphics

Efficient rendering and collision detection.



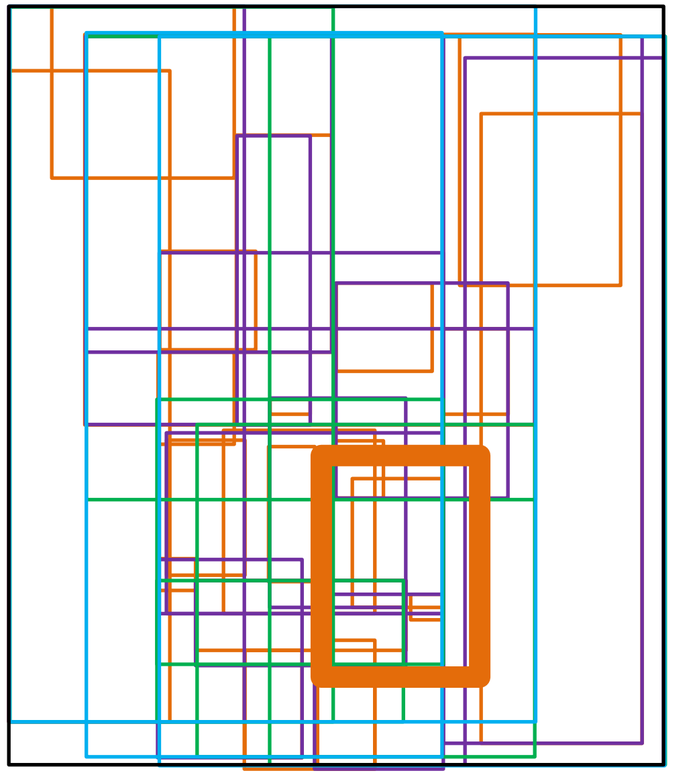
Computer Graphics

Efficient rendering and collision detection.



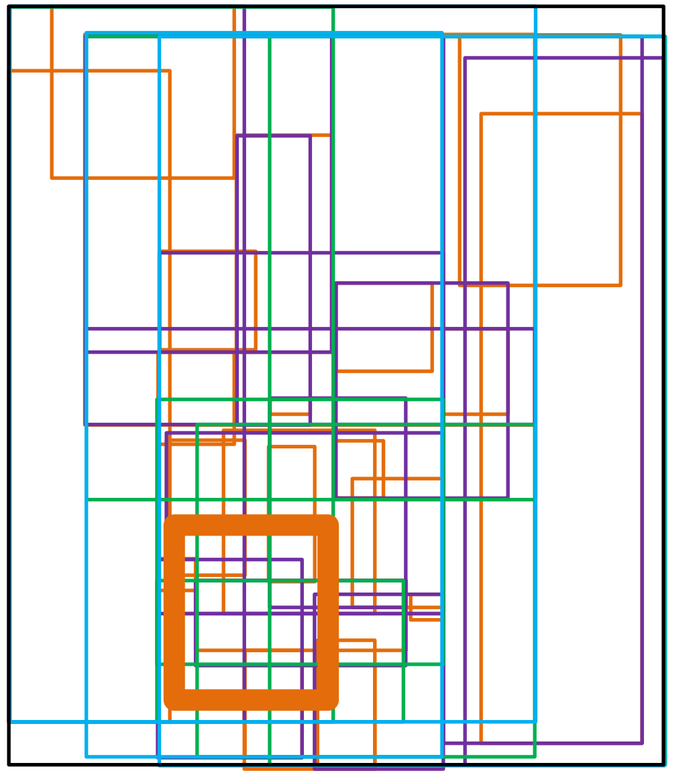
Computer Graphics

Efficient rendering and collision detection.



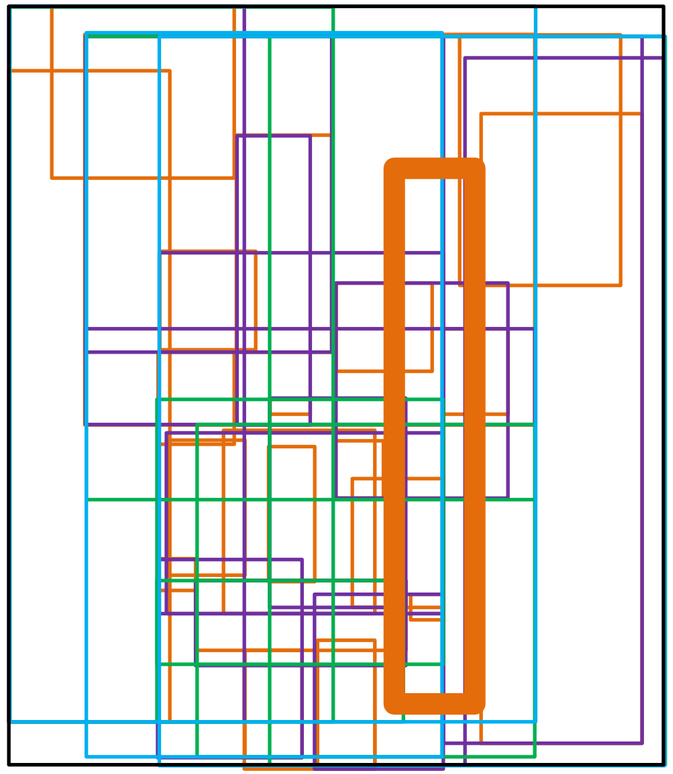
Computer Graphics

Efficient rendering and collision detection.



