UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

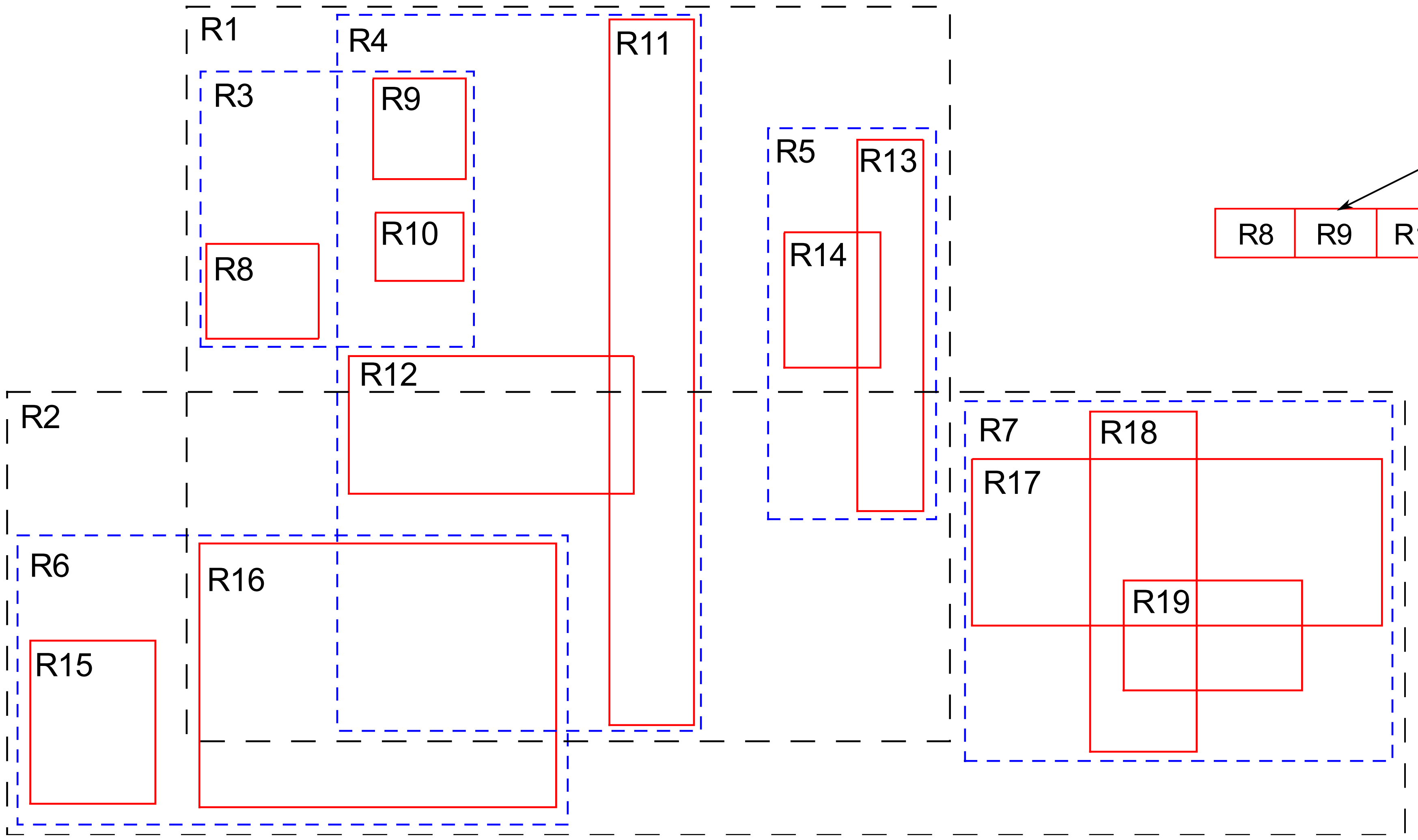
Sesión 4.2: R-Tree

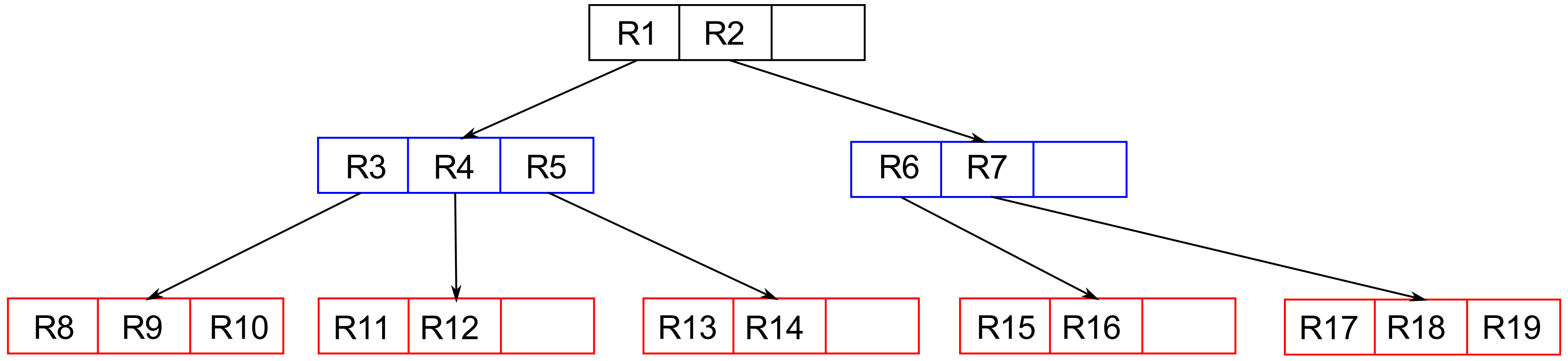
CS3102 EDA

Índice

1. R Tree

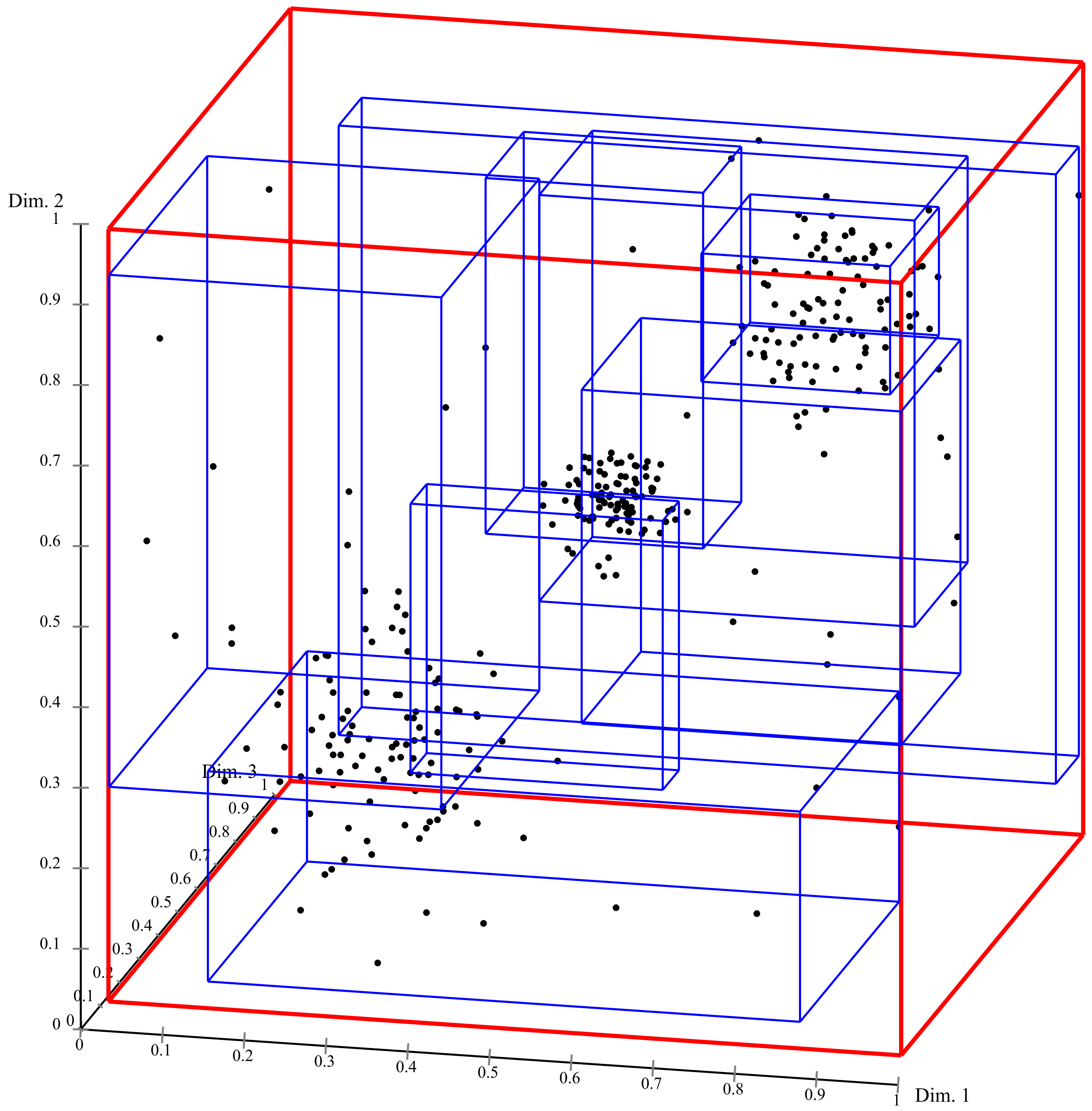
R-*Tree*





Guttman, Antonin (1984) "R-trees: A dynamic index structure for spatial searching".
Proceedings of the 1984 ACM SIGMOD international conference on Management of data. 1984. p. 47-57.

