FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

TRABAJO PRÁCTICO N°1

IMPLEMENTACIÓN DE ÁRBOLES FÍSICOS

Universidad de Buenos Aires
Facultad de Ingeniería
Departamento de Computación
75.06 – Organización de Datos - Cátedra Servetto

Grupo 4
Tutor: Nicolás Fernández Theillet
Integrantes:

Cristian Martín Westergaard	80483
Maximiliano Alexis Montiel	93157

Índice

1 Descripción General

1.1 (Breve) Introducción a los árboles de punto óptimo

En 1991, Jeffrey Uhlmann publicó un paper acerca de un nuevo árbol de espacio métrico: el árbol de punto óptimo. Como todos los demás árboles de espacio métrico, tiene como finalidad facilitar la resolución de los problemas de proximidad. Se basan en la división recursiva del espacio de estudio. Un árbol famoso de este tipo es el k-d.

Al contrario del k-d y otros, que dividen el espacio en secciones rectangulares, el árbol de punto óptimo utiliza un enfoque totalmente distinto: Secciona el espacio en “burbujas”: es decir, separa lo que queda dentro de la burbuja de lo que queda afuera.

Si lo pensamos desde un enfoque meramente geométrico, veremos que estas burbujas agrupan de forma natural los puntos del espacio, en torno de sus centros, que denominaremos pivotes.

Más adelante veremos que aplicando la desigualdad triangular a los pivotes, podremos descartar secciones completas del árbol, logrando de modo eficiente y elegante, resolver el problema de las búsquedas de proximidad.

1.2 Características básicas

Como ya vimos, los árboles de punto óptimo dividen el espacio en burbujas. Para ello, se valen de una función distancia. Es decir, basta con definir una función distancia para poder construir un árbol de este tipo. A diferencia de los otros árboles, se pueden aplicar en espacios no euclidianos, e inclusive, no cartesianos, lo que les confieren un gran abanico de aplicaciones en los más diversos ámbitos.

Sin embargo, se debe tener en cuenta que la función distancia no puede ser cualquier cosa. Debe cumplir con lo siguiente:

$d(x,\ x)\ =\ 0$	Reflexión
$d(x,\ y)\ =\ d(y,\ x)$	Conmutatividad
$d(x,\ y)\ \leq\ d(x,\ z)\ +\ d(x,\ y)$	Desigualdad triangular

Cabe notar que la única restricción sobre el conjunto imagen es sea un conjunto que tenga definida la suma, y que se pueda determinar, para todo par de elementos, la igualdad y precedencia (menor y mayor).

Sin más preámbulos…

1.3 Árbol de punto óptimo persistente

Ésta es una versión adaptada del concepto original, que se puede persistir a disco, lo que le permite ser usado como sistema de organización de datos.

Se diseñó de manera que el árbol sea un archivo de datos maestros auto-organizado y auto-balanceado, con registros de longitud variable, organizados por clave.

Entre sus características, podemos mencionar:

- Todos los nodos tienen una bandera indicadora. Si está en falso, el nodo es una hoja.
- Los registros se guardan tanto en hojas como en nodos internos.
- Los nodos internos se componen de un pivote, un radio, 2 referencias a nodos (designados izquierdo y derecho) y un conjunto de registros de datos.
- Los nodos tienen un factor de carga mínima de ⅓ de la capacidad máxima.
- En todo momento, ningún nodo, a excepción de la raíz (en estado de hoja), puede tener menos de la carga mínima.
- Todo pivote posee la misma estructura que cualquier identificador de registro.
- Un pivote no necesariamente coincide con un identificador de registro existente.
- El contenido del uno de los nodos izquierdo se encuentra dentro de la burbuja del pivote. El contenido del derecho se encuentra fuera de la burbuja del pivote.
- En el nodo interno se guardan registros, que pueden o no estar dentro de la burbuja.

2 Métodos ABM

Aquí no sólo hablaremos sobre los métodos ABM (Altas, Bajas, Modificaciones), sino cómo operar con el árbol. Comencemos:

2.1 Creación del Árbol

En la creación del árbol, se debe indicar por parámetro:

- El tamaño del bloque. Idealmente debería ser una potencia de 2 por 512 bytes.
- El descriptor de registro de datos. Donde se toma el primer campo como la clave principal.