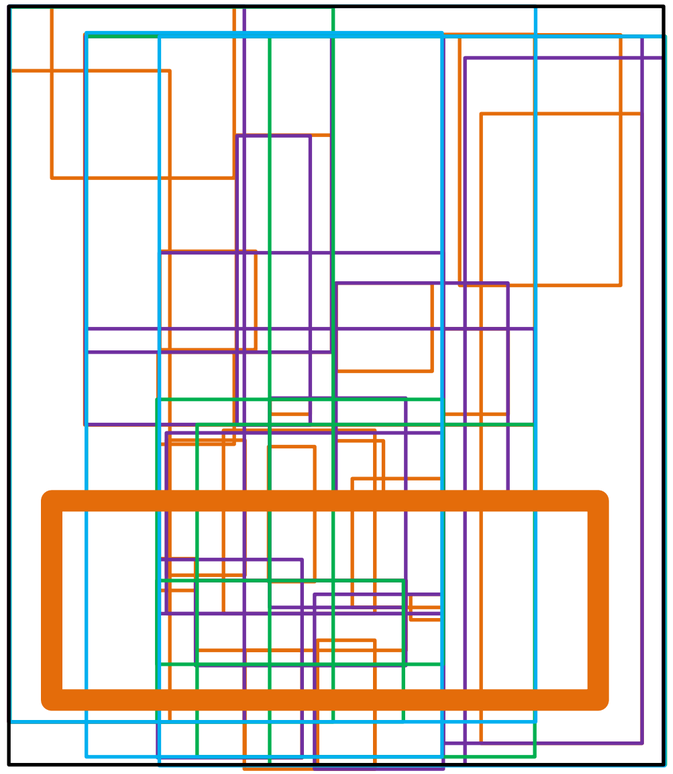
Computer Graphics

Efficient rendering and collision detection.



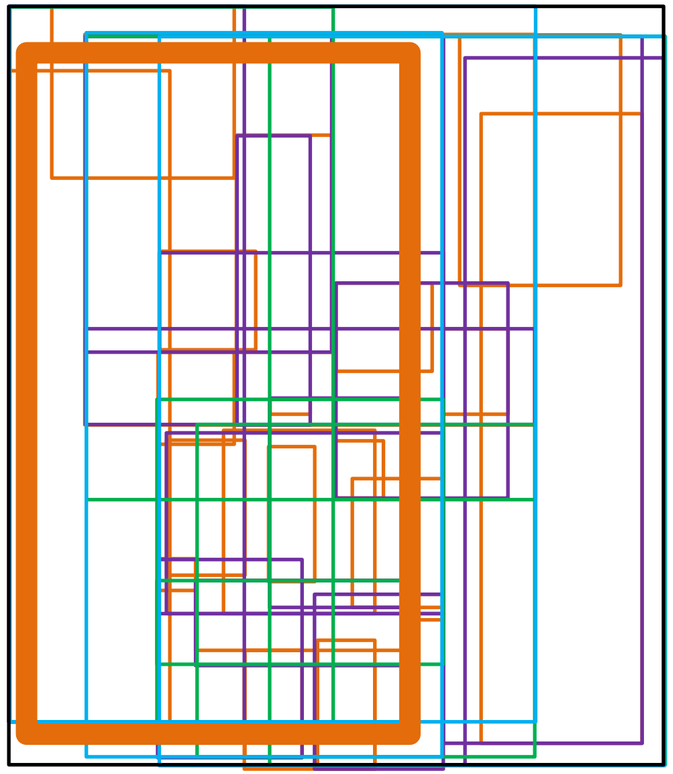
Computer Graphics

Efficient rendering and collision detection.



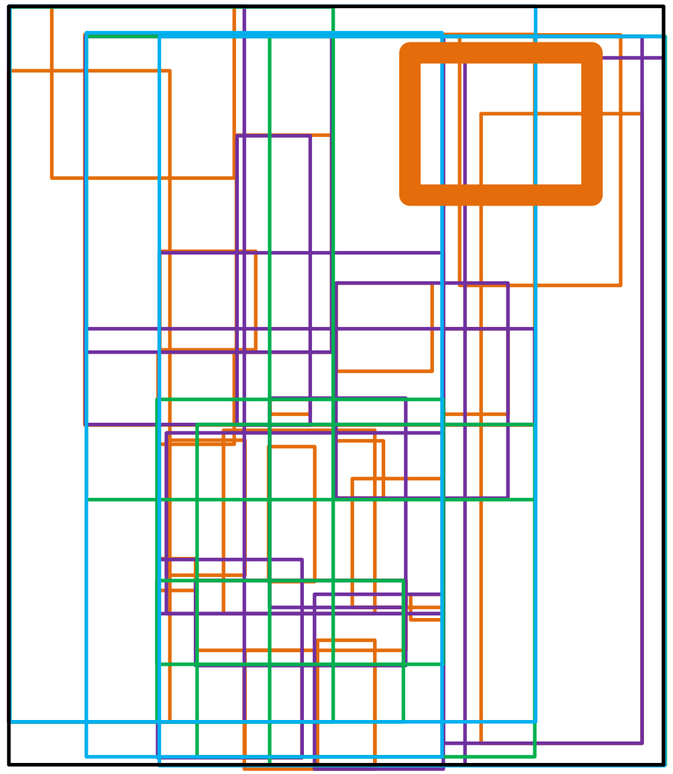
Computer Graphics

Efficient rendering and collision detection.