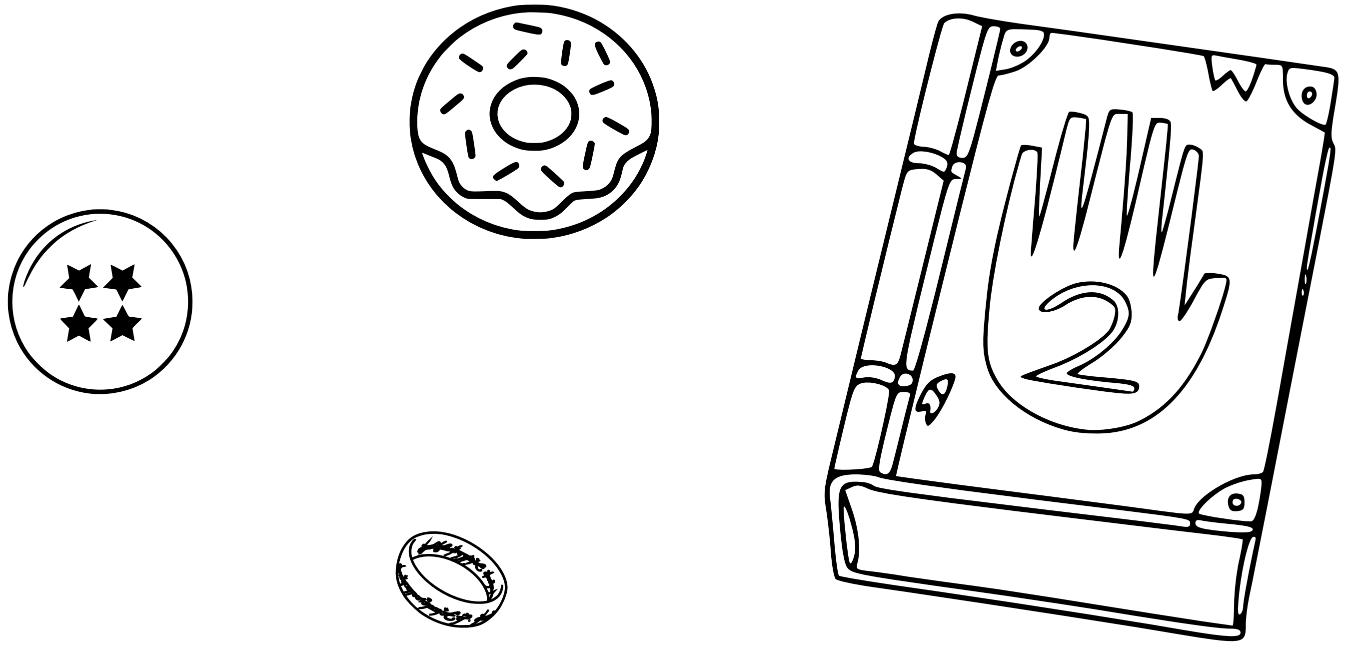
R-*Tree*



Universidad de Ingeniería y Tecnología

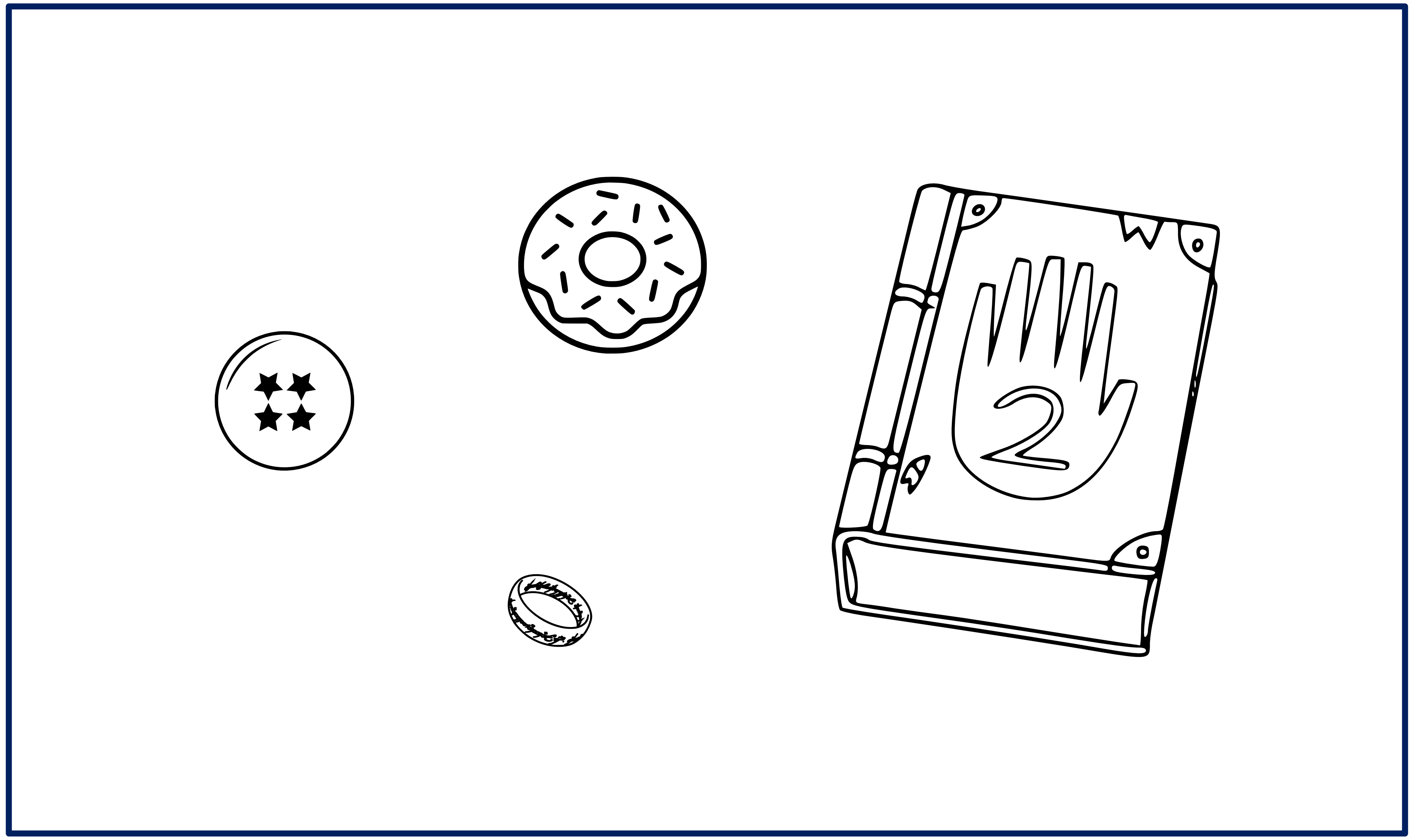
Guttman, Antonin (1984) "R-trees: A dynamic index structure for spatial searching".
Proceedings of the 1984 ACM SIGMOD international conference on Management of data. 1984. p. 47-57.

R-*Tree*



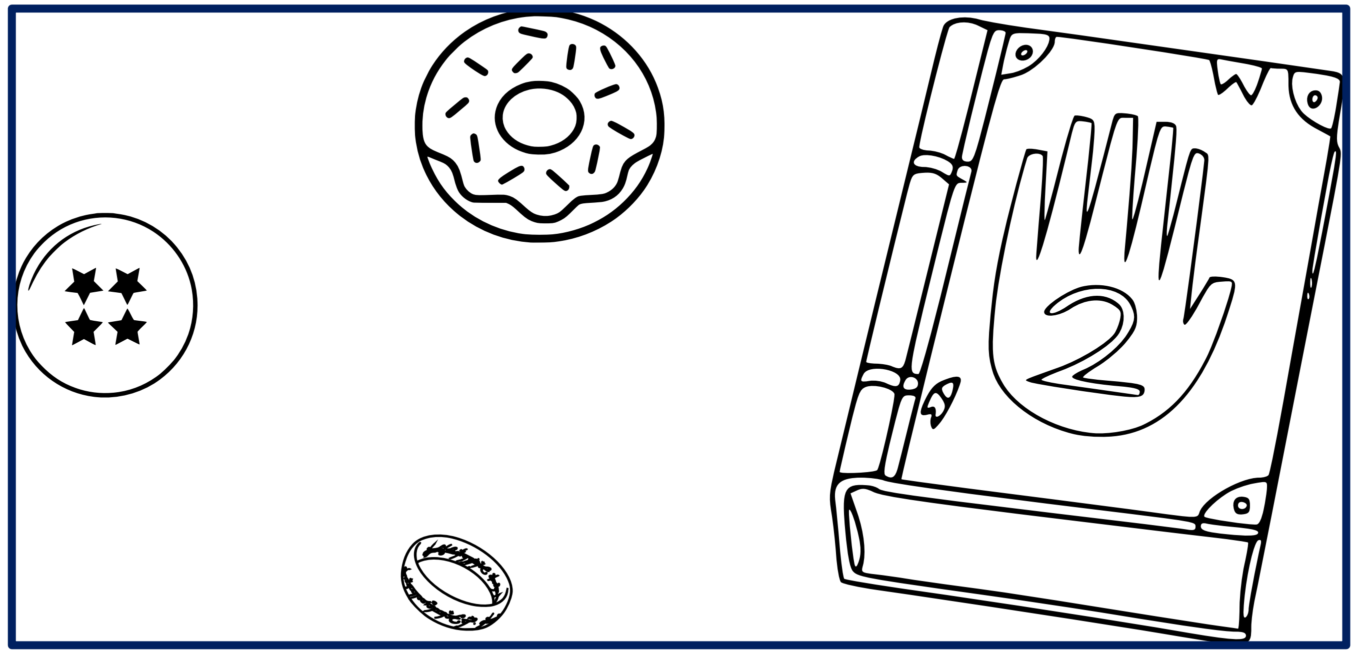
Objetos!