Dependiendo del tipo de la clave, se selecciona internamente la función distancia. Si se trata de una cadena, se usa la distancia de edición de palabra, si en cambio, se trata de un campo numérico, se usa la norma euclídea.

Dado que el campo es uno solo, y se requieren 2 coordenadas, el sistema los construye a partir del siguiente método:

2.1.1 Generación de coordenadas a partir de una clave numérica:

Se generan 2 números de tamaño igual a la mitad del tamaño de la clave en bytes, tomando los bits pares para uno, y los bits impares para otro. De esta manera, se asegura que para un conjunto de claves numeradas en orden, se obtienen coordenadas de puntos distribuídos de manera uniforme.

2.2 Interfaz de usuario del árbol

Las opciones que soporta nuestro árbol son:

- Búsquedas
- Altas
- Bajas
- Modificaciones

Pasamos a describir cada una de ellas a continuación.

2.2.1 Búsquedas

Las búsquedas de registros son una parte fundamental de todos los árboles.

En este tipo de árbol, todas las búsquedas se realizan mediante una clave, comienzan con la raíz y se recorren los nodos internos hasta llegar a las hojas.

Se pueden distinguir varios tipos de búsquedas en un árbol de punto óptimo.

2.2.1.1 Búsquedas puntuales

Las más simples, existen en todos los árboles. El objetivo es hallar un único registro, identificado por una clave.

Se basa en el invariante más importante de éste árbol:

$$X \in P_i \Leftrightarrow d(X, P_i) < r_i \qquad (1)$$

Es decir, si un registro pertenece a un pivote, la distancia al mismo es menor al radio, y viceversa, si no pertenece, es porque se encuentra a una distancia mayor que el radio.

Para recorrer los nodos internos, se busca primero el registro dentro de los contenidos del nodo. Si no se encuentra, se calcula la distancia de la clave al pivote. Si es menor al radio, se continúa la búsqueda en el nodo izquierdo. Caso contrario, se prosigue la búsqueda con el derecho. Este recorrido es similar a las búsquedas de árbol, llevando un tiempo O(log n), siendo n la altura del árbol.

Llegados a la hoja, se recorren todos los registros, comparando las claves. El registro que coincida es el registro buscado, y la búsqueda terminará con éxito. Si no se encuentra, y en virtud de (1), el método terminará reportando no encontrado, ya que el registro no se podrá encontrar en ninguna otra hoja. Este tipo de búsqueda es lineal, de orden O(k), siendo k la cantidad de registros. En promedio se recorren la mitad de los registros.

2.2.1.1.1 Búsqueda puntual aproximada

Es una variante del método de búsqueda puntual. En este caso, para cada nodo se registra el de menor distancia. Al finalizar la función de búsqueda (de carácter recursivo), el registro devuelto será el de menor distancia a la clave. Debido a la naturaleza de la función distancia, se garantiza de que se devolverá el registro asociado con la clave (en caso de que exista), dado que la distancia de una clave con sigo misma devuelve 0, la menor de las distancias.

En caso contrario, el registro devuelto será el más próximo encontrado. Al igual que el caso anterior, el recorrido del árbol es de orden O(log n) y es de O(k) para la hoja.

Estas dos maneras de buscar se usan para satisfacer las consultas normales de un sistema de datos. Sin embargo, el propósito principal de estos árboles es la de poder hacer búsquedas por proximidad.

2.2.1.2 Búsquedas por proximidad

Este conjunto de técnicas de búsquedas son propias de los árboles de espacios métricos, cuyo propósito principal es justamente ayudar a resolver el problema de encontrar un grupo de registros próximos a un punto, o inclusive todos los registros que se encuentren a menor distancia que una cota.

Consisten básicamente en establecer el rango de búsqueda, o distancia máxima a la cual debe encontrarse un registro de una clave en particular para poder ser considerado vecino.

