Location grpc 介绍

zhaoweipu@p1staff.com

Contents

Not Included:

- PostgreSQL PostGIS

Included:

- R-Tree
- GeoJson
- tile38 Set, Intersects command along with our region query
- Nearby, KNN, Haversine
- Brief intro to other tile38 functions

Background

- From lon, lat to region

```
[developer@staging1.p1staff.com ~]$grpcurl -plaintext -d '{"language": 1, "location":{"longitude":116.445711, "latitude":39.912763}}' 127.0.0.1:21520 region.LocationRegion.GetRegion
{
  "region": {
    "country": {
      "id": "3142",
      "name": "中国"
    },
    "province": {
      "id": "3121",
      "name": "北京"
    },
    "city": {
      "id": "2890",
      "name": "北京"
    },
    "district": {
      "id": "2236",
      "name": "朝阳"
    }
  }
}
```

- Nearby (not in Location grpc yet)

Background

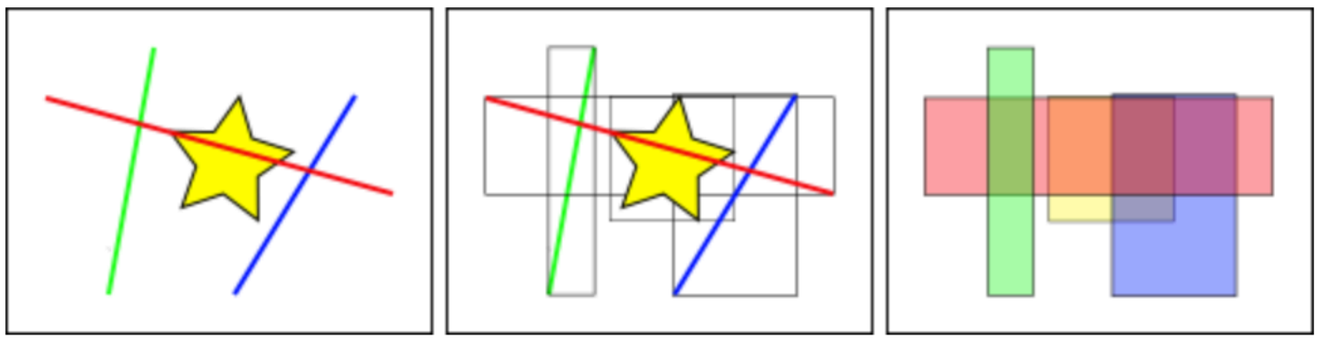
With PostgreSQL PostGIS

```
#+BEGIN_SRC go
  regionColumns                = `id, type, name_zh, name_en, name_ko, name_ja, parent_id, status, ST_X(center_bd) as c_lon, ST_Y(center_bd) as c_lat`
  selectRegionsByLocationQuery = `SELECT ` + regionColumns + ` FROM regions WHERE ST_Contains(boundary_bd, ST_POINT(?,?))`
#+END_SRC
#+BEGIN_SRC sql
  CREATE TYPE yay.region_type AS ENUM (
      'country',
      'province',
      'city',
      'district'
  );

  CREATE TABLE yay.regions (
      id integer NOT NULL,                                    --region的主键，目前业务中在使用，这个暂时不要变更
      type yay.region_type NOT NULL,                          -- 这行数据的类型：国，省，市，区
      boundary_bd public.geometry,                            -- region的地理围栏
      center_bd public.geometry,                              -- region的几何中心经纬度坐标
      name_zh character varying NOT NULL,                     -- 中文
      status yay.status DEFAULT 'default'::yay.status NOT NULL,   -- 状态目前没用，默认default就行
      name_en character varying NOT NULL,                     -- 上级的region主键id,例如，当前行type是city，那么parent_id指向province的id
      parent_id integer DEFAULT 0 NOT NULL,
      name_ko character varying NOT NULL,
      name_ja character varying NOT NULL
  );

  CREATE INDEX regions_boundary_gix ON yay.regions USING gist (boundary_bd);

  `SELECT ` + regionColumns + ` FROM regions WHERE ST_Contains(boundary_bd, ST_POINT(?,?))`
#+END_SRC
```

PostGIS using R-Tree

15.1. How Spatial Indexes Work

Standard database indexes create a hierarchical tree based on the values of the column being indexed. Spatial indexes are a little different – they are unable to index the geometric features themselves and instead index the bounding boxes of the features.



In the figure above, the number of lines that intersect the yellow star is **one**, the red line. But the bounding boxes of features that intersect the yellow box is **two**, the red and blue ones.

The way the database efficiently answers the question “what lines intersect the yellow star” is to first answer the question “what boxes intersect the yellow box” using the index (which is very fast) and then do an exact calculation of “what lines intersect the yellow star” **only for those features returned by the first test**.

For a large table, this “two pass” system of evaluating the approximate index first, then carrying out an exact test can radically reduce the amount of calculations necessary to answer a query.

Both PostGIS and Oracle Spatial share the same “R-Tree” [1] spatial index structure. R-Trees break up data into rectangles, and sub-rectangles, and sub-sub rectangles, etc. It is a self-tuning index structure that automatically handles variable data density and object size.

https://postgis.net/workshops/postgis-intro/indexing.html

R-Tree

R-Trees A Dynamic Index Structure For Spatial Searching Anotnin Guttman 1984

R-Tree Index Structure

An R-Tree is a height-balanced tree similar to a B-tree with index records in its leaf nodes containing pointers to data objects.

A spatial database consists of a collection of tuples representing spatial objects and each tuple has a unique identifier which can be used to retrieve it.

叶节点：　　包含 (I, tuple-identifier) 形式的 index record entries.
tuple-identifier refers to a tuple in the database,
$I = (I_0, I_1, ., I_{n-1})$,
n: 纬度，　$I_i = [a, b]$ object 在纬度i上的范围

非叶节点:　　包含(I, child-pointer) 形式的 entries.
child-pointer is the address of a lower node in the R-tree
I covers all rectangles in the lower node's entries.

M:　　一个节点所包含的最大entries数目
$m \leq \frac{M}{2}$:　　一个节点所包含的最少entries数目

R-tree 满足如下属性:
1.每一个叶节点包含m到M个 index records, 除非它是root
2.对于叶节点中的每一个(I, tuple-identifier) index record, I 是包含 tuple-identifier 所表示的n维data object的最小的rectangle
3.每一个非叶节点有m到M个children, 除非它是root
4.对于非叶节点中的每一个 entry (I, child-pointer), I 是包含这个child节点的所有rectangles的最小的rectangle
5.root节点包含至少两个children，　除非它是叶节点
6.所有的叶节点都在同一level上

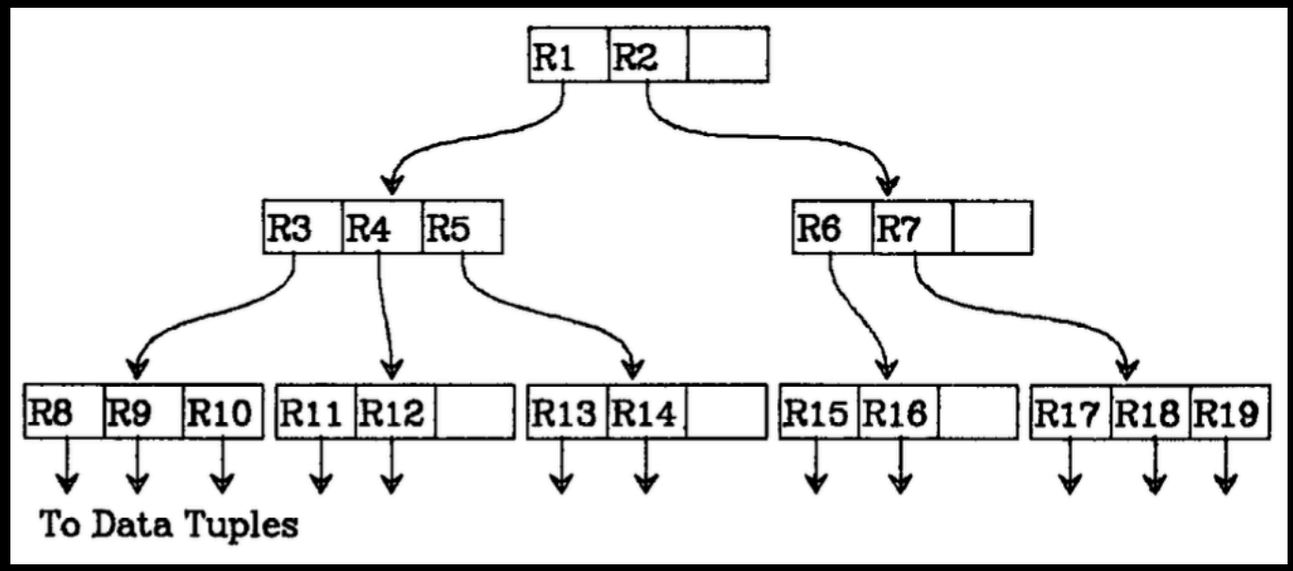
R-Tree

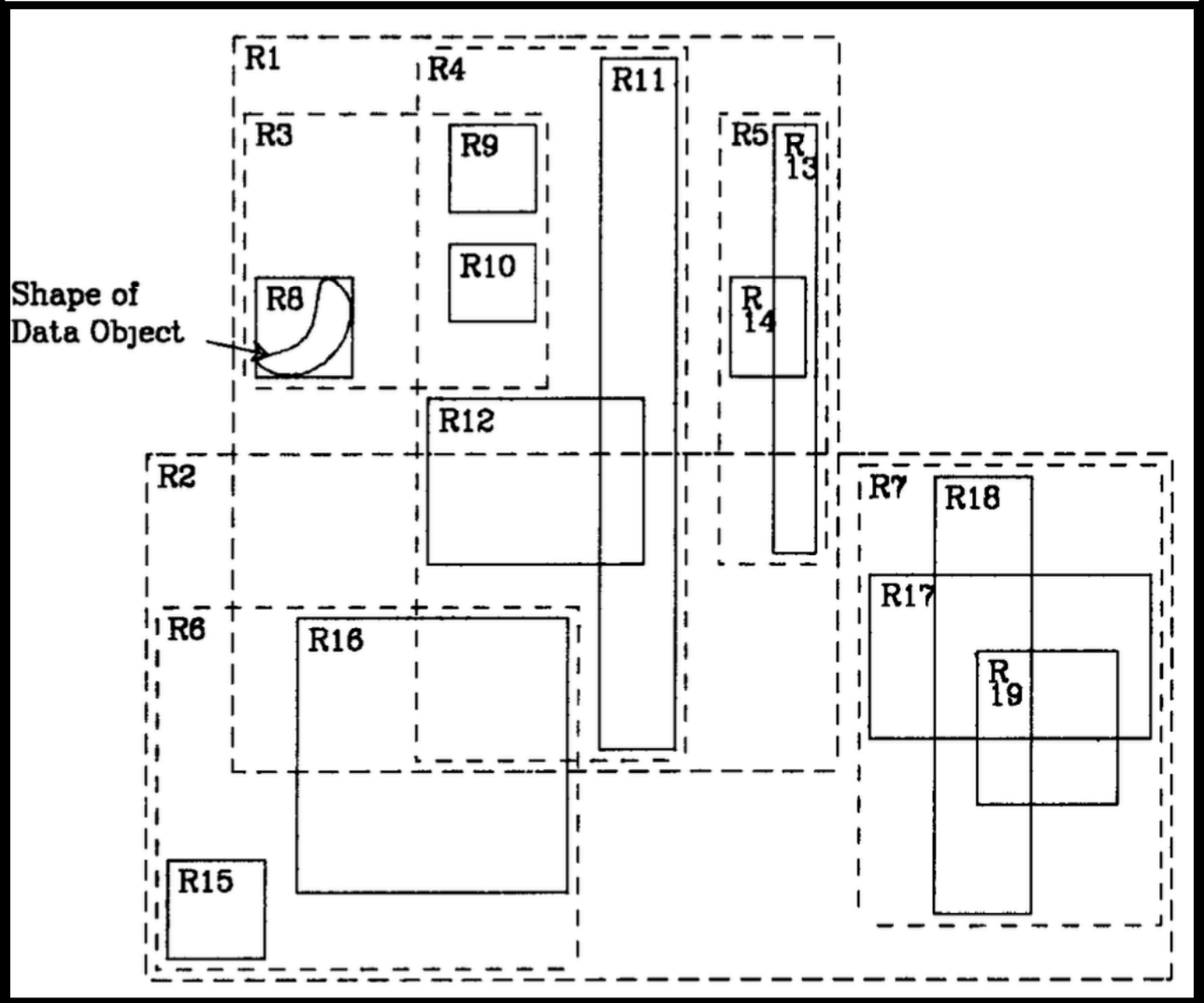
An R-tree containing N index records

Height at most: $|\log_m N| - 1$

Space utilization for all nodes except the root: $\frac{m}{M}$

If nodes have more than 3 or 4 entries the tree is very wide, and almost all the space is used for leaf nodes