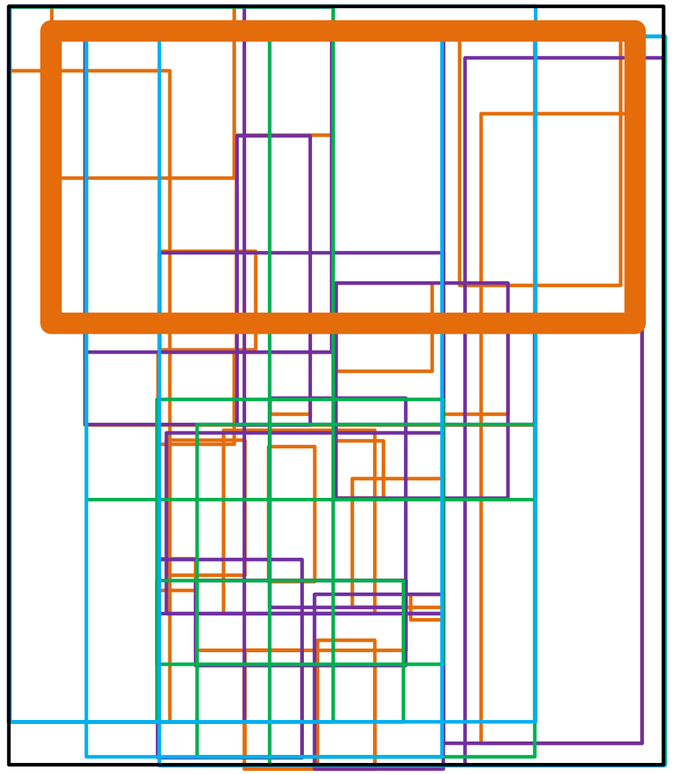
Computer Graphics

Efficient rendering and collision detection.



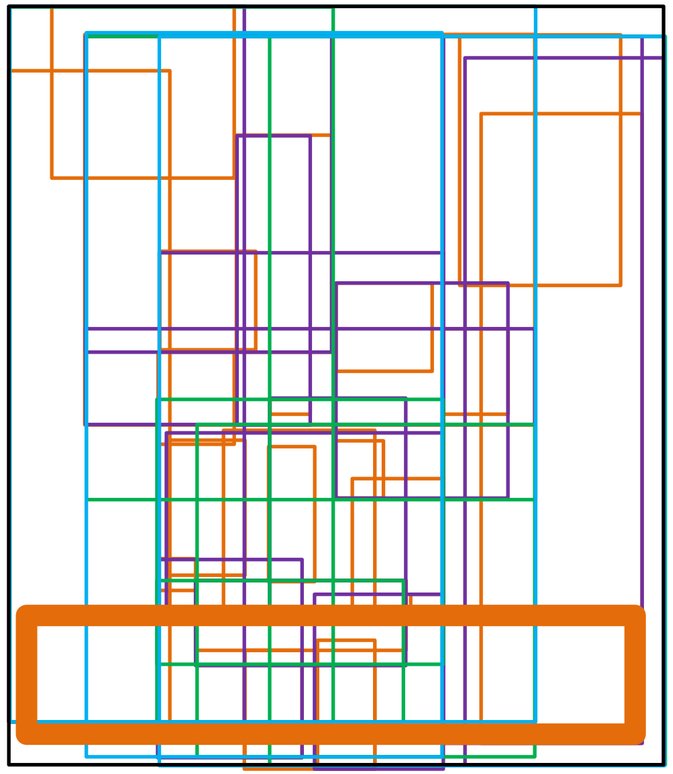
Computer Graphics

Efficient rendering and collision detection.



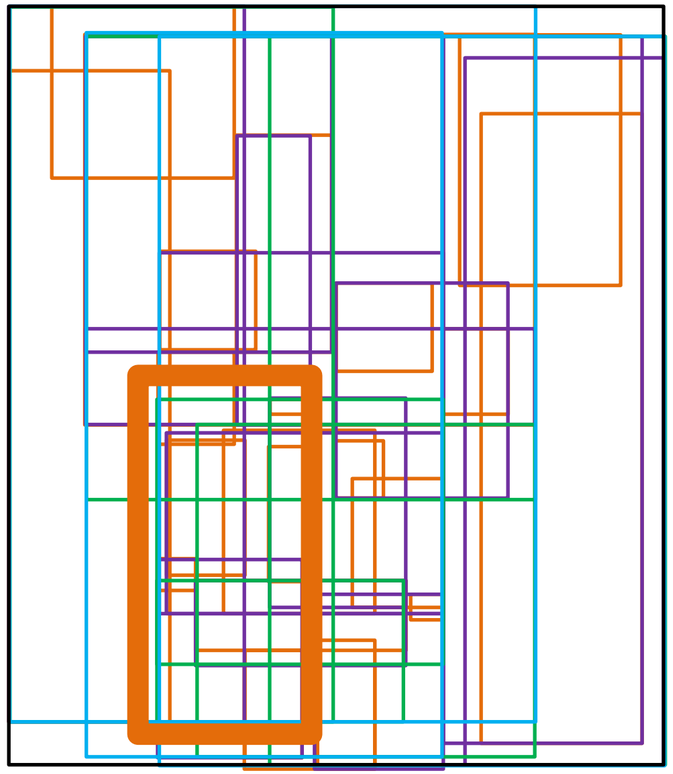
Computer Graphics

Efficient rendering and collision detection.