Para este árbol hay de 2 tipos, exhaustivos y por pivotes:

- El método exhaustivo, como su nombre lo indica, implica recorrer todos nodos del árbol, algo impracticable para árboles persistidos a disco, que pueden llegar a contener más registros que los que podrían caber en la memoria. De hecho, los árboles de espacio métrico se diseñaron para reducir la búsqueda de orden O(n) a un mero O(k log n), con un k tentativo en 1.

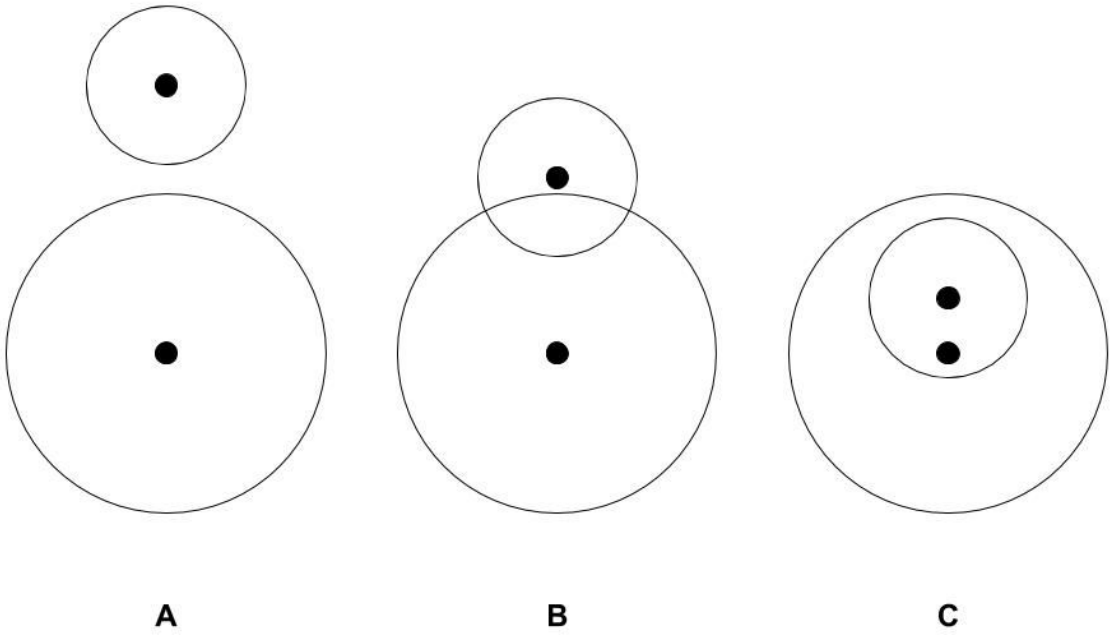
- Los métodos por pivotes, que son los que veremos a continuación, se basan en aplicar la desigualdad triangular para descartar de manera temprana pivotes, y con ello, secciones enteras del árbol, a modo de poda (en analogía con los árboles de decisión).

Repasemos brevemente el concepto de desigualdad triangular:

$$d(a,b) \leq d(a,c) + d(b,c) \qquad (2)$$

Que implica: Dado todo par de puntos, la distancia entre ellos es menor o igual a la suma de las distancias del par con un tercero.

Dado que estaremos haciendo intersecciones de burbujas (la burbuja de búsqueda, centrada en la clave y radio igual al rango, con la burbuja de los pivotes), debemos tener presente los siguientes 3 casos:



Caso A: Intersección vacía.
Caso B: Intersección parcial / Inclusión parcial.
Caso C: Inclusión total.

Aplicando la desigualdad triangular (1), pasemos ahora a analizar los 3 casos:

Caso A: Si la intersección es vacía, quiere decir que la distancia de la clave al pivote es mayor a la suma del rango y del radio:

$$d(x,\ P)\ > rango\ +\ r_p \qquad (2)$$

En este caso, ningún vecino se podría encontrar dentro de la burbuja del pivote, ya que la distancia a la clave excedería por lejos el radio de búsqueda.