R-*Tree*



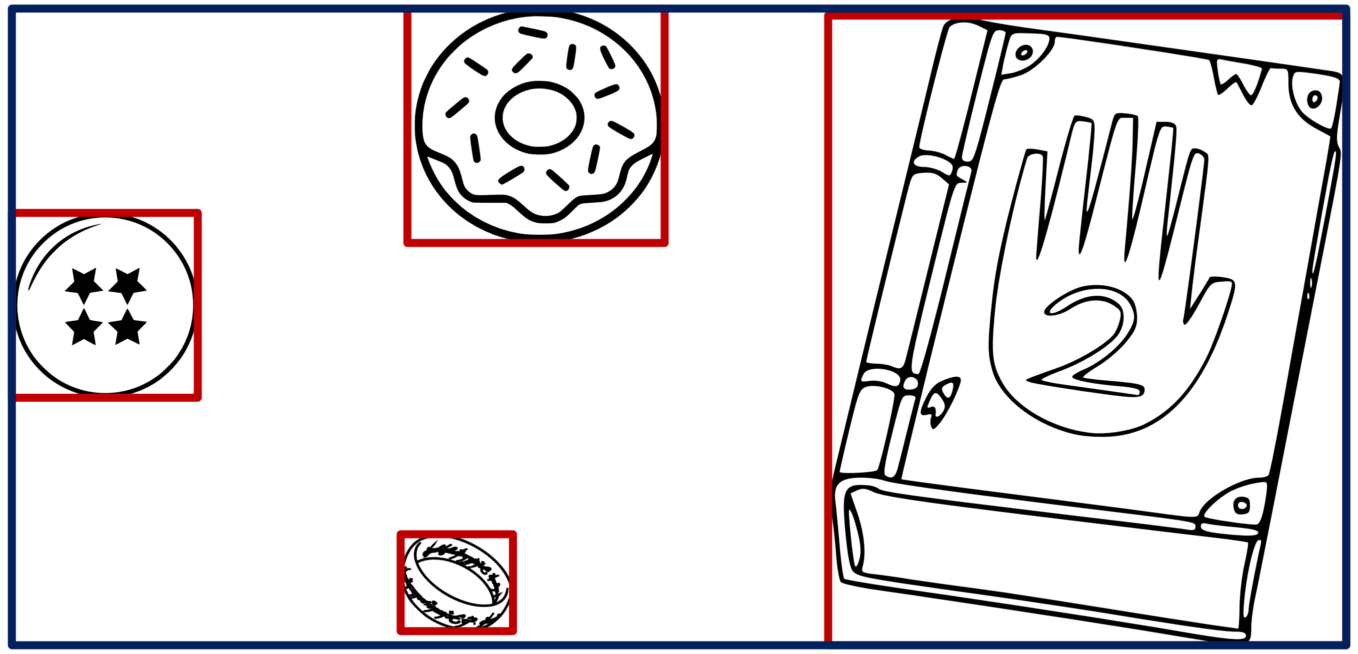
Bounding Box

R-*Tree*



Minimum Bounding Box

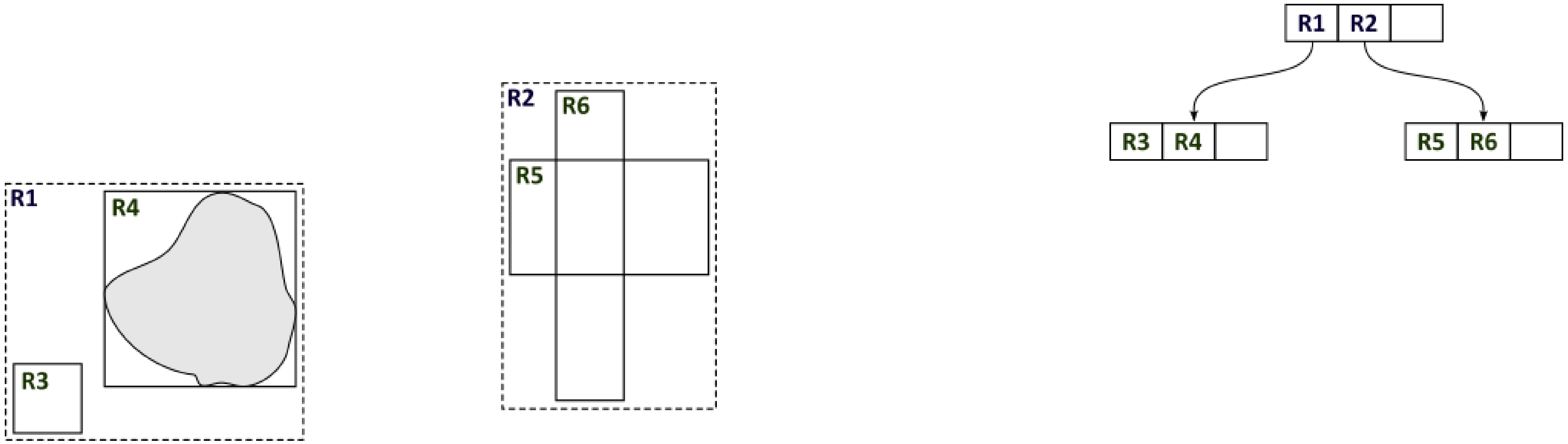
R-*Tree*



Minimum Bounding Box

R-Tree: *Nodo*

Formato del R-tree: (bounding box, puntero del nodo hijo)



R-*Tree*

Formato del R-tree: (bounding box, puntero del nodo hijo)

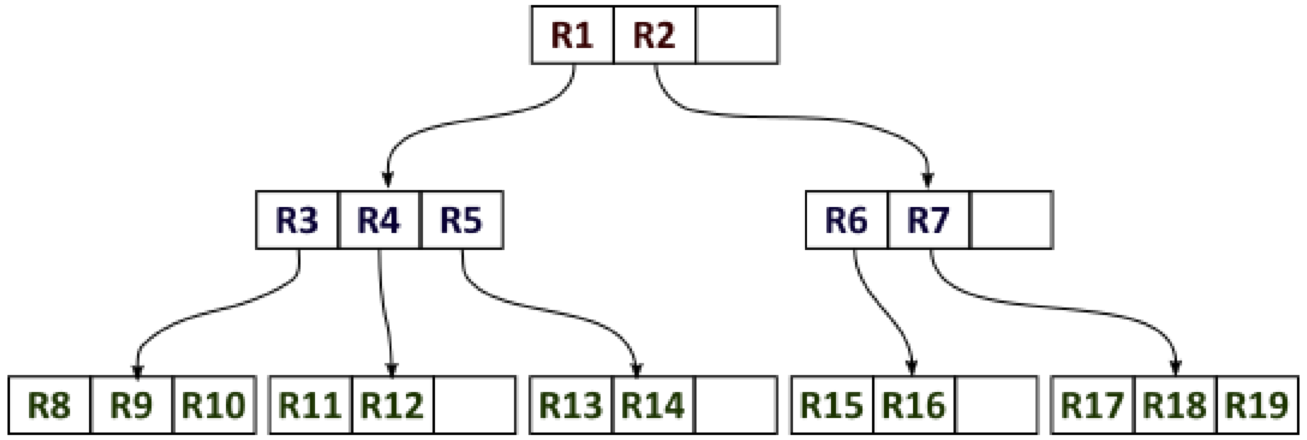
1. Cada nodo hoja (a menos que sea la raíz) puede albergar hasta M entradas, mientras que el número mínimo de entradas permitido es $m \leq M/2$.

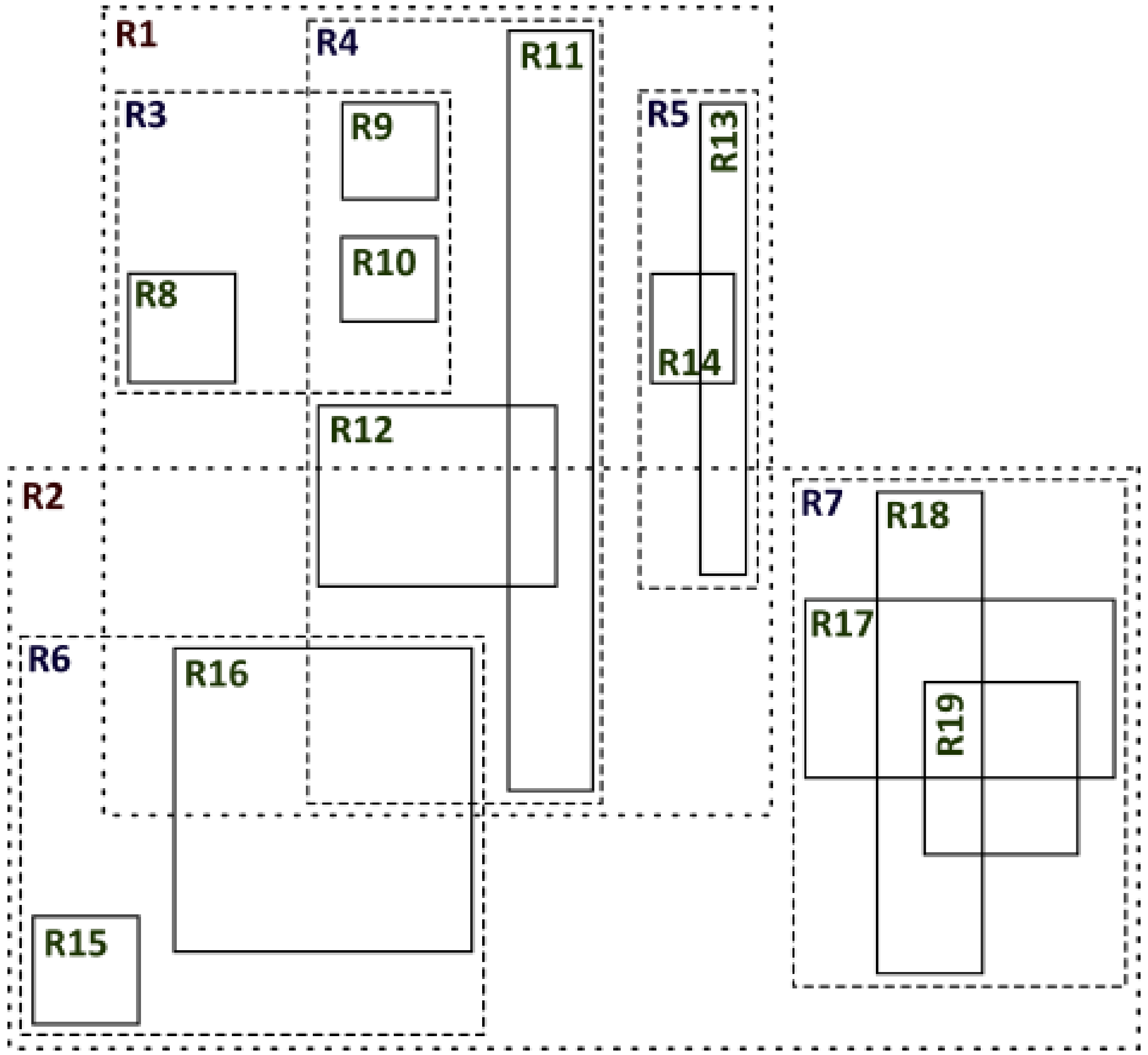
2. El número de entradas que cada nodo interno puede almacenar está de nuevo entre $m \leq M/2$ y M.

3. El número mínimo permitido de entradas en el nodo raíz es 2, a menos que sea una hoja (en este caso, puede contener cero o una sola entrada).