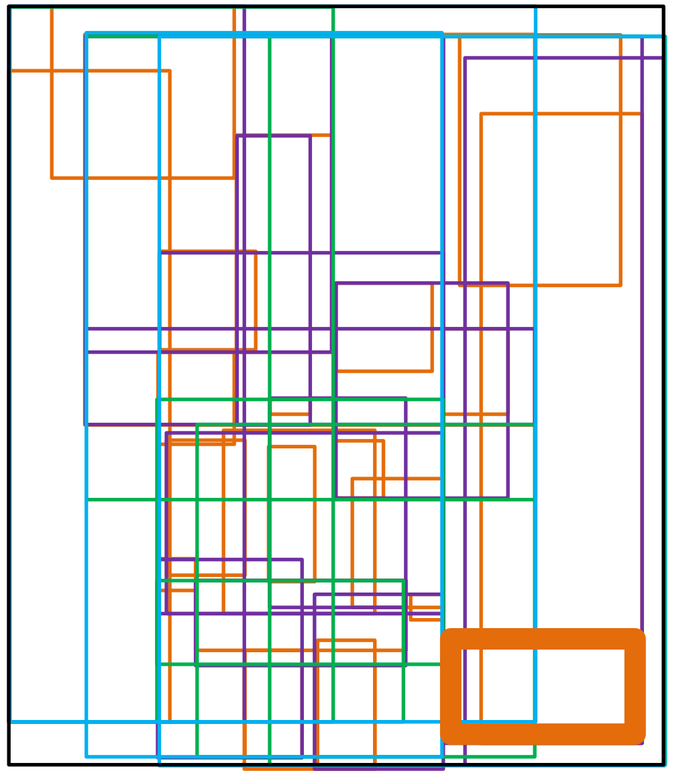
Computer Graphics

Efficient rendering and collision detection.



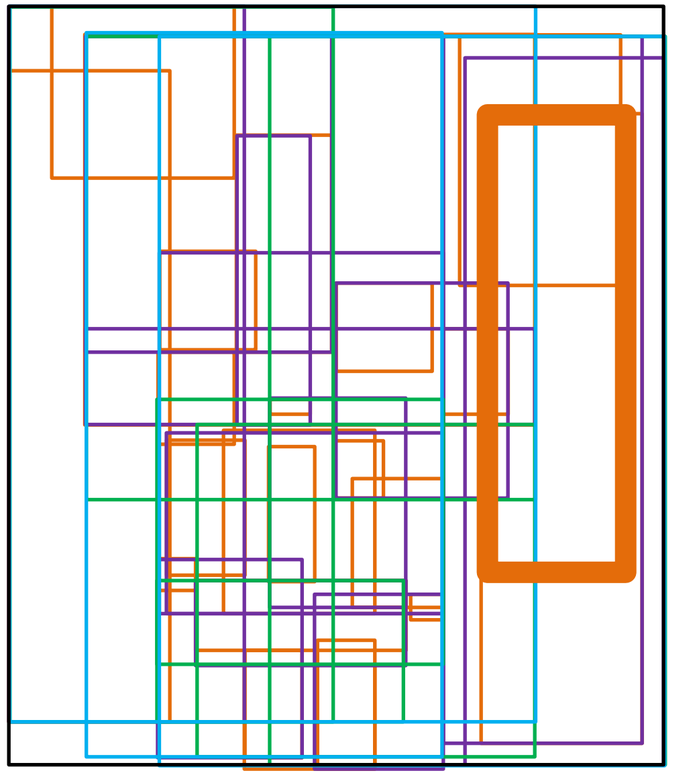
Computer Graphics

Efficient rendering and collision detection.



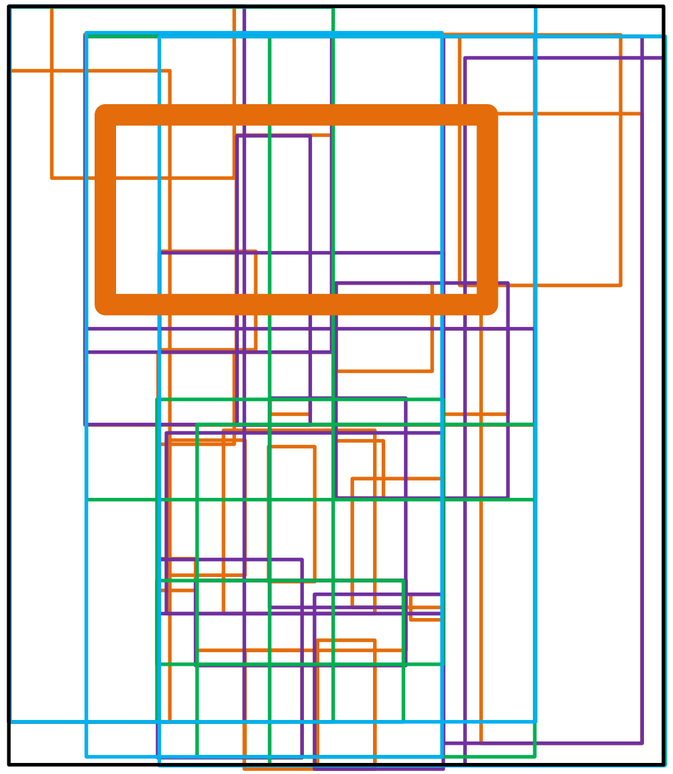
Computer Graphics

Efficient rendering and collision detection.



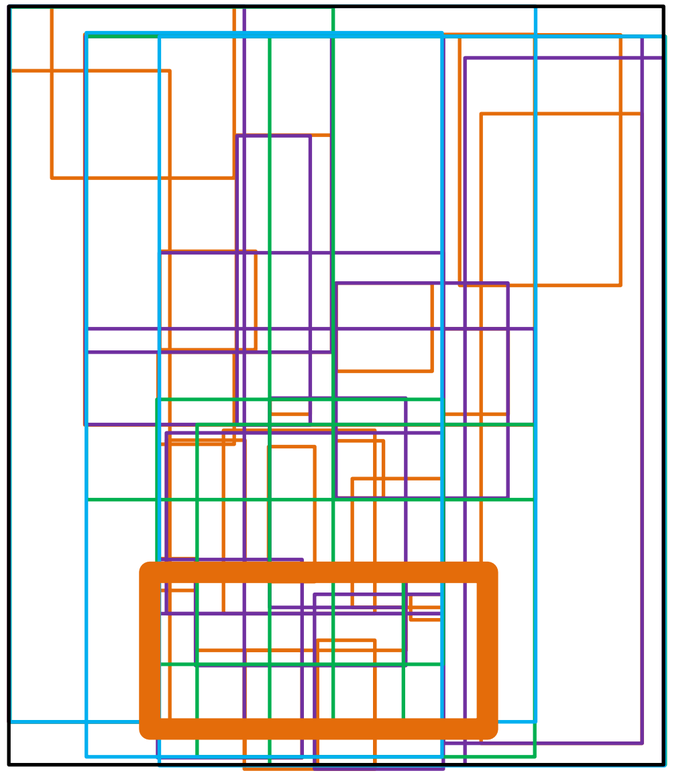
Computer Graphics

Efficient rendering and collision detection.