Caso B: En el caso de la intersección parcial, parte del área de las dos burbujas se solapan. Esto se traduce en el opuesto de (2):

$$d(x,\ P)\ \leq rango\ +\ r_p \qquad (3)$$

Eso significa, que pueden existir vecinos dentro de la burbuja del pivote. Pero asimismo, también pueden existir fuera, dado que la intersección es parcial.

Caso C: En el caso de la inclusión total del área de búsqueda dentro de la burbuja del pivote, observamos que para que un registro sea vecino, necesariamente tiene que estar dentro de la burbuja del pivote:

$$d(x, P) + rango < r_p \qquad (4)$$

Esto se puede traducir como el opuesto de A, ningún vecino se puede encontrar fuera de la burbuja del pivote.

Ahora estamos listos para encarar la búsqueda por rango.

2.2.1.2.1 Búsqueda por rango

La búsqueda por rango es una búsqueda pre-order del árbol, con las siguientes particularidades:

- Se busca primero en el nodo, registrando los vecinos.
- Sólo se entra en el nodo izquierdo si no se cumple (2).
- Sólo se entra en el nodo derecho si no se cumple (4).
- En los nodos hojas se registran los registros vecinos.

2.2.1.2.2 Búsqueda por rango con relajación

Es similar a la búsqueda por rango, pero en vez de fallar de inmediato, vuelve a probar con un rango nuevo, igual a la suma del rango con la relajación, los registros resultantes se agregan a una segunda lista.

Si bien esto es similar a la búsqueda por rango con rango+relajación, nos permite tener una lista a los registros dentro del rango, más una lista de registros obtenidos con el radio relajado.

2.2.2 Altas

Se recorre el árbol como si se tratara de una búsqueda puntual, hasta encontrar el primer nodo con espacio libre. Se insertará allí.

Si el nodo se encuentra lleno, y no es una hoja, se prosigue con el nodo correspondiente, siguiendo el criterio de la burbuja. De este modo, no se producen overflows en los nodos internos. En el caso de que no tenga el hijo correspondiente, se crea una nueva hoja y se llena hasta la carga mínima, usando los contenidos del nodo (de acuerdo al criterio de la burbuja). Puede ocurrir que no se pueda llegar a cubrir la capacidad mínima de la hoja. En este caso se reestructura el nodo.

Una vez insertado el registro nuevo en el nodo, de tratarse de una hoja, puede ocurrir que el nodo una vez serializado exceda el tamaño de bloque. En este caso, se tratará de un overflow.

2.2.3 Bajas

Se recorre el árbol como si se tratara de una búsqueda puntual.

Si no se encuentra el registro en un nodo interno, se prosigue de acuerdo con el criterio de la burbuja. Si ya se encuentra en una hoja, se reporta inexistente.

Una vez eliminado el registro, puede ocurrir que el nodo serializado quede con menos de la carga mínima. En este caso, se trata de un underflow.

2.2.4 Modificaciones

Es similar a una alta, pero se reemplaza el contenido del registro por el nuevo.
Se puede llegar tanto a una situación de desborde como a una de underflow.

A continuación mostraremos los métodos de resolución de overflow y de underflow.

2.2.5 Resolución de Overflow

En este caso, lo que se busca resolver es el problema de un nodo con más registros de los que puede almacenar. Debido a la estructura de este árbol, sólo se puede producir en una hoja, lo que se resuelve con una partición:

Se promueve la hoja a nodo interno y se genera un pivote, usando el criterio del hipercubo. Luego se calculan las distancias de los registros al pivote y se ubican en un heap de mínimos.

El primer tercio se ubica en el nuevo nodo izquierdo, registrando la distancia más grande, y el último tercio en el nodo derecho, registrando la menor distancia.

El resto se queda en el nodo. El radio se calcula entonces, usando el promedio de las distancias del último registro del nodo izquierdo, y el primer registro del nodo derecho.

Esto nos asegura que los siguientes próximos registros que se agreguen se repartan de manera uniforme entre los hijos.

2.2.6 Resolución de Underflow

En este caso, distinguimos el problema, según se trate de un nodo interno o de una hoja.