R-Tree struct in Tile38

叶节点： 包含 (I, tuple-identifier) 形式的 index record entries.
tuple-identifier refers to a tuple in the database,
$I = (I_0, I_1, ., I_{n-1})$,
n: 纬度， $I_i = [a, b]$ object 在纬度i上的范围

非叶节点: 包含(I, child-pointer) 形式的 entries.
child-pointer is the address of a lower node in the R-tree
I covers all rectangles in the lower node's entries.

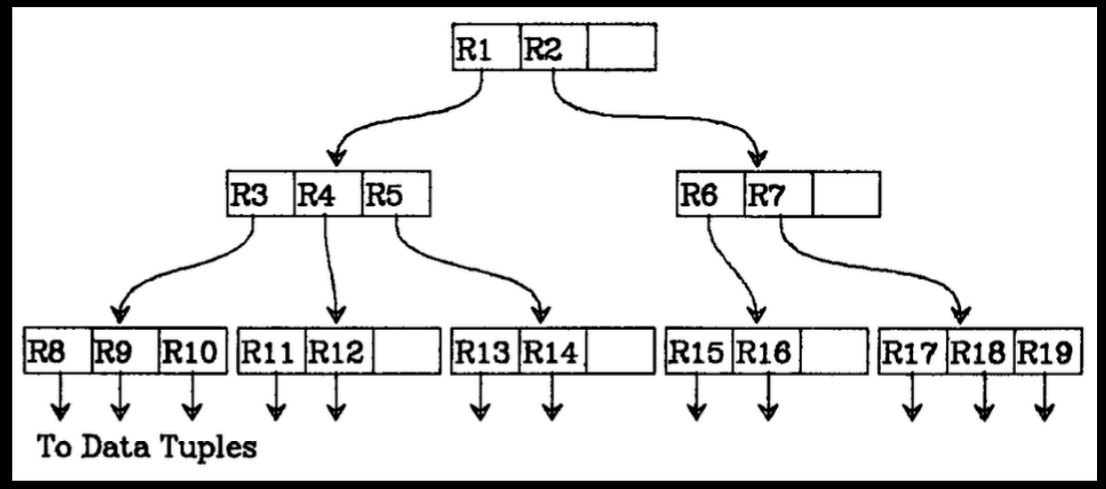
```
const (
    maxEntries = 32
    minEntries = maxEntries * 40 / 100
)

type rect struct {
    min, max [2]float64
    data     interface{}
}

type node struct {
    count int
    boxes [maxEntries + 1]rect
}

// RTree ...
type RTree struct {
    height   int
    root     rect
    count    int
    reinsert []rect
}
```

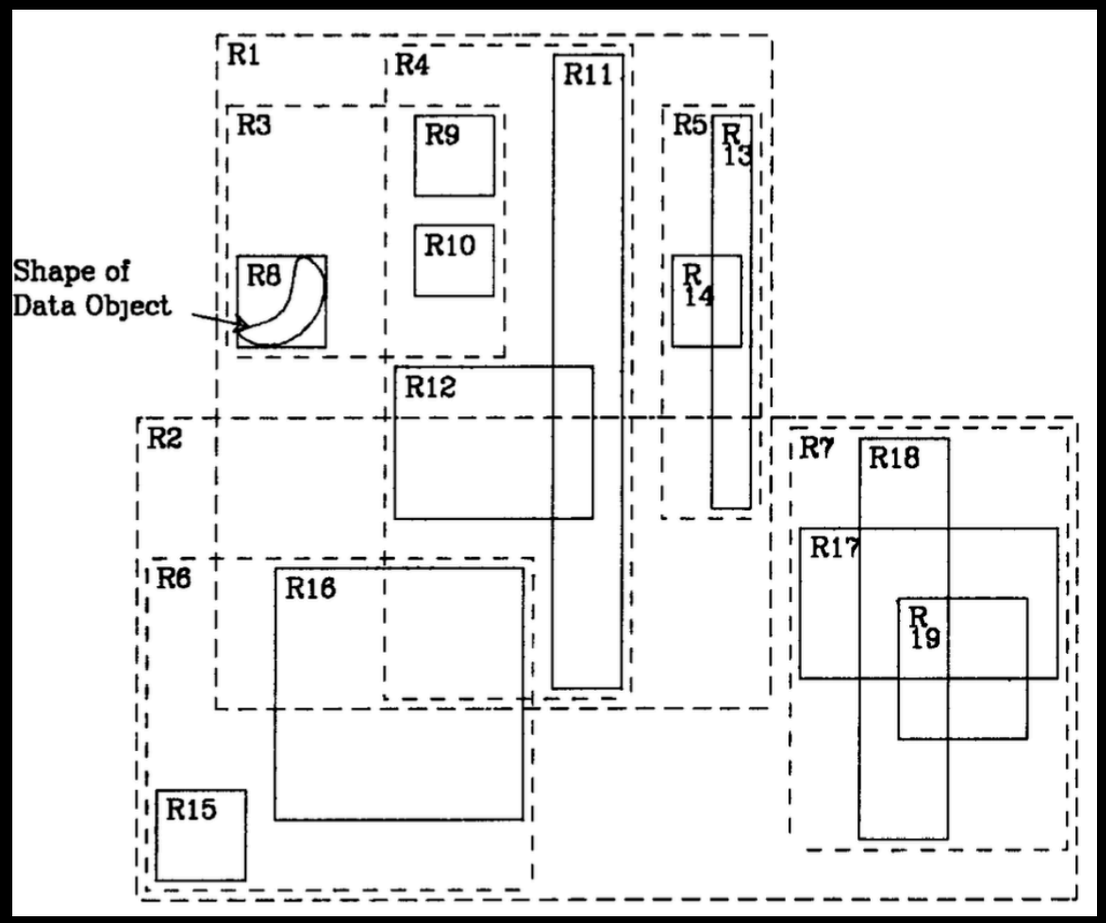
tuple-identifier. (*itemT)

child-pointer. (*node)

note:

根节点： tr.root.data.(*node)
rect. -> entry (either index record entry like R8, R9,..
or entry not in Leaf node: R1, R3,…)
min, max [2]float -> I
count in RTree: num of index record
count in node: num of entries in this node





R-Tree Searching

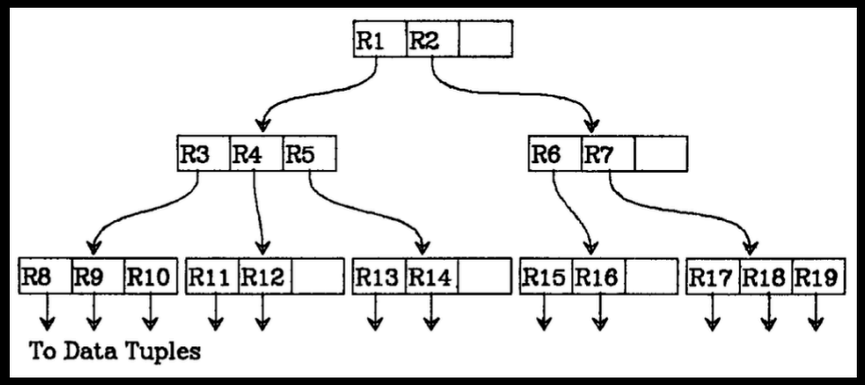
EI: the rectangle part of an index entry E

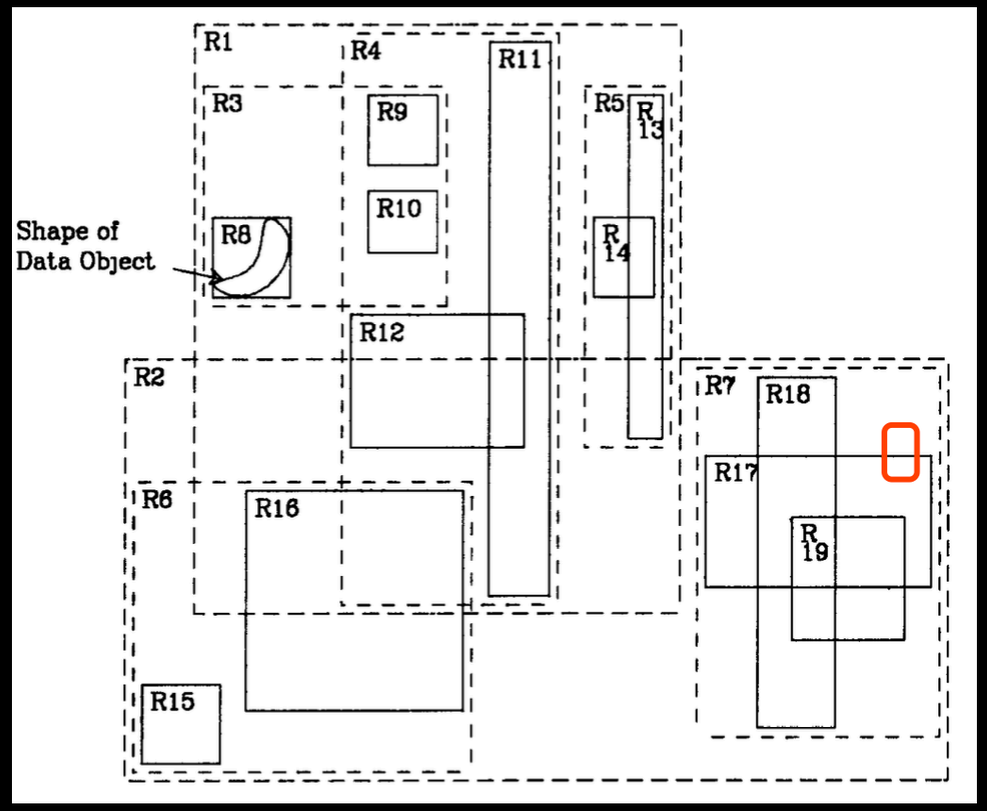
Ep: the tuple-identifier or child-pointer

Algorithm **Search**: Given an R-tree whose root node is T, find all index records whose rectangles overlap a search rectangle S.