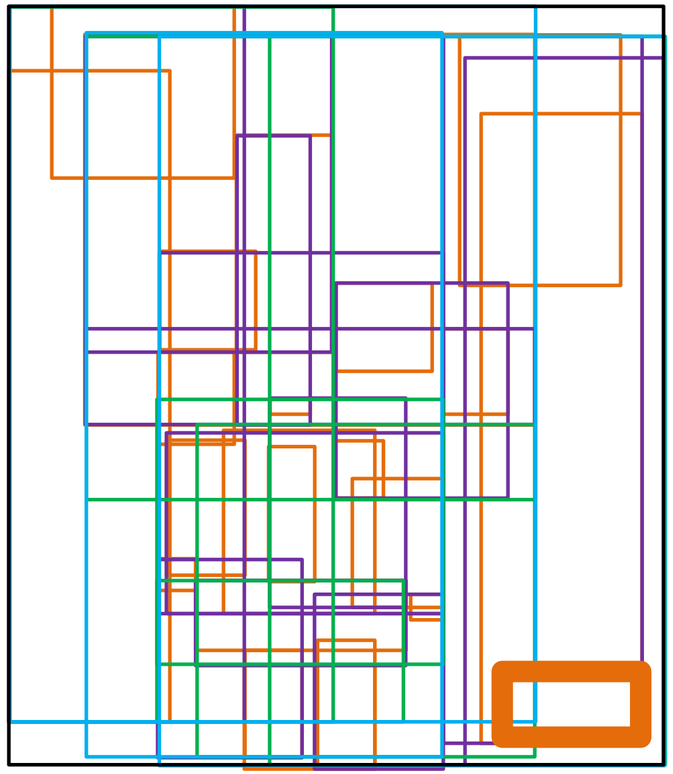
Computer Graphics

Efficient rendering and collision detection.



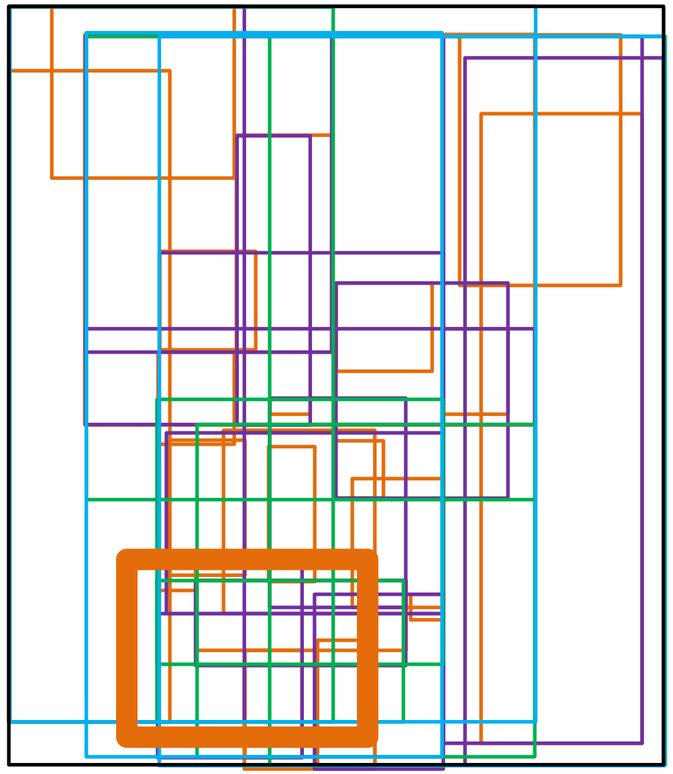
Computer Graphics

Efficient rendering and collision detection.



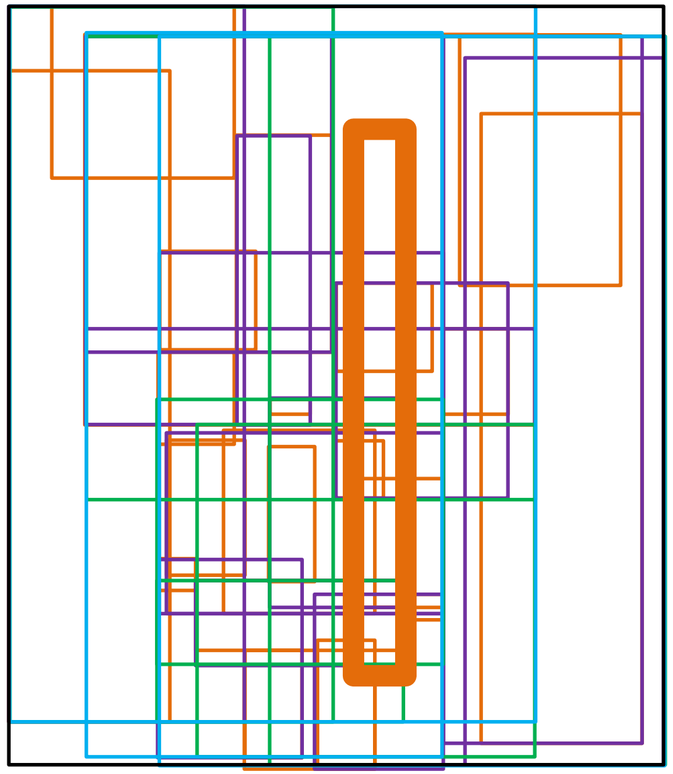
Computer Graphics

Efficient rendering and collision detection.