4. Todas las hojas del R-tree están en el mismo nivel.

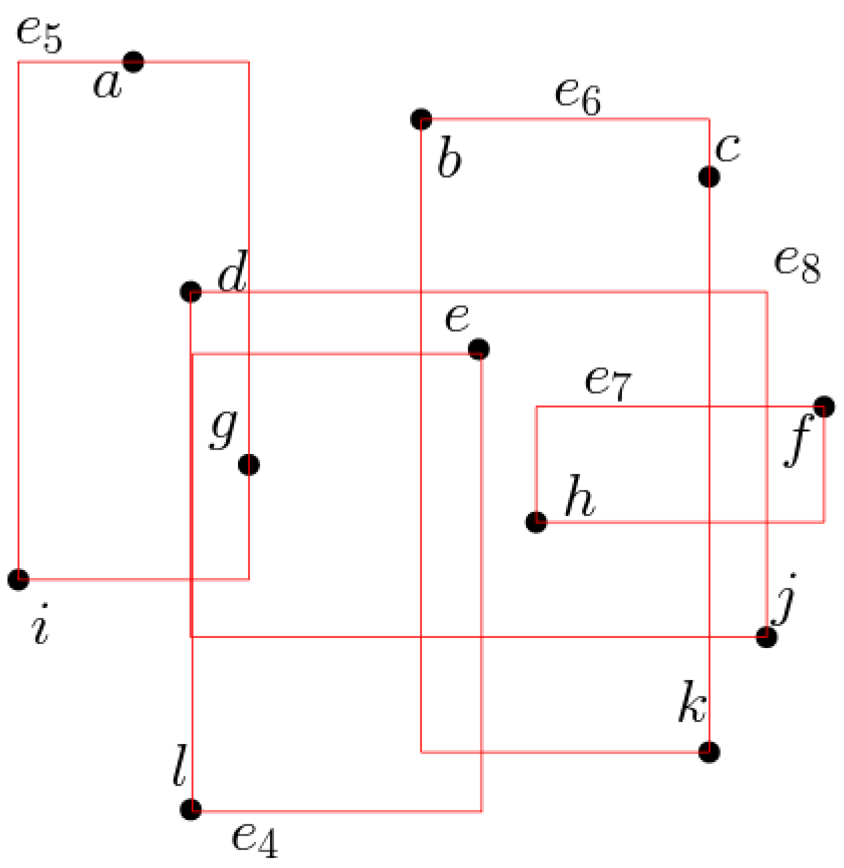
R-Tree: *Árbol*


R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19


R1
R4
R11
R3
R9
R5
R13
R10
R8
R14
R12
R2
R7
R18
R17
R6
R16
R19
R15

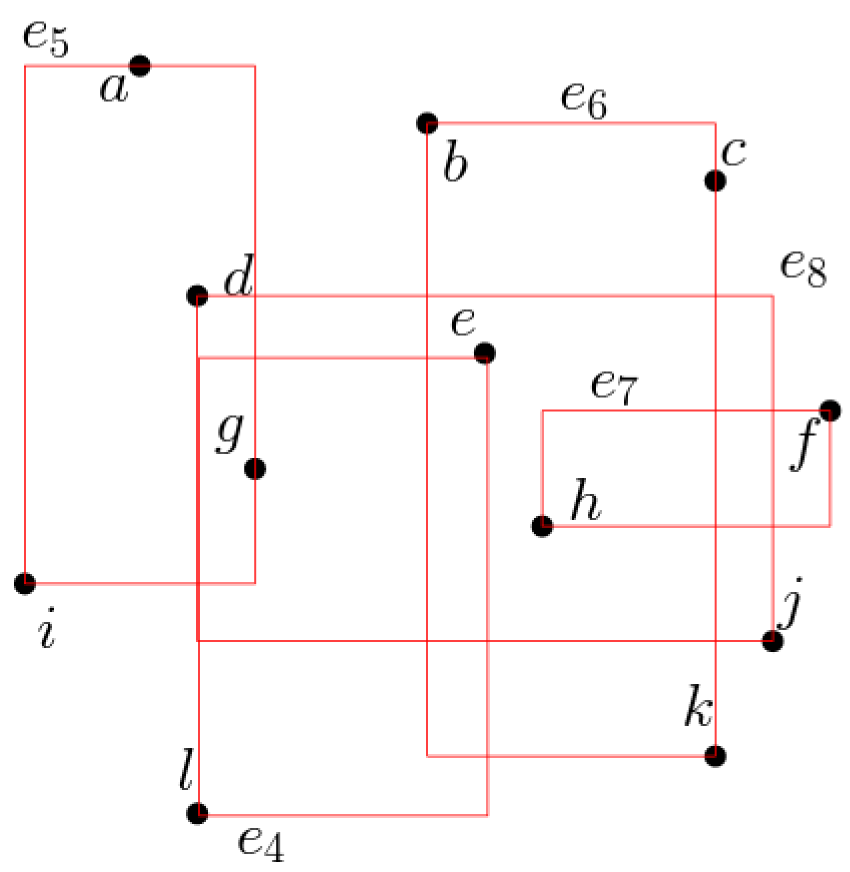
R-Tree: *Construcción*

Solución 1

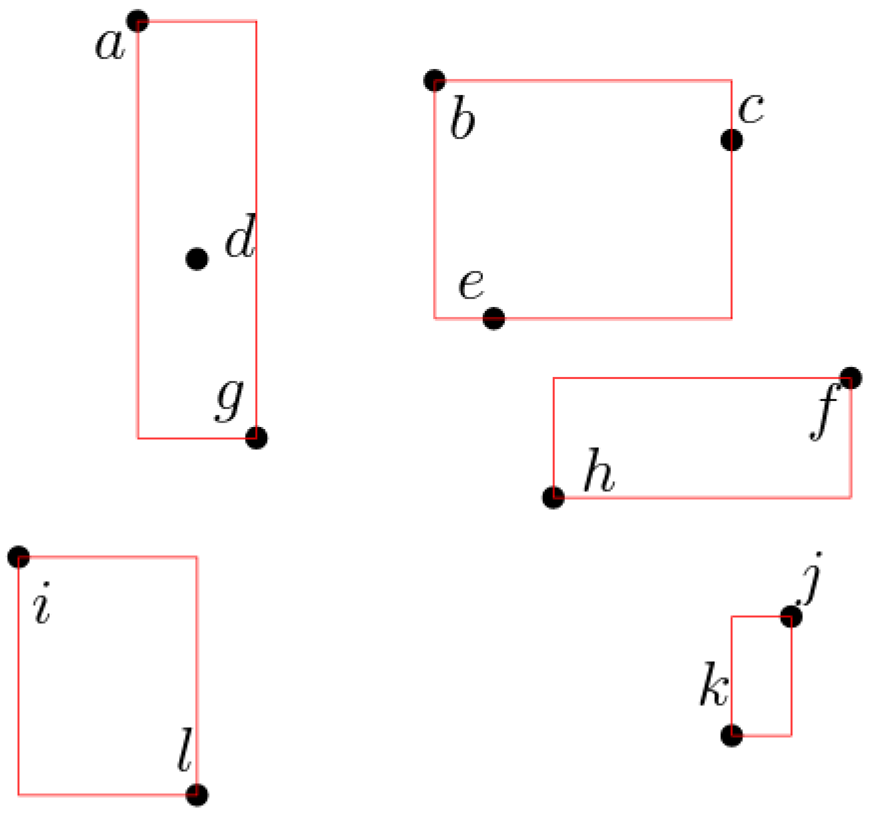


R-Tree: *Construcción*

Solución 1

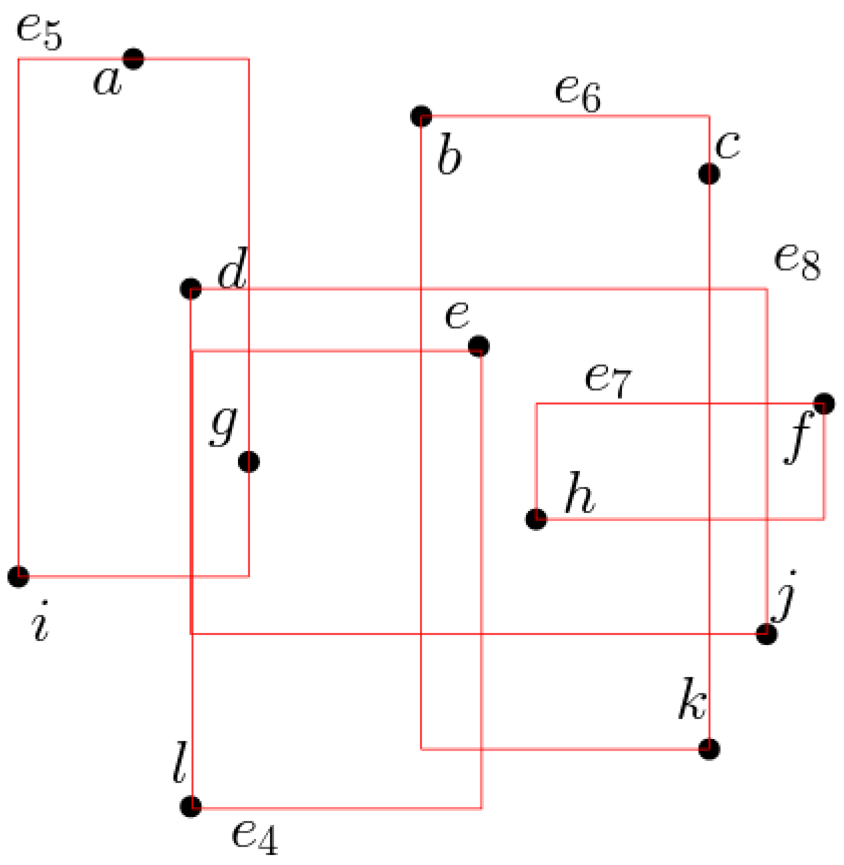


Solución 2

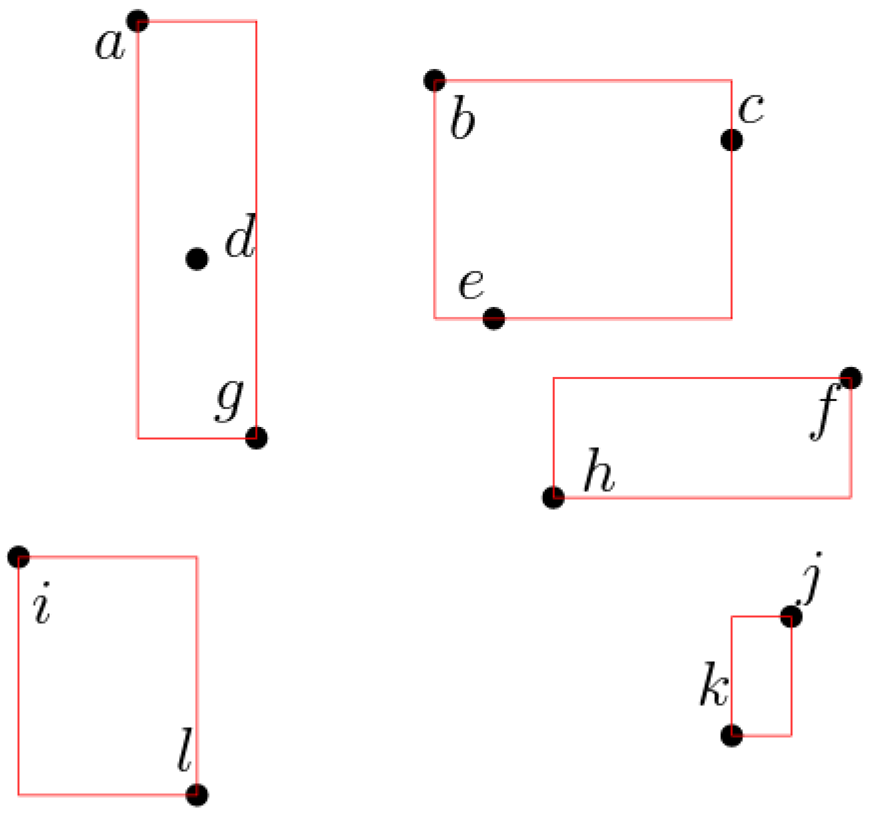