- S1 [Search subtrees] if T is not a leaf, check each entry E to determine whether EI overlaps S. For all overlapping entries, invoke **Search** on the tree whose root node is pointed to by Ep

- S2 [Search leaf node] if T is a leaf, check all entries E to determine whether EI overlaps S. If so, E is a qualifying record.

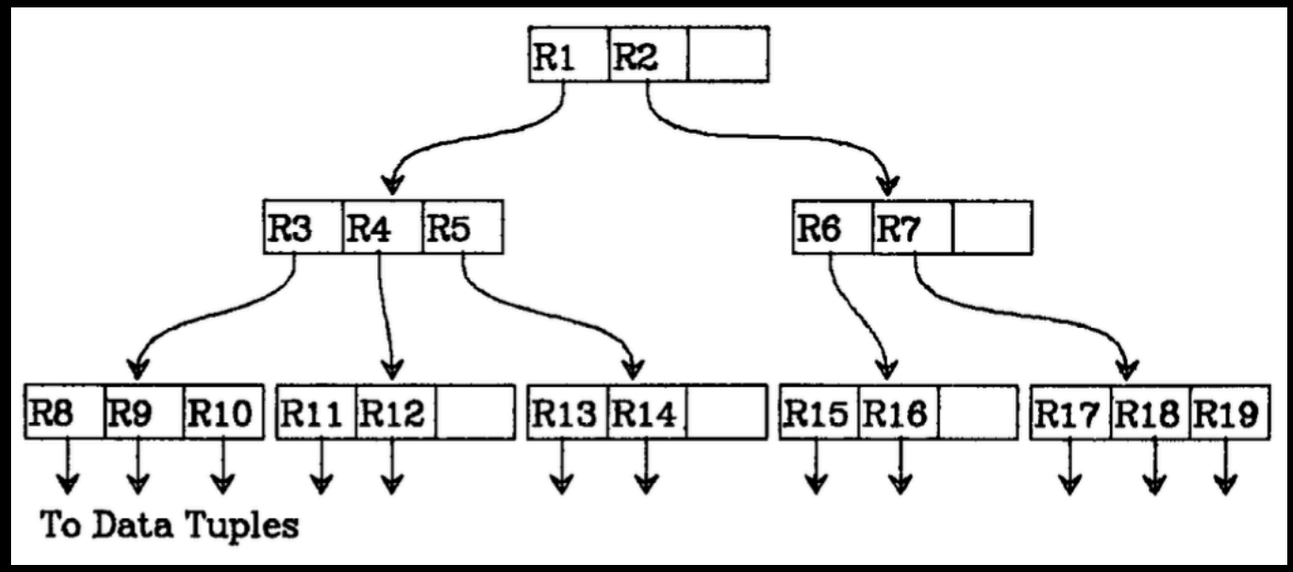


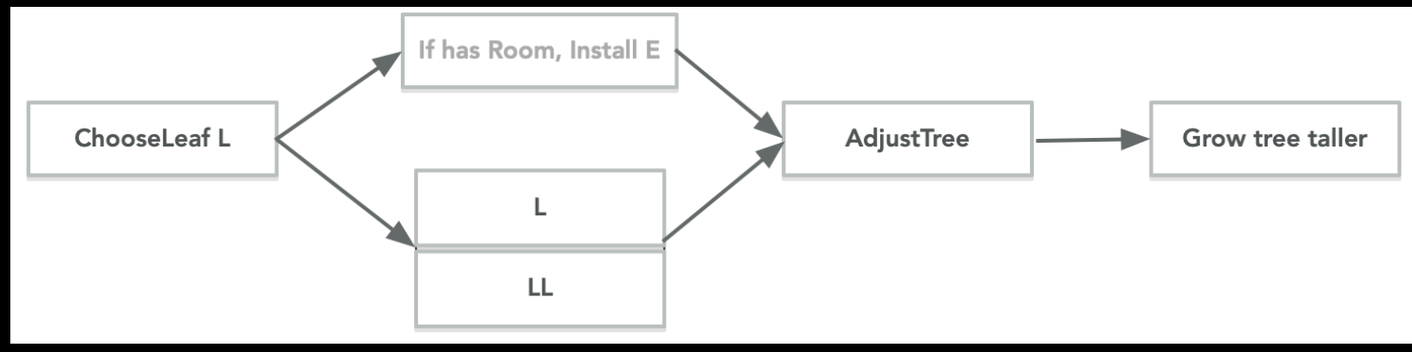


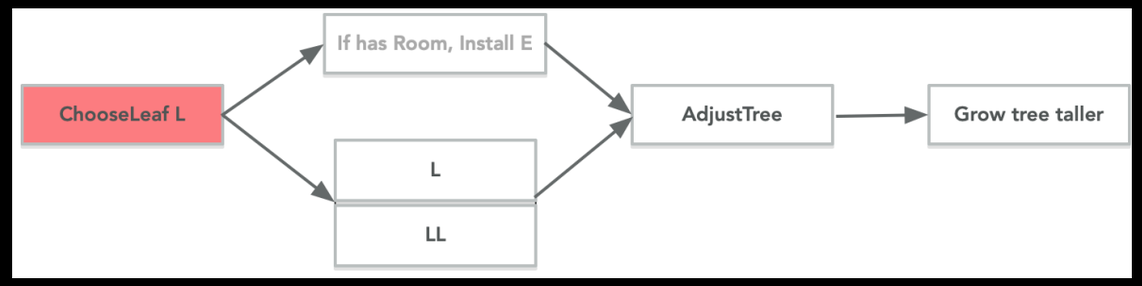
R-Tree Insertion

Algorithm **Insert.** Insert a new index entry E into an R-tree

- I1 [Find position for new record] invoke **ChooseLeaf** to select a leaf node L in which to place E
- I2 [Add record to leaf node] if L has room for another entry, install E. Otherwise invoke SplitNode to obtain L and LL containing E and all the old entries of L.
- I3 [Propagate changes upward] Invoke **AdjustTree** on L, also passing LL if a split was performed
- I4 [Grow tree taller] if node split propagation caused the root to split, create a new root whose children are the two resulting nodes