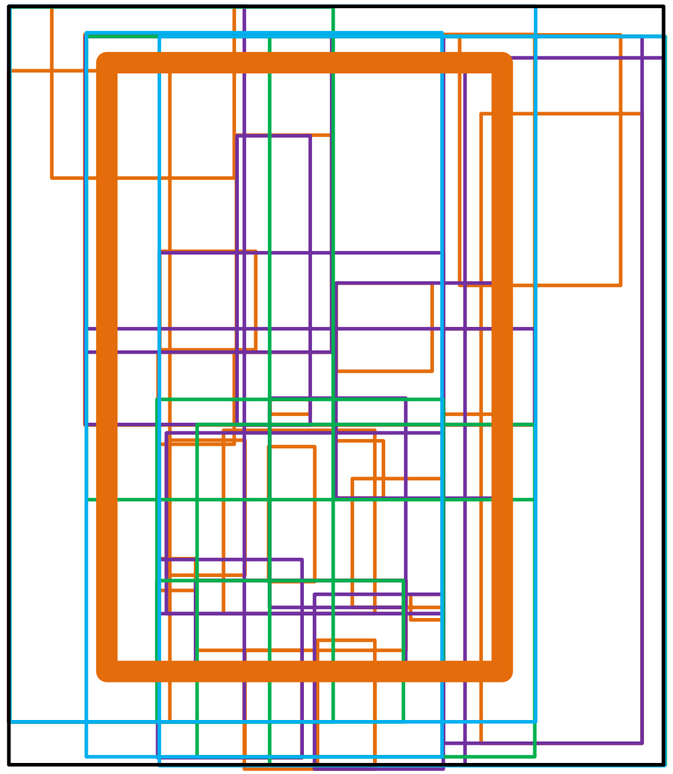
Computer Graphics

Efficient rendering and collision detection.



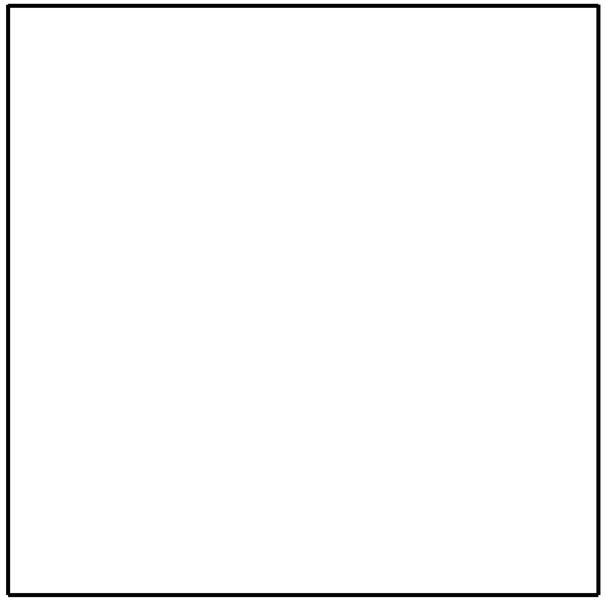
Computer Graphics

Efficient rendering and collision detection.



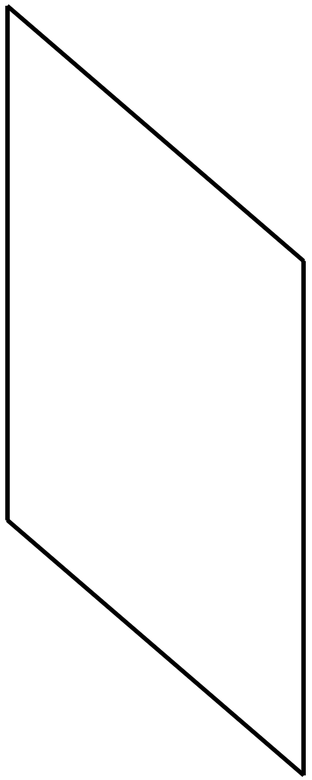
Database Systems

Managing multidimensional spatial data.