R-Tree: *Construcción*

Solución 1



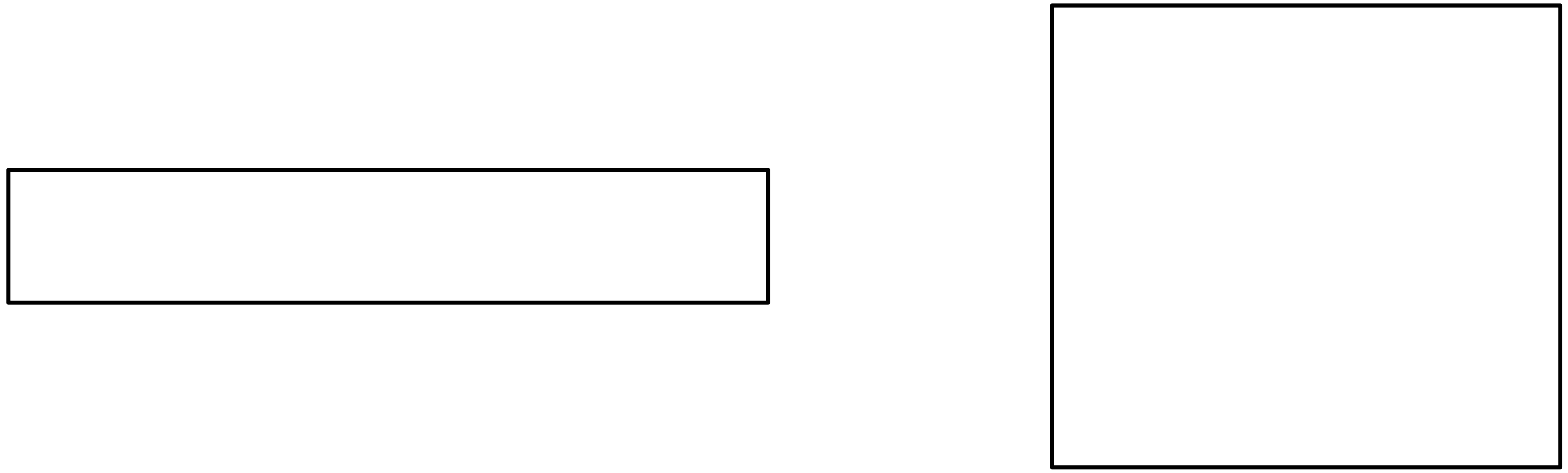
Solución 2



¿Cuál es mejor?

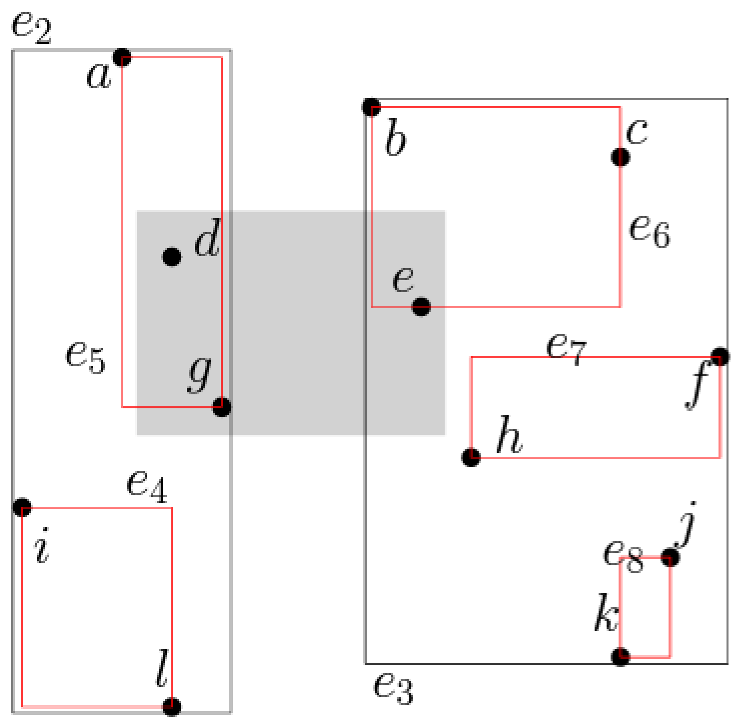
R-Tree: *Construcción*

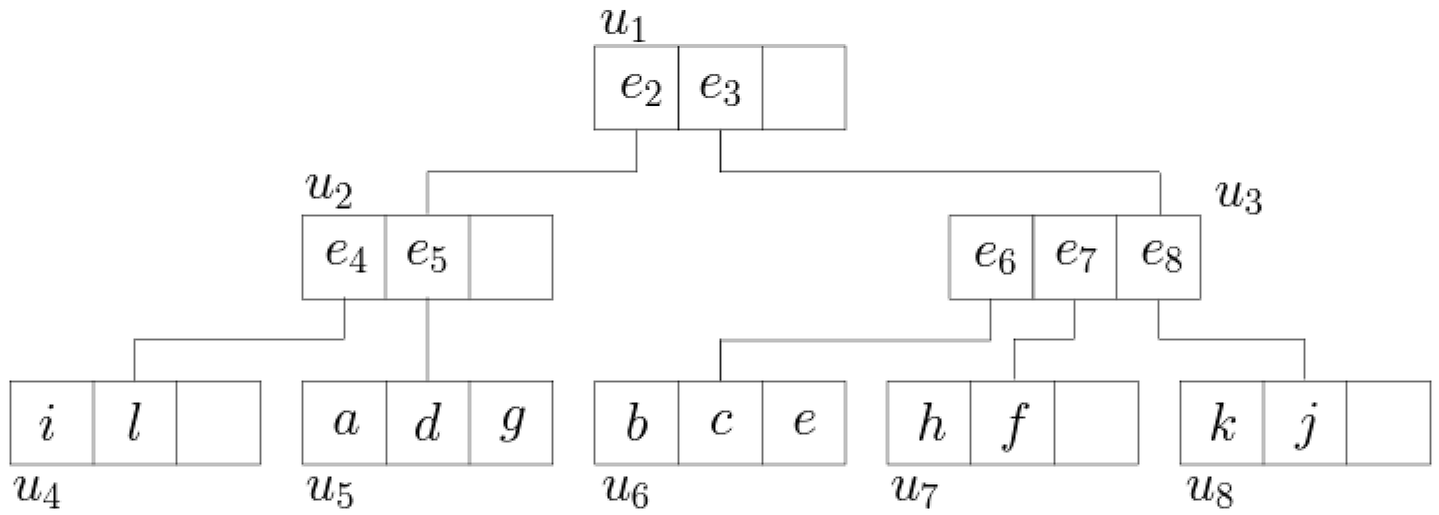
En general, el algoritmo de construcción del R-tree tiene como objetivo **minimizar** la suma de **áreas** o **perímetros** de todos los MBB.



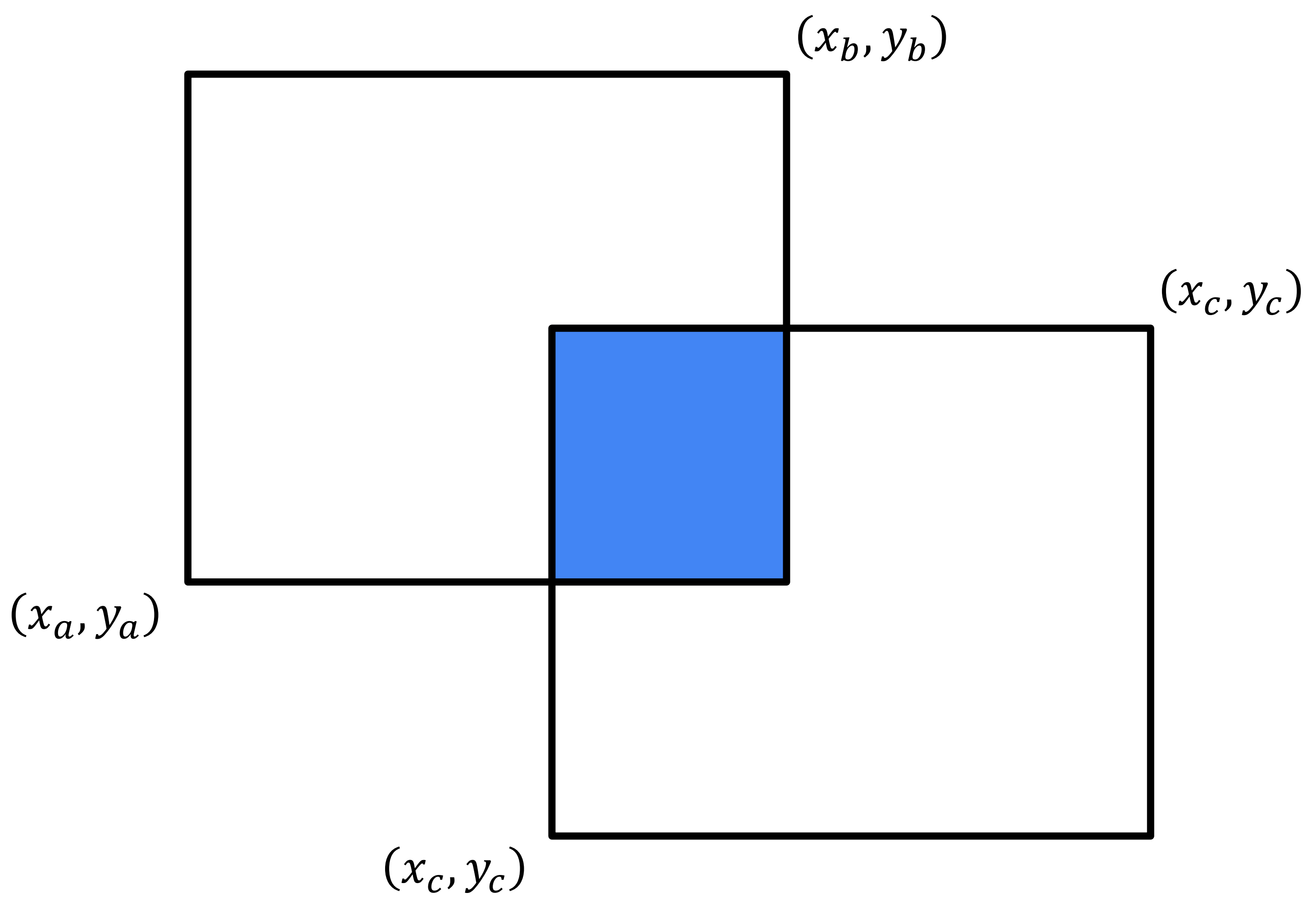
Un rectángulo con un perímetro más pequeño ***por lo general*** tiene un área más pequeña, pero no al revés.