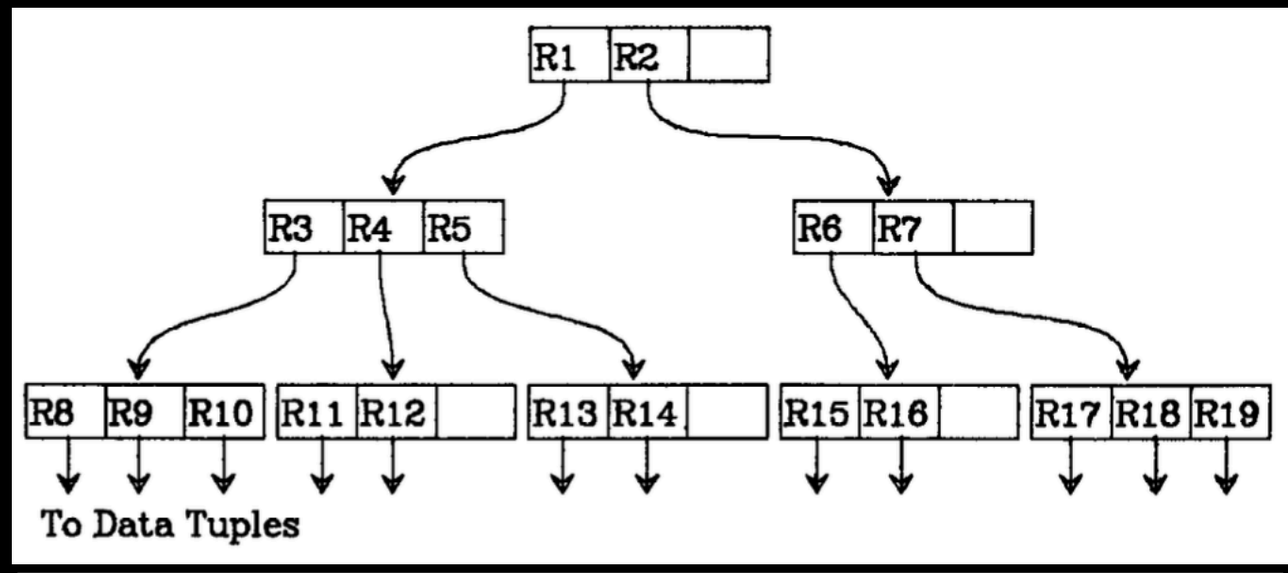


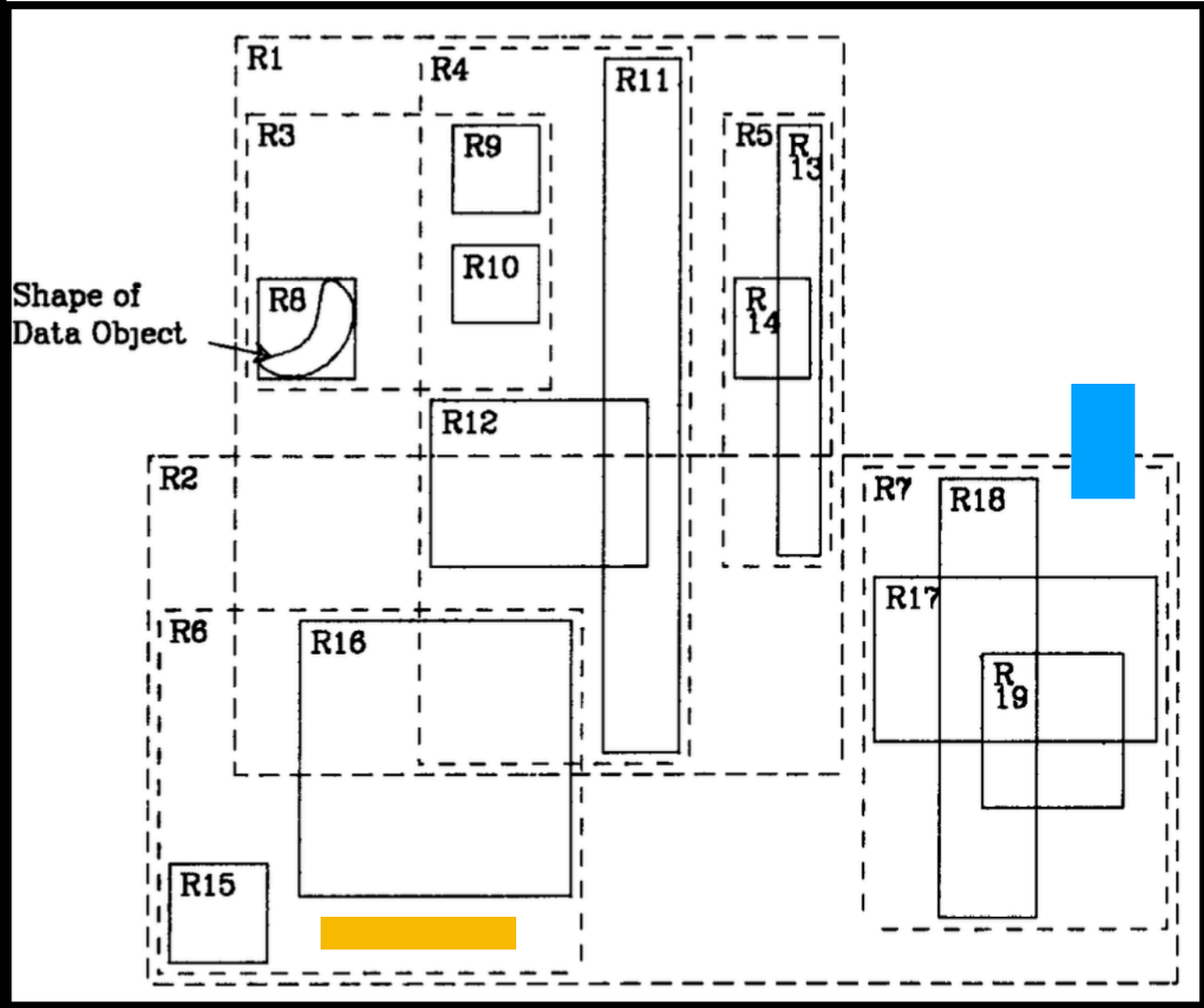


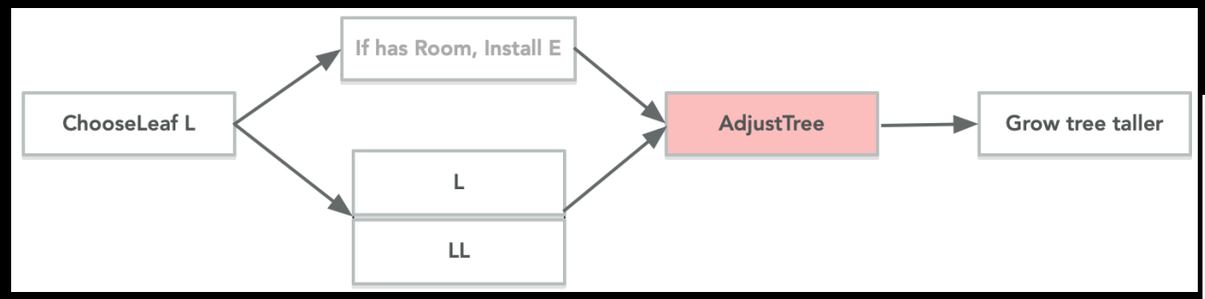


Algorithm **ChooseLeaf** Select a leaf node in which to place a new index entry E

- ★ CL1 [Initialize] Set N to be the root node
- ★ CL2 [Leaf check] if N is a leaf, return N.
- ★ CL3 [Choose subtree] if N is not a leaf, let F be the entry in N whose rectangle FI needs least enlargement to include EI. Resolve ties by choosing the entry with the rectangle of smallest area
- ★ CL4 [Descend until a leaf is reached] Set N to be the child node pointed to by Fp and repeat from CL2



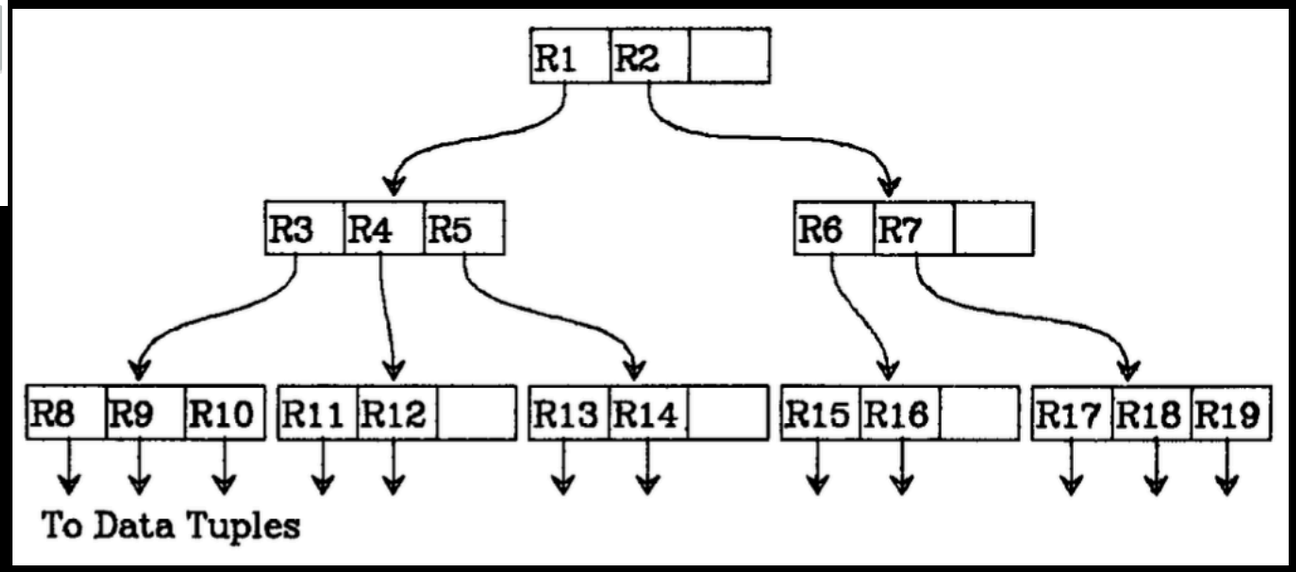


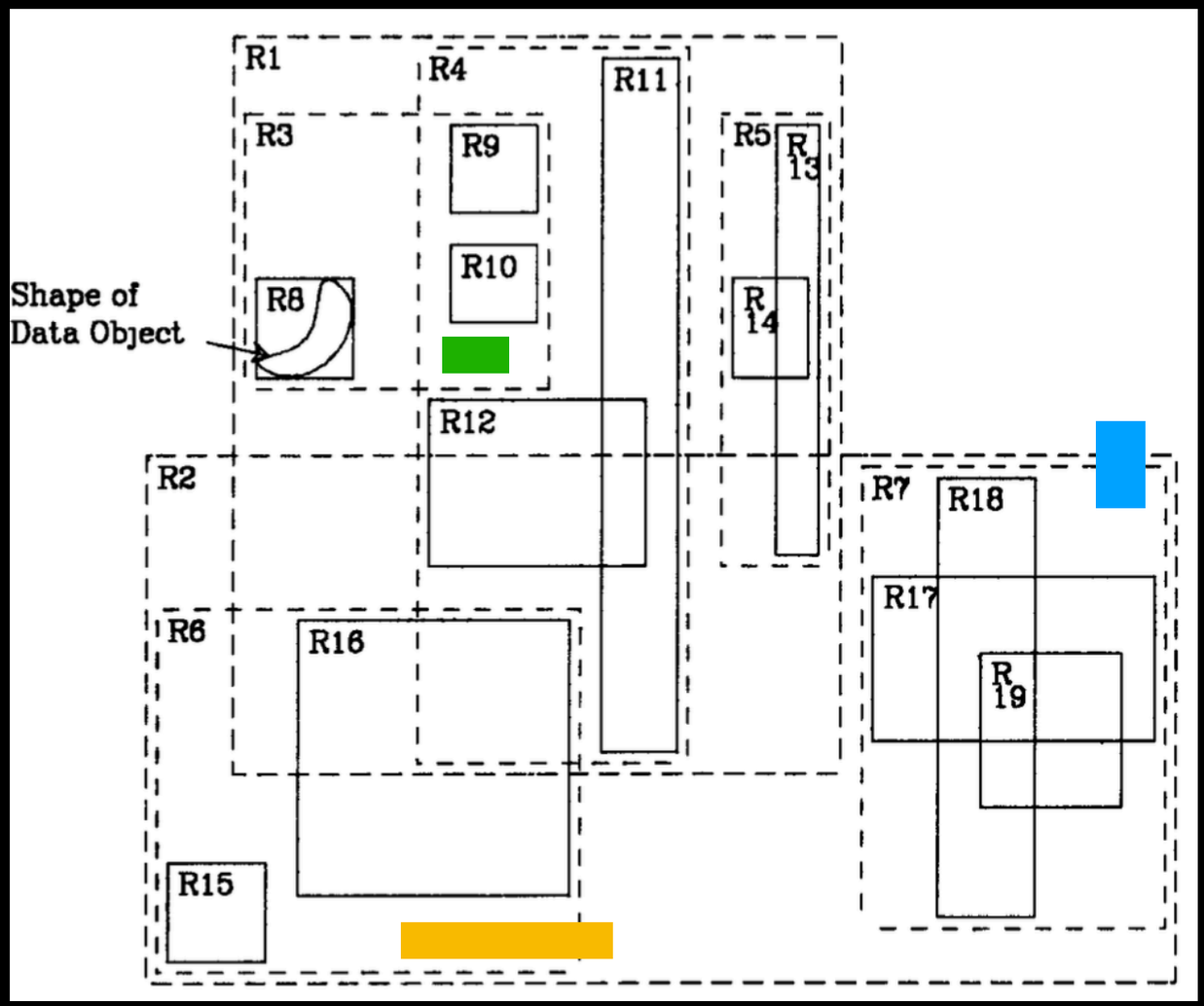


Algorithm **AdjustTree** Ascend from a leaf node L to the root, adjusting covering rectangles and propagating node split as necessary

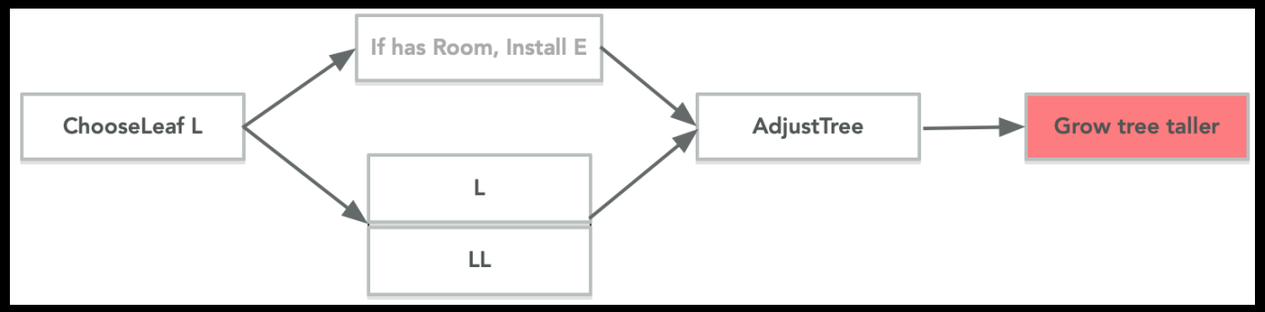
- AT1 [Initialize.] Set N=L If L was split previously, set NN to be the resulting second node
- AT2 [Check if done] if N is the root, stop
- AT3 [Adjust covering rectangle in parent entry] Let P be the parent node of N, and let E_N be N's entry in P. Adjust E_NI so that it tightly encloses all entry rectangles in N.
- AT4 [Propagate node split upward] if N has a partner NN resulting from an earlier split, create a new entry E_{NN} with E_{NN}P pointing to NN and E_{NN}I enclosing all rectangles in NN. Add E_{NN} to P if there is room. Otherwise, invoke **SplitNode** to produce P and PP containing E_{NN} and all P's old entries
- AT5 [Move up to next level] Set N=P and set NN=PP if a split occurred. Repeat from AT2.