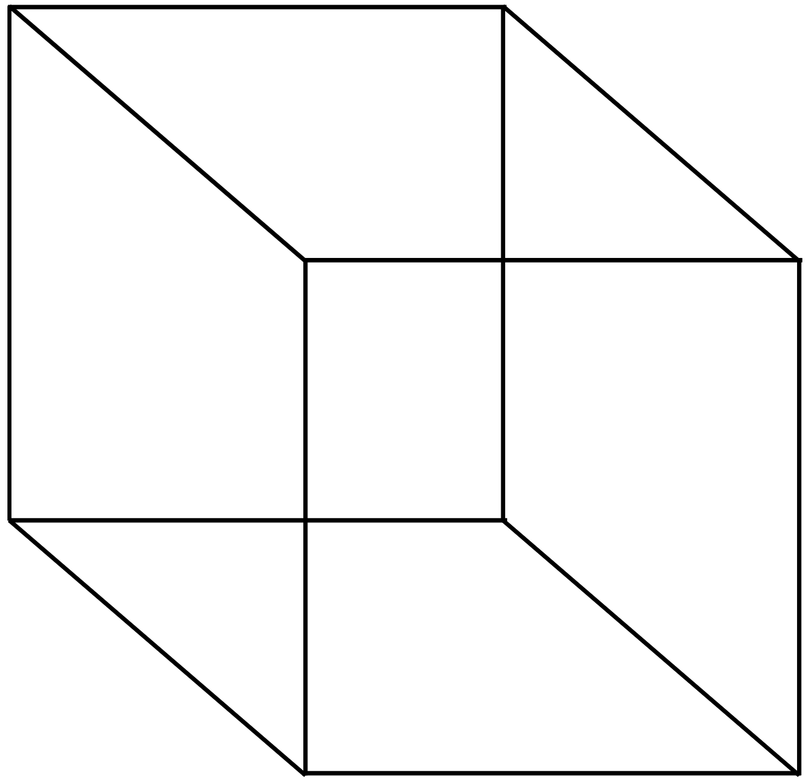
Database Systems

Managing multidimensional spatial data.



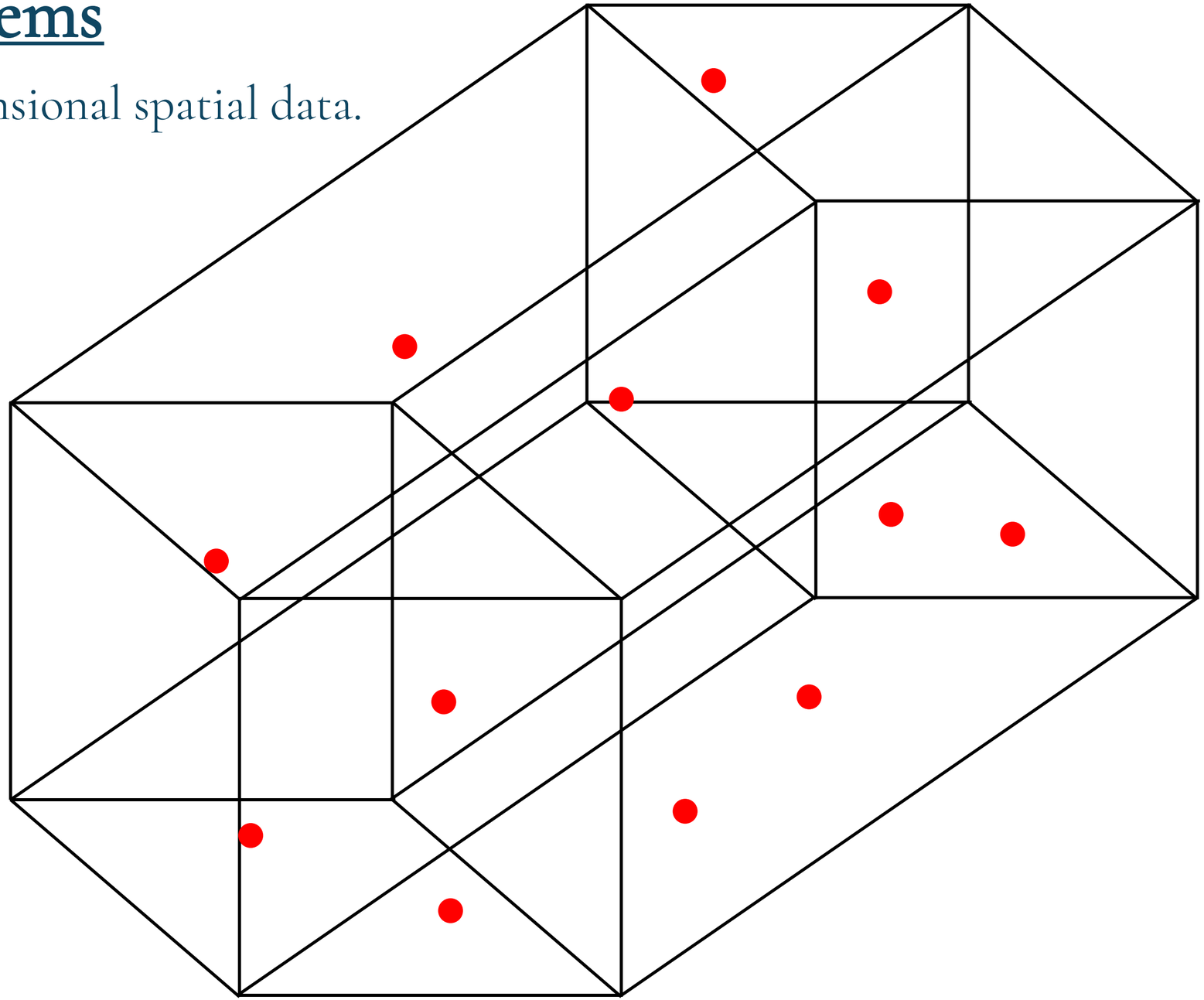
Database Systems

Managing multidimensional spatial data.



Database Systems

Managing multidimensional spatial data.



Quizzes

VI. Quizzes

Question 1:

What is the primary purpose of an R-tree data structure?

A) To store one-dimensional numerical data.

B) To manage multi-dimensional spatial data.

C) To store text-based data.

D) To perform fast searching on strings.

VI. Quizzes

Question 1:

What is the primary purpose of an R-tree data structure?

A) To store one-dimensional numerical data.

B) To manage multi-dimensional spatial data.

C) To store text-based data.

D) To perform fast searching on strings.

Question 2:

In an R-tree, the leaf nodes contain:

A) Only pointers to the child nodes.

B) Bounding boxes that cover regions.

C) Actual spatial data or references to them.

D) Pairs of keys and values.

Question 2:

In an R-tree, the leaf nodes contain:

A) Only pointers to the child nodes.