En el caso de una hoja, si el nodo padre se halla como mucho cargado a ⅔ de la capacidad máxima, se traen todos los registros de la hoja, liberándola. Si el otro nodo también se encuentra vacío (referencia vacía), se convierte el nodo padre a hoja (eliminando el pivote y radio, y cambiando de valor la bandera indicadora al valor de nodo hoja).

Si en cambio, el nodo padre se encuentra a más de ⅔ de su capacidad, se busca el primer registro que se pueda bajar al nodo en underflow. En caso de no encontrarse, se desencadena una reestructuración del nodo.

En el caso de un underflow en un nodo interno, se carga el hijo izquierdo. Si no tiene carga mínima, se trae un registro. En caso contrario, se busca hacer lo mismo con el derecho.
Si también resulta con carga mínima y ambos son hojas, se traen los contenidos y se convierte el nodo a hoja.

Si en cambio, los hijos fueran nodos internos, y ambos estuviesen con carga mínima, se traerá un registro de uno de ellos, generando un underflow que se resolverá de la misma manera. Dado que todo recorrido en el árbol termina en las hojas, se garantiza que el método siempre termina con éxito.

2.2.7 Reestructuración de nodos

En este caso, ocurre que un nodo termina con más registros de un lado de la burbuja que del otro. Esto significa que ocurrió un desequilibrio, y es necesario seleccionar otro pivote.

Se revisa el nodo que no haya provocado la reestructuración.
Si se trata de una hoja, se traen los contenidos, y se selecciona el nuevo pivote, tratando de repartir todos los registros de manera equitativa.

En el caso de un nodo interno, se copia el contenido del mismo y se intenta reinsertar los registros del nodo desencadenante y del padre. Esta es una suerte de “rotación”, que no sólo soluciona el problema, sino que cualquier reestructuración que ocurra a consecuencia de la reinserción de los registros, alcanzará a menos registros, garantizando que siempre termina con éxito.

3 Persistencia en disco

Para poder persistir el árbol de manera eficiente en disco, se utilizan nodos con un tamaño fijo de bloque, consistente en una potencia de 2 por 512 bytes. 512 bytes corresponden a la menor unidad de asignación física del disco. Debido a eso, los nodos poseen una gran capacidad, por lo que cada nodo puede almacenar una gran cantidad de registros. Debido a que los registros son de longitud variable y para reducir las posibilidades de que una modificación provoque el desborde de un nodo hoja, se reserva una parte del nodo, llamada “colchón”. Este espacio no se usa para las altas normales (ni se toma en cuenta en el cálculo de la carga mínima).

Los nodos se persisten con los siguientes datos:

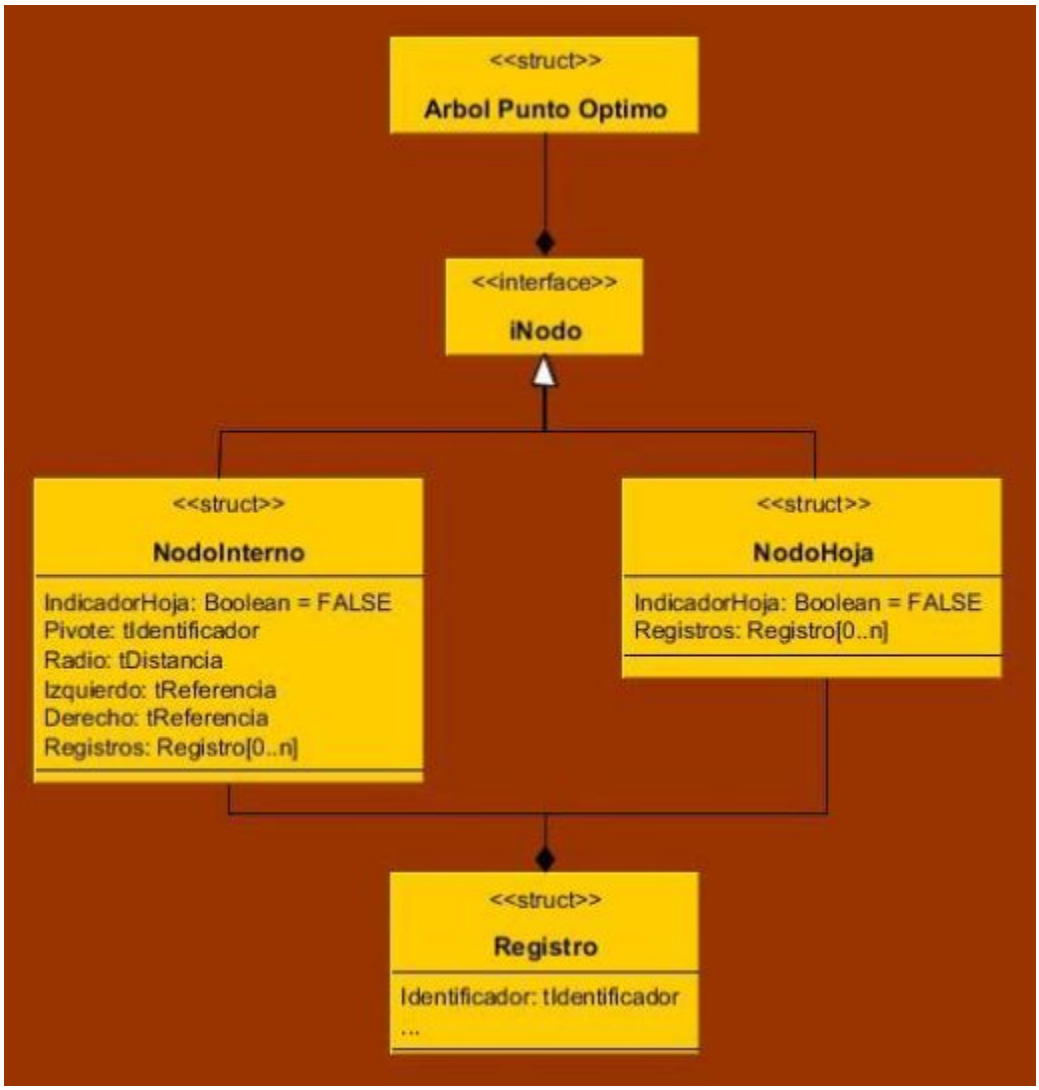
- La bandera indicadora de hoja.
- Una lista de offset a los registros (termina con un elemento en 0).
- Los registros.
- Para el caso de los nodos internos, se guardan además el pivote, el radio y las referencias a los nodos izquierdo y derecho.

En el caso de persistir la raíz, además se tiene en cuenta lo siguiente:

- Se guarda en el bloque lógico 0.

4 Diagrama de alto nivel

Esquema de la estructura del árbol a nivel lógico:



Donde tIdentificador es el tipo del identificador de la clave del registro.
tReferencia es un tipo que permite almacenar números de nodo.
tDistancia es un tipo que permite almacenar y comparar distancias.

5 Conclusiones

El árbol de punto óptimo no es una estructura pensada para guardar datos maestros o inclusive, para un uso intensivo de las altas y bajas. Como se pudo ver en los apartados de los métodos ABM, la reestructuración de un nodo es una operación muy costosa, debido a la lectura/escritura de muchos nodos, sumado al uso intensivo de la función distancia.

Asimismo, el rendimiento del árbol está condicionado por la distribución de las claves, la selección del pivote, y la forma de evaluar de la función distancia, además de la conversión de la clave en coordenadas del espacio métrico y viceversa.

6 Ejemplos

Léase:

NH por indicador de estado de nodo en modo Nodo Hoja.
NI por indicador de estado de nodo en modo Nodo Interno.

Los casos no consideran la bufferización de la raíz (nodo 0).