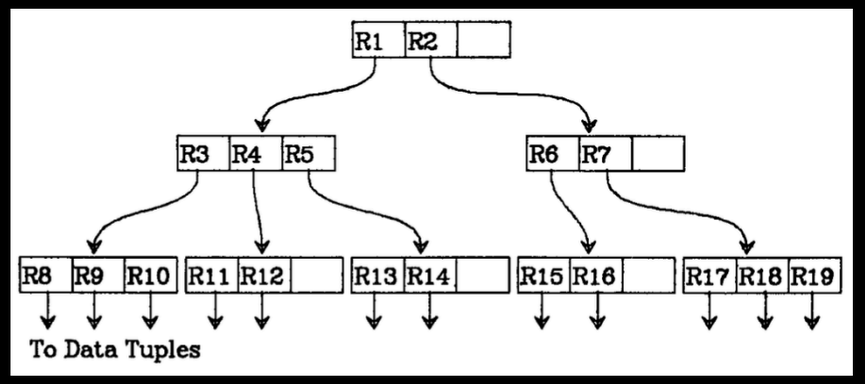
Algorithm **SplitNode** is described later

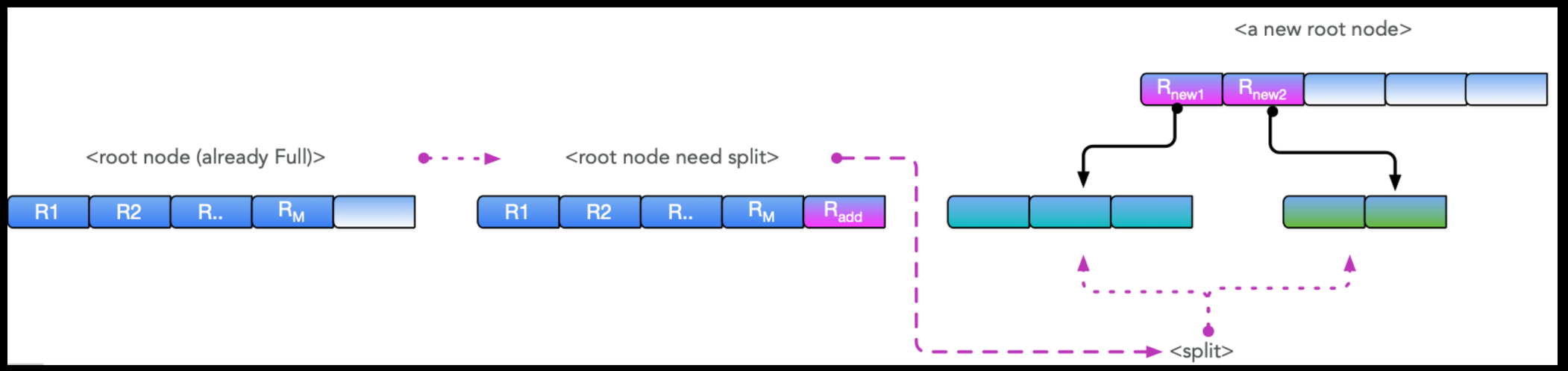




Grow tree taller

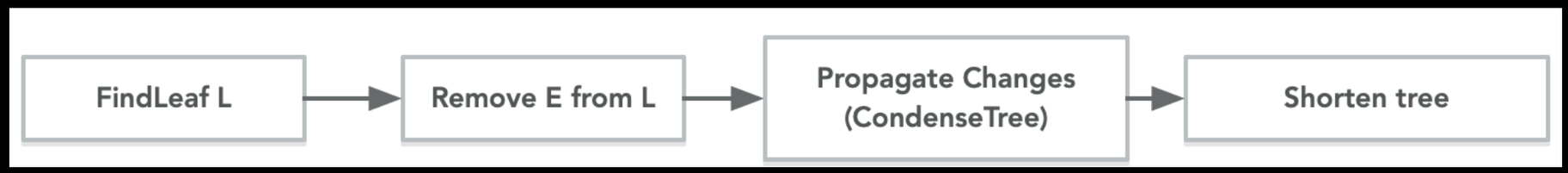







```
if tr.root.data.(*node).count == maxEntries+1 {
    newRoot := new(node)
    tr.root.splitLargestAxisEdgeSnap(&newRoot.boxes[1])
    newRoot.boxes[0] = tr.root
    newRoot.count = 2
    tr.root.data = newRoot
    tr.root.recalc()
    tr.height++
}
```

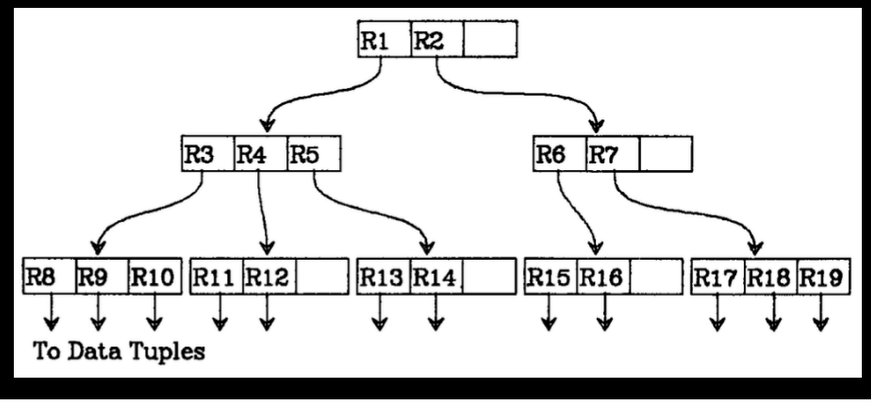
R-Tree Delete

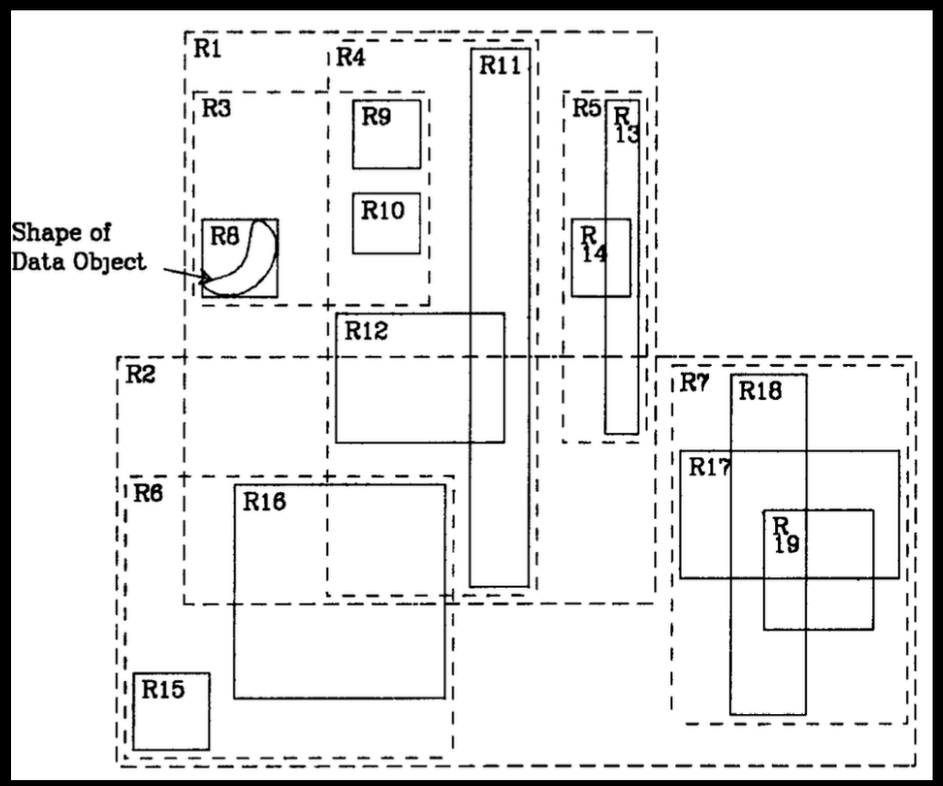


Algorithm **Delete** Remove index record E from an R-tree

- D1 [Find node containing record] Invoke **FindLeaf** to locate the leaf node L containing E. Stop if the record was not found
- D2 [Delete record] Remove E from L
- D3 [Propagate changes] Invoke **CondenseTree**, passing L
- D4 [Shorten tree.] If the root node has only one child after the tree has been adjusted, make the child the new root

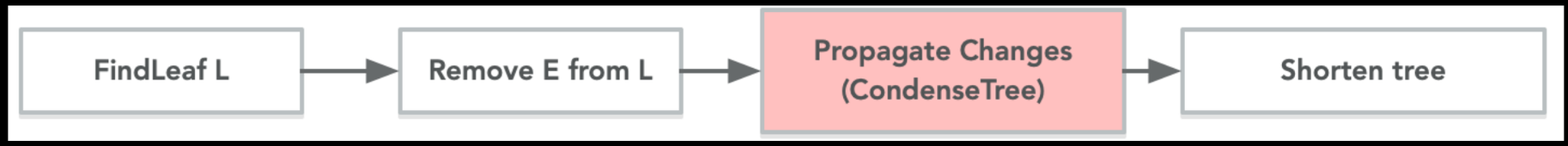
Algorithm **FindLeaf**. Given an R-tree whose root node is T, find the leaf node contains the index entry E.





- FL1 [Search subtrees] if T is not a leaf, check each entry F in T to determine if FI overlaps EI. For each such entry invoke FindLeaf on the tree whose root is pointed to by Fp until E is found or all entries have been checked.

- FL2 [Search leaf node for record] if T is a leaf, check each entry to see if it matches E. If E is found return T



Algorithm **CondenseTree** Given a leaf node L from which an entry has been deleted, eliminate the node if it has too few entries and relocate its entries. Propagate node elimination upward as necessary. Adjust all converting rectangles on the path to the root, making them smaller if possible

- CT1 [Initialize] Set N=L Set Q, the set of eliminated nodes, to be empty
- CT2 [Find parent entry] If N is the root, go to CT6. Otherwise let P be the parent of N, and let E_N be N's entry in P
- CT3 [Eliminate under-full node] If N has fewer than m entries, delete E_N from P and add N to Set Q
- CT4 [Adjust covering rectangle] If N has not been eliminated. Adjust E_NI to tightly contain all entries in N
- CT5 [Move up on level in tree] Set N=P and repeat from CT2
- CT6 [Re-insert orphaned entries] Re-insert all entries of nodes in set Q. Entries from eliminated leaf nodes are re-inserted in tree leaves as described in Algorithm Insert, but entries from higher-level nodes must be placed higher in the tree, so that leaves of their dependent subtrees will be on the same level as leaves of the main tree.