B) Bounding boxes that cover regions.

C) Actual spatial data or references to them.

D) Pairs of keys and values.

Question 3:

What type of data is most commonly indexed using an R-tree?

A) Linear numerical data.

B) Multi-dimensional spatial data (e.g., geographic coordinates, rectangles).

C) Text strings.

D) Time series data.

Question 3:

What type of data is most commonly indexed using an R-tree?

A) Linear numerical data.

B) Multi-dimensional spatial data (e.g., geographic coordinates, rectangles).

C) Text strings.

D) Time series data.

Question 4:

Which of the following is a key characteristic of an R-tree?

A) It is always unbalanced.

B) It can only handle one-dimensional data.

C) It uses bounding boxes to represent groups of spatial objects.

D) It stores data as linked lists.

Question 4:

Which of the following is a key characteristic of an R-tree?

A) It is always unbalanced.

B) It can only handle one-dimensional data.

C) It uses bounding boxes to represent groups of spatial objects.

D) It stores data as linked lists.

Programming assignments

VI. Programming assignments

Count MBRs in an R-Tree (Difficulty: )

In an R-Tree, each node contains a **Minimum Bounding Rectangle (MBR)** — the smallest axis-aligned rectangle that completely contains all its child MBRs or data rectangles. The R-Tree consists of internal nodes and leaf nodes:

- Leaf nodes contain data entries, each with its own MBR.
- Internal nodes contain children nodes, each with their own MBR.
- Every node, including the root, has an MBR that encompasses all its children's MBRs.

Your task is to count the **total number of MBRs** in the entire R-Tree.

Constraint: The R-Tree has a maximum fanout (number of children per node) of 4. The number of nodes in the tree does not exceed 10,000.

```cpp
//Exercise 01
int RTree::count_mbrs() {
    return count_mbrs_recursive(root);
}

int RTree::count_mbrs_recursive(Node* node) {
    if (!node) return 0;
    if (node->is_leaf) {
        return node->children.size(); //Leaf nodes store MBRs directly
    }
    int total=0;
    for (Node* child : node->children) {
        total += count_mbrs_recursive(child);
    }
    return total;
}
```

VI. Programming assignments

Find Overlapping MBRs in a Range (Difficulty: 😳)

In an R-Tree, each node has a Minimum Bounding Rectangle (MBR). Two MBRs are considered overlapping if their rectangular regions intersect in 2D space.

Given:

- The root of a valid R-Tree.
- A query rectangle (with coordinates x_min, y_min, x_max, y_max).

You are asked to find and return all MBRs that intersect with the query range, along with their coordinates.

Constraint: The R-Tree is balanced and has at least 10,000 nodes. The maximum fanout per node is 4.

```cpp
//Exercise 02
void RTree::count_overlaps(const MBR& range) {
    vector<Node*> overlaps;
    search_recursive(root, range, overlaps);
    cout << "Number of MBRs overlapping with (" << range.x_min << ", " << range.y_min << ", " << range.x_max << ", " << range.
    y_max << "): " << overlaps.size() << std::endl;
    for (const Node* node : overlaps) {
        cout << "  Overlapping MBR: (" << node->mbr.x_min << ", " << node->mbr.y_min << ", " << node->mbr.x_max << ", " <<
        node->mbr.y_max << ")\n";
    }
}
```

VI. Programming assignments

Calculate Total Area Covered by MBRs (Difficulty: )

In an R-Tree, each leaf node contains data entries represented by Minimum Bounding Rectangles (MBRs). The root node also has an MBR, which covers the union of all entries below.

Your task is to compute:

- The sum of areas of all leaf-level MBRs (data entries).
- The area of the root MBR.

Constraint: The R-Tree is balanced and has at least 10,000 nodes. The maximum fanout per node is 4.

MBR coordinates may be floating-point numbers (e.g., x_min = 1.234567).

```cpp
//Exercise 03
void RTree::total_area() {
    if (root->children.empty()) {
        std::cout << "Tree is empty. Root area: 0, Total leaf area: 0" << std::endl;
        return;
    }
    int root_area = root->mbr.area();
    int leaf_total = total_area_recursive(root);
    std::cout << "Root MBR Area (" << root->mbr.x_min << ", " << root->mbr.y_min << ", " << root->mbr.x_max << ", " <<
    root->mbr.y_max << "): " << root_area << std::endl;
    std::cout << "Total Leaf MBR Area: " << leaf_total << std::endl;
}

int RTree::total_area_recursive(Node* node) {
    if (node->is_leaf) {
        int total = 0;
        for (Node* child : node->children) {
            total += child->mbr.area();
        }
        return total;
    }
    int total = 0;
    for (Node* child : node->children) {
        total += total_area_recursive(child);
    }
    return total;
}
```

VI. Programming assignments

Calculate Total Area Covered by MBRs (Difficulty: )

In an R-Tree, the depth is defined as the maximum number of levels (or layers) from the root node down to any leaf node.

You are given the root of an R-Tree.

Your task is to compute the maximum depth of the tree.

Constraint: The R-Tree is balanced and has at least 10,000 nodes. The maximum fanout per node is 4.

```cpp
//Exercise 04
int RTree::tree_depth() {
    if (root->children.empty())
        return 0;
    return tree_depth_recursive(root);
}

int RTree::tree_depth_recursive(Node* node) {
    if (node->is_leaf)
        return 1;    //Leaf is at depth 1
    int max_depth = 0;
    for (Node* child : node->children) {
        int child_depth = tree_depth_recursive(child);
        if (child_depth > max_depth)
            max_depth = child_depth;
    }
    return 1 + max_depth;   //Add 1 for current level
}
```

Thank you for listening our presentation