R-Tree: *range query*



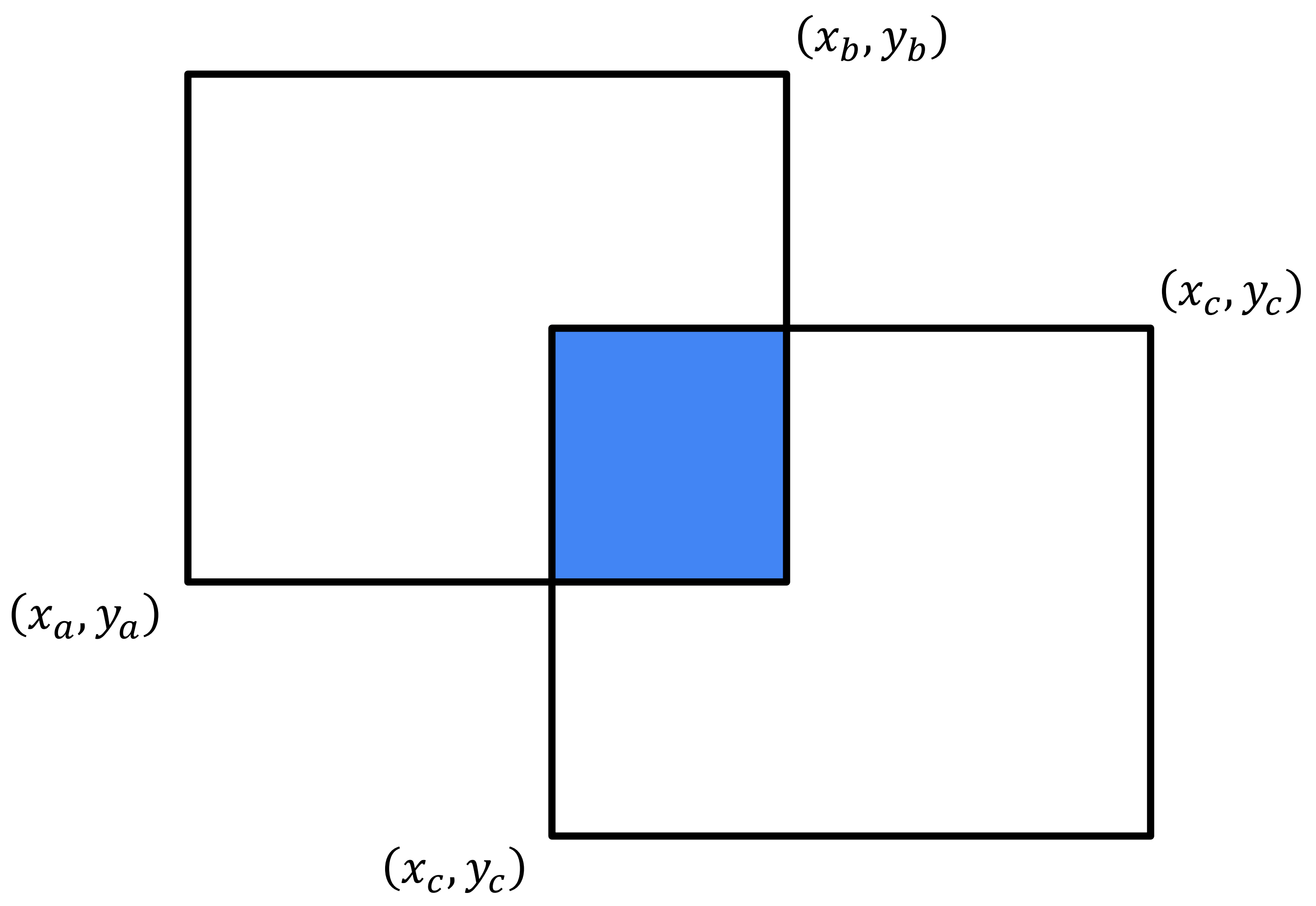


R-Tree: *range query*



¿Como determinamos el solapamiento entre dos MBB?

R-Tree: *range query*

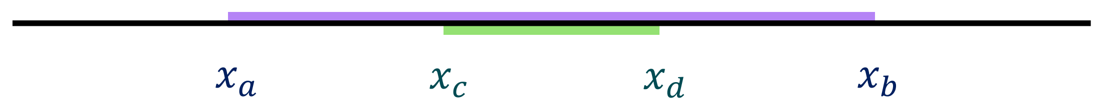


Una estrategia es analizar dimensión por dimensión

Sin perder generalidad, hacemos: $x_c = \max(x_a, x_c)$

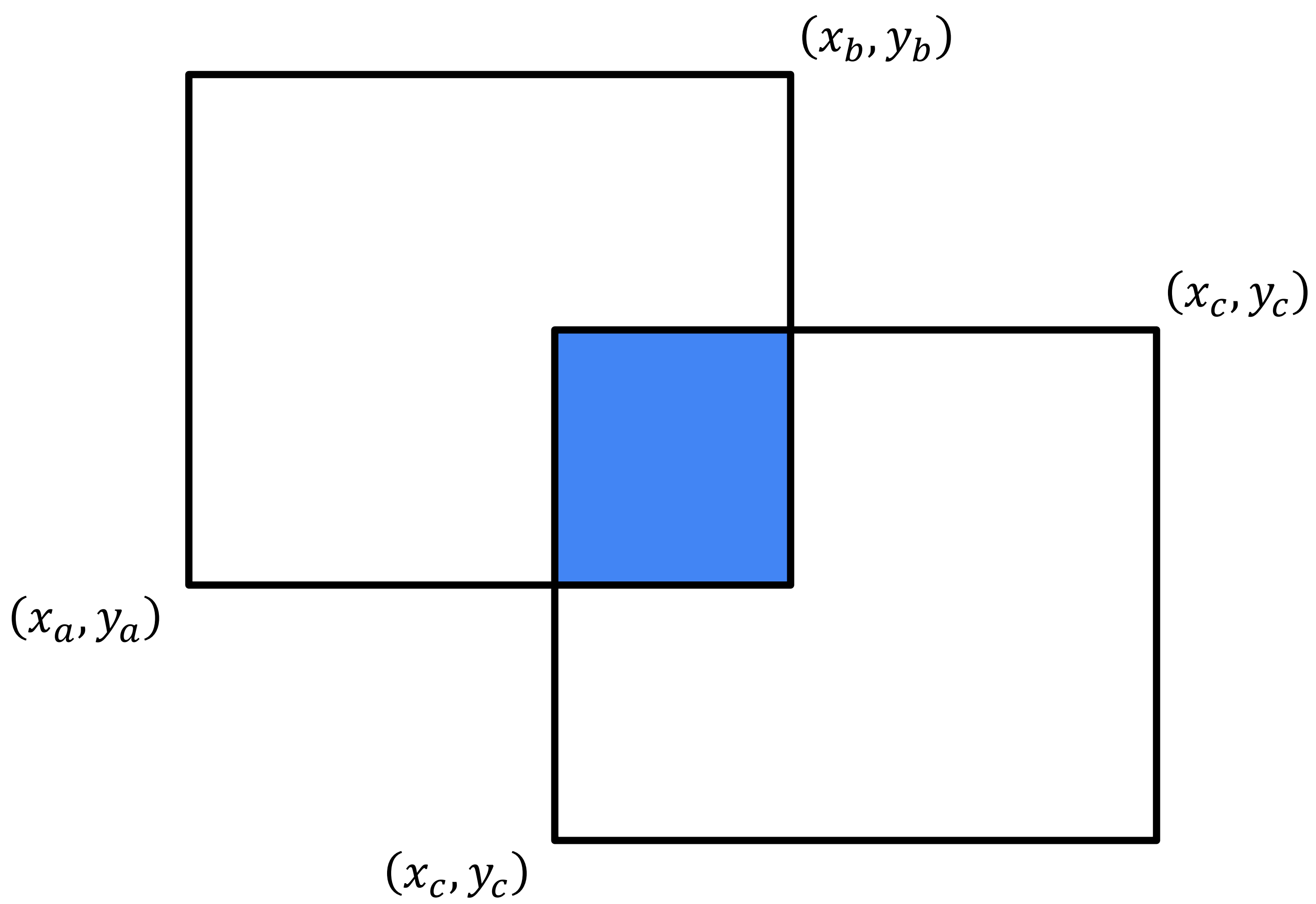
Caso 1:

Caso 2:



Caso 3:

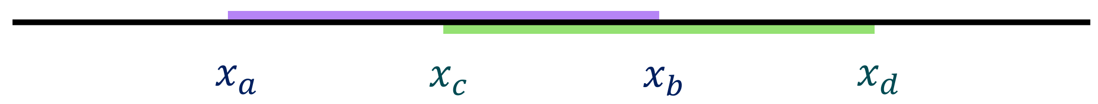
R-Tree: *range query*



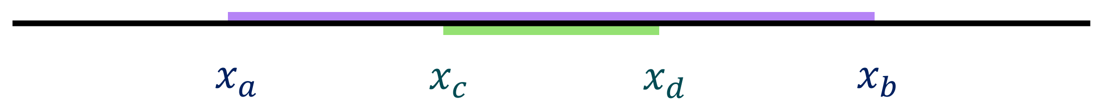
Una estrategia es analizar dimensión por dimensión

Sin perder generalidad, hacemos: $x_c = \max(x_a, x_c)$

Caso 1:



Caso 2:



Caso 3:

$$\ell_x = \max(0, \min(x_b, x_d) - \max(x_a, x_c))$$
$$\ell_y = \max(0, \min(y_b, y_d) - \max(y_a, y_c))$$

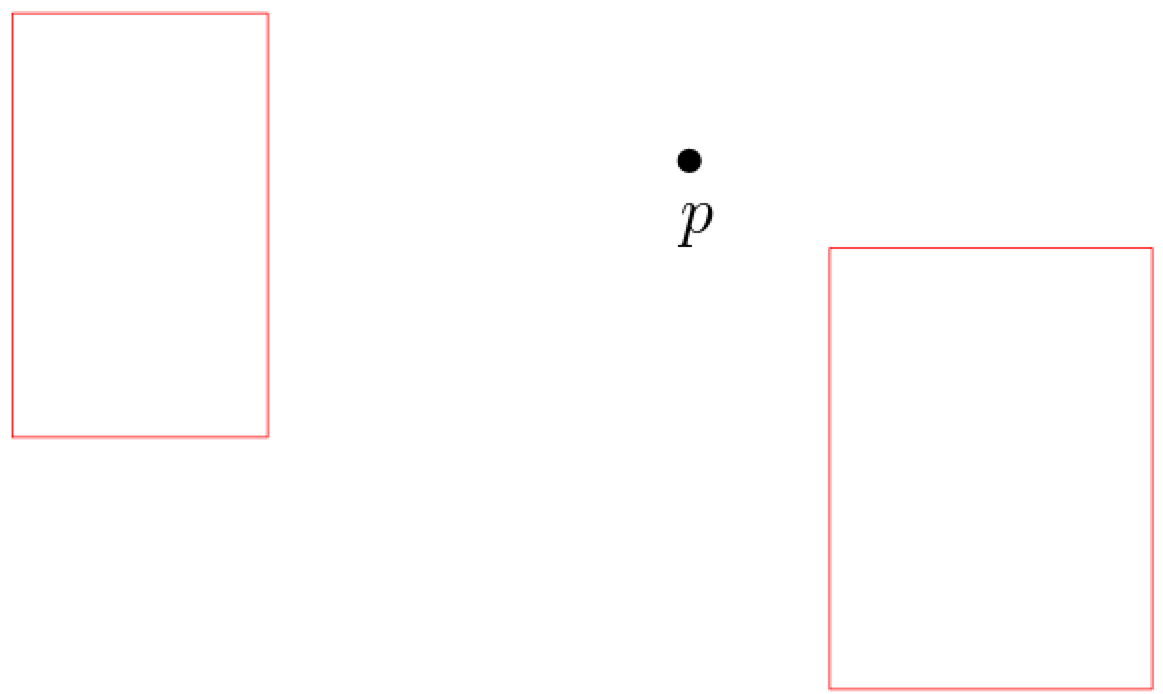
$$\Rightarrow \quad S = \ell_x \ell_y$$

R-Tree: *insert*

Algorithm `insert`(u, p)