R-tree Updating and Other ops

Update: delete , updated, re-inserted

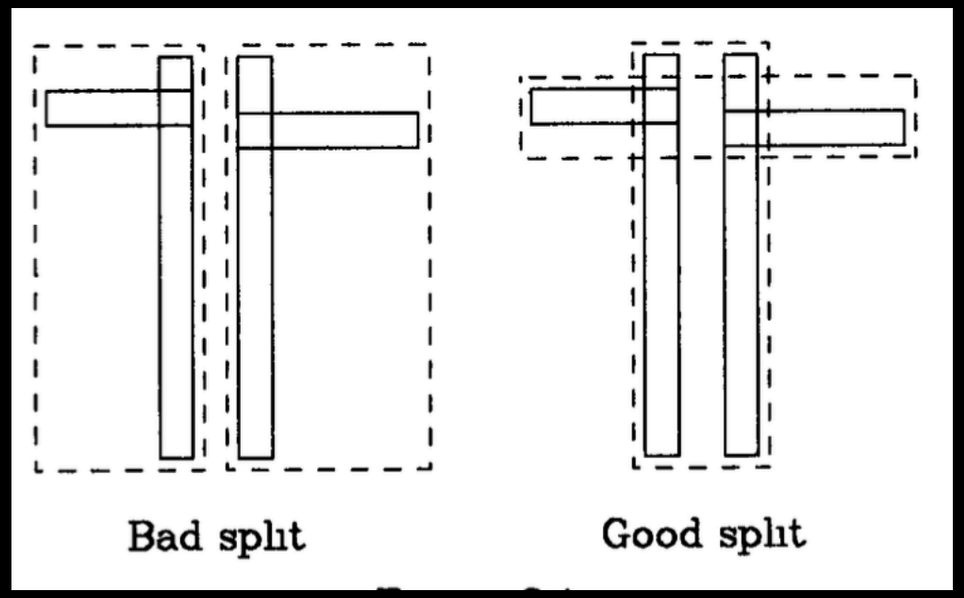
Other kinds of searches: Variations on the algorithm given.

Ex: KNN (k-Nearest Neighbor)

R-tree Node Splitting

Target: The total area of the two covering rectangles after split should be minimized.

- ★ Exhaustive Algorithm: 2^{M-1}
- ★ Quadratic-Cost Algorithm
- ★ Linear-Cost Algorithm



R-tree SplitNode Quadratic-Cost Algorithm

Target: to find a small-area split

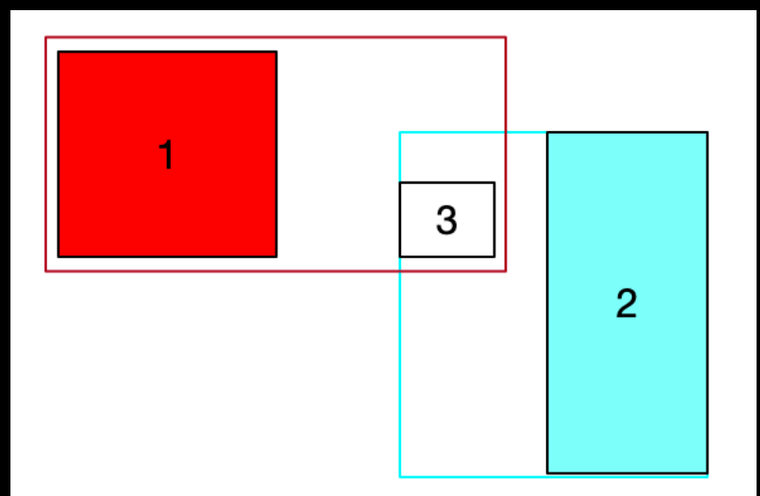
Cost: quadratic in M and linear in the number of dimensions

Algorithm **Quadratic Split** Divide a set of M+1 index entries into two group

- QS1 [Pick first entry for each group] Apply Algorithm **PickSeeks**

- QS2 [Check if done] if all entries have been assigned, stop. If one group has so few entries that all the rest must be assigned to it in order for it to have the the minimum number m, assign them and stop

- QS3 [Select entry to assign] Invoke Algorithm **PickNext** to choose the next entry to assign. Add it to the group whose covering rectangle will have to be enlarged least to accommodate it. Resolve ties by adding the entry to the group with smaller area, then to the one with fewer entires, then to either. Repeat from QS2



R-tree SplitNode Quadratic-Cost Algorithm

Algorithm **PickSeeds** Select two entries to be the first elements of the groups

- PS1 [Calculate inefficiency of grouping entries together] For each pair of entries E_1 and E_2, compose a rectangle J including E_1I, E_2I. Calculate d = area(J) - area(E_1I) - area(E_2I)
- PS2 [Choose the most wasteful pair] Choose the pair with the largest d

Algorithm **PickNext** Select one remaining entry for classification in a group

- PN1 [Determine cost of putting each entry in each group] For each entry E not yet in a group, calculate d_1= the area increase required in the covering rectangle of Group 1 to include EI. Calculate d_2 similarly for Group 2.
- PN2 [Find entry with greatest preference for one group] Choose any entry with the maximum difference between d_1 and d_2.

R-tree SplitNode Linear-Cost Algorithm

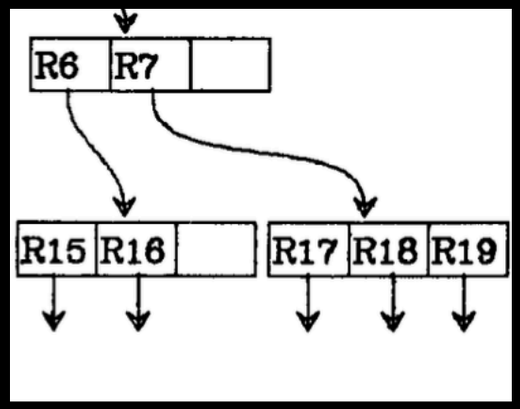
Identical to Quadratic Split but use different PickSeeds and PickNext

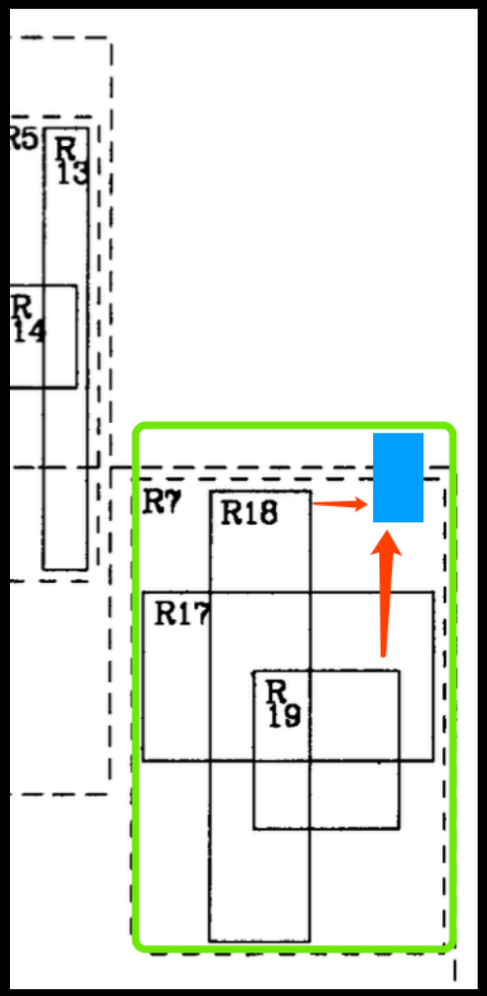
PickNext simply choose any of the remaining entries.

Algorithm **LinearPickSeeds** Select two entries to be the first elements of the groups

- LPS1 [Find extreme rectangles along all dimensions] Along each dimension, find the entry whose rectangle has the highest low side, and the one with the lowest high side. Record the separation
- LPS2 [Adjust for shape of the rectangle cluster] Normalize the separations by dividing by the width of the entire set along the corresponding dimension
- LPS3 [Select the most extreme pair] Choose the pair with the greatest normalized separation along any dimension

Conclusions: The linear node-split algorithm proved to be as good as more expensive techniques It was fast, and the sightly worse quality of the splits did not affect search performance noticeably





R-tree SplitNode in tile38

Two values, `min-dist` and `max-dist`, are calcuated for each child.

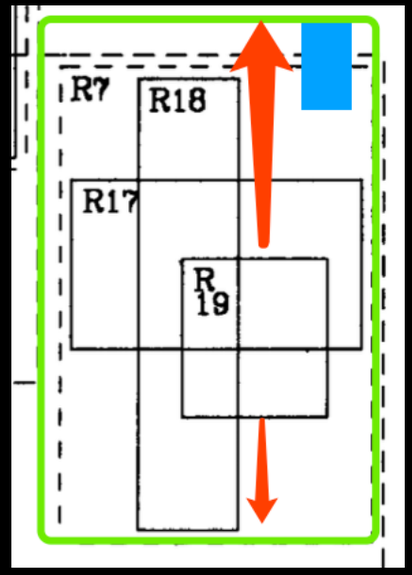
- `min-dist` is the distance from the parent's minumum value of it's largest axis to the child's minumum value of the parent largest axis.
- `max-dist` is the distance from the parent's maximum value of it's largest axis to the child's maximum value of the parent largest axis.

When the `min-dist` is less than `max-dist` then the child is placed into the `left` box.
When the `max-dist` is less than `min-dist` then the child is placed into the `right` box.
When the `min-dist` is equal to `max-dist` then the child is placed into an `equal` bucket until all of the children are evaluated.
Each `equal` box is then one-by-one placed in either `left` or `right`, whichever has less children.

Performance

In my testing:

- Insert show similar performance as the quadratic R-tree and ~1.2x - 1.5x faster than R*tree.
- Search and Delete is ~1.5x - 2x faster than quadratic and about the same as R*tree.



Tile38 intro

https://tile38.com/

Commands

All **Search** Keys Webhooks Channels Connection Replication Scripting Server

INTERSECTS
Searches for ids that intersect an area

NEARBY
Searches for ids that are nearby a point

SCAN
Incrementally iterate though a key

SEARCH
Search for string values in a key

WITHIN
Searches for ids that completely within the area

Tile38 to region

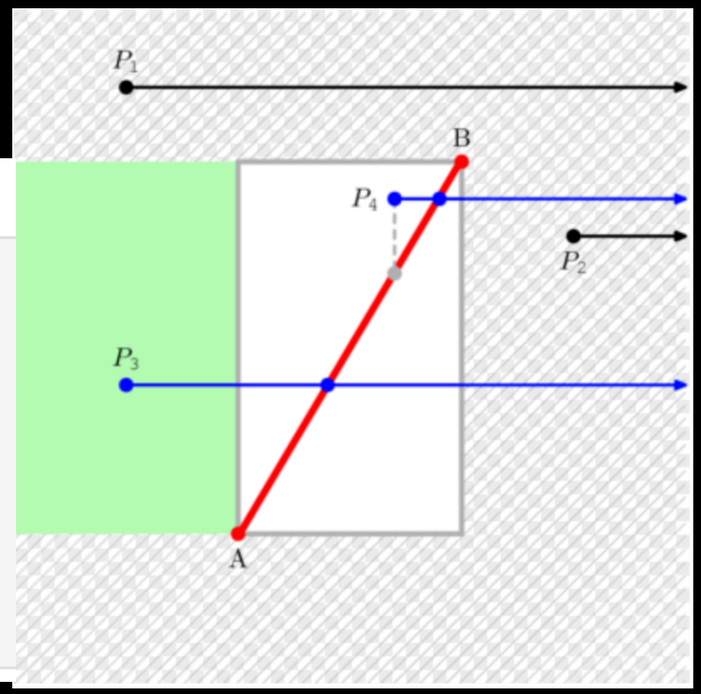
- Intersects (Rect to Rect for example)

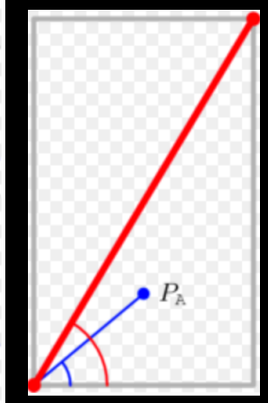
```
288 func (r *rect) intersects(b *rect) bool {
289     if b.min[0] > r.max[0] || b.max[0] < r.min[0] {
290         return false
291     }
292     if b.min[1] > r.max[1] || b.max[1] < r.min[1] {
293         return false
294     }
295     return true
296 }
297
```