6.1 OVERFLOW EN UNA HOJA CON CAPACIDAD MAX 6 REG Y CARGA MIN 1/3

–----> K7 0: NH K1,K2,K3,K4,K5,K6

0: NI 1 P,R K3,K4,K5 2

1: NH K1,K2 2: NH K6,K7

K1,K2 representan el tercio de las claves más cercanas al pivote P
K6,K7 representan el tercio de las claves más lejanas al pivote P

lecto-escrituras de nodos: E1,E2,E0

6.2 RE-ESTRUCTURACION CON CAPACIDAD MAX 6 REG Y CARGA MIN 1/3
UNDERFLOW EN HOJA DERECHA
NODO INTERNO A SU IZQUIERDA

0: NI 1 P,R K1,K2,K3,K4,K5,K6 2

1: NI 3 P',R' K7,K8,K9,K10,K11,K12 4 2: NH K13

0: NI 1 P',R' K7,K8,K9,K10,K11,K12 2

1: NI 3 P',R' K1,K2,K3,K4,K13 4 2: NH K5,K6

K1,K2,K3,K4,K13 resultan ser las claves dentro de la esfera de centro P' y radio R'
K5,K6 resultan ser las claves fuera de la esfera de centro P' y radio R'

lecto-escrituras de nodos: L0,L1,E0,E1,E2

6.3 RE-ESTRUCTURACION CON CAPACIDAD MAX 6 REG Y CARGA MIN 1/3
UNDERFLOW EN HOJA DERECHA
NODO HOJA A SU IZQUIERDA

0: NI 1 P,R K1,K2,K3,K4,K5,K6 2

1: NH K7,K8,K9,K10,K11,K12 2: NH K13

0: NI 1 P',R' K2,K3,K4,K6,K13 2

1: NH K1,K5,K11,K12 2: NH K7,K8,K9,K10

K1,K5,K11,K12 representan el tercio de las claves más cercanas al pivote P'
K7,K8,K9,K10 representan el tercio de las claves más lejanas al pivote P'

lecto-escrituras de nodos: L0,L1,E0,E1,E2

6.4 UNDERFLOW EN HOJA CON CAPACIDAD MAX 6 REG Y CARGA MIN 1/3 PADRE COMPLETO

0: NI 1 P,R K1,K2,K3,K4,K5,K6 2

1: NH K7,K8,K9,K10,K11,K12

2: NH K13

0: NI 1 P,R K1,K3,K4,K5,K6 2

1: NH K7,K8,K9,K10,K11,K12

2: NH K2,K13

K2 primera clave que se encuentra fuera de la esfera de centro P y radio R

lecto-escrituras de nodos: L0,E0,E2

6.5 UNDERFLOW EN HOJA CON CAPACIDAD MAX 6 REG Y CARGA MIN 1/3 PADRE CON CARGA MINIMA

0: NI 1 P,R K1,K2 2

1: NH K3,K4.K5,K6,K7,K8

2: NH K9

0: NI 1 P,R K1,K2,K9 -1

1: NH K3,K4.K5,K6,K7,K8

2: libre

lecto-escrituras de nodos: L0,E0

6.6 UNDERFLOW EN NODO INTERNO CON CAP MAX 6 REG Y CARGA MIN 1/3 HIJOS HOJA CON CARGA MIN

0: NI 1 P,R K1 2

1: NH K2,K3

2: NH K4,K5

0: NH K1,K2,K3,K4,K5

1: libre

2: libre

lecto-escrituras de nodos: L1,L2,E0

6.7 UNDERFLOW EN NODO INTERNO CON CAP MAX 6 REG Y CARGA MIN 1/3 HIJOS INTERNOS UNO CON CARGA MIN

0: NI 1 P,R K1 2

1: NI 3 P',R' K2,K3,K4.K5 4

2: NI 5 P'',R'' K6,K7 6

0: NI 1 P,R K1,K5 2

1: NI 3 P',R' K2,K3,K4 4

2: NI 5 P'',R'' K6,K7 6

K5 última clave del hijo con carga no mínima

lecto-escrituras de nodos: L1,E1,E0

7 Bibliografía

https://en.wikipedia.org/wiki/Vantage-point_tree
http://pnylab.com/pny/papers/vptree/main.html
http://lbd.udc.es/Repository/Thesis/1348132590960_tese_luis_g_ares.pdf