1. **if** u is a leaf node **then**
2. add p to u
3. **if** u overflows **then**
   /* namely, u has $B + 1$ points */
4. `handle-overflow`(u)
5. **else**
6. $v \leftarrow$ `choose-subtree`(u, p)
   /* which subtree under u should we insert p into? */
7. `insert`(v, p)

R-Tree: *Choose-subtree*



Devolver al nodo cuyo MBB requiera el **mínimo** aumento de perímetro para cubrir p.

En caso de empate, retorna el MBB más pequeño.

R-Tree: *Overflow Handling*

Algorithm `handle-overflow`(u)

`split`(u) into u and u'
if u is the root **then**
    create a new root with u and u' as its child nodes
else
    $w \leftarrow$ the parent of u
    update $MBR(u)$ in w
    add u' as a child of w
    if w overflows **then**
        `handle-overflow`(w)

R-Tree: *Splitting*

Linear Split

1. Elija dos objetos como semillas, de modo que estén lo más separados posible.

2. Considera cada objeto restante en un orden aleatorio y asígnalo al nodo que requiera la menor ampliación de su MBB.

Quadratic Split

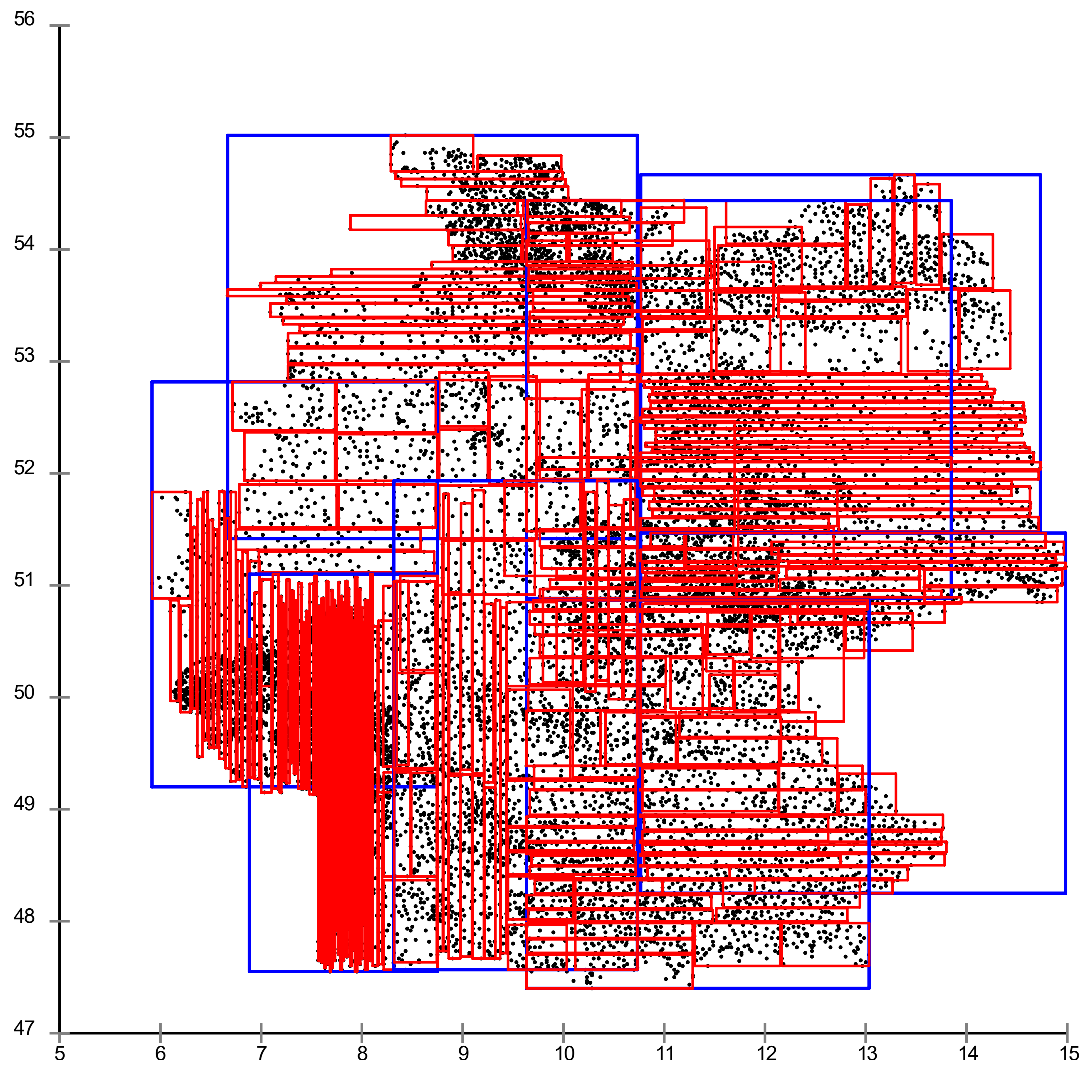
1. Elija dos objetos como semillas para los dos nodos, de modo que crean el **mayor espacio muerto** posible.

2. Asigne los objetos restantes a uno de los dos grupos. Para cada objeto, calcule el aumento en el área de la MBB que resultaría de añadir el rectángulo a cada grupo. Asigne el objeto al grupo que suponga el **menor aumento de área**. En caso de empate, asigna el rectángulo al grupo con menor área o menor número de elementos.

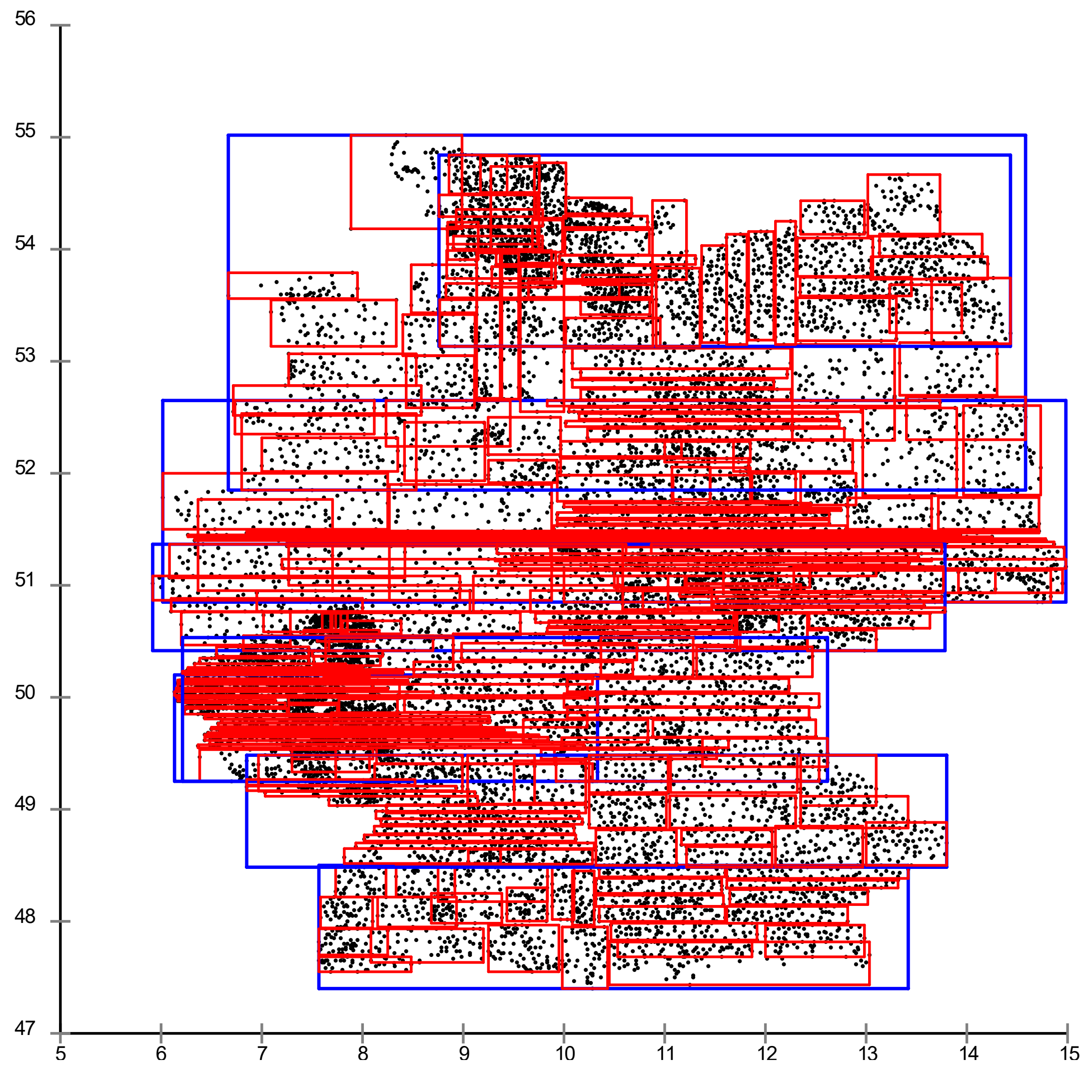
Evite que cualquier nodo tenga menos elementos que el valor mínimo asignado!