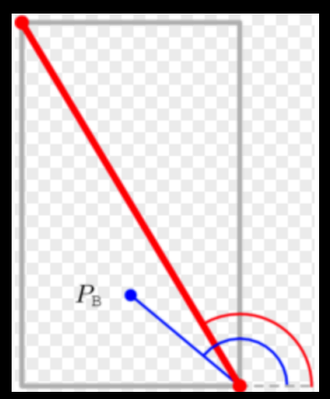
- Ray-casting algorithm

A pseudocode can be simply:

```
count ← 0
foreach side in polygon:
  if ray_intersects_segment(P,side) then
    count ← count + 1
if is_odd(count) then
  return inside
else
  return outside
```





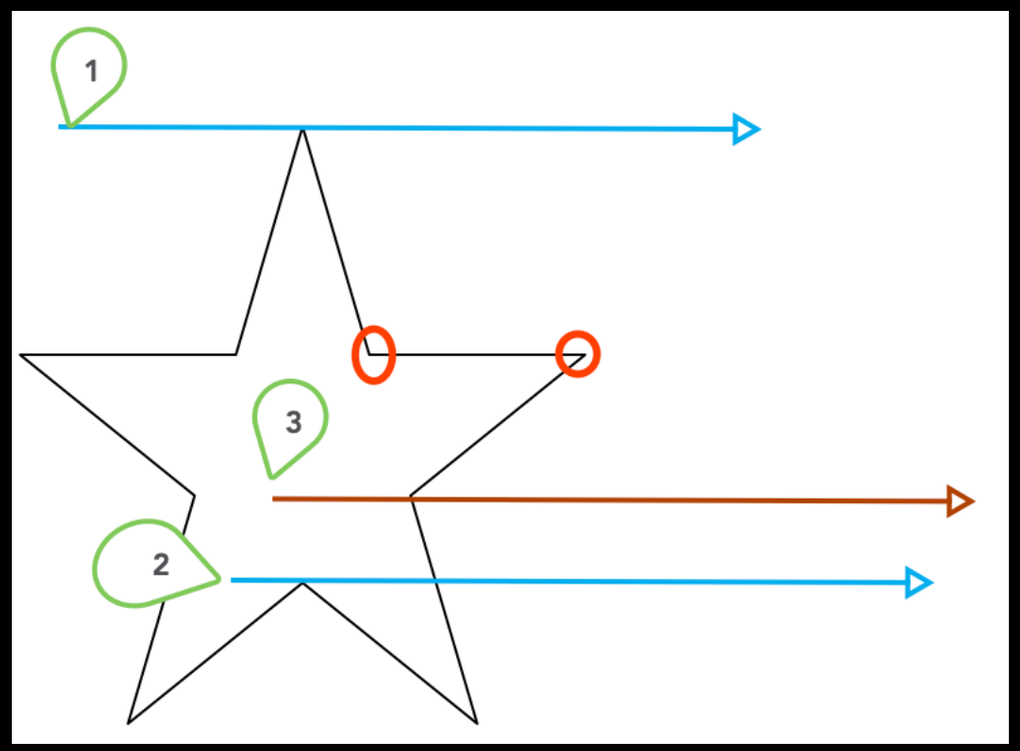


Ray-Casting implementation

- 探探程序设计大赛 Problem G: R-Tree Quadratic-cost + Ray-Casting

```
func (p *Polygon) Contains(pt Point) (inside bool) {
    if pt[0] < p.xmin || pt[0] > p.xmax || pt[1] < p.ymin || pt[1] > p.ymax {
        return
    }
    for i := 0; i < p.N-1; i++ {
        Pi, Pj := p.Ring[i+1], p.Ring[i]
        if (pt[1] < Pi[1]) != (pt[1] < Pj[1]) && (pt[0] < (Pj[0]-Pi[0])*(pt[1]-Pi[1])/(Pj[1]-Pi[1])+Pi[0]) { // core tricks here
            inside = !inside
        }
    }
    return
}
```

```
    for p.Y == a.Y || p.Y == b.Y {
        p.Y = math.Nextafter(p.Y, math.Inf(1))
    }
```

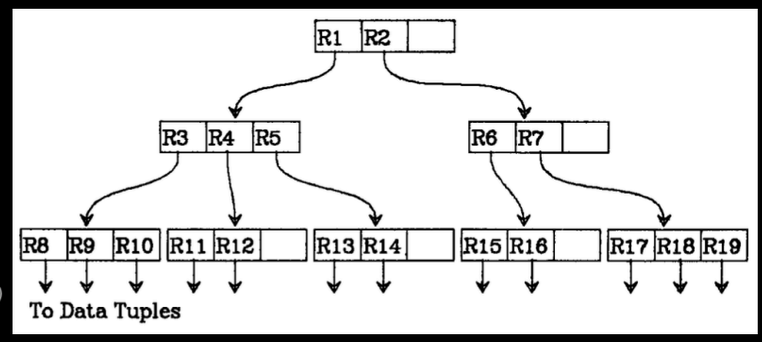


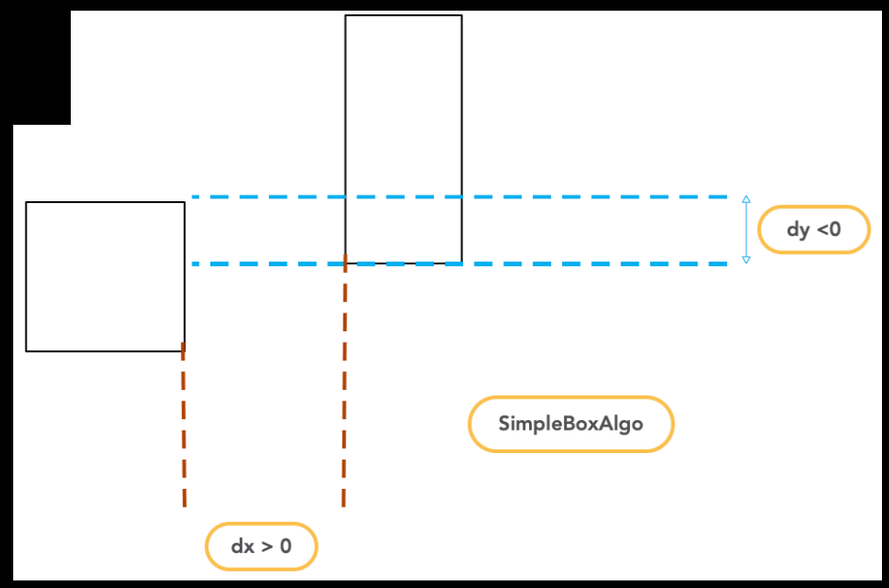
Tile38 Nearby

- KNN Operation

```
for {
    // gather all children for parent
    children = index.tree.Children(parent, children[:0])
    for _, child := range children {
        added = added[:0]
        algo(child.Min, child.Max, child.Data, child.Item, add)
        for _, node := range added {
            q.push(node)
        }
    }
    for {
        node, ok := q.pop()
        if !ok {
            // nothing left in queue
            return
        }
        if node.child.Item {
            if !iter(node.child.Min, node.child.Max,
                node.child.Data, node.dist) {
                return
            }
        } else {
            // gather more children
            parent = node.child.Data
            break
        }
    }
}
```

```
// Priority Queue ordered by dist (smallest to largest)

type qnode struct {
    dist  float64
    child child.Child
}

type queue struct {
    nodes []qnode
    len   int
    size  int
}
```





- Haversine

```
// DistanceToHaversine ...
func DistanceToHaversine(meters float64) float64 {
    // convert the given distance to its haversine
    sin := math.Sin(0.5 * meters / earthRadius)
    return sin * sin
}
```

```
maxDist := target.Haversine()
```

```
dist := target.HaversineTo(o.Center())
if maxDist > 0 && dist > maxDist {
    return false
}
items = append(items, iterItem{id: id, o: o, fields: fields, dist: dist})
```

```
// Haversine ...
func Haversine(latA, lonA, latB, lonB float64) float64 {
    φ1 := latA * radians
    λ1 := lonA * radians
    φ2 := latB * radians
    λ2 := lonB * radians
    Δφ := φ2 - φ1
    Δλ := λ2 - λ1
    sΔφ2 := math.Sin(Δφ / 2)
    sΔλ2 := math.Sin(Δλ / 2)
    return sΔφ2*sΔφ2 + math.Cos(φ1)*math.Cos(φ2)*sΔλ2*sΔλ2
}
```

Repositories

```
/Users/zhaoweipu/go/pkg/mod/github.com/tidwall:
total used in directory 0 available 2447927785
drwxr-xr-x   24 zhaoweipu  staff   768 Dec 30 16:56 .
drwxr-xr-x  106 zhaoweipu  staff  3392 Dec 11 22:21 ..
-rw-r--r--    1 root       staff     0 Dec 27 20:19 .projectile
dr-x------    7 zhaoweipu  staff   224 Dec 11 22:21 btree@v0.0.0-20170113224114-9876f1454cf0
dr-x------    8 zhaoweipu  staff   256 Dec 11 22:21 buntdb@v1.1.0
dr-x------    6 zhaoweipu  staff   192 Dec 11 22:21 cities@v0.0.0-20190730194520-dbe1ae0b862c
dr-x------    7 zhaoweipu  staff   224 Dec 11 22:21 geoindex@v1.1.0
dr-x------   37 zhaoweipu  staff  1184 Dec 11 22:21 geojson@v1.1.8
dr-x------   13 zhaoweipu  staff   416 Dec 11 22:21 gjson@v1.3.2
dr-x------    6 zhaoweipu  staff   192 Dec 11 22:21 grect@v0.0.0-20161006141115-ba9a043346eb
dr-x------    6 zhaoweipu  staff   192 Dec 11 22:21 lotsa@v0.0.0-20180225195211-a03631ac7f1c
dr-x------    6 zhaoweipu  staff   192 Dec 27 12:07 lotsa@v0.0.0-20190911211615-e96c7fea75f1
dr-x------    7 zhaoweipu  staff   224 Dec 11 22:21 match@v1.0.1
dr-x------    7 zhaoweipu  staff   224 Dec 11 22:21 pretty@v1.0.0
dr-x------    7 zhaoweipu  staff   224 Dec 11 22:21 rbang@v1.1.0
dr-x------    5 zhaoweipu  staff   160 Dec 11 22:21 redbench@v0.0.0-20181110173744-17c5b5b864a4
dr-x------   11 zhaoweipu  staff   352 Dec 11 22:21 redcon@v0.0.0-20171003141744-3df12143a4fe
dr-x------   14 zhaoweipu  staff   448 Dec 11 22:21 resp@v0.0.0-20160908231031-b2b1a7ca20e3
dr-x------    8 zhaoweipu  staff   256 Dec 11 22:21 rhh@v1.1.0
dr-x------    8 zhaoweipu  staff   256 Dec 11 22:21 rtree@v0.0.0-20180113144539-6cd427091e0e
dr-x------   10 zhaoweipu  staff   320 Dec 11 22:21 sjson@v1.0.2
dr-x------   18 zhaoweipu  staff   576 Dec 16 11:47 tile38@v0.0.0-20191211181953-d48dd2278afb
dr-x------    6 zhaoweipu  staff   192 Dec 11 22:21 tinybtree@v0.0.0-20181217131827-de5932d649b5
dr-x------    6 zhaoweipu  staff   192 Dec 11 22:21 tinyqueue@v0.0.0-20180302190814-1e39f5511563
```

Tile38 Set && GeoJson

- Set

SET key id [FIELD name value ...] [EX seconds] [NX|XX] (OBJECT geojson)|(POINT lat lon [z])|(BOUNDS minlat minlon maxlat maxlon)|(HASH geohash)|(STRING value)

GeoJSON is a geospatial data interchange format based on JavaScript Object Notation (JSON)

rfc7946

Table of Contents

数据源

- pg regions GeometryType

```
putong=# select ST_GeometryType(boundary_bd) FROM regions group by ST_GeometryType(boundary_bd);
 st_geometrytype
-----------------
 ST_MultiPolygon
 ST_Polygon

(3 rows)
```

- select id, ST_AsGeoJson(boundary_bd) as boundary_bd FROM regions WHERE id in (1)


```
 1 | {"type":"Polygon","coordinates":[[[93.498773,32.506218],[93.508084,32.509583],[93.522894,32.481909],[93.536472,32.479559],[93.542091,32.492965],[93.561
543,32.49182],[93.571494,32.50009],[93.570445,32.506042],[93.578947,32.507292],[93.57996,32.513079],[93.586352,32.511226],[93.614342,32.524213],[93.61658,32.
529317],[93.626274,32.52936],[93.636955,32.548053],[93.632275,32.557302],[93.647782,32.559242],[93.659824,32.578503],[93.676457,32.581263],[93.685193,32.5739
33],[93.705893,32.58233],[93.728424,32.584453],[93.749668,32.579098],[93.756162,32.569169],[93.7695,32.574523],[93.787266,32.562841],[93.80071,32.564376],[93
.827689,32.555004],[93.836179,32.535827],[93.83414,32.52727],[93.843883,32.520035],[93.85738,32.51636],[93.857164,32.501802],[93.866062,32.4925],[93.861865,3
2.481845],[93.865804,32.478229],[93.862377,32.472707],[93.865137,32.470898],[93.876045,32.480062],[93.889906,32.477495],[93.903206,32.478909],[93.914136,32.4
70823],[93.935484,32.482675],[93.966303,32.491284],[93.978713,32.477141],[93.981717,32.46666],[94.002793,32.469216],[94.005001,32.461831],[94.017766,32.45195
7],[94.036819,32.45381],[94.038816,32.457744],[94.035955,32.465351],[94.047059,32.465339],[94.057495,32.476168],[94.073625,32.476466],[94.078153,32.471876],[
```


- select id, boundary_bd as boundary_bd FROM regions WHERE id in (3121);


```
 3121 | 0103000000001000000580200001426 8C6665595D4039EE940ED61D44408315A75A0B575D4013B9E00CFE1E44401250E10852565D40C095ECD808204440A1DAE044F4555D40DD28B2D6502
44440016A6AD95A535D40F5D555815A2444409D84D21742535D4045A165DD3F2644402E910BCEE0525D4076E107E753274440D34B8C65FA515D405CE674594C284440581EA4A7C8515D408638D6C5
6D2A4440BC40498105525D40FDF7E0B54B2B4440FED7B96933515D409F8F32E2022C4440F12A6B9BE24F5D404D2EC6C03A3044402EE23B31EB4E5D405F251FBB0B30444071AB2006BA4E5D4066BCA
DF4DA30444012A27C410B4F5D402BC0779B37324440FEEDB25F774F5D40CEA96400A832444093718C648F4F5D40363D2828453344400A4D124BCA4F5D40A514747B4933444080608E1EBF4F5D4044
```

3.1.6. Polygon

To specify a constraint specific to Polygons, it is useful to introduce the concept of a linear ring:

- A linear ring is a closed LineString with four or more positions.
- The first and last positions are equivalent, and they MUST contain identical values; their representation SHOULD also be identical.
- A linear ring is the boundary of a surface or the boundary of a hole in a surface.
- A linear ring MUST follow the right-hand rule with respect to the area it bounds, i.e., exterior rings are counterclockwise, and holes are clockwise.

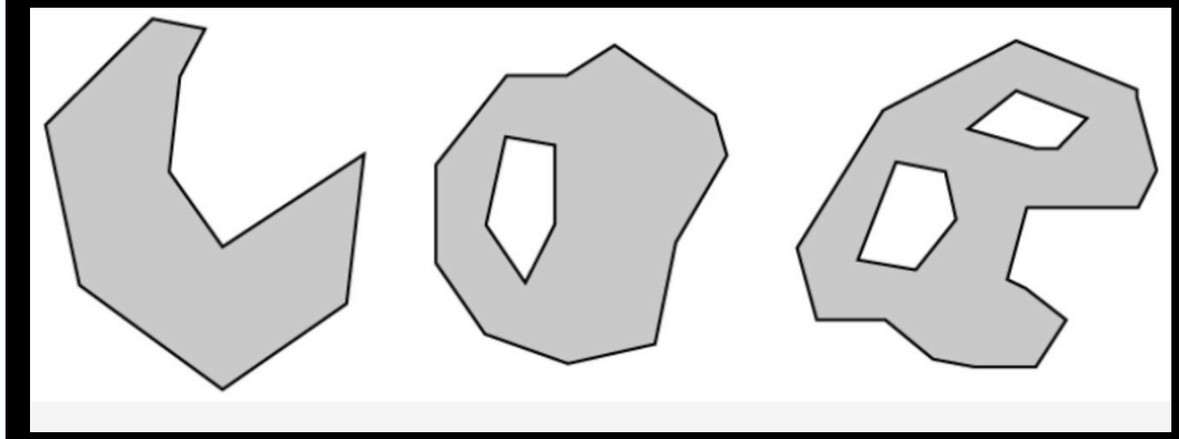
Note: the [GJ2008] specification did not discuss linear ring winding order. For backwards compatibility, parsers SHOULD NOT reject Polygons that do not follow the right-hand rule.

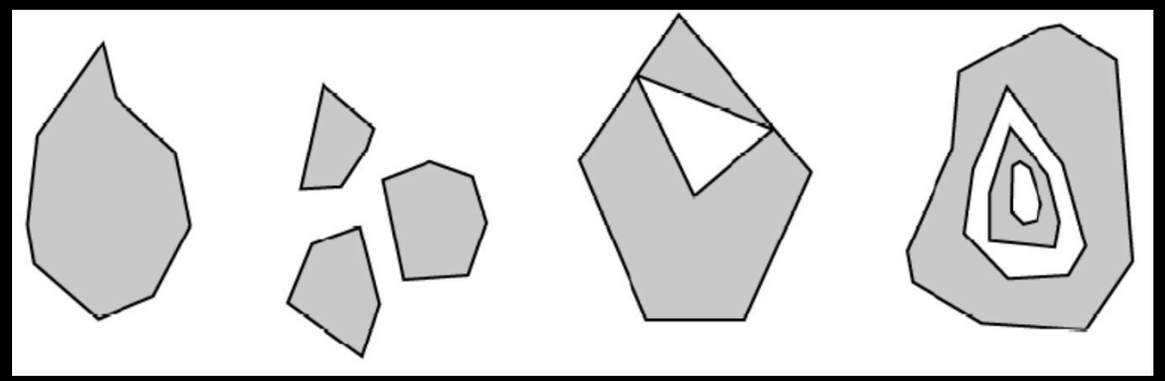
Though a linear ring is not explicitly represented as a GeoJSON geometry type, it leads to a canonical formulation of the Polygon geometry type definition as follows:

- For type "Polygon", the "coordinates" member MUST be an array of linear ring coordinate arrays.
- For Polygons with more than one of these rings, the first MUST be the exterior ring, and any others MUST be interior rings. The exterior ring bounds the surface, and the interior rings (if present) bound holes within the surface.

3.1.7. MultiPolygon

For type "MultiPolygon", the "coordinates" member is an array of Polygon coordinate arrays.





Location grpc with tile38

Set

- SET type regionId OBJECT geoJsonString
  - type: country, province, city, district

Intersects

- INTERSECTS key HASH geoHash

More about tile38

https://github.com/tidwall/tile38

https://tile38.com/documentation/

Tile38 – Realtime geofencing and geospatial index, v1.7.0

Roaming Geofences test

```
Last login: Tue Jan  7 20:23:34 on ttys007
til%

   (py3) ➜  ~ tile38-cli
127.0.0.1:9851> NEARBY people FENCE ROAM people * 5000
{"ok":true,"live":true}
{"command":"del","id":"bob","time":"2020-01-08T11:35:47.051384+08:00"}
{"command":"del","id":"alice","time":"2020-01-08T11:36:02.210275+08:00"}
```

```
{"command":"set","group":"5e154e5248d0aaea6701def6","detect":"roam","key":"pe
ople","time":"2020-01-08T11:36:50.362829+08:00","id":"alice","object":{"type"
:"Point","coordinates":[-115.02,33.02]},"nearby":{"key":"people","id":"bob","
object":{"type":"Point","coordinates":[-115.01,33.01]},"meters":1451.138}}
```

```
{"command":"set","group":"5e154f1f48d0aaea6701def7","detect":"roam","key":"pe
ople","time":"2020-01-08T11:40:15.708682+08:00","id":"bob","object":{"type":"
Point","coordinates":[-115.01,33.01]},"nearby":{"key":"people","id":"alice","
object":{"type":"Point","coordinates":[-115.02,33.02]},"meters":1451.138}}
```

```
{"command":"set","group":"5e154f3e48d0aaea6701def8","detect":"roam","key":"pe
ople","time":"2020-01-08T11:40:46.002783+08:00","id":"jhon","object":{"type":
"Point","coordinates":[-115.03,33.03]},"nearby":{"key":"people","id":"alice",
"object":{"type":"Point","coordinates":[-115.02,33.02]},"meters":1451.07}}
{"command":"set","group":"5e154f3e48d0aaea6701def8","detect":"roam","key":"pe
ople","time":"2020-01-08T11:40:46.002783+08:00","id":"jhon","object":{"type":
"Point","coordinates":[-115.03,33.03]},"nearby":{"key":"people","id":"bob","o
bject":{"type":"Point","coordinates":[-115.01,33.01]},"meters":2902.208}}
```

```
Last login: Wed Jan  8 11:28:33 on ttys006
(py3) ➜  ~ tile38-cli
127.0.0.1:9851> del people bob
{"ok":true,"elapsed":"23.55µs"}
127.0.0.1:9851> del people alice
{"ok":true,"elapsed":"12.061µs"}
127.0.0.1:9851> del people jhon
{"ok":true,"elapsed":"4.538µs"}
127.0.0.1:9851> SET people bob POINT 33.01 -115.01
{"ok":true,"elapsed":"22.434µs"}
127.0.0.1:9851> SET people alice POINT 33.02 -115.02
{"ok":true,"elapsed":"15.873µs"}
127.0.0.1:9851> SET people bob POINT -33.02 115.02
{"ok":true,"elapsed":"17.732µs"}
127.0.0.1:9851> SET people bob POINT 33.01 -115.01
{"ok":true,"elapsed":"20.267µs"}
127.0.0.1:9851>
127.0.0.1:9851> SET people jhon POINT 33.03 -115.03
{"ok":true,"elapsed":"19.979µs"}
127.0.0.1:9851>
```

pg开关测试

https://confluence.p1staff.com/pages/viewpage.action?pageId=44058139

More tests

region查询去除pg的测试

location grpc 服务线上压测