R-Tree: *Splitting*

Linear Split



Quadratic Split



R-_Tree_

Pero… Hay otra forma…

R-Tree: *Splitting a leaf node*

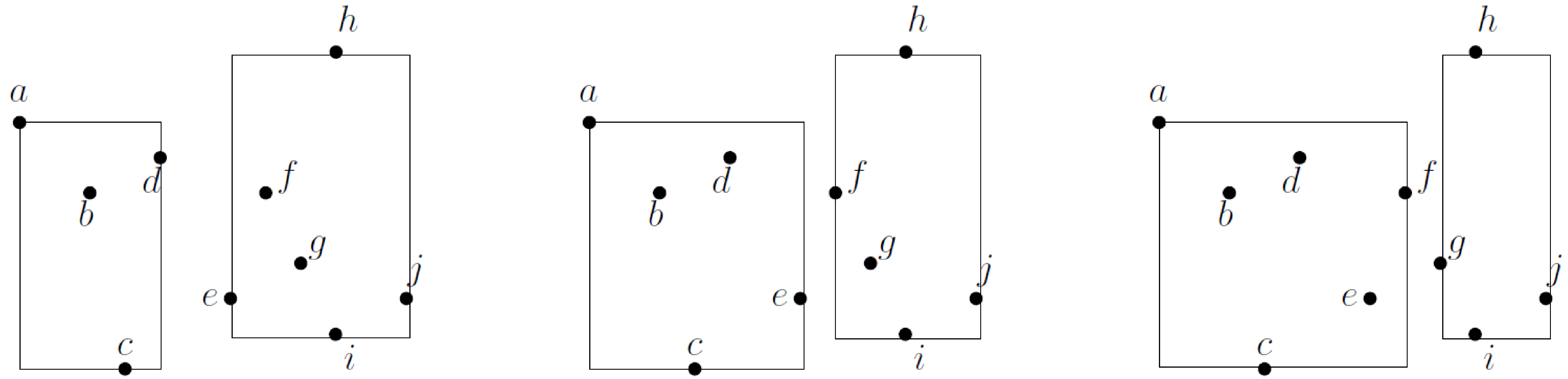
Sea S un conjunto de $B+1$ puntos. Divida S en dos conjuntos disjuntos S_1 y S_2 para **minimizar** la suma del perímetros de MBR(S_1) y MBR(S_2), sujeto a la condición de que $|S_1| \geq 0.4M$ y $|S_2| \geq 0.4M$

Algorithm `split`(u)

1. $m =$ the number of points in u
2. sort the points of u on x-dimension
3. **for** $i = \lceil 0.4B \rceil$ to $m - \lceil 0.4B \rceil$
4. $\quad S_1 \leftarrow$ the set of the first i points in the list
5. $\quad S_2 \leftarrow$ the set of the other i points in the list
6. $\quad$ calculate the perimeter sum of $MBR(S_1)$ and $MBR(S_2)$; record it if this is the best split so far
7. Repeat Lines 2-6 with respect to y-dimension
8. **return** the best split found

R-Tree: *Splitting a leaf node*

Sea S un conjunto de $B+1$ puntos. Divida S en dos conjuntos disjuntos S_1 y S_2 para **minimizar** la suma del perímetros de MBR(S_1) y MBR(S_2), sujeto a la condición de que $|S_1| \geq 0.4M$ y $|S_2| \geq 0.4M$



Hay **3** posibles divisiones a lo largo de la dimensión x. Recuerde que cada nodo debe tener al menos $0.4M = 4$ puntos (aquí $M = 10$).

R-Tree: *Splitting a internal node*

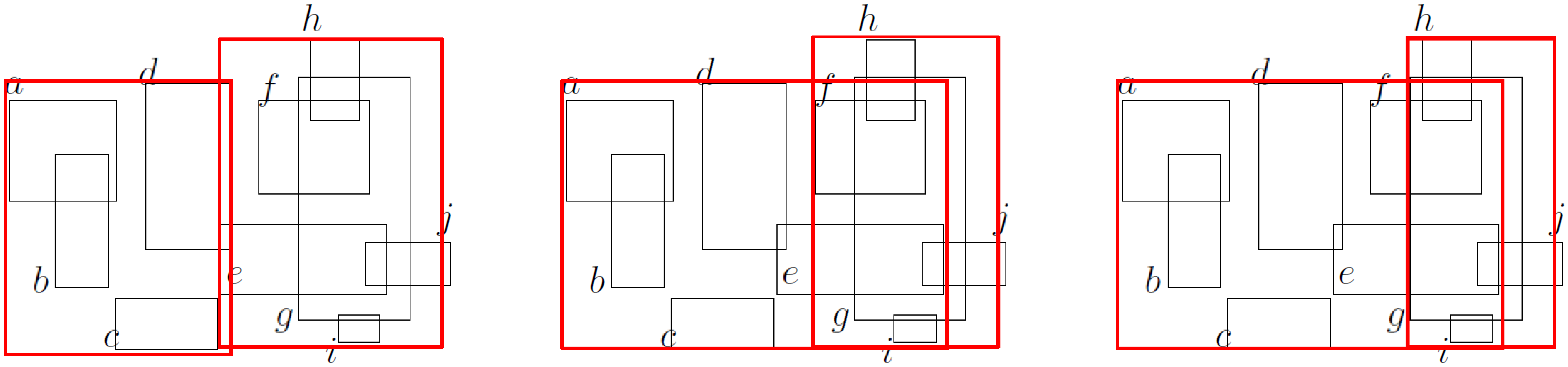
Sea S un conjunto de $B+1$ rectángulos. Divida S en dos conjuntos disjuntos S_1 y S_2 para **minimizar** la suma del perímetros de MBR(S_1) y MBR(S_2), sujeto a la condición de que $|S_1| \geq 0.4M$ y $|S_2| \geq 0.4M$

Algorithm split(u)
/* u is an internal node */

1. m = the number of points in u
2. sort the rectangles in u by their left boundaries on the x-dimension
3. **for** $i = \lceil 0.4B \rceil$ to $m - \lceil 0.4B \rceil$
4. $\quad S_1 \leftarrow$ the set of the first i rectangles in the list
5. $\quad S_2 \leftarrow$ the set of the other i rectangles in the list
6. $\quad$ calculate the perimeter sum of $MBR(S_1)$ and $MBR(S_2)$; record it if this is the best split so far
7. Repeat Lines 2-6 with respect to the right boundaries on the x-dimension
8. Repeat Lines 2-7 w.r.t. the y-dimension
9. **return** the best split found

R-Tree: *Splitting a internal node*

Sea S un conjunto de $B+1$ rectángulos. Divida S en dos conjuntos disjuntos S_1 y S_2 para **minimizar** la suma del perímetros de MBR(S_1) y MBR(S_2), sujeto a la condición de que $|S_1| \geq 0.4M$ y $|S_2| \geq 0.4M$



Hay **3** posibles divisiones con respecto a los límites izquierdos en la dimensión x. Recuerda que cada nodo debe tener al menos $0.4M = 4$ puntos (aquí $M = 10$).

R-Tree: *Delete*

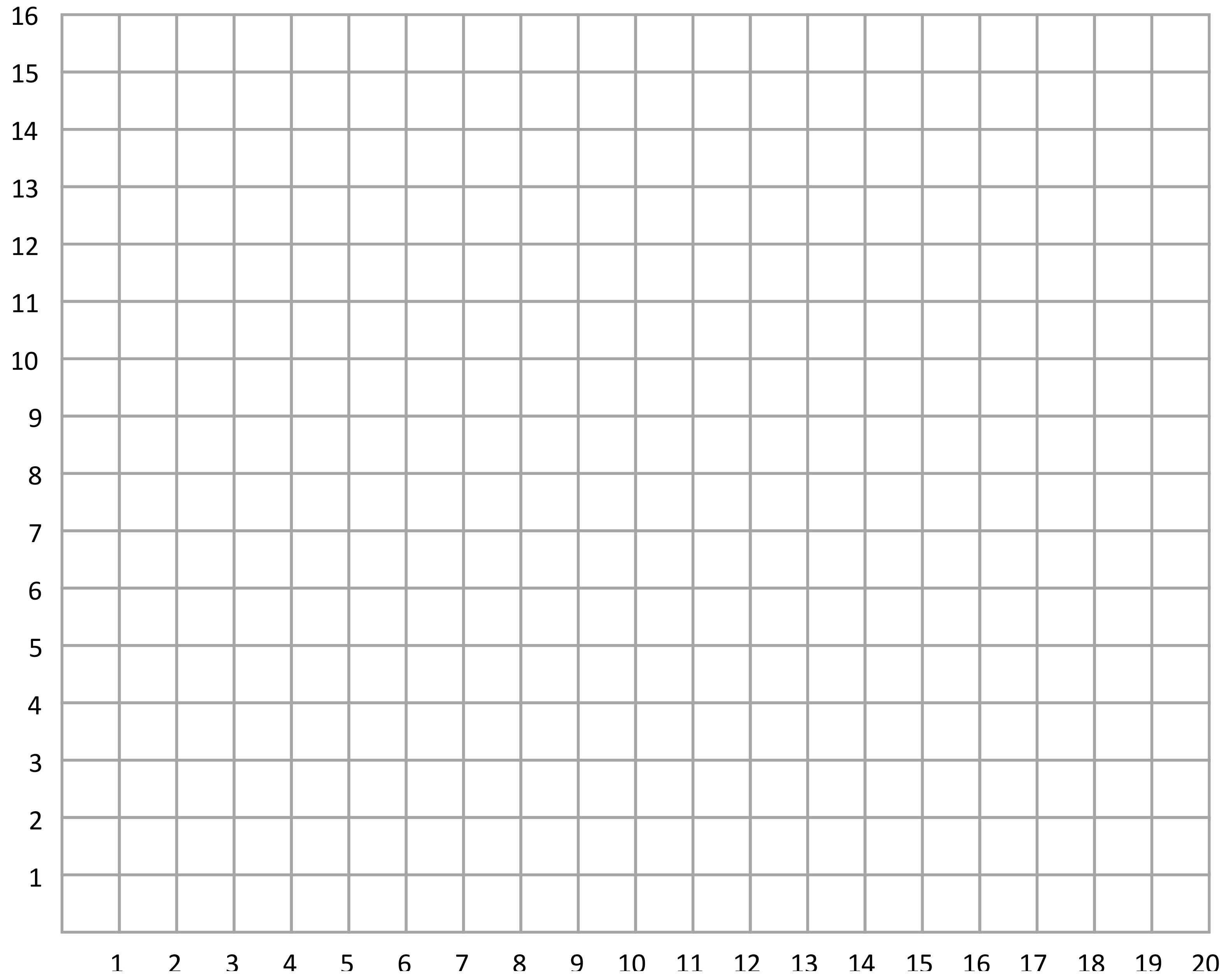
1. Encuentra el nodo hoja que contiene la entrada E

2. Eliminar E de este nodo

3. Si está incompleto:
    - Elimine el nodo y su referencia en el padre
    - Vuelva a insertar los huérfanos (otras entradas) en el árbol usando el algoritmo de inserción.

4. Si durante este proceso el nodo raíz tiene un solo elemento, la altura del árbol puede disminuir.

R-*Tree*


16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA
INGENIERÍA
MECATRÓNICA
BIOINGENIERÍA
CIENCIA DE
LA COMPUTACIÓN
AMBIENTAL